UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Dottorato di Ricerca in
Automatica e Ricerca Operativa

MAT/09

XIX Ciclo

Models and Algorithms for Combinatorial Optimization Problems arising in Railway Applications

Valentina Cacchiani

Il Coordinatore
Prof. Claudio Melchiorri

I Tutor
Prof. Paolo Toth e Prof. Alberto Caprara

A.A. 2003–2006

Keywords

Train Timetabling

Train Unit Assignment

Integer Linear Programming

Linear Programming Relaxation

Lagrangian Relaxation

Column Generation

Constraint Separation

Branch and Cut and Price

Heuristic Algorithm

Contents

Acknowledgments

First of all, I want to thank my Tutors, Professor Paolo Toth and Professor Alberto Caprara, for giving me the great opportunity of working with them, for showing me every day the desire of tackling new problems and for encouraging me with their precious teaching. They have given me important responsibility in applied projects, and have always pushed me to obtain better results. This has contributed a lot to my research. They have always been ready to answer my questions, and have shown me a very strong enthusiasm for doing research. They have been and are a model to be followed.

Many thanks go to all the other members of the Operations Research group at the University of Bologna for their professional help and friendship. In particular, I want to thank Professor Silvano Martello, Professor Daniele Vigo and Professor Andrea Lodi for their important suggestions in my research. I want to thank my colleagues Dr. Michele Monaci, Dr. Manuel Iori, and Ph.D. Enrico Malaguti, who have shared my office with me during my Ph.D. and have been very nice and friendly. A particular thanks goes to Claudia D'Ambrosio, currently a Ph.D. student in our group, who is always very helpful when I have had difficulties. I want to thank Laura Galli and Andrea Tramontani, two Ph.D. students in our group; moreover, I don't want to forget the people at DEIS at the University of Bologna for their important work and for being nice and friendly during these years.

I also want to thank all of those people who have worked for the realization of the European Project Partner and Arrival, and have helped in stating real-world problems, which are studied in my Ph.D. thesis.

During my Ph.D. I spent some months abroad at the Massachusetts Institute of Technology in Boston (USA), working with Professor Dimitris Bertsimas. Even if this work is not included in this thesis because it concerns a health care problem that is far from the main topic of

my thesis, I want to thank Professor Bertsimas for this period of research and for extending my view of Operations Research through new techniques and approaches. I want to thank other researchers at MIT and Harvard Medical School, who have worked with me on this health care problem, given me very important suggestions for my research activity and given me a great satisfaction for my work, in particular Professor Thomas Bortfeld, David Craft, Omid Nohadani and Timothy Chan. I want also to thank other researchers at MIT, who have stimulated my curiosity and have given me new hints for future research, in particular Professor Andreas Schulz, Dan Iancu, Nicoleta Neagu and Dan Stratila.

Bologna, March 8, 2007

Valentina Cacchiani

List of Figures

List of Tables

Preface

This Ph.D. thesis includes the research activity developed together with my Advisors Prof. Paolo Toth and Prof. Alberto Caprara during my Ph.D. course in "Aut. and O.R." ("Automatica e Ricerca Operativa") at the University of Bologna.

The topic of my thesis concerns railway optimization problems, and arises from real-world applications studied in collaboration with some Italian Railway companies and within European Commission Projects.

The study of applied problems has given me both the opportunity of facing the difficulties that arise when dealing with real-world problems and the possibility of finding solutions to problems that are not only of academic interest.

Two main problems are examined in this thesis, and for both of them we proposed mathematical models and implemented algorithms which provide satisfactory solutions.

We introduce the problems and give a literature review in Chapter 1.

The first problem we present is the Train Timetabling Problem. It is a very important problem in the railway system planning, because it strongly affects the customers' satisfaction. In its basic version, it aims at determining an optimal timetable for a set of trains, on a single one-way line, satisfying some track capacity constraints.

For this problem, we first describe the basic version, where a single one-way line is considered. We propose an Integer Linear Programming (ILP) formulation, based on a graph representation of the problem. As the resulting model has exponentially many variables and a huge number of constraints, we solve the corresponding Linear Programming (LP) relaxation by means of column generation and constraint separation techniques. We then present some constructive heuristic algorithms based on the LP-relaxation of the proposed ILP model. These algorithms are able to find better solutions than those obtained by previous methods based on different models and Lagrangian relaxation of the constraints. We also describe an exact branch-and-cut-and-price algorithm that finds the optimal solution for small-size real-world instances.

Two other methods for separating the huge number of constraints are then described. These procedures take into account some particular cases that can occur in real-world appli-

cations of the Train Timetabling Problem. Moreover, they are able to solve a more general class of problems, concerning the determination of the Stable Set of Maximum Weight and the Clique of Maximum Weight in a class of particular graphs, given by the edge intersection of a comparability graph and a k-partite graph.

The column generation approach to the Train Timetabling Problem for the single one-way line, together with the branch-and-cut-and-price algorithm, based on a simple constraint separation, the constructive heuristics, and the other methods for the separation are described in Chapter 2.

In Chapter 3, we consider the extension of the problem to a railway network. For this extended problem, we concentrate mainly on freight transportation, even if the method is general and can deal with passenger trains as well. We propose a graph formulation of the problem and an Integer Linear Programming model. The problem is then relaxed in a Lagrangian way and a subgradient procedure is used to find near-optimal Lagrangian multipliers. At each iteration of the subgradient procedure, we apply a constructive heuristic algorithm, based on this Lagrangian relaxation, and a refinement procedure, which provide good feasible solutions in short computing times.

Finally, in Chapter 4 we present the second problem: the Train Unit Assignment Problem. This is another very important problem for railway system planning: it also affects the customers' satisfaction and concerns a long term and expensive choice, because the composition of the fleet is decided consequently. We propose a graph formulation for the problem and some Integer Linear Programming models. We consider the Linear Programming relaxation of one model, which contains a class of inequalities describing the convex hull of the corresponding constraints. The LP-relaxation is solved by column generation and a constructive heuristic algorithm is derived. We then present a Lagrangian relaxation of another model and build a constructive heuristic algorithm based on this relaxation. Finally we present an alternative model for obtaining a lower bound on the optimal value, which is based on some properties of the comparability graphs.

Chapter 1

Introduction

1.1 Railway Optimization

Railway systems present many planning and operational problems that can be studied by Operations Research. In the last years the reorganization of the Railway system in Europe has brought an increasing interest in the study of automated decision support tools capable to solve optimization problems with applications in railways. Many European Projects have focused on modelling and solving problems arising in this field, according to the regulations of the European Commission, which specify that the management of the infrastructure should be the responsibility of the governments, but operating trains should be carried out by independent Train Operators on a commercial basis. The use of mathematical models and of solution techniques coming from Operations Research allows one to obtain fast, close to optimality and effective solutions to the problems and a consequent better use of the Railway Infrastructure. Nevertheless, a lot of research is still to be carried out, since railway optimization problems are highly complex.

The design of a complete railway system is highly complex and it is naturally divided into sub-problems, which are not independent and the choices taken to solve one of the sub-problems heavily influence the others. Most of these problems turn out to be NP-hard and have a huge size, but the decomposition helps to apply mathematical techniques in order to find more effective solutions.

Among railway optimization problems, the most known ones concern the main phases that are needed in the complete construction of a Railway System. These phases are performed in sequence since the solution of a problem containing all the different features would be too complex. The most studied areas consider Line Planning, Train Timetabling, Train Platforming, Rolling Stock Circulation, Train Unit Shunting and Crew Planning.

For surveys on railway optimization problems see, e.g., [10, 16, 20, 24, 32].

Line Planning has the aim of deciding which are the routes for the passenger trains, specifying the types and the frequencies of the trains on each route.

Train Timetabling is the following phase, in which the actual timetable for each train has to be fixed, according to the requests of the Train Operators but satisfying some operational constraints.

Once the timetables are fixed, trains have to be assigned to platforms in the stations: this phase is called Train Platforming.

Rolling Stock Circulation deals with the assignment of locomotives and carriages (or self-contained train units) to the trains, with a fixed timetable and platform.

Outside the rush hours, the rolling stock is to be parked on a shunting area: this process is called Train Unit Shunting.

Finally, the rosters of the drivers and conductors have to be defined in order to operate the specified timetable: this process is studied by Crew Planning.

In this chapter, we focus on these most studied railway problems and give a description of the models and solution techniques which have been proposed to deal with them. A more complete description is dedicated to the Train Timetabling Problem and the Train Unit Assignment Problem, which are the main part of this Ph.D. thesis.

1.1.1 Line Planning

Many European operators of passenger trains operate a cyclic timetable, i.e. a timetable in which the scheduled trains can be partitioned into subsets such that the trains in each subset have the same routes and the same stop stations and the difference between the departure times is one cycle time (usually one hour).

Each subset of trains is called a line, and trains belonging to the same line differ only for the departure and arrival times, according to the cycle.

The frequency of a line denotes the number of trains that run in each direction per cycle on the common route.

The main aim of the Line Planning Problem is to design a line system such that all travel demands are satisfied, and two conflicting objectives are possible: one is to maximize the service for the passengers and the other is to minimize the operational costs of the system.

When designing a line system, it is important to maximize the number of passengers that can travel from their origin to their destination without having to change train; this suggests the use of long lines. However, the use of long lines turns out to induce problems such as the propagation of delays. Thus, a trade-off between the two objectives has to be made. Moreover, in order to satisfy the passenger demand, one can choose either a high frequency of the line or trains with a high capacity.

There is a limited set of papers dealing with the Line Planning Problem and most of them deal with one line type only.

1.1.2 Train Timetabling

The general aim of the Train Timetabling Problem is to determine an optimal timetable for a set of trains on a railway network, satisfying a set of operational constraints, mainly due to safeness rules.

Two main paths appear in the literature: one considers the cyclic version of the problem and the other the noncyclic one.

In the cyclic timetabling, each trip is operated in a cyclic way, according to the period of the timetable (e.g. one hour). This is an advantage for the passengers, because it helps to remember the timetables. However, it produces higher costs.

The noncyclic timetabling is relevant for scenarios with a great traffic and a competitive environment among Train Operators. The Infrastructure Manager collects all the requests of timetables from the Train Operators and constructs the overall actual timetable. Each Train Operator associates a profit to each train, indicating the priority of the train and gives his ideal timetable, defining the departure and arrival times for the visited stations, and the allowed changes. The Infrastructure Manager constructs the optimal timetable by maximizing the overall profit, given by the global priority of the scheduled trains decreased by a penalty function that takes into account the modifications with respect to the ideal timetable.

1.1.3 Train Platforming

The Train Platforming deals with the decision of the routes that each scheduled train must follow when entering, stopping and then leaving a station. In particular, each train has to perform a path from the point where it enters the station to the point where it leaves the station. This path passes through a platform.

Given a set of directions for train arrivals and departures, a set of platforms and a set of trains, with the corresponding timetable, directions and set of platforms where the train can stop, the problem determines the optimal routing of the trains, avoiding incompatible routes.

There are two main types of constraints which avoid incompatibilities: one imposes a minimum time interval between the departure of a train from a platform and the arrival of the next train at the same platform; the other one forbids overlapping between routes, i.e. the occupation of two incompatible routes by two trains in the same time instant.

The problem calls for the determination of the arrival and departure routes, together with the assigned platform, and with the arrival and departure times that may be slightly different from the ideal ones.

1.1.4 Rolling Stock Circulation

A Train Operator has to decide on the type and the number of rolling stock units to be assigned to the scheduled trains, with the constraint of satisfying the passenger demand. This is an important choice, also because this problem can be used to determine the composition of the fleet, which is a long term and very expensive choice.

Two main cases concerning the equipment can occur: the first one considers locomotives and carriages while the second one considers aggregated modules called train units.

In the first case the problem calls for determining for each trip the locomotive types and their number, and the carriage types and their number. These numbers depend on each other as the locomotives must have enough power.

In the second case the types of the train units must be determined and their number.

In order to satisfy the passenger demand, it is possible to change the equipment by adding or removing equipment from the trains, even if this operation is expensive.

Another important distinction that can be made concerns the nature of the railway network and the decisions about maintenance: in a sparse network with long distances it is necessary to take into account the maintenance operations and build a schedule for them; on the contrary, in a dense network, the maintenance operations can be handled easily, since there are time intervals when two identical rolling stock units are at the same station, and consequently they can be exchanged.

A second difference regarding sparse and dense networks concerns the seat requests: namely, in a sparse network, usually a seat reservation system exists, and then the requests are known; in a dense network, this is not the case, and decisions must be taken without this knowledge.

1.1.5 Train Unit Shunting

Besides the rolling stock circulation, which determines the operating phase during the rush hours, where locomotives and carriages are covering the timetabled trips or are under maintenance, the train unit shunting determines the parking phase of the rolling stock during the night hours.

In particular, it concerns the routing of the rolling stock between the station area and the shunting area, the short term maintenance and the inside and outside cleaning.

The problem considers a railway station and the corresponding shunting area, it takes a timetable for each train, indicating the arrival or departure time and platform and its composition and the routing costs from each platform to each shunting track and viceversa. It determines a matching of the arriving train units that have to be parked on a shunt yard with the departing train units, such that the overall cost is minimized.

1.1.6 Crew Planning

The last problem to be faced is to build the workload for the crew (drivers and conductors), in order to cover the planned timetable and with the objective of minimizing the number of crews.

The problem takes in input the planned timetable for the train services to be performed every day of a time horizon. Each train service is split into a sequence of trips, which correspond to journeys that must be executed by the same crew without rest.

Each trip has a departure time and an arrival time, and a departure station and an arrival station. Each crew performs a sequence of trips, by satisfying a set of rules given by the contracts. This sequence of trips is called a roster.

The Crew Planning Problem aims at determining a set of rosters such that each trip is covered, all the operational constraints are satisfied and the cost is minimized.

The problem is usually solved by a two phases approach. First a convenient set of duties covering all the trips is constructed: each duty represents a sequence of trips to be covered by a single crew within a short time period (e.g. one day). This first phase is called Crew Scheduling. Then, the duties selected in the first phase are sequenced to obtain the final rosters. This second phase is called Crew Rostering.

1.2 Train Timetabling Problem

Management of main railway lines is increasingly becoming an important issue of European transport systems. Several reasons motivate better usage and planning of the rail infrastructure, particularly on the so-called European corridors, where track resource is limited due to greater traffic densities, and competitive pressure among the train operators is expected to increase in the near future. The availability of effective, computer-aided tools to improve the planning ability of railways over traditional methods is consequent to the new market scenario. First, the re-organization of the European rail system, following the EU policy directives, has separated the activities of the Infrastructure Manager (who is responsible for train planning and real-time control) from the train operators (who provide rolling stock and transport services). This introduces the so-called access-to-infrastructure problem, in which several operators request capacity on a common railway line. Second, European railways are being transformed into more liberalized and privatized companies, which are expected to compete on a more profit-oriented basis. Third, the rail transport system is subject to increasing pressure by governments and social interest groups to improve its overall efficiency and quality of service. Finally, the strategic character of the sector is highlighted in view of ecological impacts and national policies aiming at spilling freight traffic shares from roads to

rails.

In this context, the ability to undertake infrastructure planning in a very timely, smooth and efficient way is becoming one of the most important tasks of the infrastructure manager, who at the same time has to optimize the use of the infrastructure and to provide track allocation, through rational and transparent procedures, as required in normative directives, introduced in the European Union since the early 1990s.

The general aim of Train Timetabling Problem is to implement scheduling algorithms which can provide a timetable plan on heavy-traffic, long-distance corridors and more generally on railway networks. Through this model the Infrastructure Manager can allocate "optimally" the paths requested by all Train Operators and proceed with the overall timetable design process, possibly with final local refinements and minor adjustments, as in the tradition of railway planners. In brief, this allows each Train Operator to submit requests for paths on the given railway line or network, and allows the Infrastructure Manager to collect all the requests, run the optimization algorithm to allocate (if possible) all of them at maximum profit, and eventually respond to Train Operators with the proposed plan of track allocation and relative "access fees".

The essential characteristics of the process can be summarized as follows:

- train paths are given a value (i.e., a priority), an ideal timetable, with ideal departure and arrival times, and tolerances within which they can be "moved";
- the optimal allocation is found by maximizing the difference between the values of the trains scheduled and a cost penalty function, which takes into account the deviations from the ideal timetables;
- a subset of paths can be fixed and shall not be moved, being either prioritary (as it is the case for long-term or clock-phased services) or already allocated (i.e., "sold" to some train operator company);
- the scheduling algorithm uses typical parameters as in the current timetable planning (e.g., minimum headway between trains, available tracks at each station);
- maintenance operations, also called "possessions", can impose constraints or forbid regular operations of the planned timetable during some intervals;
- signaling failures can cause some degraded operational mode, switching from an automatic to a manual block system.

In addition, under the assumption of competitive market, the process can be iterated if some operator does not accept the solution and asks for a re-evaluation by the infrastructure

manager, e.g., by using modified path values, since the paths allocated take into account the value or access fee that the train operator is willing to pay.

The train timetabling problem has received considerable attention in the literature.

We can distinguish between two main classes of problems: the cyclic timetabling and the noncyclic timetabling.

In the cyclic timetabling trains are grouped in subsets, where trains in each subset have the same route and their departure and arrival times differ for a fixed time period (cycle). An advantage of the cyclic timetables is that it is easier to remember the schedule of the trains for the passengers. On the other hand, this system is much more expensive because even in period of the day where a few trains are needed one must execute the same timetable used in the peak hours.

The noncyclic timetabling is particularly used for corridors or networks where the demand of trains to be scheduled is high, and many different Train Operators ask for scheduling trains in a competitive environment.

We next examine in details the different possible approaches for dealing with the Train Timetabling Problem, both in the cyclic and noncyclic cases.

Cyclic Timetabling

The cyclic timetabling was introduced by Serafini and Ukovich ([52]): they describe a mathematical model for the so-called Periodic Event Scheduling Problem (PESP). The PESP considers the scheduling of repetitive events under periodic time window constraints. The events are scheduled for one cycle (e.g. one hour) and then the cycle is repeated.

A timetable must satisfy a set of requirements which are basically related to safety regulations. In particular, they require that for any pair of trains using the same track a minimum headway time must be respected; moreover, they forbid overtaking along a track linking two consecutive stations and crossing of trains travelling in opposite directions along the same track.

These requirements can be modelled by using periodic constraints. In particular, each constraint defines a relation between two *events*, e.g. arrival of a train at a station and departure of another train from the same station. The constraint states that one event can occur only a certain time interval later than another one. For example, a minimum time interval between an arrival of a train at a station and a departure of another train from the same station must be respected. The time difference between two events can either be a fixed time or described as a time window. These time intervals are process times: for example, the travel time from a station to the next one, and the minimum and maximum stopping time at a station are process times.

Once defined the set N of all events that have to be scheduled, we introduce integer variables $v_i \in \{0, \ldots, T\}$, for each event $i \in N$, where T represents the time period. These variables represent the time instant at which the event $i \in N$ takes place.

Each constraint deals with a pair of events and with a time window, which imposes lower and upper bounds on the time interval between the two events. As the events occur in a cyclic way, the time windows must be considered modulo T (i.e. $mod(T)$).

A general constraint can be expressed as follows:

$$l_{ij} \leq (v_j - v_i) mod(T) \leq u_{ij} \tag{1.1}$$

where $i, j \in N$, v_i and v_j are the corresponding decision variables, and l_{ij} and u_{ij} are the lower and upper limits, respectively, for the process time.

The decision variables can assume integer values $\in [0, \ldots, T]$, due to the cyclic nature of the problem.

A timetabling model which includes the modulo operation is very hard to solve. Hence, the modulo operation is substituted by introducing a binary variable for each constraint: $p_{ij} \in \{0, 1\}$.

The constraints can then be rewritten as:

$$l_{ij} \leq (v_j - v_i) + p_{ij} T \leq u_{ij} \tag{1.2}$$

It is very useful to define the so-called *Constraint Graph* G for representing the problem. It has a node set N corresponding to the event set and an arc set A, where each arc corresponds to a constraint.

The PESP can be formulated as follows: given a set of events N, a cycle time T and time windows (l_{ij}, u_{ij}) for all $(i, j) \in A$, find a solution (v, p) such that

$$l_{ij} \leq (v_j - v_i) + p_{ij} T \leq u_{ij}, (i, j) \in A \tag{1.3}$$

$$v_i \in \{0, \ldots, T\}, i \in N \tag{1.4}$$

$$p_{ij} \in \{0, 1\}, (i, j) \in A \tag{1.5}$$

This is a feasibility problem, but can be easily extended by considering a function to be minimized, taking into account for example the costs that arise because of the use of the infrastructure.

Most models that consider cyclic train timetabling start from the PESP.

We review the most relevant works concerning the cyclic Train Timetabling Problem.

Schrijver and Steenbeek [54] developed a constraint propagation algorithm for solving PESP. Their algorithm has been implemented in the DONS system, that has become an indispensable tool in the Dutch long term railway timetabling process of Netherlands Railways and ProRail. Schrijver and Steenbeek also develop local optimization techniques to improve a feasible solution for fixed values of the variables p_{ij} in (1.3). Instances with up to 250 trains (all trains running in one hour of the Dutch timetable) can be solved usually within reasonable computing time.

In order to cope with the weak LP relaxation of models based on constraints (1.3), Nachtigall [43], Lindner [37], and Peeters [48] also describe a formulation of PESP based on cycle bases. This formulation is somewhat easier to solve than the formulation based on constraints (1.3), because of the lower number of integer variables and the somewhat better LP relaxation.

Nachtigall and Voget [44] use PESP to generate cyclic timetables with minimal passenger waiting times.

Odijk [46] uses the PESP at a strategic level to determine the capacity of the infrastructure around railway stations.

Kroon and Peeters [35] describe a PESP model including variable trip times. This may result in additional flexibility leading to a higher probability of obtaining a feasible solution.

Finally, Kroon et al. [34] describe a stochastic optimization variant of PESP. Their model explicitly takes into account stochastic disturbances of the railway processes, distinguishing between a planned timetable and several realizations of the timetable under pre-determined stochastic disturbances. The model can be used to allocate time supplements and buffer times to the processes in the planned timetable in such a way that the average delay of the realizations of the trains is minimal. In order to keep the computation times at an acceptable level, they start with an existing timetable and they fix the variables p_{ij}. They show that, by taking into account stochastic disturbances in the design of the timetable, an increase in robustness can be achieved.

Noncyclic Timetabling

When dealing with a competitive market, the noncyclic version of the problem becomes more appropriate, as several Train Operators ask for scheduling a set of trains according to a preferred timetable and it becomes harder to respect a fixed time period while satisfying most of the requests.

Moreover, when considering the scheduling of freight trains, one has usually to face the already planned timetable for the passenger trains, and it is not much relevant to have a periodic timetable for the freight trains, and sometimes not even possible.

We review the most relevant works concerning the noncyclic Train Timetabling Problem (TTP).

Szpigel [56] considers a variant of these models in which the order of the train departures from a station is not represented by binary variables but by disjunctive constraints. Small instances of the problem are solved by branch-and-bound by computing bounds through the relaxation of these disjunctive constraints.

Jovanovic and Harker [33] solve, by branch-and-bound techniques, a version of these models that calls for a feasible schedule rather than for the optimization of a suitable objective function.

Cai and Goh [14] illustrate a constructive greedy heuristic driven by one of these models.

Carey and Lockwood [18] define a heuristic that considers the trains one at a time (in appropriate order), and for each train solves a MILP analogous to these models in order to schedule the train optimally, keeping the path of the previously scheduled trains partially fixed. More precisely, the relative order of the train departures for these trains is kept fixed, whereas their arrival and departure times may be changed.

Higgins et al. [31] define local search, tabu search, genetic and hybrid heuristics, finding a feasible solution by using a model in the family above.

Brännlund et al. [8] discretize the time into one-minute *time slots* and subdivide the line into *blocks*. Operational constraints impose that two trains cannot be in the same block in the same time slot. They define an ILP model with a binary variable x_{it}^r each time the timetable constraints allow train r to be in block i in time slot t. This model is not suited for large size instances as those arising for the main European corridors.

Oliveira and Smith [47] model TTP as a special case of the Job-Shop Scheduling Problem, considering trains as jobs to be scheduled on lines regarded as resources, and present a hybrid algorithm devised under the Constraint Programming paradigm, showing how to adapt this framework in some special real-life applications.

Different ILP models based on a graph representation of the problem were presented by Caprara et al. [15], and Caprara et al. [17]. In both papers, time is discretized (i.e., expressed in minutes from 1 to 1440) and Lagrangian relaxation is used to derive bounds on the optimal solution value as well as to drive a heuristic procedure. This approach, whose main features are outlined in the following, produces good relaxations and heuristic solutions also for large-size instances.

Vansteenwegen and Van Oudheusden [57] study the problem of improving the passenger service, taking into account waiting times and delays, on a small part of the Belgian railway network.

Semet and Schoenauer [55] present an effective evolutionary algorithm for real-world train timetabling, where the main focus is on the reconstruction of the schedule following a small

perturbation, with the aim of minimizing the total delay.

Mistry and Kwan [42] present a cooperative coevolutionary train timetabling problem algorithm concerned with the automatic generation of planning timetables.

Cacchiani, Caprara and Toth ([11]) consider the (periodic and non-periodic) train timetabling on a corridor (single one-way line connecting two major stations). They present an ILP formulation in which each variable corresponds to a full timetable for a train. This is in contrast with previous approaches to the same problem, which were based on ILP formulations in which each variable is associated with a departure and/or arrival of a train at a specific station in a specific time instant, whose LP relaxation is too expensive to be solved exactly. They propose heuristic and exact algorithms for the problem, that are based on the solution of the LP relaxation of the ILP formulation. Experimental results on real-world instances of the problem show that the proposed approach is capable of producing heuristic solutions of better quality than those obtained by these previous approaches, and of solving some small-size instances to proven optimality. This approach will be described in details in Chapter 2.

Cacchiani, Caprara and Toth ([12]) study the problem of freight transportation in railway networks, where a certain number of passenger trains have a prescribed timetable which cannot be changed and train operators want to insert new freight trains, according to their *suggested* timetables. The freight trains can choose different routes in the railway network and their timetables can be changed in order to respect safeness operational constraints. The aim is to introduce as many new freight trains as possible, by trying to change the timetables as less as possible and by satisfying all the operational constraints. They present an Integer Linear Programming (ILP) formulation, and solve the problem by using a heuristic algorithm based on a Lagrangian relaxation of the ILP model, presenting computational results on real-world instances which show the effectiveness of their approach. This problem and the solution method are described in Chapter 3.

As an example of model for the noncyclic timetabling, we describe the model by Brännlund et al. [8].

Consider the time discretized in minutes and let x^r_{it} be a binary variable, where $x^r_{it} = 1$ if train r occupies block i at time t, and $x^r_{it} = 0$ otherwise. Let x^r denote the corresponding vector. Let T^r denote the set of vectors x^r, which result in technically and logically feasible schedules for train r, not considering the effect of other trains on the track (including the empty schedule, i.e. the train r is not scheduled). Let $v^r(x^r)$ denote the total value of a given timetable x^r for train r, where the value is given by the profit of the departure time minus a per minute cost for unnecessary waiting along the track.

The problem can be stated as follows:

$$\max \sum_r v^r(x^r) \tag{1.6}$$

$$\sum_r x^r_{it} \leq 1, \forall i, t \tag{1.7}$$

$$x^r \in T^r, \forall r \tag{1.8}$$

The constraints 1.7 impose that on each block, there can be only one train at each time period.

The constraints 1.8 ensure technical feasibility.

The objective function maximizes the total value of all the trains.

1.3 Train Unit Assignment Problem

The efficient circulation of railway rolling stock is a very important issue, also because the solution of the problem can be used for determining the composition of the fleet and changing the fleet is a long term and expensive choice. For the same reason, the rolling stock assignment cannot be changed very frequently. The choice of how many carriages or train units must be bought for covering all the scheduled trips with the corresponding seat demands is much relevant.

As described, given a set of timetabled trips and a set of locomotives and carriages or train units, the Rolling Stock Circulation Problem consists of finding the assignment of the train units (TUs) to the trips, with the aim of minimizing the number of locomotives and carriages or train units used and satisfying the seat demand.

This can be considered as the basic version of the problem: many variants can occur, based on the specific real-world problems. These variants contain many additional constraints that make the problem very difficult and increase the need of optimization tools for solving it.

For example, one is usually allowed to combine more train units, possibly of different types, in order to satisfy the passenger seat covering constraints on a given trip.

A first distinction can be done concerning the rolling stock circulation problem: some papers deal with locomotives and (locomotive-hauled) carriages while others study the so-called train units, which are self-contained trains with an engine and passenger seats.

The literature presents several variants of the problem, most dealing with specific real-life characteristics, which impose different classes of constraints. We next list some of the possible variants and briefly review the related literature.

Besides the basic version, other constraints can arise: for example, some trips can have only a subset of train unit types or locomotives that can be used to satisfy the request; some

stations can be the beginning or the end of a daily assignment, but not all of them (i.e., for capacity reasons).

Moreover, the train unit maintenance is often included in the solution approaches, as it is a very important issue. Namely, every fixed number of days, the train units must be recovered in a particular depot, where some maintenance operations are executed on the engine.

It is often the case that different types of maintenance must be distinguished: namely, a more frequent maintenance for cleaning the machines, and a longer one for repairing possible damages.

Another important element of this problem concerns the variety of the objectives. Not only the minimization of the number of train units is important: if the problem is infeasible, the aim is to obtain a solution which is not too far from the satisfaction of all the requests.

Moreover, it is important that every train unit travels about the same amount of miles, as this is one of the reasons that cause maintenance. Another expensive operation in terms of time is the coupling and uncoupling of the machines: sometimes, also this is view as an objective, in order to try to satisfy the requests but trying to avoid coupling and uncoupling operations, if not necessary.

Concerning coupling, it is important to underline that, in general, not all couplings are feasible, since some train unit types can be combined only with a subset of the other types, and this imposes additional constraints.

Every time that two trips are in sequence, the arrival station of the first trip must be the same as the departure station of the second one. However, there can be exceptions, also because otherwise a feasible solution might not exist. The exception consists in using the so-called deadhead trips. This means that the train unit is moved from a station to another one, in order to cover a trip, but without covering any trip along this travel.

The use of deadhead trips is not always possible, because of the capacity of the tracks and the other train units travelling along the tracks. Moreover, the use of deadhead trips is expensive, since the train units travel consuming power and the units themselves. Sometimes, this can as well be viewed as an objective, so as to minimize the deadhead mileage.

This problem, in all its variants, has found a lot of interest in the literature.

For surveys on the specific problem as well as on the use of combinatorial optimization techniques in railway planning see, e.g., [10, 16, 20, 24, 32].

Most of the approaches in the literature consider the case in which locomotives and (locomotive-hauled) cars have to be assigned to trains [9, 21, 22, 23, 24, 40, 51].

In particular, Brucker et al [9] consider the problem of routing railway carriages through a railway network, so that each service is satisfied according to the corresponding request of passenger seats; they don't consider the order of the carriages and minimize a non-linear cost function. The problem is solved through a simulated annealing procedure.

In [21], Cordeau et al present a locomotive and carriage assignment problem, which is formulated as a large integer linear program and solved by a Benders decomposition approach.

Cordeau et al. ([23]) extend the model by considering real-life aspects, such as maintenance operations and propose a heuristic branch and bound approach for solving it, where the LP-relaxation is solved by column generation techniques.

Lingaya et al. ([40]) present a model for operational management of locomotive-hauled railway carriages, where the order of the carriages is taken into account. They are given the order of the trains. The problem is solved using a Dantzig-Wolfe reformulation.

In [5], Ben-Khedher et al. consider the case in which there is a unique type of TUs. The objective is to maximize the expected profit for the company and the problem is solved by means of stochastic optimization, branch and bound and column generation.

There are a few references that consider the assignment of TUs: [1, 2, 26, 50, 53]. Most of them consider the case in which there is a very small number of distinct TU types (two in most cases). On the other hand, in most of these cases, the rules for composing TUs for a trip are quite difficult.

In [1], Abbink et al. present an integer linear programming formulation, with the objective of minimizing the seat shortages during the rush hours.

Alfieri et al. ([2]) propose an integer linear programming model to determine the rolling stock circulation for multiple rolling stock types, while satisfying the demand and with the aim of minimizing the carriage-kilometers. The problem is solved by decomposition into subproblems.

Schrijver ([53]) presents a problem where a single day workload is considered. The objective is to minimize the number of train units. The problem is formulated as an integer linear program and is solved by commercial solvers.

Peeters and Kroon ([50]) present a problem in which the train series concept is introduced: given two endpoints between which several trains run up and down according to the timetable, for a train series, the available rolling stock consists of the same material type with different subtypes, which differ in number of carriages and in capacity. The order of the units in a composition is considered. They take into account three evaluation criteria: the kilometer-shortages, the number of shunting operations and the carriage-kilometers. They model the problem by using a transition graph, which represents the set of feasible transitions between compositions. They solve the problem by using a Dantizg-Wolfe reformulation and applying a branch-and-price algorithm. They solve real-world instances of NSR (the main Dutch Train Operator) in very short computing times.

Fioole et al. ([26]) present a mixed integer linear programming model, that can be seen as an extended version of the model described by Schrijver ([53]). They apply several methods to improve the LP relaxation and manage to solve to near optimality complex instances by

CPLEX.

Cacchiani, Caprara and Toth ([13]) study a real-world train unit assignment problem for an operator running trains in a regional area. Given a set of timetabled train trips, each with a required number of passenger seats, and a set of train units, each with a given number of available seats, the problem calls for an assignment of the train units to trips, possibly combining more than one train unit for a given trip, that fulfills the seat requests. With respect to analogous case studies previously faced in the literature, this is characterized by the fairly large number of distinct train unit types available (in addition to the fairly large number of trips to be covered). They present an approach based on an ILP formulation in which the seat requirement constraints are stated in a "strong" form, derived from the description of the convex hull of the variant of the knapsack polytope arising when the sum of the variables is restricted not to exceed two. This approach is described in details in Chapter 4.

We describe, as an example of Train Unit Assignment Problem, the model proposed by Schrijver ([53]).

Let M be the set of rolling stock subtypes and $G = (V, A)$ the time-space graph whose arc set is given by the union of the trip arcs, denoted by A_T, and the inventory arcs, denoted by A_I. For every trip arc $a \in A_T$, the number of first and second class passengers is denoted by p_a^1 and p_a^2, respectively, and the maximum train length as ℓ_a. With every subtype $m \in M$, we associate the parameters q_m^1, q_m^2, c_m, and w_m, denoting, respectively, the capacities of first and second class seats, the cost and the length of a train unit of subtype $m \in M$. With every arc $a \in A$ and every subtype $m \in M$, we associate an integer variable x_a^m, representing the number of units of subtype m deployed on the trip if $a \in A_T$, and staying in the station if $a \in A_I$.

The objective pursued in the model is to minimize the global cost of the train units deployed on the trains. To this end, the number of train units on the night inventory arcs is minimized. Letting A_N be the set of night inventory arcs, and $\delta^+(v)$ and $\delta^-(v)$ the sets of arcs entering and leaving vertex v, respectively, the model reads as follows:

$$\min \sum_{m \in M} \sum_{a \in A_N} c_m x_a^m \tag{1.9}$$

subject to

$$\sum_{a \in \delta^-(v)} x_a^m = \sum_{a \in \delta^+(v)} x_a^m, \qquad v \in V, m \in M, \tag{1.10}$$

$$\sum_{m \in M} q_m^k x_a^m \geq p_a^k, \qquad a \in A_T, k = 1, 2, \tag{1.11}$$

$$\sum_{m \in M} w_m x_a^m \leq \ell_a, \qquad a \in A_T, \tag{1.12}$$

$$x_a^m \geq 0, \text{ integer}, \qquad a \in A, m \in M. \tag{1.13}$$

Constraints (1.10) are flow conservation constraints in each vertex of G. Constraints (1.11) impose that the capacity on each arc must not be less than the expected number of first ($k = 1$) and second ($k = 2$) class passengers. Constraints (1.12) impose that the sum of the lengths of the train units deployed on a trip does not exceed the maximum train length.

Chapter 2

A Column Generation Approach to Train Timetabling on a Corridor

2.1 Introduction

The train timetabling problem (TTP) is a fundamental problem in railway optimization, and has received considerable attention in the optimization literature in recent years, mainly due to the ongoing reorganization and privatization of European railways, which is a strong incentive towards efficient use of resources. The general aim of TTP is to provide a timetable for a number of trains on a certain part of the railway network. According to the current situation, an Infrastructure Manager is handling the railway network and receiving requests from Train Operators, concerning trains to be operated for a given time horizon. Each of these requests specifies a path for a train along with the arrival and departure times for all stations along the path. Given that these requests are mutually incompatible, the Infrastructure Manager has to solve a TTP, modifying the arrival and/or departure times of some trains (and possibly canceling some other trains), in order to come up with a proposed feasible scenario for the Train Operators, who may either accept it (given that they will pay less for the requests that were modified), or come up with new proposals. The process is iterated until the scenario proposed by the Infrastructure Manager is accepted by all Train Operators. We refer to [7] for a more detailed description of this process in the German case.

Given the practical relevance of the problem and the different organization rules of the various Infrastructure Managers, some variants of the problem were formulated and solved. Although it is out of the scope of this chapter to present in detail all the methods proposed in the literature to tackle these variants (we refer the interested reader to the survey by Caprara, Kroon, Monaci, Peeters and Toth [16]), they can be roughly classified according to the following criteria: (a) the method focuses on a single line or on a general network, and (b)

the method takes advantage or not of the fact that the solution to be constructed is periodic, if this is the case. As to (a), the reason for focusing on a single (main) line, often called corridor, is that, in many cases, once the timetable for the trains on the corridor has been determined, it is relatively easy to find a convenient timetable for the trains on the other lines of the network. On the other hand, there are other situations in which there are a few congested lines, and solving the problem on each line separately would be a too rough approach. As to (b), in some real cases (nearly) all trains are repeated every period (typically one hour). When this holds, not only the size of the problem is widely reduced by considering only one period, but also the mathematical formulation can be stronger since the "big M" needed to write disjunctive constraints can actually be as small as the period itself (see again, e.g., [16] for details). As a result, there are methods that take advantage of the periodicity and are not suited for cases in which the problem is non-periodic. At present, given their effectiveness, these appear to be the majority of the methods that have been tested on networks as opposed to single lines.

Among the methods that consider a single line and can be used also in the non-periodic case, Szpigel [56], Jovanovic and Harker [33], Cai and Goh [14], Carey and Lockwood [18], and Higgins, Kozan and Ferreira [31], present mixed ILP formulations in which the arrival and departure times are represented by continuous variables and there are binary variables expressing the order of the train departures from each station, and heuristic algorithms based on these formulations. Moreover Brännlund, Lindberg, Nöu and Nilsson [8], Caprara, Fischetti and Toth [15], and Caprara, Monaci, Toth and Guida [17] illustrate pure ILP formulations obtained by discretizing time and representing the problem on a suitable graph. Due to the large sizes of the instances considered, these approaches avoid the explicit solution of the associated LP relaxation, resorting to computationally faster approaches based on Lagrangian relaxation combined with subgradient optimization.

The references dealing with approaches tested on networks and that assume periodicity build on the seminal paper on the periodic event scheduling problem by Serafini and Ukovic [52]. Schrijver and Steenbeek [54] and Odijk [46] consider a simple but rather weak ILP formulation in which, again, there are continuous (bounded) variables representing arrival and departure times modulo the period, and there are binary variables expressing the order of the train arrival/departures at stations. The formulation is solved by branch-and-bound. Lindner [37], Peeters [48], Peeters and Kroon [49], Kroon and Peeters [35], and Lindner and Zimmermann [39] deal with analogous ILP formulations in which the binary variables are removed at the cost of introducing a larger and more complex set of constraints related with the notion of cycle bases. The resulting LP relaxations are much stronger, leading to much smaller solution times as discussed in [36], again by using branch-and-bound.

In this chapter, we illustrate a solution method for TTP on a single line, that can be used

also when the problem is non-periodic, although the case on which we focus is periodic with period one day. The method uses an ILP formulation that is a natural variant of the ones presented in [15, 17], and is based on the explicit solution of the LP relaxation of this ILP formulation. The LP relaxation is used to drive both heuristic and exact branch-and-bound algorithms. The chapter is organized as follows. The version of TTP we consider is defined formally in Section 2.2, and its canonical representation on a space-time graph, common to many of the approaches mentioned above, is described in Section 2.3. The new ILP formulation, with variables associated with paths (rather than nodes/arcs) in this graph, is given in Section 2.4, and the solution of its LP relaxation by separation and column generation techniques is discussed in Section 2.5. Sections 2.6 and 2.7 discuss heuristic and exact algorithms based on this LP relaxation, respectively. In Section 2.8 we report computational results on real-world instances of the problem, showing that our heuristics provide improved solutions with respect to previous approaches, whereas our exact method is capable of solving to proven optimality for the first time some of the instances in the testbed. Finally, in Section 2.9, we propose a stronger version of the overtaking constraints, involving an arbitrary number of trains, and the associated separation methods, whose study concerns the problem of computing Stable Set and Clique of Maximum Weight.

2.2 The Train Timetabling Problem

We face the same problem as in [15, 17], therefore its description is a synthesis of the one given in those references.

We consider a single, one-way track corridor linking two major stations, with a number of intermediate stations in between together with a set of trains that are candidate to be run every day of a given time horizon along the corridor. Let $S = \{1, \ldots, s\}$ represent the set of stations, numbered according to the order in which they appear along the corridor for the running direction considered, and $T = \{1, \ldots, t\}$ denote the set of candidate trains. For each train $j \in T$, a *first* (departure) station f_j and a *last* (destination) station l_j ($l_j > f_j$) are given. Let $S^j := \{f_j, \ldots, l_j\} \subseteq S$ be the ordered set of stations visited by train j ($j \in T$).

The *track capacity constraints* impose that overtaking between trains occurs only within a station. Furthermore, for each station $i \in S$, there are lower bounds a_i and d_i on the time interval between two consecutive arrivals and two consecutive departures, respectively. A *timetable* defines, for each train $j \in T$, the departure time from f_j, the arrival time at l_j, and the arrival and departure times for the intermediate stations $f_j + 1, \ldots, l_j - 1$. Each train is assigned by the train operator an *ideal timetable*, representing the most desirable timetable for the train, that may be modified in order to satisfy the track capacity constraints. In particular, with respect to the ideal timetable, one is allowed to modify (anticipate or delay)

the departure time of each train from its first station, and to increase (but not decrease) the stopping time interval at the (intermediate) stations. Moreover, one can also *cancel* the train. The timetable for train j in the final solution will be referred to as the *actual timetable*.

Observe that, differently from [15], we consider the version of the problem in which the travel time between consecutive stations must be the same in the ideal and actual timetables; see [17] for a comparison with the version in which this time can be increased as well. In this new version one can write the overtaking constraints in the ILP in a form that is much stronger for the LP relaxation than in the old version.

The *objective* is to maximize the sum of the profits of the scheduled trains, defined as follows. The *profit* achieved for each train $j \in T$ is given by $\pi_j - \alpha_j\nu_j - \gamma_j\mu_j$, where π_j is the *ideal profit*, that is the profit that is achieved if the train travels according to its ideal timetable, ν_j is the *shift*, that is the absolute difference between the departure times from station f_j in the ideal and actual timetables, μ_j is the *stretch*, that is the (nonnegative) difference between the travel time from f_j to l_j in the actual and ideal timetables (equal to the sum of the stopping time increases over all intermediate stations), and α_j, γ_j are given nonnegative parameters.

For convenience, we will use the acronym TTP to denote the specific problem defined above. It was shown in [15] that TTP is NP-hard in the strong sense, being a generalization of the well-known maximum stable set problem.

Note that it would be easy to impose in the model additional constraints on the train timetables, such as the requirement that a train does not arrive at a station later than a certain time instant, to guarantee a connection with a train on a different line whose timetable is fixed. However, we did not consider these constraints in our case study.

2.3 A Graph Representation

We next outline the representation of the problem on a graph described in [15, 17]. Times are here discretized and expressed as integers from 1 to $q := 1440$ (the number of minutes in a day), though a finer discretization would also be possible (e.g., 1/2, 1/4 minute) without changing the model, although the time and space complexity of the associated algorithms could increase considerably.

Let $G = (V, A)$ be the (directed, acyclic) space-time multigraph in which nodes represent arrivals/departures at a station in given time instants, and paths represent feasible timetables for trains, defined as follows (an illustrative picture taken from [15] is given in Figure 2.1). The node set V has the form $\{\sigma, \tau\} \cup (U^2 \cup \ldots \cup U^s) \cup (W^1 \cup \ldots \cup W^{s-1})$, where σ and τ are an *artificial source node* and an *artificial sink node*, respectively, whereas sets U^i, $i \in S \setminus \{1\}$, and W^i, $i \in S \setminus \{s\}$, represent the set of time instants in which some train can arrive at and

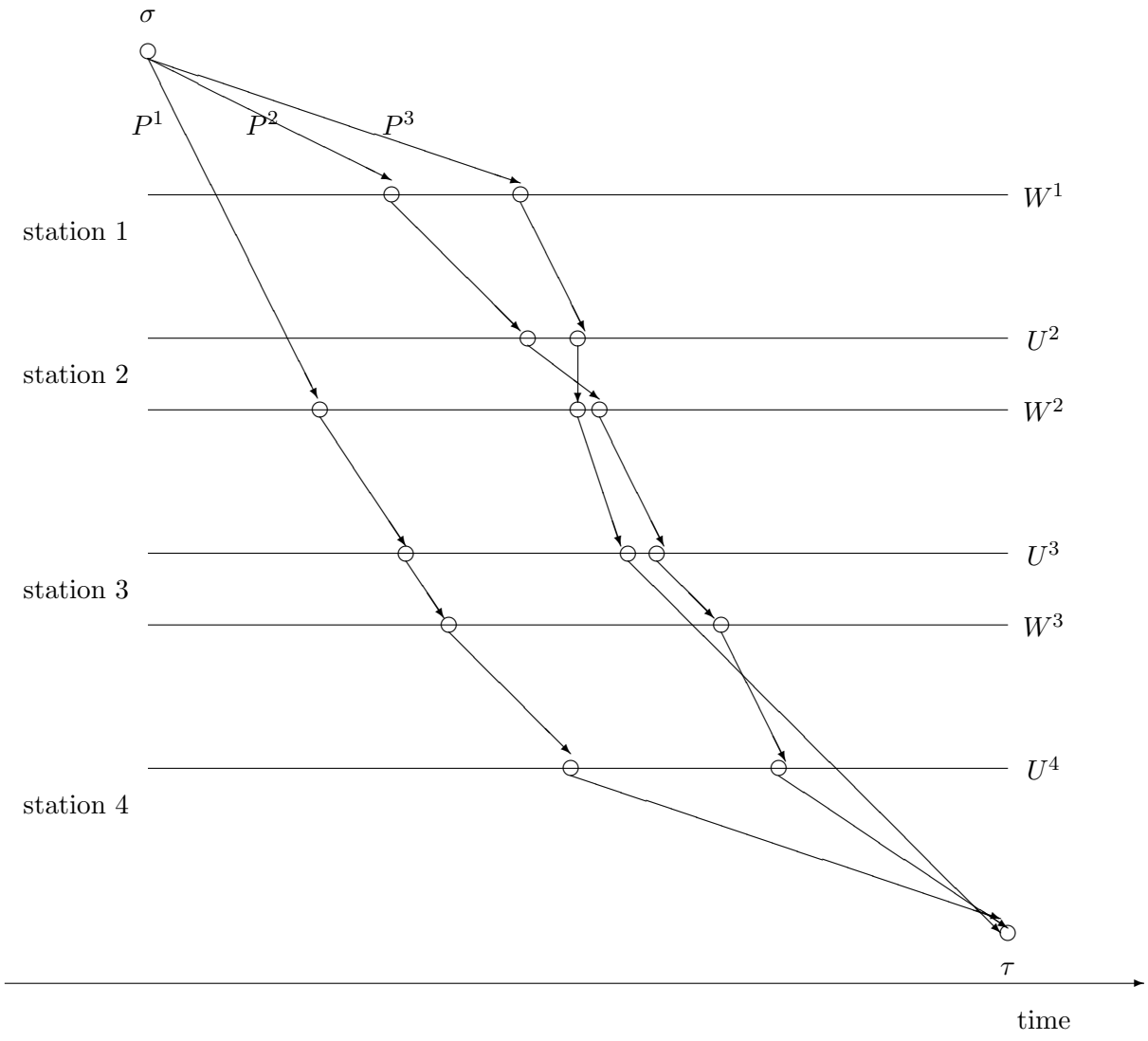

Figure 2.1: An example of train paths in graph G (with $s = 4$, $t = 3$, $f_1 = 2, l_1 = 4, f_2 = 1, l_2 = 4, f_3 = 1, l_3 = 3$).

depart from station i, respectively. We call the nodes in $U^2 \cup \ldots \cup U^s$ and $W^1 \cup \ldots \cup W^{s-1}$ *arrival* and *departure* nodes, respectively. Let $\theta(v)$ be the time instant associated with a given node $v \in V$. Moreover, let $\Delta(u,v) := \theta(v) - \theta(u)$ if $\theta(v) \geq \theta(u)$, $\Delta(u,v) := \theta(v) - \theta(u) + q$ otherwise. Because of the "cyclic" nature of the time horizon, we say that node u *precedes* node v (i.e., $u \preceq v$) if $\Delta(v,u) \geq \Delta(u,v)$ (i.e., if the cyclic time interval between $\theta(v)$ and $\theta(u)$ is not smaller than the cyclic time interval between $\theta(u)$ and $\theta(v)$).

Note that not all time instants correspond to possible arrivals/departures of train j at a station $i \in S^j$. Accordingly, let $V^j \subseteq \{\sigma, \tau\} \cup (U^{f_j+1} \cup \ldots \cup U^{l_j}) \cup (W^{f_j} \cup \ldots \cup W^{l_j-1})$ denote the set of nodes associated with time instants corresponding to possible arrivals/departures of train j in a positive-profit timetable (see [17] for a detailed definition of V^j).

The arc set A is partitioned into sets $A^1, \ldots, A^t$, one for each train $j \in T$. In particular, for every train $j \in T$, A^j contains:

- a set of *starting* arcs (σ, v), for each $v \in W^{f_j} \cap V^j$, whose profit is $p_{(\sigma,v)} := \pi_j - \alpha_j \nu(v)$, with $\nu(v) := \min\{\Delta(v^*, v), \Delta(v, v^*)\}$, where v^* is the node associated with the departure of train j from station i in the ideal timetable;

- a set of *station* arcs (u, v), for each $i \in S^j \setminus \{f_j, l_j\}$, $u \in U^i \cap V^j$ and $v \in W^i \cap V^j$ such that $\Delta(u,v)$ is at least equal to the minimum stop time of train j in station i, whose profit is $p_{(u,v)} := -\gamma_j \mu(u,v)$, with $\mu(u,v) := \Delta(u,v) - \Delta(u^*, v^*)$, where u^* and v^* are the nodes associated, respectively, with the arrival and departure of train j at station i in the ideal timetable;

- a set of *segment* arcs (v, u), for each $i \in S^j \setminus \{l_j\}$, $v \in W^i \cap V^j$ and $u \in U^{i+1} \cap V^j$ such that $\Delta(v,u)$ is equal to the travel time of train j from station i to station $i+1$, whose profit is $p_{(v,u)} := 0$;

- a set of *ending* arcs (u, τ), for each $u \in U^{l_j} \cap V^j$, whose profit is $p_{(u,\tau)} := 0$.

By construction, for each train $j \in T$, any path from σ to τ in G which uses only arcs in A^j and has profit p corresponds to a feasible timetable for train j having profit p.

To satisfy the track capacity constraints, one should impose that certain pairs of arcs, associated with different trains, cannot be selected in the overall solution. This is discussed in detail in the next section.

2.4 An ILP Model with Path Variables

The solution approaches in the literature that are based on the graph representation of the previous section define ILP formulations in which variables are associated with nodes and/or

arcs of the graphs [8, 15]. As the associated LP relaxations turn out to be extremely expensive to solve exactly, even by the state-of-the-art LP solvers, a heuristic solution of their dual is obtained by combining Lagrangian relaxation of the track capacity constraints with subgradient optimization.

In this section we illustrate an alternative solution approach, based on an alternative (and equally natural) ILP formulation in which variables are associated with paths in G.

In the new ILP model there is a binary variable for each possible path for a train, indicating if the path is chosen or not in the solution. Considering that for each train $j \in T$ the corresponding path can be shifted and/or stretched with respect to the *ideal path*, i.e., the path associated with the ideal timetable of train j, there is an exponentially large number of possible paths for a train, as will be discussed next. Therefore, our solution approach is based on column generation techniques, that are easy to apply as the column generation problem calls for an optimal path in the (acyclic) graph considered. Moreover, we will have a very large number of constraints, as was the case for the previous ILP formulation. These constraints will be handled by separation algorithms, which in this case have to deal with fractional solutions, differently from the case in which Lagrangian relaxation is used.

Recall the definitions of Section 2.3. Moreover, given two consecutive stations i and $i+1$ along with two trains j and k such that $i, i+1 \in S^j \cap S^k$, let

$$b_i^{jk} := \max\{d_i, a_{i+1} + r_j - r_k\}$$

denote the minimum time interval between a departure of j from i and a departure of k from i (in this order) in a feasible solution, where r_j and r_k are the travel times of j and k from i to $i+1$, respectively. Note that if j and k depart from i within a time interval smaller than b_i^{jk}, either their departures are too close in time, or their arrivals are too close in time, or k overtakes j between i and $i+1$.

For each $j \in T$, let $\mathcal{P}^j$ be the collection of possible paths for train j, each associated with a path from σ to τ in G containing only arcs in A^j (paths are seen as arc subsets) and having positive profit. Furthermore, let $\mathcal{P} := \mathcal{P}^1 \cup \ldots \cup \mathcal{P}^t$ be the overall (multi-)collection of paths, denoting by $p_P := \sum_{a \in P} p_a$ the actual profit for path $P \in \mathcal{P}$. Finally, let $\mathcal{P}_w^j \subseteq \mathcal{P}^j$ be the (possibly empty) subcollection of paths for train j that visit node $w \in V^j$, and $\mathcal{P}_w := \mathcal{P}_w^1 \cup \ldots \cup \mathcal{P}_w^t$ be the subcollection of paths that visit node $w \in V$.

Letting x_P, $P \in \mathcal{P}$, be a binary variable that is equal to 1 if and only if the path $P \in \mathcal{P}$ is selected in an optimal solution, the ILP model is the following

$$\max \sum_{P \in \mathcal{P}} p_P x_P \tag{2.1}$$

subject to

$$\sum_{P\in\mathcal{P}^j} x_P \le 1, \qquad j \in T, \tag{2.2}$$

$$\sum_{w\in U^i: w\succeq u, \Delta(u,w)<a_i} \sum_{P\in\mathcal{P}_w} x_P \le 1, \qquad i \in S\setminus\{1\}, u \in U^i, \tag{2.3}$$

$$\sum_{w\in W^i: w\succeq v, \Delta(v,w)<d_i} \sum_{P\in\mathcal{P}_w} x_P \le 1, \qquad i \in S\setminus\{s\}, v \in W^i, \tag{2.4}$$

$$\sum_{w\in W^i\cap V^j: v_1\preceq w\prec v_2} \sum_{P\in\mathcal{P}^j_w} x_P + \sum_{w\in W^i\cap V^j: w\succeq v_2, \Delta(v_2,w)<b_i^{kj}} \sum_{P\in\mathcal{P}^j_w} x_P + \\ + \sum_{w\in W^i\cap V^k: w\succeq v_2, \Delta(v_1,w)<b_i^{jk}} \sum_{P\in\mathcal{P}^k_w} x_P \le 1,$$

$$i \in S\setminus\{s\}, j,k \in T \text{ (with } j\neq k; i, i+1 \in S^j\cap S^k),$$

$$v_1, v_2 \in W^i \text{ (with } v_1 \preceq v_2, \Delta(v_1,v_2) < b_i^{jk}), \tag{2.5}$$

$$x_P \ge 0, \qquad P \in \mathcal{P}, \tag{2.6}$$

$$x_P \text{ integer}, \qquad P \in \mathcal{P}. \tag{2.7}$$

The interpretation of the objective function (2.1) and of constraints (2.2), that impose that at most one path for each train is selected, poses no problem. The *arrival time* constraints (2.3) and the *departure time* constraints (2.4), which have analogous counterparts in [15, 17], prevent two consecutive arrivals and departures at the same station i to be too close in time. The *overtaking* constraints (2.5), which are the immediate counterpart of constraints (13) in [17], are formally more complex. These constraints are defined by specifying a station $i \in S\setminus\{s\}$, two trains $j,k \in T$ such that $i, i+1 \in S^j\cap S^k$, and two nodes $v_1, v_2 \in W^i$ such that $v_1 \preceq v_2$ and $\Delta(v_1,v_2) < b_i^{jk}$. By definition of b_i^{jk}, we must have $x_{P_1} + x_{P_2} \le 1$ for all $P_1 \in \mathcal{P}^j_{v_1}$ and $P_2 \in \mathcal{P}^k_{v_2}$, which would be the weakest possible form of overtaking constraints. A very simple strengthening would be $\sum_{P\in\mathcal{P}^j_{v_1}} x_P + \sum_{P\in\mathcal{P}^k_{v_2}} x_P \le 1$. By proceeding further, taking also into account the arrival and departure constraints, one gets to (2.5), which are maximal in the sense that no other variable x_P for $P \in \mathcal{P}^j \cup \mathcal{P}^k$ could be added to the left-hand side maintaining validity. Specifically, nodes v_1 and v_2 represent the earliest departure times from i for trains j and k, respectively, involved in (2.5). Stated in words, constraints (2.5) forbid the simultaneous departure from station i of train j at a time instant between $\theta(v_1)$ and $\theta(v_2) + b_i^{kj} - 1$ and of train k at a time instant between $\theta(v_2)$ and $\theta(v_1) + b_i^{jk} - 1$. Since the departure of j at time $\theta(v_2) + b_i^{kj}$ is compatible with the departure of k at time $\theta(v_2)$, and the departure of k at time $\theta(v_1) + b_i^{jk}$ is compatible with the departure of j at time

$\theta(v_1)$, the time windows in the constraint cannot be enlarged. Finally, note that, although the ILP model would be valid also without constraints (2.3) and (2.4), as arrival/departures too close in time are forbidden also by (2.5), the former lead to a stronger LP relaxation and are also much faster to separate in practice, so it is much better to keep them in the formulation.

Note that the ILP model above is of *set packing type*, and is associated with the stable set problem defined on the graph with one node for each path $P \in \mathcal{P}$, with associated profit p_P, and one edge joining each pair of nodes corresponding to paths that cannot be both selected in the solution, i.e., that have both coefficient 1 in at least one of the inequalities.

2.5 Solution of the LP Relaxation

As already mentioned, the number of variables in the new ILP formulation can be very large. Rough bounds on the number of paths associated with each train $j \in T$ are the following, letting $\nu_j^{\max} := \max\{\nu : \pi_j - \alpha_j \nu > 0\}$ and $\mu_j^{\max} := \max\{\mu : \pi_j - \gamma_j \mu > 0\}$ be the maximum shift and stretch, respectively, in a timetable for train j that has positive profit. It is not difficult to see that, once the shift for a path of train j is fixed, the number of paths that accumulate a total stretch not larger than $\mu_j^{\max}$ in the $m_j := l_j - f_j - 1$ intermediate stations for train j is equal to the number of vectors with m_j integer nonnegative components whose sum is at most $\mu_j^{\max}$, which is equal to $\binom{\mu_j^{\max}+m_j}{m_j}$. Accordingly, we have that $|\mathcal{P}^j|$ is at least equal to the number of paths with zero shift and stretch not larger than $\mu_j^{\max}$, i.e., $|\mathcal{P}^j| \geq \binom{\mu_j^{\max}+m_j}{m_j}$, and at most equal to the number of paths with shift not larger than $\nu_j^{\max}$ and stretch not larger than $\mu_j^{\max}$, i.e., $|\mathcal{P}^j| \leq (2\nu_j^{\max}+1)\binom{\mu_j^{\max}+m_j}{m_j}$. In any case, $|\mathcal{P}^j|$ may be exponential in m_j, and for the instances considered in Section 2.8 this makes it absolutely impractical to consider explicitly all paths in $\mathcal{P}^j$.

The natural approach to tackle such a large number of variables is to use column generation techniques for the solution of the LP relaxation, working with an LP with only a subset of the variables (at the beginning, e.g., only the variables associated with the ideal path for each train), adding variables with positive reduced profit to the LP at each iteration. This is illustrated in detail in Section 2.5.1.

Although not exponential, also the number of constraints in the new ILP formulation is fairly large. Specifically, we have $O(|V|)$ arrival and departure constraints (2.3) and (2.4) and $O(t^2|V|b^{\max})$ overtaking constraints (2.5), where $b^{\max}$ is the maximum value of b_i^{jk} over all stations i and train pairs j, k. Accordingly, rather than imposing all these constraints since the beginning, it is much better to handle them by using separation techniques, as illustrated in Section 2.5.2.

Note that the column generation problem simply amounts to the computation of an optimal path in an acyclic graph, and the effect of adding new constraints with nonnegative dual

variables simply corresponds to changing the "penalties" of some nodes in this path computation. Accordingly, the combination of separation and column generation poses no serious problem in our case (i.e., the addition of new constraints does not destroy the structure of the column generation problem, as opposed to what is frequently the case, see, e.g., [4]).

The general structure of our method to solve the LP relaxation is the following:

1. We initialize a reduced LP with only the t variables associated with the ideal paths for each train, and constraints (2.2) and (2.6);
2. We solve the reduced LP, obtaining the primal solution x^* and the dual solution y^*;
3. We apply column generation: if variables with positive reduced profit with respect to y^* are found, we add them to the reduced LP and go to Step 2.;
4. We apply separation for constraints (2.3) and (2.4): if constraints violated by x^* are found, we add them to the reduced LP and go to Step 2.;
5. We apply separation for constraints (2.5): if constraints violated by x^* are found, we add them to the reduced LP and go to Step 2.;
6. We terminate since x^*, y^* is an optimal primal-dual pair for the whole LP (2.1)–(2.6).

The reason for separating constraints (2.3) and (2.4) before constraints (2.5) is that the former are stronger as well as faster to check for violation. On the other hand, the choice to perform column generation before separation was motivated by experimental evidence, showing that convergence is faster if we start separation only when the current primal solution is optimal with respect to the constraints in the reduced LP. Of course, in the first iteration there are no variables with positive reduced profit, and separation amounts to checking feasibility of the solution defined by the ideal timetables.

2.5.1 Column Generation

The column generation problem is solved by considering in turn each train $j \in T$ and checking if there exists a path $P \in \mathcal{P}^j$ with positive reduced profit.

Let the dual variables associated with constraints (2.2), (2.3), (2.4), and (2.5) be γ_j ($j \in T$), α_u ($i \in S \setminus \{1\}, u \in U^i$), β_v ($i \in S \setminus \{s\}, v \in V^i$), and δ_{j,k,v_1,v_2} ($i \in S \setminus \{s\}, j,k \in T, v_1, v_2 \in W^i$ satisfying the requirements in (2.5)). Moreover, for $i \in S \setminus \{1\}$ and $w \in U^i$, let $\zeta_w := \sum_{u \in U^i : u \preceq w, \Delta(u,w) < a_i} \alpha_u$, and, similarly, for $i \in S \setminus \{s\}$ and $w \in W^i$, let $\eta_w := \sum_{v \in W^i : v \preceq w, \Delta(v,w) < d_i} \beta_v$. Finally, for $\ell \in T$, $i \in S \setminus \{l_\ell, \ldots, s\}$ and $w \in W^i$, let $\xi_{\ell w}$ be the sum of variables δ_{j,k,v_1,v_2} over all constraints for which ℓ is either j or k and node w is one of the

nodes in the summations for train ℓ in (2.5) (i.e., the first two summations if $\ell = j$ and the third one if $\ell = k$).

For a path $P \in \mathcal{P}^j$, let U_P and W_P be, respectively, the set of arrival and departure nodes visited by path P. The reduced profit of path P is given by:

$$p_P - \gamma_j - \sum_{u \in U_P} \zeta_u - \sum_{v \in W_P} (\eta_v + \xi_{jv}).$$

Accordingly, one may find the path $P \in \mathcal{P}^j$ with maximum reduced profit by finding, in $O(|A^j|)$ time, the path of maximum profit from σ to τ in G that uses only arcs in A^j, proceeding in a completely analogous way as in the solution of the Lagrangian relaxation in [15], where the profit of a path $P \in \mathcal{P}^j$ is now given by

$$\sum_{a \in P} p_a - \sum_{u \in U_P} \zeta_u - \sum_{v \in W_P} (\eta_v + \xi_{jv}).$$

(Note that both in [15] and here profits are associated both with nodes and arcs of G, and they can be handled in a way completely analogous to the case of arc profits only.)

If the maximum profit is not larger than γ_j, then all variables x_P, $P \in \mathcal{P}^j$, have nonpositive reduced profit. Otherwise, the path found is associated with the variable with the most positive reduced profit.

2.5.2 Separation

Separation of constraints (2.3), (2.4) and (2.5) is done by trivial enumeration, as it is the case within the relax-and-cut procedure associated with Lagrangian relaxation, with the difference that we have to deal with fractional variable values in this case.

Specifically, for constraints (2.3) and (2.4), for each node $v \in V$ we initialize a value y^*_w to 0. Then, we consider in turn each positive variable x^*_P in the reduced LP solution and increase the value y^*_w of all nodes $w \in U_P \cup W_P$ by x^*_P. Finally, for each station $i \in S \setminus \{1\}$ and node $u \in U^i$, we test if $\sum_{w \in U^i : w \succeq u, \Delta(u,w) < a_i} y^*_w > 1$, in which case constraint (2.3) associated with node u is violated. The computation of the node values can be carried out in $O(sn)$ time, where n is the number of positive variables in the reduced LP solution, and the subsequent check is easily done in $O(|V|)$ time by using accumulators to store, for each node $u \in U^i$, the sum of the values of consecutive nodes (i.e., of nodes associated with consecutive time instants) in U^i, starting from an (arbitrarily chosen) initial node and ending in u. The check for constraints (2.4) is perfectly analogous.

For constraints (2.5), for each train $j \in T$ and for each node $v \in V^j$ we first initialize a value z^*_{jv} to 0, and consider in turn each positive variable x^*_P in the reduced LP solution

for $P \in \mathcal{P}^j$, increasing the value of all nodes in W_P by x^*_P. Then, we enumerate all pairs of trains $j, k \in T$ whose paths may be in conflict (by determining, for each train $j \in T$ and once for all, the set of trains whose paths may be in conflict with a path of j), and all stations $i \in \{f_j, \ldots, l_j - 1\} \cap \{f_k, \ldots, l_k - 1\}$. For each pair of nodes $v_1, v_2 \in W^i, v_1 \preceq v_2, \Delta(v_1, v_2) < b_i^{jk}$ we have that constraint (2.5) corresponding to j, k, v_1, v_2 is violated if and only if

$$\sum_{w \in W^i \cap V^j : v_1 \preceq w \prec v_2} z^*_{jw} + \sum_{w \in W^i \cap V^j : w \succeq v_2, \Delta(v_2, w) < b_i^{kj}} z^*_{jw} + \sum_{w \in W^i \cap V^k : w \succeq v_2, \Delta(v_1, w) < b_i^{jk}} z^*_{kw} > 1.$$

The time complexity of the above procedure to separate constraints (2.5) is again $O(sn)$ for the computation of the values, and then $O(t^2|V|b^{\max})$ for the subsequent check, since, by using accumulators similar to those used for the separation of constraints (2.3) and (2.4), each constraint can be checked in constant time.

2.6 Heuristic Algorithms Based on the LP Relaxation

In this section we present some constructive and local search heuristics, based on the optimal solution of the LP relaxation of the ILP model (2.1)–(2.7). We present several variants all of which have been implemented and tested.

2.6.1 Constructive Heuristics

The construction of heuristic solutions from the LP relaxation is done following two main principles: the first one is to fix to 1 or 0 some variables x_P associated with train paths, the second one is to construct the path of each train step-by-step, as it is done in the enumerative algorithm of the previous section.

In all cases, we fix some part of the LP solution and then solve again the LP relaxation taking into account what was fixed (using dual simplex).

For the cases in which x_P variables are fixed, given an optimal LP solution x^*, we consider the following policies:

- fix to 1 all the variables x_P whose value x^*_P is 1 as well as the variable x_P with the highest fractional value x^*_P;
- fix to 1 the variable x_P with the highest fractional value x^*_P.
- fix to 1 all the variables x_P whose value x^*_P is 1 as well as the variable x_P with fractional value x^*_P whose fixing to 1 leads to the highest value for the resulting LP solution;

- fix to 1 all the variables x_P whose value x_P^* is 1 as well as the variable x_P with fractional value x_P^* such that $p_P x_P^*$ is maximum.

For the cases in which the paths in the solution are constructed step-by-step, we consider one train at a time, and construct the path for the train by fixing the departure nodes in stations $\{f_j, f_j + 1, \ldots, l_j - 1\}$ in this order. Starting from the optimal LP solution x^*, we use the following two criteria to determine the next train to be considered:

- consider the train j such that $\sum_{P \in \mathcal{P}^j} p_P x_P^*$ is maximum;
- consider the train j such that $\max_{P \in \mathcal{P}^j} x_P^*$ is maximum.

Moreover, once the departure nodes in stations $\{f_j, \ldots, i-1\}$ for the current train j have been fixed, and the associated LP solved, yielding solution x^*, we use the following two criteria to fix the departure node in station i:

- fix the node $v \in W^i \cap V^j$ such that $\sum_{P \in \mathcal{P}_v^j} x_P^*$ is maximum;
- fix the node $v \in W^i \cap V^j$ such that $\max_{P \in \mathcal{P}_v^j} x_P^*$ is maximum.

If, after fixing a departure node for a train, either the resulting LP becomes infeasible or the profit of train j in the LP becomes nonpositve, we cancel the train.

The procedure above is stopped when the optimal LP solution is integer. Note that in this solution some trains may be canceled either by fixing, or simply since all the associated variables are 0.

After extensive testing of the presented possibilities on our testbed instances, we decided to apply only the following two path fixing policies (the first two described policies):

- fix to 1 all the variables x_P whose value x_P^* is 1, and additionally the variable x_P with the highest fractional value x_P^*;
- fix to 1 the variable x_P with the highest fractional value x_P^*.

2.6.2 Local Search

We try to improve the solutions constructed by the heuristics of the previous section by using the following local search procedures.

The first procedure, which is analogous to the one used in the Lagrangian heuristic proposed in [15], considers, in increasing order of train index, each train j whose associated path has positive shift and/or stretch. For each j, the procedure removes the associated path from the solution and finds the optimal path for j with the paths of the other trains fixed (as

usual, by computing an optimal path in the acyclic graph G). Once all trains have been considered, the trains that are currently canceled are considered, again in increasing order of train index, and the optimal path (if any) with the other train paths fixed is found. If, after having computed the optimal path for all these trains, the solution has been improved, the procedure is re-applied.

The second procedure is similar, with the difference that, each time a train j with shifted and/or stretched path is considered, we define the ILP model (2.1)–(2.7) for the problem in which all the paths of the other trains in the solution are fixed, and all the variables associated with paths for train j *as well as* with those of all the canceled trains are present. This ILP is solved heuristically by the constructive methods described in Section 2.6.1, and the best solution is taken.

A straightforward generalization of the second procedure is to consider, rather than a single train with shifted/stretched path, k trains, all of which are scheduled in the solution and at least one of which is shifted/stretched, and heuristically solve the ILP in which the paths for all the other scheduled trains are fixed. In order to limit the computational effort, we consider only the cases $k = 2$ and $k = 3$ ($k = 1$ being covered by the procedure above). Moreover, we consider only sets of k trains such that, by removing each of the k associated paths, at least one of the paths for the remaining trains can be improved. For each of the k paths, this is checked by removing the path from the solution and, for each remaining train j, recomputing the optimal path if all the other paths in the solution are fixed.

2.7 An Exact Branch-and-Cut-and-Price Algorithm

In this section we illustrate an exact enumerative algorithm based on the ILP formulation proposed in Section 2.4, capable of finding the provably optimal solution for some (small-size) real-world instances. We use a canonical branch-and-bound approach in which upper bounds are computed by solving the LP relaxation (amended by the branching constraints) by means of column generation and separation. Accordingly, our method falls within the category of the so-called branch-and-cut-and-price algorithms.

Having specified how we solve the LP relaxation in Section 2.5, the other main issue of the method is the way in which branching is performed.

Our branching method amounts to specifying, for some station, which departure node must be visited by the path of a train – as the travel times between consecutive stations are fixed, as explained in Section 2.2, the specification of all departure nodes for a train uniquely identifies its path in the solution (if any, i.e., if the train is not canceled).

In order to decide the train on which to branch, starting from the optimal LP solution x^*, we proceed as follows: for each train $j \in T$, let $\rho_j := \sum_{P \in \mathcal{P}^j} x^*_P$ and $\omega_j := \max_{P \in \mathcal{P}^j} x^*_P$

be, respectively, the sum and the maximum of the values taken by the variables associated with the train in the LP solution. We branch on the train j for which ρ_j/ω_j is maximum. The reason for this choice is that a train with a large value of the ratio is a train for which the sum of the associated variables is large but, at the same time, many of these variables are fractional. Then, branching is aimed at trying to limit the number of paths whose variables can have positive value for the train, hopefully improving the upper bound.

After having decided the train j, we choose a station $i \in S^j$ and then branch by imposing that the path for train j visits a specific departure node $v \in W^i \cap V^j$, by setting $x_P = 0$ for all paths P that do not visit that node. Moreover, we explore, in this order, the subproblems associated with node $u \in W^i$ such that: (i) $\Delta(u, v) = 0$ (i.e., $u = v$), (ii) $\Delta(u, v) = 1$, (iii) $\Delta(v, u) = 1$, (iv) $\Delta(u, v) = 2$, (v) $\Delta(v, u) = 2$, and so on, excluding of course the cases in which $u \notin V^j$ as well as those in which forcing the path for train j to visit node u would lead to a violation of the track capacity constraints along with the constraints already imposed by branching. When we perform the first branching concerning train j, we also consider the possibility to have no path for the train, i.e., to cancel it, exploring this subproblem as the last one among those generated by the branching.

The choice of i and v works as follows. If there is no node already fixed for train j, we let $i := f_j$ and v be such that $\sum_{P \in \mathcal{P}_v^j} x_P^*$ is maximum, recalling that P_v^j is the set of paths for train j that visit node v. Otherwise, i.e., if there are nodes already fixed for train j, we consider the path $P \in P^j$ such that x_P^* is maximum. We let v be the node in V^j such that $(\eta_v + \xi_{jv})$ is maximum and i be the station associated with v, recalling that $(\eta_v + \xi_{jv})$ is the "penalty" associated with node v, as defined in Section 2.5.1. The reason for this choice is to try to identify a "good" path for train j, and at the same time a node with a high "penalty" in order to have some impact on the LP solution.

The decision tree associated with the subproblems generated by branching is explored according to a depth-first policy, in order to limit the storage requirement of the method. Before branching at the root subproblem, we apply the heuristic algorithms of the previous section in order to have a valid (and hopefully tight) lower bound on the optimum to use in the bounding part.

Note that, differently from most algorithms in which column generation has to be performed for each subproblem, in our case the branching conditions are easily dealt within the column generation procedure. Indeed, in the associated optimal path computation for train j, the visit of departure nodes that have been already fixed is forced by skipping the exploration of nodes that are incompatible with these departure nodes.

2.8 Experimental Results

The algorithms proposed in Sections 2.5 to 2.6 were implemented in C and tested on a set of real-world instances from Rete Ferroviaria Italiana (the Italian railway infrastructure management company, RFI for short), a company of Ferrovie dello Stato (FS holding group).

2.8.1 The Real-World Instances

For each station i, the lower bounds a_i and d_i on the time intervals between two consecutive arrivals and departures of trains are set to 4 and 2 minutes, respectively, according to the current operational rules. The coefficients used in the objective function are illustrated in Table 2.1. In particular, the profit coefficients are identical for the trains of the same type. For instance, the profit achieved if a Eurostar train j is scheduled according to its ideal timetable is $\pi_j = 200$, and there is a penalty $\alpha_j = 7$ for each minute of shift as well as a penalty $\gamma_j = 10$ for each minute of stretch. Note that the shift is penalized less than the stretch for all train types.

Train Type	π_j	α_j	γ_j
Eurostar	200	7	10
Euronight	150	7	10
Intercity	120	6	9
Express	110	5	8
Combined	100	6	9
Direct	100	5	8
Local	100	5	6
Freight	100	2	3

Table 2.1: Train profit coefficients depending on the train type.

The main characteristics of the instances are outlined in Table 2.2. Column **# stations** gives the total number of stations along the considered corridor. Column **# trains** gives the total number of trains on input followed by an array with the number of Eurostar, Euronight, Intercity, Express, Combined, Direct, Local and Freight trains, respectively. Furthermore, Column **ideal profit** gives the sum of the profits achievable by scheduling each train according to its ideal timetable. We remark that the instances used in this chapter are slightly different with respect to those considered in [15]. In particular, the original train type "Intercity" has been split into train types "Intercity" and "Combined", with different profit coefficients. In addition, few intermediate stations in which no overtaking can occur are now explicitly considered.

Most of these instances contain a small portion of the trains that are daily run along the

Instance	first station	last station	# stations	# trains	ideal profit
PC-BO-1	Piacenza	Bologna	17	40 (6,0,10,0,0,12,2,10)	4800
MU-VR	Munich	Verona	48	54 (0,0,0,7,0,0,47,0)	5470
BZ-VR	Bolzano	Verona	27	128 (34,0,0,10,1,11,38,34)	16300
PC-BO-2	Piacenza	Bologna	17	93 (12,3,13,10,5,20,6,24)	11010
PC-BO-3	Piacenza	Bologna	17	60 (12,1,10,0,4,12,7,14)	7450
PC-BO-4	Piacenza	Bologna	17	221 (28,3,17,35,35,28,14,61)	25740
CH-RM	Chiasso	Rome	102	41 (17,0,11,0,0,0,7,6)	6020
BN-BO	Brennero	Bologna	48	68 (1,0,5,13,0,11,38,0)	7130
CH-MI	Chiasso	Milan	16	194 (20,1,20,8,9,19,66,51)	21930
MO-MI-1	Modane	Milan	54	16 (1,0,3,0,0,4,5,3)	1760
MO-MI-2	Modane	Milan	54	100 (19,0,26,0,0,13,22,20)	12420

Table 2.2: Characteristics of the instances considered.

associated lines, considering a limited time window within the day, and in some cases only some of the various train types.

2.8.2 The Results

In this section we present a comparison of the upper bound and heuristic solution values obtained by the method of [17] and by the methods discussed in this chapter (Sections 2.5 and 2.6) as well as the results of the exact algorithm (Section 2.7) on the instances that it was able to solve. All times reported are expressed in CPU seconds on a PC Pentium IV 2.4 GHz with 512Mb RAM. We solved the LPs using CPLEX 9.0.

The Lagrangian method proposed in [17] is an improvement over the one in [15]. The results reported in [17] were obtained by running the code on a different architecture (with a different floating point arithmetic). Accordingly, rather than reporting those result with a rough time conversion, we decided to run the code from [17] on the machine mentioned above. This explains why the results given here are slightly different.

As far as the heuristic solution values are concerned, among the many possible combinations of the options discussed in Section 2.6, we report the results for the following two heuristics. Heuristic H1 first constructs a solution by using the first fixing rule described in Section 2.6.1, and then applies the three local search procedures described in Section 2.6.2, in the order in which they were described. Heuristic H2 first constructs a solution by using the second fixing rule described in Section 2.6.1, and then applies only the first two local search procedures described in Section 2.6.2, again in that order, these two procedures being much faster than the third one.

Table 2.3 reports the results on the instances illustrated above. The columns in the table

have the following meaning:

- **Lagr. UB** is the Lagrangian upper bound obtained by the method in [17].
- **Lagr. heur** is the value of the best heuristic solution found by the method in [17].
- **Lagr. % gap** is the percentage gap between the best upper bound and the value of the best heuristic solution found by the method in [17].
- **LP UB** is the optimal value of the LP relaxation discussed in this chapter.
- **LP H1** and **LP H2** are the solutions found by heuristics H1 and H2, respectively, illustrated above.
- **LP % gap** is the percentage gap between the best upper bound and the value of the best solution found by H1 and H2.

Table 2.3 shows that, on the one hand, the LP upper bound, which dominates the Lagrangian upper bound, is often significantly stronger and can be computed within much smaller computing times. The solutions found by heuristic H2 are of quality comparable to those found by the Lagrangian heuristic and require, on average, a similar computing time, although the time taken by the former has a much higher variance. On the other hand, in almost all cases, the solutions found by heuristic H1 are better than those found by the Lagrangian heuristic, but the associated computing time is much higher.

Name	Lagr. UB		Lagr. heur		Lagr. % gap	LP UB		LP H1		LP H2		LP % gap
	value	time	value	time		value	time	value	time	value	time	
PC-BO-1	4314	65	3629	54	% 15.9	4091	14	3776	5836	3725	137	% 7.7
MU-VR	5032	128	4211	97	% 16.3	4894	19	4253	5210	3968	42	% 13.1
BZ-VR	16152	259	15994	260	% 0.98	16102	11	15977	2297	15997	12	% 0.65
PC-BO-2	10953	75	10861	78	% 0.84	10914	6	10882	504	10727	3	% 0.29
PC-BO-3	7235	73	7135	73	% 1.38	7200	4	7153	644	7138	6	% 0.65
PC-BO-4	24243	616	21425	523	% 11.6	23894	761	22041	43200	21753	6029	% 7.7
CH-RM	5850	392	5567	383	% 4.8	5823	26	5574	5530	5407	73	% 4.3
BN-BO	6909	126	6774	123	% 1.95	6852	6	6716	1811	6649	5	% 1.98
CH-MI	21259	232	20816	232	% 2.08	21131	15	21022	13653	20919	31	% 0.5
MO-MI-1	1727	17	1684	15	% 2.49	1708	2	1684	36	1634	3	% 1.4
MO-MI-2	11336	415	9318	354	% 17.8	11185	288	9453	48051	9498	1239	% 15.08

Table 2.3: Comparison of the results obtained with the Lagrangian heuristic of [17] and the LP-based heuristic of the present chapter.

The exact algorithm described in Section 2.7, which initially applies heuristic H1, could solve 3 of the 11 instances within a time limit of $100,000$ seconds (around 28 hours). It does

not appear that, by increasing the time limit by, say, a factor two or three, the method could solve to proven optimality other instances among those considered. The associated optimal value, computing time, and number of subproblems explored are reported in Table 2.4. This is the first time that some instances in this testbed are solved to proven optimality.

Name	opt	time	# subpr.
PC-BO-2	10882	683	6611
PC-BO-3	7161	77562	699687
MO-MI-1	1684	44	680

Table 2.4: Results with the branch-and-cut-and-price algorithm.

In previous (unpublished) experiments we observed that the solution of the LP relaxation of the ILP models in [15, 17] is extremely cumbersome in practice. This chapter shows that an equivalent LP relaxation in which the number of variables is exponentially large can be solved within a computing time that is not only reasonable (as one might have expected) but also significantly smaller than the time needed to compute good Lagrangian bounds by the approaches proposed in [15, 17], yielding notably better bounds. On the other hand, although the solution values returned by the heuristic method based on the new LP relaxation tend to be better than those found by the Lagrangian heuristics of [15, 17], the corresponding solution times are much larger, which seems to indicate that the Lagrangian approach is preferable for practical purposes.

Future work will be concerned with facing real-world instances that contain (most of) the trains that should be run along the considered corridor on a day. Moreover, we will test our approach on instances associated with a network rather than a corridor, comparing it with the other approaches that are suited for this case.

2.9 Train Timetabling and Comparability Graphs

2.9.1 Introduction

Starting from the LP relaxation of the ILP model for the Train Timetabling Problem described in Section 2.4, we propose a stronger version of the overtaking constraints, involving an arbitrary number of trains, and the associated separation methods. The study of this separation concerns the problem of computing Stable Set and Clique of Maximum Weight in a particular graph given by the edge intersection of a comparability graph and a complete $k-$partite graph.

2.9.2 Graph Description

For sake of clarity, we reformulate the model by using a new graph that we call the *path compatibility graph*.

We consider each line segment l connecting two consecutive stations $(i, h) \in S$ on the corridor and define such a graph for each line segment. We call this graph $G_l = (V_l, A_l)$.

This graph is constructed starting from the original graph $G = (V, A)$ described in Section 2.3.

In particular, arcs that connect two consecutive stations with the line segment l in the graph G become nodes in the new graph G_l. There exists an arc $a \in A_l$ connecting nodes a_{1l} and $a_{2l} \in V_l$ if the corresponding arcs a_1 and $a_2 \in A$ are such that they are associated with different trains, $a_1 \preceq a_2$ and they are compatible in terms of arrival/departure and overtaking constraints.

Formally, G_l has vertex set given by $P^1 \bigcup P^2 \ldots \bigcup P^{|T|}$ and is the edge intersection of:

a. the complete $|T|$-partite graph with color classes $P^1, \ldots, P^{|T|}$

b. the comparability graph defined by the transitive relation of *precedence* $\preceq$ as defined in Section 2.3

Let $\mathcal{S}$ be the collection of maximal stable sets of G_l (neglecting, to be precise, the orientation of the arcs).

Given the graph G_l the ILP model described in Section 2.4 can be reformulated as follows.

$$\max \sum_{P \in \mathcal{P}} p_P x_P \tag{2.8}$$

$$\sum_{P \in S} x_P \leq 1, \qquad S \in \mathcal{S}, \tag{2.9}$$

$$x_P \in \{0, 1\}, \qquad P \in \mathcal{P} \tag{2.10}$$

For the separation of (2.9), each node $a_l \in V_l$ in G_l is assigned a weight w_{a_l} given by the sum of the values of the variables x_P associated with paths in the graph G that contain the arc $a \in A$ corresponding to the node a_l. In order to find the most violated overtaking constraint (if it exists) we want to compute the *maximum weighted stable set* and check if this weight is greater than 1, that implies a violated constraint has been found.

We next describe two methods for separating constraints (2.9): the first one is a heuristic approach and the second one is exact.

2.9.3 Separation by using stable set

Note that G_l is a non-perfect graph. However, for solving the maximum weighted stable set in the auxiliary graph G_l we can use some results valid on the perfect graphs and on the comparability graphs. In particular, we consider the graph G_p obtained by computing the transitive closure of the graph G_l.

The corresponding auxiliary graph G_p is a perfect graph (if orientations are neglected) and consequently the value of the solution of the maximum stable set corresponds to the value of the minimum number of cliques that cover all the nodes.

Moreover, in this case, graph G_p is a comparability graph and consequently cliques correspond to paths in the graph. So the problem of finding the maximum stable set on graph G_p is equivalent to the problem of finding the minimum number of paths that cover all the nodes. This number can be computed as a minimum flow with demands on the nodes: let graph G_p^1 be the auxiliary graph G_p where we have introduced a source s and a sink t and where we have connected the source to the nodes without ingoing arcs and the sink to the nodes without outgoing arcs; let each demand associated with a node be equal to 1: computing the minimum flow with this demands on the nodes we obtain the minimum number of paths that cover all the nodes.

In our case, we want to find the maximum weighted stable set; let $w_1, \dots w_n$ be the demands associated with each node in the graph G_p^1; the maximum weighted stable set corresponds to a set of cliques such that each node i is contained in at least w_i cliques; this corresponds to a set of paths such that each node i is contained in at least w_i paths and can be obtained calculating the minimum flow with a demand associated with each node i equal to w_i.

Since the collection of stable sets of G_p is a (strict) subcollection of the stable sets of G_l, the described method for separating overtaking constraints is a heuristic procedure and some violated constraints involving paths that belong to the same train could not be found. The linear programming model that we use for the minimum flow problem with demands on the nodes is the following.

$$\min \sum_{a \in \delta^+_{(s)}} f_a \tag{2.11}$$

$$\sum_{a \in \delta^-_{(i)}} f_a = \sum_{a \in \delta^+_{(i)}} f_a, \quad i \in V_l \backslash \{s,t\} \tag{2.12}$$

$$\sum_{a \in \delta^-_{(i)}} f_a \geq w_i, \quad i \in V_l \backslash \{s,t\} \tag{2.13}$$

$$f_a \geq 0, \quad a \in A_p \tag{2.14}$$

As already said, this problem is equivalent to the problem of finding a set of paths such that each node i is contained in at least w_i paths. A linear programming model for this problem is the following.

$$\min \sum_{p \in P} x_p \tag{2.15}$$

$$\sum_{p \in P_i} x_p \geq w_i, \quad i \in V_l \tag{2.16}$$

$$x_p \geq 0, \quad p \in P \tag{2.17}$$

The dual problem of the last model is the problem of finding the maximum weighted stable set in the graph G_l; a model for this problem is the following, where we introduce a variable y_i for each node $i \in V_l$.

$$\max \sum_{i \in V_l} w_i y_i \tag{2.18}$$

$$\sum_{i \in p} y_i \leq 1, \quad p \in P \tag{2.19}$$

$$y_i \geq 0, \quad i \in V_l \tag{2.20}$$

Hence, the solution of the minimum flow problem gives the maximum weighted stable set of G_l. In particular the dual variables associated to the constraints (2.13) give the information on which are the nodes belonging to the stable set of maximum weight.

The algorithm is repeated for each pair of stations s_1 and $s_2 \in S$. We consider all the paths that visit both s_1 and s_2 and we construct the corresponding auxiliary graph G_l. If the value of the solution is greater than 1 we have found an overtaking violated constraint; so we create the constraint and add it to the model of the original problem, considering that nodes which constitute the maximum weighted stable set are associated to arcs in the graph G that are involved in the constraint. Moreover, we introduce in the constraint other paths, even if they have a null variable, that can be involved in the constraint.

2.9.4 Separation by using dynamic programming

The method described in 2.9.3 is a heuristic procedure for separating the overtaking constraints. Now, we show an exact method, based on the dynamic programming technique.

This approach is polynomial in the size of the network. For each couple of consecutive stations we repeat the same routine.

Given G_l, we let s_1 and $s_2 \in S$ be the associated stations. We consider the set of trains which visit both the stations and for each train the set of its paths. We want to find the most violated constraint, which involves variables corresponding to paths that are incompatible between this two stations and has the higher value for the sum of involved variables. Lets call *stable set* a set of incompatible variables which are candidate to be involved in a constraint.

In the easiest case where we have only one path for each train we can insert a new path (variable) in a stable set if it is incompatible with all the paths already in the stable set and the global profit becomes higher.

We consider the paths ordered by decreasing speed between stations s_1 and s_2. As the paths are ordered by decreasing speed, there are two extreme cases in which we can add a path to a stable set and all other cases are in between. The following conditions corresponding to these two extreme cases must be true *simultaneously* in order to add a path to a stable set.

The first one is due to departure constraints: a path p can be added to a stable set if the first departure instant (from the station s_1) b of paths inserted in the stable set is incompatible with the departure instant u from the station s_1 of p.

The second one is due to arrival constraints: a path p can be added to a stable set if the last arrival instant c (in the station s_2) of paths inserted in the stable set is incompatible with the arrival instant z in the station s_2 of p.

All other possibilities of adding a path to a stable set are those in between these two extreme cases. In particular, given a stable set, let call b the first departure instant of paths already inserted in the stable set and c the last arrival instant of paths already inserted in the stable set.

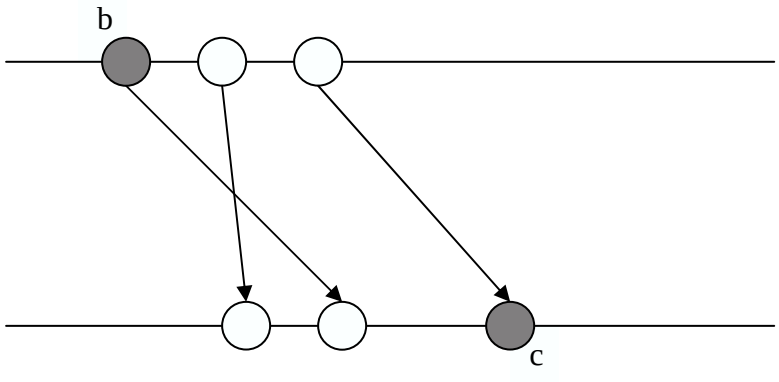

Figure 2.2: Example of a *stable set* and the associated b,c

Let p be a path with departure instant u from the station s_1 and arrival instant z at the

station s_2: it can be added to the stable set if $u - d_{s_1} + 1 \preceq b$ and $c \preceq z + a_{s_2} - 1$, where d_{s_1} and a_{s_2} are the minimum interval time between two consecutive departures from station s_1 and two consecutive arrivals in station s_2 respectively.

Note that if $d_{s_1} \neq a_{s_2}$, we have to consider other possibilities of adding a path to a stable set and the number of other cases depends on the difference between the two time intervals.

Consider for example the case (that is the case of the Italian Railways) in which the minimum time interval between two arrivals is four minutes and the minimum time interval between two departures is two. Accordingly, we will illustrate the methods for the case $a_{s_2} - d_{s_1} = 2$.

A path can be added to a stable set even if it has a departure instant compatible with one or more paths in the stable set but has an arrival instant incompatible with all the paths in the stable set. In this case, we have to consider also stable set with $b = u - d_{s_1}$ but we must pay attention not to add paths compatible with some path in the stable set. For this reason, when we consider the case of $b = u - d_{s_1}$ we have to look at the first arrival instant (let it be k) for arcs starting in instant b and to be sure this instant is incompatible with z.

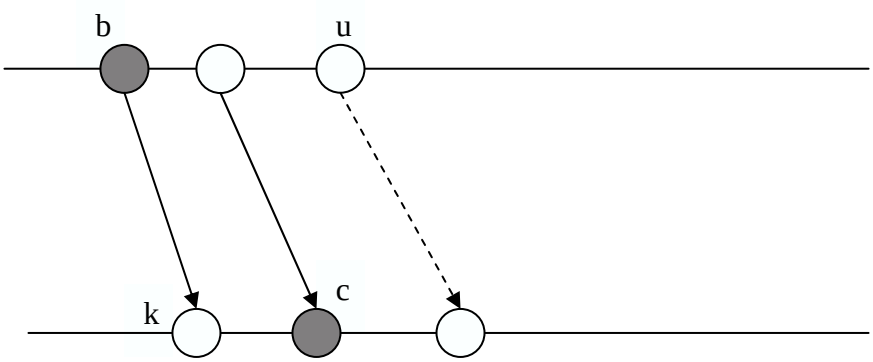

Figure 2.3: Considering $b = u - d_{s_1}$

Moreover, we have to consider the case of $b = u - d_{s_1} - 1$ (which is the last case, as the difference between the time intervals is two) and we can insert a path in the stable set only if the first arrival instant (let it be l) for arcs starting in b or in $b + 1$ is incompatible with z (in this case we have to consider also arcs starting in $b + 1$ as they are compatible with the path as regard as the departure instant and can be compatible also for the arrival instant).

As already said, we can have more than one path for each train, as we are considering the continuous relaxation of the problem and we can have fractional variables. In this case, after having ordered the paths for all the train visiting the couple of stations s_1 and s_2 by decreasing train speed, we consider for each train j all the $(|V^j \bigcap W^{s_1}|^2)$ possible subsets of paths corresponding to departures of train j from station s_1 within a given time window and

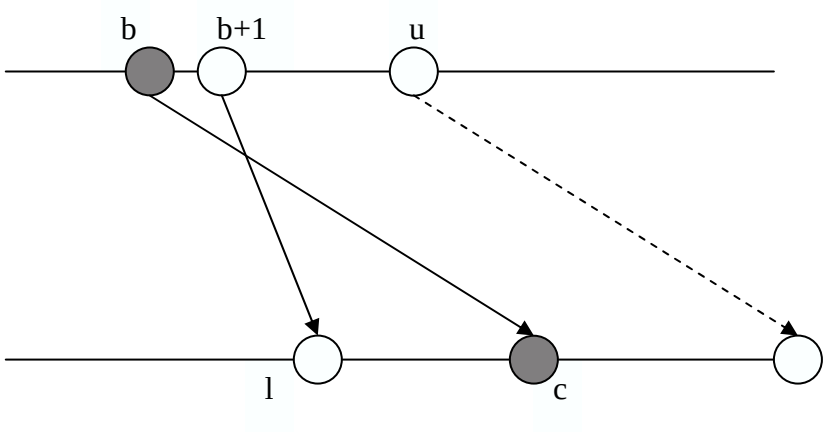

Figure 2.4: Considering $b = u - d_{s_1}$, we need to check the arrival corresponding to $b + 1$

we try to add them to the existing stable sets.

Let u be the departure instant from station s_1 for the first path in the current subset and z its arrival instant in station s_2 and let v be the departure instant for the last path in the current subset and y its arrival instant.

This subset of paths, that are obviously all incompatible each other as they belong to the same train, can be added to a stable set if every paths in the subset is incompatible with all the paths already inserted in the stable set; this is true for such stable sets that have $v - d_{s_1} + 1 \preceq b$ and $c \preceq z + a_{s_2} - 1$.

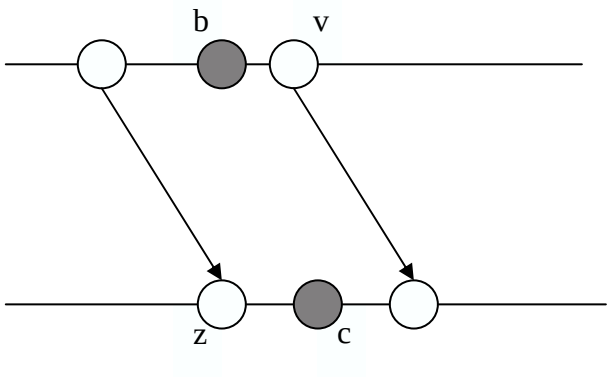

Figure 2.5: Adding to a stable set more than a path for a train

As before, if $a_{s_2} - d_{s_1} = 2$ as in our case study, we have to consider also the other two cases, the one with $b = v - d_{s_1}$ and we can add the set of paths if $k \succeq y - a_{s_2} + 1$ and the one with $b = v - d_{s_1} - 1$ and we can add the set of paths if $l \succeq y - a_{s_2} + 1$ (in both cases,

$b \preceq c \preceq z + a_{s_2} - 1$). Naturally, if we want to add a set of paths to a stable set that does not still exist (as it happens at the beginning for example) we just construct a stable set with this set of paths.

What we remember as a *state* of a stable set is given by the first departure instant from station s_1 of a path inserted in the stable set, the last arrival instant in station s_2 of a path inserted in the stable set, the first arrival instant in station s_2 for paths starting from station s_1 in instant b and the first arrival time in station s_2 for paths starting form station s_1 in instant b or $b + 1$. Then we store the *profit* of the stable set which is given by the sum of the values of the variables corresponding to paths inserted in the stable set. Knowing all this information, we can decide if a set of paths is to be added to a stable set, as we want to add only incompatible paths and to obtain an increasing global profit.

Every time we add a set of paths to a stable set, we update the state and the profit and we keep a link to the previous stable set: precisely, we have to know which is the set of paths added at last and the state of the previous stable set, so that at the end we can identify which are the variables involved in the constraint.

At the end of the execution for a pair of stations we find the stable set with maximum profit and we construct the real constraint, using a backward procedure: we insert in the constraint the set of variables corresponding to the set of paths added at last in the stable set and we consider the previous stable set through the stored information; then the procedure is repeated until we arrive at the first train; if we do not have any path for a train to be inserted in a stable set we simply go back to the previous train.

As known, a problem with dynamic programming is the great amount of memory necessary to store the states during the execution. In order to reduce the quantity of memory used, we decide not to use memory for each possible instant k (first arrival instant for paths starting from instant b in the stable set) but a flag which allows to know if a set of paths can be added or not to a stable set. The same is done for instant l. Moreover, we do not consider all the possible instants in a day in which an overtaking constraint can be violated all together, but we consider *overlapping windows* whose width is equal to the maximum time distance between instant b and c for all the stable sets as regard as stations s_1 and s_2. Each window starts in an instant $t \in \{1, \ldots, q\}$ where we have a path and such that t is greater than that of the previous window. The procedure is repeated for each window.

Chapter 3

Freight Transportation in Railway Networks

3.1 Introduction

Freight transportation is increasingly moving towards the choice of using the railway infrastructure instead of highways. The main reason is that there is a great saving in terms of energy, evaluated on about the 50% of the corresponding cost on road. Another fundamental reason which favorites the railroad transportation is safeness: in normal conditions, the train has no interaction with other trains; moreover, trains are operated by expert drivers, who can also make use of automatic control systems that avoid human errors. Of course, an increase in the freight transportation on railroad would bring to a reduction of the quantity of trucks on road, with a corresponding reduction of traffic and accidents.

As far as freight transportation assumes a relevant role, it becomes important to optimize the usage of the railway infrastructure, dealing with a competitive pressure among train operators who desire to schedule an increasing number of freight trains.

Freight transportation has to face the scheduling of passenger trains: passenger trains have to respect prescribed timetables, as they give a service which has generally priority above freight transportation. Moreover, passengers should not be subject to delays and long stops in the stations. In this context, it is important that freight trains are scheduled taking into account requirements of passenger trains. This means that freight trains should be able to be driven in less congested lines, even with an increase in the travel time.

For this reason, we have to analyze the topology of the railway network and take into account all the possible alternative routes which link pairs of stations, in order to be able to decide which could be the best route for each freight train.

Moreover, we must start from a planned timetable for passenger trains which cannot be

changed in any way, and try to add as many new freight trains as possible, by satisfying all the safeness operational constraints. These constraints impose to respect safety margin between consecutive departures and arrivals of trains in the same station, and to avoid overtaking between two consecutive stations of the line (further details will be given in the next sections).

Even if freight trains have a lower priority than passenger trains, train operators propose a preferred timetable for each freight train, by giving, in addition to the starting and ending stations, the other stations which must be visited and the time instants which would be better for running the train. We will call this timetable *ideal timetable*.

The safeness operational constraints can force to modify the preferred timetables of the freight trains, but the aim is to change the timetables proposed by the train operators as less as possible.

The described problem consists in defining the timetables of the new freight trains, following the requests of the train operators, by satisfying the safeness operational constraints and keeping the timetables of the passenger trains as they are. Viewing the problem under this aspect, it can be treated as a case of Train Timetabling Problem, where we have to deal with a railway network and with prescribed timetables.

The considered problem belongs to the class of *Train Timetabling Problems*, for which different approaches have been studied. Vansteenwegen and Van Oudheusden [57] study the problem of improving the passenger service, taking into account waiting times and delays, on a small part of the Belgian railway network. Semet and Schoenauer [55] present an efficient evolutionary algorithm for real-world train timetabling, where the main focus is on the reconstruction of the schedule following a small perturbation, with the aim of minimizing the total delay. Kroon, Dekker and Vromans [34] study a stochastic optimization variant of the well-known periodic event scheduling problem (PESP) and Kroon and Peeters [35] describe how variable trip times can be included into an existing cyclic railway timetabling model based on the PESP. Mistry and Kwan [42] present a cooperative coevolutionary train timetabling problem algorithm concerned with the automatic generation of planning timetables. Barber et al. [3] present a tool for solving and plotting train schedules developed for the Spanish Railways, by using preprocessing steps and heuristics.

Many references consider mixed integer linear programming formulations in which the arrival and departure times are represented by continuous variables and there are logical (binary) variables expressing the order of the train departures from each station. Szpigel [56] considers a variant of these models in which the order of the train departures from a station is not represented by binary variables but by disjunctive constraints. The problem is then solved by branch-and-bound for small size instances by computing bounds through the relaxation of these disjunctive constraints. Jovanovic and Harker [33] solve a version of these models, that calls for a feasible schedule rather than for the optimization of a suitable objective function,

by using branch-and-bound techniques. Cai and Goh [14] illustrate a constructive greedy heuristic driven by one of these models. Carey and Lockwood [18] define a heuristic that considers the trains one at a time (in appropriate order), and for each train solves a mixed integer linear program analogous to those mentioned above in order to schedule the train optimally, keeping the path of the previously scheduled trains partially fixed. More precisely, the relative order of the train departures for these trains is kept fixed, whereas their arrival and departure times may be changed. Higgins, Kozan and Ferreira [31] define local search, tabu search, genetic and hybrid heuristics, finding a feasible solution by using a model in the family above.

Brännlund, Lindberg, Nöu and Nilsson [8] discretize the time into 1-minute *time slots* and subdivide the track line into *blocks*. Operational constraints impose that two trains cannot be in the same block in the same time slot. There is a binary variable x_{sbj} each time the timetable constraints allow train j to be in block b in time slot s. This model is not suited for large size instances as those arising for the main European corridors.

Oliveira and Smith [47] model the problem as a special case of the Job-Shop Scheduling Problem, considering trains as jobs to be scheduled on tracks regarded as resources, and presented a hybrid algorithm devised under the Constraint Programming paradigm, showing how to adapt this framework in some special real-life applications.

Schrijver and Steenbeek [54], Lindner and Zimmermann [38], and Peeters and Kroon [49] consider the case in which the timetable is identical with a period of one hour (rather than one day as it is the case of the problem considered in the other references), and address the general case of a railway network instead of a single (main) line. The problem is solved through a mixed integer linear programming formulation in which the times are again represented by continuous variables and integer variables are used to impose that the differences between pairs of time variables belong to a certain interval modulo one hour. Further references on this version of the problem can be found in Peeters [48].

Caprara, Fischetti and Toth [15] propose a mathematical formulation and an associated Lagrangian heuristic capable of handling hundreds of stations and trains.

In this chapter, we illustrate and computationally evaluate the algorithm developed for the problem, first describing a graph representation which is used to introduce an Integer Linear Programming (ILP) model for the problem and then proposing a heuristic algorithm based on its Lagrangian relaxation. This is an extension of the algorithm proposed in [15] for the simplified case of a single-track line on which trains travel only in one direction. Both the ILP model and the heuristic algorithm have been extended for dealing with the case of a generic railway network, taking into account new classes of constraints.

The considered problem is defined in Section 3.2. A description of the railway network and of the representation we use are given in Section 3.3. A graph representation and an

ILP formulation are presented in Sections 3.4 and 3.5, respectively. An associated Lagrangian heuristic is illustrated in Section 3.6. In Section 3.7, some computational results for some real-world case studies, corresponding to an international network involving alternative routes and station nodes, are presented.

In a more general context, the formulation of the Integer Linear Programming model and the solution method that will be described in the next sections can be used to solve the Train Timetabling Problem on a railway network, when no prescribed timetables are available, considering at the same time requests for scheduling passenger and freight trains, i.e., all the timetables for the entire railway network have to be defined.

3.2 Problem Description

In this section we illustrate the problem which can be viewed as an extension of the Train Timetabling Problem (TTP). The TTP in its basic version focuses on a linear one-way, single-track line and has the aim of determining an optimal timetable for a set of trains, by satisfying some operational constraints. These operational constraints impose to respect safety margin between consecutive departures and arrivals of trains in the same station and to avoid overtaking between two consecutive stations of the line.

In particular, we have to satisfy the following constraints:

1. a minimum time interval between two consecutive departures of trains from the same station on the same line is necessary;

2. a minimum time interval between two consecutive arrivals of trains at the same station on the same line is necessary;

3. a minimum time interval between an arrival (departure) and a departure (arrival) of trains which are consecutive at the same station on the same line is necessary;

4. overtaking is allowed only in correspondence of a station or by using an alternative line;

5. crossing is not allowed for trains on bi-directional tracks.

The considered Freight Transportation Problem (FTP) has the aim of scheduling as many new freight trains as possible on a Railway Network, and to follow the ideal timetable for the freight trains if possible, by satisfying all the operational constraints, where the timetables of the passenger trains are fixed and cannot be changed in any way.

We are allowed to operate modifications on the ideal timetable of the freight trains. In particular, we can: i) modify the departure instant from the starting station of a train, ii) choose any possible route which links the starting station to the ending station going through

the other prescribed stations even if it causes an increase in the total travel time (i.e. it is not the shortest route), iii) increase the stopping time in one or more of the visited stations. The first operation is called *shift* and the other two operations are called *stretch*. We distinguish between stretch caused by increase in the travel time and stretch due to increase in the stopping time.

The FTP considers a network instead of a one-way single-track line: in a network both bi-directional tracks and alternative routes between stations are present.

3.3 A General Railway Network

In this section we illustrate the extension of the TTP to the general case in which (a) the railway network topology is arbitrary (rather than a single line) (b) some tracks can be used for traveling in both directions (as it is the case for line sections with a unique track).

The trains have to be run every day of a given time horizon (e.g., 6-12 months). Some trains (e.g. passenger trains) have a prescribed timetable that cannot be changed in any way.

Leaving these timetables as they are, the train operators want to add some freight trains on the network. The main aim is to add as many freight trains as possible. Nevertheless, the train operators suggest ideal timetables for these trains.

For each freight train we know the starting and the ending stations together with the other stations which must be visited by the train (possibly none). These trains are allowed to choose any possible route which goes from the starting station to the ending station, visiting every prescribed station in between. Of course, it is better to follow the suggested timetable, and to change it as less as possible.

3.3.1 Representation of a Railway Network

A general railway network can be represented by a mixed multi-graph $N = (S, E \cup A)$ in which the vertex set S is given by the set of stations in the network, and E and A are the edge and the arc set, respectively. Each edge $e = \{h, i\}$ represents a *bi-directional track segment* between station h and station i, with no intermediate station in between, that may be used by trains travelling in both directions (i.e., from h to i or from i to h).

Analogously, each arc $a = (h, i)$ represents a *mono-directional track segment* between station h and station i, with no intermediate station in between, that may be used by trains travelling from h to i only. N is a multi-graph since there may be multiple track segments (edges and/or arcs) between adjacent stations. For convenience, for each station $i \in S$, we will let $\delta_N^+(i) \subseteq E \cup A$ denote the set of edges incident to i and of arcs leaving i, and $\delta_N^-(i) \subseteq E \cup A$ denote the set of edges incident to i and of arcs entering i.

We let T denote the set of trains that we consider in the railway network, where $T = T_1 \bigcup T_2$, and we let T_1 denote the set of passenger trains which have a prescribed timetable and T_2 denote the set of new possible freight trains which are candidate to be run every day of the given time horizon.

For each train $j \in T_1$ a complete timetable is given: a *first* (departure) station f_j and a *last* (destination) station l_j ($l_j > f_j$) are given. Let $S^j := \{f_j, \ldots, l_j\} \subseteq S$ be the ordered set of stations visited by train $j \in T_1$. The prescribed timetable defines, for each train $j \in T_1$, the departure time from f_j, the arrival time at l_j, and the arrival and departure times for the intermediate stations $f_j + 1, \ldots, l_j - 1$.

For each train $j \in T_2$, a starting station $f_j \in S$ and an ending station $l_j \in S$ are given, along with a set $M_j \subseteq S$ of stations that the train *must* visit, with $f_j, l_j \in M_j$, and a set $C_j \subseteq S$ of stations that the train *can* visit. For each pair of stations in $M_j \cup C_j$ that are connected by track segment(s), there is only one direction in which train j may travel, which is specified along with the corresponding travel time. (Note that we may easily forbid a specific track for a given train, e.g., a "fast" track for a "slow" train, by setting this time to infinity for the train.) Actually, as it is natural, we put restrictions on possible ways of going from f_j to l_j through the other stations in M_j. To be precise, letting N^j denote the *auxiliary* network with vertex set $M_j \cup C_j$ and one arc for each pair of stations as above, oriented according to the train direction, we require that N^j is *acyclic* and that *each* path in N^j from f_j to l_j visits *all* the stations in M_j, i.e., the removal of an arbitrary station in M_j from N^j disconnects f_j from l_j.

A *timetable* for train $j \in T_2$ defines a path in N^j from f_j to l_j that visits only stations in $M_j \cup C_j$ and in particular all stations in M_j, traversing each edge and arc in the direction allowed for train j, along with the departure time from f_j, the arrival time at l_j, and the arrival and departure times for all the intermediate stations in the path. Acyclicity of N^j guarantees that the order in which the stations in M_j are visited is the same for all paths. Furthermore, each station in C_j, if visited, will appear between the same two stations in M_j along the path. This allows us to associate with each station in C_j a *predecessor and a successor compulsory station* in M_j (with respect to train j).

On input, each train $j \in T_2$ is assigned an *ideal (partial) timetable*, that specifies the desired departure time from station f_j, the minimum stopping time for stations in M_j, the set C_j of stations that can be visited by the train and the arrival station l_j. Moreover, for each train $j \in T_2$ the travel time along each track segment is given. Usually the travel time is not associated directly to the train, but it depends only on the train type, so the description of the travel times is given once at all, for the whole network for each train type.

As a train $j \in T_2$ could have some forbidden routes in the network, we compute the travel time between consecutive stations h, i in M_j as the *shortest possible* one, i.e., it corresponds

to the shortest path from h to i in N^j that visits only nodes in C_j and where distances are equal to the travel times for each segment mentioned above.

The most desirable timetable for train $j \in T_2$ would be obtained by extending the ideal timetable to a complete one, by specifying the corresponding path in N^j and the arrival and departure times for all stations visited by the path.

However, the specifications of the ideal timetable may be modified in order to satisfy the operational constraints. In particular, one may increase the stopping time at some stations in M_j, choose a path between consecutive stations in M_j in N^j that does not correspond to the shortest one, and force train j to stop in some intermediate stations in C_j (we do not allow a train to slow down between two consecutive stations: we take a slowing down between stations h, i into account by considering an increase in the stopping time in station i). The resulting timetable will be referred to as the *actual timetable* for train j.

Track capacity constraints impose a minimum distance between trains travelling on the same track segment. First of all, it is not allowed that two trains travelling in opposite directions are on the track at the same time nor that two trains travelling in the same direction overtake each other on the track. Moreover, minimum time intervals between consecutive departures/arrivals on the same track segment are specified. More precisely, for each bi-directional track segment represented by edge e and each endpoint (station) i of e, we have:

$d(i, e)$: minimum time interval between consecutive departures (of trains travelling in the same direction) from i on (the track segment represented by) e;

$a(i, e)$: minimum time interval between consecutive arrivals (of trains travelling in the same direction) in i on e;

$f(i, e)$: minimum time interval between an arrival in i and a departure from i on e (of trains travelling in opposite directions);

$g(i, e)$: minimum time interval between a departure from i and an arrival in i on e (of trains travelling in opposite directions).

In most cases, the value $g(i, e)$ is not imposed as an explicit constraint but turns out to be equal to

$$\text{minimum travel time from } i \text{ to } h \text{ on } e + f(h, e) + \text{minimum travel time from } h \text{ to } i \text{ on } e,$$

where h is the other endpoint of e. Nevertheless, for notational convenience we treat $g(i, e)$ as an explicit constraint.

For a mono-directional track segment represented by arc $a = (h, i)$ the situation is similar but of course only $d(h, a)$ and $a(i, a)$ are defined.

The objective function is aimed at optimizing the insertion of new freight trains, according to the requests of train operators who require that the trains are scheduled with the smallest possible "deviation" from the ideal timetable given in input.

For this reason, we associate with each train $j \in T_2$ a *profit* given by $\pi_j - \phi_j(\nu_j) - \gamma_j \mu_j$, where

1. π_j is the *train priority*, that is the profit that would be achieved if the train travelled according to its ideal timetable,

2. ν_j is the *shift*, that is the absolute difference between the departure times from station f_j in the ideal and actual timetables,

3. μ_j is the *stretch*, that is the (nonnegative) difference between the "running times" (i.e. the differences between the arrival time at l_j and the departure time from f_j) in the actual and ideal timetables;

4. $\phi_j(\nu_j)$ is a user-defined nondecreasing function penalizing the train shift (with $\phi_j(0) = 0$),

5. γ_j is a given nonnegative parameter (i.e., the function penalizing the train stretch is assumed to be linear).

The *objective* is to maximize the overall profit, i.e., the sum of the profits of the scheduled trains. Of course, if the profit of train j turns out to be non-positive, it is better to cancel train j.

Note that the stretch has to be handled with some care since train j may take different paths along the network to reach the next station that it must visit. The fact that rather than assigning a large shift and/or stretch it is better to cancel the train defines *implicit time windows constraints* for the arrival and departure time of train j at each station in M_j, and consequently also in each station in C_j.

We also define N^j for $j \in T_1$, where N^j denotes the *auxiliary* network with vertex set S^j and one arc for each pair of stations, oriented according to the train direction. The set M_j coincides with the set S^j and the set C_j is empty.

3.4 The Space-Time Graph

Times are here discretized and expressed as integers from 1 to $q := 1440$ (the number of minutes in a day), though a finer discretization would also be possible (e.g., 1/2, 1/4 minute) without changing the model. In the following, *time instants* indicate a particular instant within the day, while *time intervals* indicate a continuous interval (not longer than q).

Our space-time directed multi-graph $G = (V, R)$, with node set V and arc set R, is defined as follows.

3.4.1 The node set of G

For each edge e in the railway network N and each endpoint i of e, we define a set $U(i, e)$ of *arrival nodes* in station i on (the track represented by) e and a set $W(i, e)$ of *departure nodes* from station i on e. There is one node in $U(i, e)$ and one node in $W(i, e)$ for each time instant in which a train may arrive in and depart from station i on e, respectively, taking into account the time discretization mentioned above. Analogously, for each arc $a = (h, i)$ in N, there are a set $U(h, a)$ of arrival nodes in h on a and a set $W(i, a)$ of departure nodes from i on a.

In addition, for notational convenience, we have in G a fictitious *source* node σ and a fictitious *sink* node τ. Summarizing,

$$V = \{\sigma, \tau\} \cup \bigcup_{e=\{h,i\}\in E} (U(h, e) \cup W(h, e) \cup U(i, e) \cup W(i, e)) \cup \bigcup_{a=(h,i)\in A} (U(i, a) \cup W(h, a)).$$

Let $\theta(v)$ be the time instant associated with a given node $v \in V$. Considering the "cyclic" nature of the time horizon, we will perform additions and subtractions on these time instants implicitly intending that the result should be taken modulo q. Let $\Delta(u, v) := \theta(v) - \theta(u)$. We say that node u *precedes* node v (i.e., $u \preceq v$) if $\Delta(v, u) \geq \Delta(u, v)$ (i.e, if the cyclic time interval between $\theta(v)$ and $\theta(u)$ is not smaller than the cyclic time interval between $\theta(u)$ and $\theta(v)$). Analogously, we will use the notation $u \prec v$, $u \succeq v$, $u \succ v$.

The set $V^j \subseteq V$ of nodes associated with each train $j \in T$ is defined by the nodes in V corresponding to possible departure and arrivals of train j in its path, plus the source and the sink.

Note that for $j \in T_1$ set V^j contains just one departure and one arrival node in $S^j \backslash \{f_j, l_j\}$, and one departure node for f_j and one arrival node for l_j, as the timetable is already defined.

3.4.2 The arc set of G

Arc set R is partitioned into sets R^j, one for each train $j \in T$. Each set R^j can be defined starting from the railway network N and the associated auxiliary network N^j for train j defined above.

For each arc (h, i) of N^j, corresponding to two stations $h, i \in M_j \cup C_j$ such that train j may travel from h to i on a track segment, consider each $e \in \delta_N^+(h) \cap \delta_N^-(i)$ and the travel time t for train j along the corresponding track segment. R^j contains $|W(h, e) \cap V^j|$ *segment* arcs of the form (v, u), one for each $v \in W(h, e) \cap V^j$ and $u \in U(i, e) \cap V^j$ such that $v \preceq u$ and

$\Delta(v,u) = t$. These arcs represent the travel of train j from station h to station i, recalling that the departure time of train j along a track segment uniquely determines its arrival time.

Moreover, for each station $i \in (M_j \cup C_j) \setminus \{f_j, l_j\}$, let t be the minimum stopping time for train j in i (possibly null, and in particular null if $i \in C_j$). For each edge $e_1 \in \delta_N^-(i)$ such that $U(i,e_1) \cap V^j \neq \emptyset$ and each edge $e_2 \in \delta_N^+(i)$ such that $W(i,e_2) \cap V^j \neq \emptyset$), R^j contains up to $|U(i,e_1) \cap V^j| \cdot |W(i,e_2) \cap V^j|$ *station* arcs of the form (u,v) for each $u \in U(i,e_1) \cap V^j$ and $v \in W(i,e_2) \cap V^j$ such that $u \preceq v$ and $\Delta(u,v) \geq t$. These arcs represent the stop of train j in station i, noting that it may arrive at i on any track and depart from i on any other track. In the current representation, there is no time charged to change track in the station, but this could easily be included in the model.

Finally, R^j contains the *starting* arcs (σ, v), for $v \in (\bigcup_{e \in \delta_N^+(f_j)} W(f_j,e)) \cap V^j$, and the *ending* arcs (u,τ), for $u \in (\bigcup_{e \in \delta_N^-(l_j)} U(l_j,e)) \cap V^j$.

Note that for trains $j \in T_1$ the arc set R^j contains just one arc for each pair of nodes in V^j.

3.4.3 Timetables in G

For each train $j \in T$, let $G^j = (V^j, R^j)$ be the subgraph of G induced by node set V^j and arc set R^j. Note that, while G is not necessarily acyclic due to the periodic nature of the problem, G^j *is acyclic* since so is the auxiliary network N^j.

A timetable for train j is now represented by a path in G^j from σ to τ. Also the converse holds, due to the requirements on the structure of N^j that guarantee that all the stations in M_j are visited (in the order specified) by an arbitrary path. Letting (σ, w) and (u,τ) be, respectively, the first and last arc in a path, the *shift* of the path is given by

$$\nu(w) := |\theta(w) - [\text{ideal departure time of } j \text{ from } f_j]|$$

and its *stretch* by

$$\Delta(w,u) - [\text{ideal travel time of } j \text{ from } f_j \text{ to } l_j].$$

While the shift of the path is uniquely determined by its starting node, computing the stretch according to the formula above requires knowing which is the last node in the path. Since it is necessary to know the stretch as long as the path is constructed, we decided to assign a stretch value to each arc in R^j, which is defined as follows. For each station arc $(u,w) \in R^j$, associated with station $i \in M_j \cup C_j$, the stretch is defined by

$$\mu(u,w) := \Delta(u,w) - [\text{ideal stop time for train } j \text{ in station } i].$$

In order to define the stretch of segment arcs, we first define the stretch $\eta(e)$ for each

$e \in E \cup A$ corresponding to an arc (h, i) in N^j, and then let the stretch $\mu(w, u)$ of each segment arc $(w, u) \in R^j$, $u \in U(h, e)$ and $w \in W(i, e)$, be equal to $\eta(e)$.

The definition of $\eta(e)$ is the following. Consider the path P in the auxiliary network N^j associated with the ideal timetable for train j, i.e. the shortest possible path linking the starting station f_j to the ending stations l_j, that visits every station in M_j. This shortest path is computed considering the *ideal departure time* of train j from station f_j (i.e. that given on input in the ideal timetable), the minimum travel time for the train along each track segment that can be visited and the minimum stopping time for each station in M_j.

For each station i visited by P, let the *ideal arrival time* and the *ideal departure time* be those in the ideal timetable, i.e. those computed considering the shortest path from f_j to l_j (if $i \in C_j$, these two values coincide and are uniquely defined by the departure time from the predecessor station in M_j). Then, for each station $i \in C_j$ which is not in P, let h be the first station in M_j that is reached if the train visits i (note that h always exists, possibly being l_j). Let θ_h be the ideal arrival time at station h, and Δ_{ih} be the minimum travel time for j from i to h. The ideal arrival time of j at i, θ_i, is set to $\theta_h - \Delta_{ih}$. This corresponds to the latest instant at which the train should arrive at (and depart from) station i in order to be able to arrive at station h at the same instant as in the ideal timetable. The ideal departure time at station i is set to θ_i as well, as the train does not have to stop in i. Once the ideal arrival and departure times for all stations have been defined, we have

$$\eta(e) := [\text{travel time on track segment } e] -$$

$$[\text{ideal arrival time in station } h - \text{ ideal departure time from station } i], \tag{3.1}$$

where the times refer of course to train j. Note that, for a "detour" with respect to the path in the ideal timetable, all the stretching due to the detour is charged to the first arc.

Finally, the stretch of each starting and ending arc is null. Therefore, the stretch of a path in G^j for train j is given by the sum of the stretches of the corresponding arcs.

More generally, we want the profit of a timetable to be equal to the sum of the profits of the arcs in the corresponding path. This is achieved by defining for each starting arc (σ, w) the profit $p_{(\sigma,w)} := \pi_j - \phi_j(\nu(w))$, for each ending arc (u, τ) the null profit $p_{(u,\tau)} := 0$, and for each other arc $(u, w) \in R^j$ the profit $p_{(u,w)} := -\gamma_j \mu(u, w)$.

As for trains $j \in T_1$, each of them has exactly one path P, corresponding to the prescribed timetable that is given on input and must not be changed.

3.5 ILP formulation

We next illustrate the *Integer Linear Programming* (ILP) formulation that we use for our problem, based on the space-time graph G. We introduce binary variables x_r associated with arcs $r \in R$ as well as binary variables y_v associated with each node $v \in V$ and z_{jv} associated with each train-node pair $j \in T$ and $v \in V^j$. We express the constraints imposing that each train is either cancelled or associated with a feasible path by means of the x_r variables, whereas the track capacity constraints are imposed via the y_v and z_{jv} variables. Trains $j \in T_1$ are considered as the other trains, since the structure of the corresponding graph itself imposes the only one possible path for each train. Moreover, they must be considered as they forbid other freight trains to visit nodes which do not satisfy safeness operational constraints.

Then, we relax the track capacity constraints in a Lagrangian way (thus associating a penalty with each node and train-node pair) and solve the relaxed problem as $|T|$ optimal path problems on G^j ($j \in T$). This implies that the arcs in R are considered only in this optimal path computation and not in the rest of the procedure.

We first introduce the x_r variables and define the model without track capacity constraints. For each train $j \in T$ and each arc $r \in R^j$, we introduce a binary variable x_r equal to 1 if and only if the path in the solution associated with train j contains arc r. Trains in T_1 have just one arc going out from σ and the corresponding binary variable x_r is set to 1. For notational convenience, for each train $j \in T$ and node $v \in V^j$, let $\delta_j^+(v)$ and $\delta_j^-(v)$ denote the (possibly empty) sets of arcs in R^j leaving and entering node v, respectively. The ILP model, without the track capacity constraints, then reads

$$\max \sum_{j \in T} \sum_{r \in R^j} p_r x_r \tag{3.2}$$

subject to

$$\sum_{r \in \delta_j^+(\sigma)} x_r \leq 1, \qquad j \in T_2, \tag{3.3}$$

$$\sum_{r \in \delta_j^+(\sigma)} x_r = 1, \qquad j \in T_1, \tag{3.4}$$

$$\sum_{r \in \delta_j^-(v)} x_r = \sum_{r \in \delta_j^+(v)} x_r, \qquad j \in T, v \in V^j \setminus \{\sigma, \tau\}, \tag{3.5}$$

$$z_{jv} = \sum_{r \in \delta_j^-(v)} x_r, \qquad j \in T, v \in V^j \setminus \{\sigma, \tau\}, \tag{3.6}$$

$$y_v = \sum_{j \in T : v \in V^j} z_{jv}, \qquad v \in V \setminus \{\sigma, \tau\}, \tag{3.7}$$

$$x_r \geq 0, \qquad r \in R, \tag{3.8}$$

$$x_r \text{ integer}, \qquad r \in R. \tag{3.9}$$

As discussed next, the capacity constraints are modeled as *clique constraints* stating that the sum of a set of pairwise incompatible variables is at most 1, where two binary variables are called *incompatible* if they cannot both take the value 1 in a feasible solution.

3.5.1 Arrival/departure constraints

For each edge $e \in E$ and endpoint i of e, the first class of additional constraints forbids arrivals/departures at station i on (the track segment represented by) edge e that are too close in time. These inequalities involve the y_v variables only and can be defined by specifying the node $w_1 \in W(i,e)$ corresponding to the earliest departure and the node $u_1 \in U(i,e)$ corresponding to the earliest arrival involved in the constraint, so that y_{u_1} and y_{w_1} are incompatible, i.e., either $u_1 \preceq w_1$ and $\Delta(u_1, w_1) < f(i,e)$ or $w_1 \preceq u_1$ and $\Delta(w_1, u_1) < g(i,e)$. Once u_1 and w_1 have been specified, let $w_2 \in W(i,e)$ be the node such that $w_2 \succ w_1$, y_{w_2} is compatible either with y_{w_1} or with y_{u_1}, and $\Delta(w_1, w_2)$ is minimum, i.e., either $\Delta(w_1, w_2) = d(i,e)$ or $\Delta(u_1, w_2) = f(i,e)$. Moreover, let $u_2 \in U(i,e)$ be the node such that $u_2 \succ u_1$, y_{u_2} is compatible either with y_{w_1} or with y_{u_1}, and $\Delta(u_1, u_2)$ is minimum, i.e., either $\Delta(u_1, u_2) = a(i,e)$ or $\Delta(w_1, u_2) = g(i,e)$. The constraint then reads

$$\sum_{u \in U(i,e) : u_1 \preceq u \prec u_2} y_u + \sum_{w \in W(i,e) : w_1 \preceq w \prec w_2} y_w \leq 1. \tag{3.10}$$

Once more, there is one such constraint for each $e \in E$, $i \in S$ endpoint of i, $u_1 \in U(i,e)$ and $w_1 \in W(i,e)$ such that y_{u_1} and y_{w_1} are incompatible.

The constraints associated with each arc $a = (h,i) \in A$ are analogous, with the difference that they involve only nodes in $W(h,a)$, in which case $\Delta(w_1, w_2) = d(i,e)$, or only nodes in $U(i,a)$, in which case $\Delta(u_1, u_2) = a(i,e)$.

3.5.2 Overtaking constraints

The second class of additional constraints forbids overtakings between trains traveling on the same direction along the same track segment. These constraints involve only the z_{jv} variables

and are based on the assumption that the travel time of a train along a track segment is fixed.

Consider a track segment joining stations h and i, represented by $e \in E \cup A$. The constraints are defined by specifying two trains j, k such that (h, i) is an arc in both auxiliary networks N^j and N^k and $t_j \geq t_k$, where t_j and t_k are the travel times of j and k on e, along with a node $w_1 \in W(h, e) \cap V^j$ and a node $w_2 \in W(h, e) \cap V^k$ such that z_{jw_1} and z_{kw_2} are incompatible, i.e., either $\min\{\Delta(w_1, w_2), \Delta(w_2, w_1)\} < d(h, e)$, $\min\{\Delta(v_1, v_2), \Delta(v_2, v_1)\} < a(h, e)$, or $w_1 \prec w_2 \prec v_2 \prec v_1$ (overtaking), where v_1 and v_2 are the arrival nodes for j and k corresponding to w_1 and w_2, respectively. Nodes w_1 and w_2 represent the earliest departure from h for trains j and k, respectively, involved in the constraint.

Let $w_3 \in W(h, e) \cap V^j$ be such that $w_3 \succ w_1$, z_{jw_3} is compatible with z_{kw_2}, and $\Delta(w_1, w_3)$ is as small as possible, i.e., $\Delta(w_2, w_3) = d(h, e)$. This means that the departure of k at $\theta(w_2)$ and the departure of j at $\theta(w_3)$ from h on e would be at the minimum feasible distance in time. (Actually, it may be the case that the departure of j from h at time $\theta(w_2)+d(h, e)$ is infeasible, in which case we let w_3 be the earliest node such that $w_3 \succ w_1$ and $w_3 \in W(h, e) \setminus V^j$.) Similarly, let $w_4 \in W(h, e) \cap V^k$ be such that $w_4 \succ w_2$, z_{kw_4} is compatible with z_{jw_1}, and $\Delta(w_2, w_4)$ is as small as possible, i.e., $\theta(w_1) + t_j + a(i, e) = \theta(w_4) + t_k$. This means that the departure of j at $\theta(w_1)$ and the departure of k at $\theta(w_4)$ from h on e would correspond to arrivals in i at the minimum feasible distance in time. (Actually, it may be the case that the departure of k from h at time $\theta(w_1) + t_j + a(i, e) - t_k$ is infeasible, in which case we let w_4 be the earliest node such that $w_4 \succ w_2$ and $w_4 \in W(h, e) \setminus V^k$.) The constraint has the form

$$\sum_{w \in W(h,e) \cap V^j : w_1 \preceq w \prec w_3} z_{jw} + \sum_{w \in W(h,e) \cap V^k : w_2 \preceq w \prec w_4} z_{kw} \leq 1. \tag{3.11}$$

There is one such constraint for each $e \in E \cup A$, $j, k \in T$ such that (h, i) is an arc in both N^j and N^k and $t_j \geq t_k$, where h, i are the endpoints of e (and $e = (h, i)$ if $e \in A$), $w_1 \in W(h, e) \cap V^j$ and $w_2 \in W(h, e) \cap V^k$ such that z_{jw_1} and z_{kw_2} are incompatible.

3.5.3 Crossing constraints

The third and last class of additional constraints forbids crossings between trains traveling in opposite directions along the same track segment. These constraints are similar to (3.11), involving only the z_{jv} variables and being based on the assumption that the travel time of a train along a track segment is fixed.

Consider a bi-directional track segment joining stations h and i, represented by $e \in E$. The constraints are defined by specifying two trains j, k such that (h, i) and (i, h), respectively, are arcs in the auxiliary networks N^j and N^k, along with a node $w_1 \in W(h, e) \cap V^j$ and a node $w_2 \in W(i, e) \cap V^k$ such that z_{jw_1} and z_{kw_2} are incompatible, i.e., either $v_2 \preceq w_1$ and

$\Delta(v_2, w_1) < f(h, e)$, or $v_1 \preceq w_2$ and $\Delta(v_1, w_2) < f(i, e)$, or $w_1 \prec v_2$ and $w_2 \prec v_1$ (crossing), where v_1 and v_2 are the arrival nodes for j and k corresponding to w_1 and w_2, respectively. Nodes w_1 and w_2 represent the earliest departure from h for train j and the earliest departure from i of k, respectively, involved in the constraint. Again, let t_j and t_k be the travel times of j and k on e.

Let $w_3 \in W(h, e) \cap V^j$ be such that $w_3 \succ w_1$, z_{jw_3} is compatible with z_{kw_2}, and $\Delta(w_1, w_3)$ is as small as possible, i.e., $\theta(w_2) + t_k + f(h, e) = \theta(w_3)$. This means that the departure of k at $\theta(w_2)$ from i on e and the departure of j at $\theta(w_3)$ from h on e would correspond to an arrival and a subsequent departure in h at the minimum feasible distance in time. (If the departure of j from h at time $\theta(w_2) + t_k + f(h, e)$ is infeasible, let w_3 be the earliest node such that $w_3 \succ w_1$ and $w_3 \in W(h, e) \setminus V^j$.) Symmetrically, let $w_4 \in W(h, e) \cap V^k$ be such that $w_4 \succ w_2$, z_{kw_4} is compatible with z_{jw_1}, and $\Delta(w_2, w_4)$ is as small as possible, i.e., $\theta(w_1) + t_j + f(i, e) = \theta(w_4)$. (If the departure of k from i at time $\theta(w_1) + t_j + f(i, e)$ is infeasible, let w_4 be the earliest node such that $w_4 \succ w_2$ and $w_4 \in W(h, e) \setminus V^k$.) The constraint has the form

$$\sum_{w \in W(h,e) \cap V^j : w_1 \preceq w \prec w_3} z_{jw} + \sum_{w \in W(i,e) \cap V^k : w_2 \preceq w \prec w_4} z_{kw} \leq 1. \tag{3.12}$$

There is one such constraint for each $e = \{h, i\} \in E$, $j, k \in T$ such that (h, i) is an arc in N^j and (i, h) an arc in N^k, $w_1 \in W(h, e) \cap V^j$ and $w_2 \in W(i, e) \cap V^k$ such that z_{jw_1} and z_{kw_2} are incompatible.

3.6 Solution Method

Based on the above model, the solution method that we adopted has the same structure as the one in [15], namely the track capacity constraints are relaxed in a Lagrangian way and near-optimal Lagrangian multipliers are determined through subgradient optimization.

As the number of relaxed constraints is far too large to be handled explicitly, we use a dynamic constraint-generation scheme sometimes referred to as *relax-and-cut*, see Fisher [27] and Escudero, Guignard and Malik [25]. We implicitly set all the Lagrangian multipliers to 0 and initialize a *constraint pool structure* to be empty. At each iteration of the subgradient optimization procedure, we identify constraints which are violated by the relaxed solution, possibly adding these constraints to the pool. More precisely, if two trains arrive at station i on the same track at time instants which are too close to each other, we check whether the pool contains any arrival constraint both variables corresponding to this station, track and these time instants. If such a constraint is found, we simply update the associated Lagrangian multiplier, otherwise we add to the pool the constraint. Train departures from a station on the same track in time instants too close to each other are handled in an analogous way. Then, if

train j overtakes train k between two consecutive stations i and $i+1$ on the same track without violating any departure constraint from station i and arrival constraint in station $i+1$, we have a violated overtaking constraint. In this case, we check whether the pool contains any overtaking constraint involving variables corresponding to these trains, stations, track and time instants. If such a constraint is found, we update its Lagrangian multiplier, otherwise we add to the pool the constraint. Finally, if a train j travels in the opposite direction of a train k between two stations i and $i+1$ on the same track without violating any constraint imposing a minimum time interval between an arrival and a departure in station $i+1$ and a departure and an arrival in station i, this means that the two trains are crossing each other. In this case, we check whether the pool contains any crossing constraint involving variables corresponding to these trains, stations, track and time instants. If such a constraint is found, we update its Lagrangian multiplier, otherwise we add to the pool the constraint.

We now describe how the subgradient procedure works.

Consider the h-th constraint in the pool, and let τ_h be the *subgradient component* of this constraint corresponding to the optimal solution of the current relaxation, given by the difference between the right-hand-side and the left-hand-side values. The associated Lagrangian multiplier λ_h is updated in a standard way by using the well-known formula introduced by Held and Karp [30]:

$$\lambda_h = \max\left\{\lambda_h + \rho\frac{UB - LB}{||\tau||^2}\tau_h, 0\right\} \tag{3.13}$$

where LB is the value of the best TTP solution found, UB is the best upper bound, $\rho > 0$ is a given *step size parameter*, and τ is the overall *subgradient vector* associated with the solution of the current relaxation. We remove the constraint from the pool if the associated multiplier reaches the value zero.

The initial value of the step size ρ is 1. We use the standard technique of halving ρ when no upper bound improvement occurs for a certain number of iterations (10 in our implementation). As to the stopping condition, it arises when ρ becomes smaller than a given threshold, and will be illustrated in detail in the next section.

At each iteration of the subgradient procedure, we apply a constructive heuristic algorithm based on the Lagrangian profits and a refinement procedure to improve the solution found. Finally, when the subgradient procedure has converged, we fix some paths in the solution and re-apply subgradient optimization to the resulting problem. Below we describe the heuristic algorithm and the main issues that are different with respect to the previous method. For a detailed description of the heuristic algorithm and of the fixing phase, see [15].

3.6.1 Heuristic algorithm

The structure of the heuristic algorithm is as follows:

1. **Constructive Phase**

   a. initially we order the trains one after the other according to decreasing profits in the Lagrangian (infeasible) solution,

   b. for each train we try to schedule it by finding the optimal path, with respect to the *Lagrangian* profits, that is compatible with the paths of the trains previously scheduled. Being the schedule of the previous trains fixed, finding the (locally optimal) schedule of the next train j in the sequence simply amounts to the solution of a maximum profit path problem in the subgraph of G resulting from the removal of the arcs of R^j which are incompatible with those corresponding to the trains already scheduled. If the actual (i.e., not Lagrangian) profit of this path is non-positive, or if no path can be found for the train, the train is not scheduled in the solution. Choosing the path with maximum Lagrangian profit, instead of the path with maximum actual profit, is an attempt to take care of the trains with lower ranking which still have to be scheduled. Besides finding better solutions than its counterpart based on actual profits, this policy yields substantially different heuristic solutions for different Lagrangian multipliers, increasing the likelihood of improving the incumbent solution.

2. **Refining Phase** For each train that is scheduled in the solution (in arbitrary order), we check if the associated timetable is shifted and/or stretched with respect to the ideal one. We try to improve the solution by considering each scheduled train j in turn, keeping all paths in the solution fixed with the exception of the one of train j, that is replaced by the path which is optimal with respect to the *original* profits (and of course compatible with the other paths). This path may turn out to have a higher profit than the initial one, which was computed so as to maximize the Lagrangian profit, in which case the old path is replaced by the new one. Afterwards, for each train that is not scheduled in the solution (in any order), we compute the path with maximum actual profit compatible with the scheduled trains, and test whether this profit is indeed positive. This may happen either because we are now considering actual profits instead of Lagrangian ones, or because the paths of the other trains have been changed by the previous phase. Whenever a path with positive profit for an unscheduled train is found, it is added to the solution. The two phases above, namely the recomputation of the paths for the trains already scheduled and the attempt to find positive profit paths for the unscheduled trains, are iterated until no improvement in the solution value is achieved in both phases.

The main difference with respect to the single one-way track case is that here we have to deal with a network, where we may have bi-directional track segments and alternative

routes between pairs of stations. Moreover, train operators could ask for having a maximum shift and a maximum stretch according to the type of the train, e.g. distinguishing between different types of freight trains. Finally, we have the fixed passenger trains which cannot be moved from their ideal timetable.

In order to consider these new features, we start constructing the paths for the fixed passenger trains.

Then, while constructing the maximum profit path for each new freight train according to the Lagrangian profits, we forbid the nodes already occupied by the previous trains, according to the safeness operational constraints.

Moreover, we allow the train to use alternative routes even if they cause a longer travel time. This means that the stretch due to the use of an alternative longer route must not be considered as the stretch due to the increase in the stopping time, otherwise no train would choose a longer route, without reaching a null or negative profit.

The maximum shift imposes an explicit time-window constraint, which can be more strict than that due to the decreasing of the profit.

The maximum stretch instead has to be dealt in a different way: when the train operators require not to overtake the maximum stretch, they refer to the overall stretch of the train (due to the increase in the stopping time) along the entire path. This means that a priori we are not able to limit the time-windows for each station, since the train can consume all the stretch in one station. In order not to have a great computational time we have decided to treat heuristically the check on the maximum stretch, just avoiding the path when it overtakes the maximum stretch.

3.7 Case study

In this section we present some computational results on real-world instances of Rete Ferroviaria Italiana (the main Italian Railway Infrastructure Company) obtained by the heuristic algorithm described in Section 3.6.

The case we consider is a very complex railway network, where we have many different alternative routes for going from a station to another one, as well as the so-called *railway nodes*, i.e., big stations where we can have many different lines and different routes also inside the station. Moreover, some tracks can be used in both directions, i.e., we are not allowed to separate the trains going in opposite directions.

The scenario presents the typical situation where we have a large number of passenger trains for which we cannot change the timetable ("fixed trains"). Then we consider the additional request of new freight trains, formulated by the train operators.

We have analyzed different situations for the requests of new trains, by considering as

fixed trains those that are scheduled in the actual timetable.

We consider the railway network composed by the corridor Kufstein-Verona Porta Nuova; the corridor Verona Porta Nuova-Bologna, which presents some double-way line segments; the railway node of Bologna; the alternative routes (some with double-way tracks) from Bologna to Rome (via Florence and via Falconara); the railway node of Florence; the different possible routes going from Florence to Rome ("direttissima", i.e., the fast line, and "lenta", i.e., the slow line); the railway node of Rome. This railway network presents many different features, and represents a good test bed for the proposed algorithm.

The railway network is represented in Figure 3.1 and Figure.

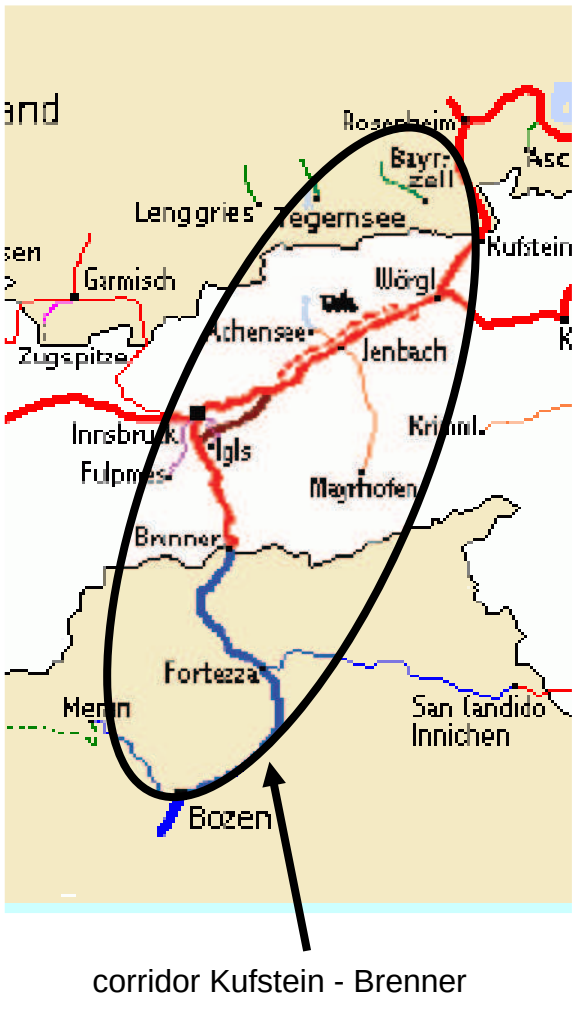

Figure 3.1: Corridor Kufstein-Brenner

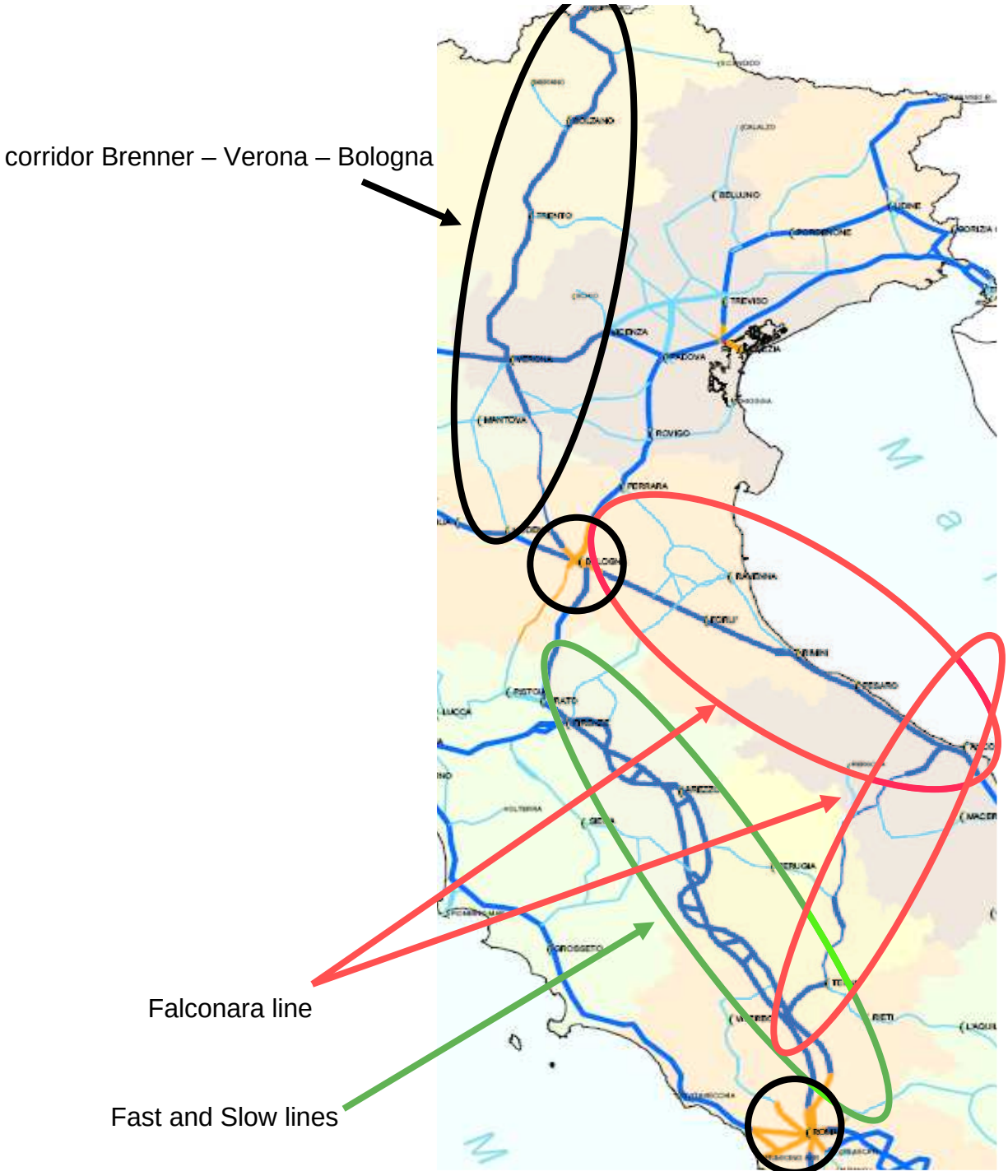

Figure 3.2: Corridor Brenner-Verona-Bologna, Falconara Line, Fast and Slow Lines

3.7.1 Results

We have different possibilities for choosing a path for a new train: in particular, we can choose a longer route in case this route is rarely visited by fixed trains, taking of course into account

the larger costs incurred because of the longer distance travelled by the train. Accordingly, we give a penalty to the choice corresponding to a longer route for reaching a station that must be visited by the train. Although such a penalty leads the algorithm to schedule most of the new trains along the fastest route, generally more trains are scheduled by using the slower routes as well.

We start with a feasible timetable which contains all the fixed trains (that cannot be changed), and we try to satisfy the requests of the train operators for adding new freight trains. We consider 679 fixed trains (230 from Kufstein to Verona Porta Nuova and 449 from Verona Porta Nuova to Rome) and two different cases obtained by adding 24 new trains (one each hour) and 48 new trains (one each half an hour), all starting from Kufstein and ending in Rome.

We have also analyzed an additional case, where more new freight trains are considered by adding 96 new freight trains (one each 15 minutes). This case is used to obtain a measure of the so-called *residual capacity* of the railway network, i.e. the number of new freight trains which can be run in the network (without changing the timetable of the fixed trains).

For each new train, we consider a maximum shift of 15 minutes and different values for the maximum stretch (20 minutes, 30 minutes and 45 minutes).

We have implemented the algorithm in C language and we have executed the tests on a PC Pentium IV, 1 GB Ram, 3.2 GHz.

In Tables 1 and 2 we present some computational results obtained in two different scenarios: the first one forbids the use of the alternative slower line via Falconara, whereas the second one gives the possibility to use it. The computational times are shorter (below 1000 seconds) for the first scenario. The computational times increase when we consider the possibility of the alternative route ia Falconara, but in this case a greater number of new freight trains can be scheduled. In order to obtain these results we applied the constructive heuristic and the refinement procedure described in Section 3.6.1.

Name	max str (20 min)	time (sec)	max str (30 min)	time (sec)	max str (45 min)	time (sec)
case 24 trains	13	(130)	21	(132)	22	(133)
case 48 trains	29	(401)	43	(427)	46	(668)
case 96 trains	50	(447)	74	(510)	90	(993)

Table 3.1: Results for the railway network Kufstein-Rome, without the use of the Falconara line.

Name	max str (20 min)	time (sec)	max str (30 min)	time(sec)	max str (45 min)	time (sec)
case 24 trains	16	(134)	22	(146)	24	(210)
case 48 trains	32	(497)	46	(635)	48	(956)
case 96 trains	56	(782)	83	(1066)	95	(2308)

Table 3.2: Results for the railway network Kufstein-Rome, with the use of the Falconara line.

3.8 Conclusion

The problem we have studied deals with the growing necessity of using the railway infrastructures for freight transportation instead of the highways, both for safety and savings reasons. In particular, we have considered the case in which new freight trains are added in a real-world railway network, where safeness operational constraints must be satisfied and a prescribed timetable for a large set of passenger trains is given.

The problem has been modelled by means of a space-time graph and an ILP-formulation based on this graph, which includes new classes of constraints due to the structure of the network. The solution of the problem has been obtained by applying a Lagrangian relaxation of the safeness operational constraints and executing a heuristic algorithm based on this relaxation.

The computational results presented in Section 3.7 for a set of case studies concerning a large scale real-world railway network show the effectiveness of the proposed approach for analyzing different scenarios in short computing times.

Chapter 4

Solving a Real-World Train Unit Assignment Problem

4.1 Introduction

The assignment of locomotives and cars to trains with published timetables is a key problem to be faced by operators of passenger trains, given that the acquisition of rolling stock is an expensive long-term investment, and that fulfilling the passenger requests, namely guaranteeing (within reasonable margins) that each passenger has a seat, is fundamental to ensure customer satisfaction.

Given its importance, the problem has been widely studied in the literature on railway optimization; for surveys on the specific problem as well as on the use of combinatorial optimization in railway planning see, e.g., [10, 16, 20, 24, 32]. In this chapter, we illustrate how we solved a real-world case of the problem for the trains operated by a passenger Train Operator operating in a regional area. In this problem, so-called *Train Units* (TUs), rather than locomotives and cars, have to be assigned to trains. A TU is a self-contained train with an engine and passenger seats, and TUs can be combined together to increase the number of available seats. Even if most of the approaches in the literature consider the case in which locomotives and (locomotive-hauled) cars have to be assigned to trains [9, 21, 22, 23, 24, 40, 51], there are a few references that consider the assignment of TUs. Among these, [5] considers the case in which there is a unique type of TU, that, as will be clear in the following, would make the specific problem in our case study elementary to be solved. Moreover, also the other references on TU assignment [1, 2, 26, 50, 53] consider the case in which there is a very small number of distinct TU types (two in most cases). On the other hand, the peculiar characteristic of our case study is the relatively large number of distinct TU types, namely eight, that are available to the train operator. (This is opposed to the rules for composing TUs

for a trip, which are much simpler in our case than in the cases addressed in the references above, e.g., at most TUs can be assigned to each trip.)

The large number of TU types, along with the fairly large number of train trips to be covered, namely a few hundred, make our case study very challenging from an optimization viewpoint. In particular, although unavoidably the mathematical programming models that one may consider are analogous to those presented in the references mentioned above, the optimal solution of these models appears to be out of reach at the moment. Moreover, even only finding a feasible solution following the classical heuristic approaches, based or not on these models, is far from trivial. On the other hand, we eventually managed to design an effective heuristic procedure based on an appropriate *Integer Linear Programming* (ILP) formulation that allowed us to find a solution about 10% better than the "manual" solution found by practitioners. Based on our previous experience on similar case studies, we found this very strange: there is a wide margin of improvement over the manual solution, but even only finding a feasible solution of the same value as the manual one (which is feasible according to our formal definition of the problem) is hard, or at least was hard for us.

As will be discussed next, the key constraints of the problem concern the minimum number of passenger seats that have to be assigned to every trip. In ILP models, this is naturally formulated as a "knapsack-type" constraint in "$\geq$" form. The numerical nature of this constraint makes it very "weak" when the *Linear Programming* (LP) relaxation of the problem is considered, as already observed in [53]. In particular, none of the approaches we tried, among those based on ILP models and LP relaxation, managed to find a feasible solution as long as we stuck to these constraints. On the other hand, taking into account the fact that in our case at most two TUs can be combined to cover a trip, we replaced the "weak" constraints above by the inequalities obtained from a complete description of the knapsack polytope for the special case in which the sum of the variables cannot exceed two. This is similar to what was done in [53], with the difference that in that case the description was found numerically, case by case, for polytopes with two variables, whereas in our case the upper bound of two on the variable sum allows a formal description that is valid for any number of variables. Our final heuristic method, based on the ILP model with these new inequalities, yields the results mentioned above.

The chapter is organized as follows. In Section 4.2 we formally define the problem considered in the chapter, whose computational complexity is analyzed in Section 4.3. ILP models are illustrated in Section 4.4, and strengthened as outlined above in Section 4.5. In Section 4.6 we illustrate our heuristic method. In Section 4.7 we consider an extension of the problem and present two ILP models, and a heuristic algorithm based on the Lagrangian relaxation of one of them. In Section 4.8 we present an additional method to compute bounds on the optimal solution value and in Section 4.9 we define additional maintenance constraints for the

problem and discuss how to deal with them. Finally, Section 4.10 presents our preliminary computational results on our case study.

4.2 Problem Description

Given a set of timetabled trips to be performed every day, and a set of TUs of different types, the *TU Assignment Problem* (TUAP) calls for the specification of the TUs to be used, and, for each of these TUs, of the associated trips. The sequence of trips associated with a TU corresponds to a possible daily workload for the TU, and must satisfy a set of sequencing constraints. For instance, in our case study, for each pair of consecutive trips in the sequence, the time elapsing between the arrival of the first one and the departure of the second one must be large enough to allow the TU to travel from the arrival station of the first one to the departure station of the second one (this is a deadhead in case the two stations do not coincide).

Given that there is an overnight break of a few hours, it is not necessarily the case that every TU used performs the same set of trips every day. Indeed, after having performed a sequence of trips on one day, a TU can perform on the following day a sequence of trips assigned to another TU of the same type (possibly performing a deadhead transfer within the night break). In other words, the, say, q trip sequences assigned to TUs of a given type can be numbered as $1, \ldots, q$ in an arbitrary way, and can be performed by q TUs of that type, all performing a different sequence on each day, and each one performing the q sequences in the cyclic order $1, \ldots, q$ over a period of q days. This is important when maintenance constraints, illustrated in Section 4.9, are introduced in the problem.

TUs can be assigned to the same trip in order to guarantee that the number of passenger seats required by the trip is reached. As our problem concerns a suburban area, there is no distinction between first and second class seats as in most references above. At the end of the trip, the TUs assigned to the trip can be uncoupled and assigned to different trips following the rules outlined above. In particular, the feasibility of a sequence of trips for a TU does not depend on the other TUs assigned to the trips, which is a notable simplification with respect to other cases of the problem addressed in the literature, see, e.g., [50]. This is related with the fact that in our case at most two TUs can be combined assigned to a trip, in order to keep coupling and uncoupling operations simple, so these operations take relatively short.

Although there are many factors contributing to the cost of a solution, such as deadheading or coupling/uncoupling operations, the dominant cost in the case we consider is related with the use of a TU, and in this chapter we will restrict ourselves to this cost. In fact, although we will formulate our model with a generic cost associated with the use of a TU of a given type, as is the case in [53], in our experiments our objective will be to minimize the overall

number of TUs used.

Formally, the problem input specifies a set of n train *trips* and a set of p TU types. Each trip $j \in \{1, \ldots, n\}$ is defined by a required number r_j of passenger seats, and a maximum number u_j of TUs that can be assigned to the trip. (Additionally, each trip is characterized by an arrival time and station and a departure time and station, and by a subset of TU types that can perform it, but this information is implicitly encoded in the graph illustrated below.) Each TU type $k \in \{1, \ldots, p\}$ is defined by a number d^k of available TUs, a cost c^k for each such TU used, and an associated capacity s^k (number of available seats). We say that a trip j is *covered* if the overall capacity of the TUs assigned to the trip is at least r_j. Finally, as is customary, the sequencing constraints are represented by a directed multigraph $G = (V, A)$, where each node corresponds to a trip, and in addition there are a dummy start node 0 and a dummy end node $n+1$, i.e., $V = \{0, \ldots, n+1\}$, and arc set A is partitioned into p subsets $A^1, \ldots, A^p$, where A^k is associated with TUs of type k for $k = 1, \ldots, p$. Given two distinct trips $i, j \in V \setminus \{0, n+1\}$, arc $(i, j)^k \in A^k$ exists if and only if a TU of type k can be assigned to i and then to j within the same day. (Specifically, arc $(i, j)^k$ exists whenever both trips i and j can be assigned to a TU of type k, and the time between the arrival of trip i and the departure of trip j allows a TU of such type to travel from the arrival station of trip i to the departure station of trip j). Moreover, the dummy nodes are connected with all other nodes, namely $(0, i)^k, (i, n+1)^k \in A^k$ for $i = 1, \ldots, n$ and $k = 1, \ldots, p$. Note that each subgraph (V, A^k) is simple and transitive. Given a node $i \in V$, we will let $\delta^k_-(i)$ and $\delta^k_+(i)$ denote, respectively, the set of arcs entering and leaving node i.

There is a one-to-one correspondence between trips assigned to a TU of type k and a path in G formed by arcs in A^k. The problem calls for the determination, for each TU type $k \in \{1, \ldots, p\}$, of up to d^k paths from 0 to $n+1$ formed by arcs in A^k, each path having cost c^k and capacity s^k, such that each trip $j \in \{1, \ldots, n\}$ is visited by at most u_j paths whose overall capacity is at least r_j, with the objective of minimizing the overall cost of the paths. In the following we will use the acronym TUAP to denote the problem just described.

In the specific application that we will consider, we have $u_j = 2$ for $j = 1, \ldots, n$, i.e., each trip can be assigned to at most two TUs. For this specific case, we will discuss how to write the constraints on the required number of seats in a way that is much stronger than the trivial one.

4.3 Problem Complexity

In this section we discuss the complexity of TUAP, proving in particular that the specific version considered in our case study is strongly NP-hard.

The first result shows that the real difficulty of the problem is due to the presence of

distinct TU types.

Observation 1 *TUAP is solvable efficiently in case $p = 1$, i.e., if there is a unique TU type.*

Proof. In this case, one can replace each trip j by $\lceil r_j/s^1 \rceil$ trips with the same timetable and request s^1: the associated problem calls for the determination of the minimum number of paths to cover all the vertices in a transitive directed acyclic graph, which is polynomially solvable by flow techniques (see, e.g., [29]). □

If distinct TU types are present, the problem is already difficult if each trip must be covered by one TU only, and the minimum connection time between two trips does not depend on the trips nor on the TU type (e.g., it is 0, as in the statement of the proposition below). This problem has already been considered in the literature as it arises in other applications, e.g., in the assignment of classrooms to timetabled classes, with the constraint that each class receives a classroom having a number of seats at least equal to the number of students attending the class. The following proposition is due to [6].

Proposition 1 *TUAP is strongly NP-hard in the special case in which $u_j = 1$ for $j = 1, \ldots, n$, and $(i, j) \in A^k$ if and only if the departure time of trip j is not smaller than the arrival time of trip i for $i, j = 1, \ldots, n$ and $k = 1, \ldots, p$.*

Proof. In this case, each TU of type k can be assigned to a set of trips S if and only if $r_j \leq s^k$ for $j \in S$ and the timetables of the trips in S do not pairwise overlap. The feasibility version of the problem, asking if it is possible to cover all the trips with the available TUs, is a possible equivalent reformulation of the following problem: given an interval graph $I = (N, E)$ (in which nodes in N correspond to intervals on the real line and an edge in E connects two nodes if and only if the corresponding intervals overlap) and a function $\mu : V \to \{1, \ldots, q\}$, decide if it is possible to color all the nodes of G with colors $\{1, \ldots, q\}$ so that adjacent nodes receive distinct colors and each node $j \in V$ receives a color at least equal to $\mu(j)$. This problem, called μ-coloring of an interval graph in [6], has been shown to be strongly NP-complete in that paper.

For instance, a trivial reduction from μ-coloring of an interval graph to the special case TUAP described in the statement is given by $n := |V|$, $r_j := \mu(j)$ and $u_j := 1$ for $j = 1, \ldots, n$, $p := q$, $d^k := 1$ and $s^k := k$ for $k = 1, \ldots, p$, $(i, j)^k \in A^k$ whenever the interval associated with node i precedes, without overlaps, the interval associated with node j, for $i, j = 1, \ldots, n$ and $k = 1, \ldots, p$. □

The above proof can be adapted to show that TUAP remains strongly NP-hard also in the case $u_j = 2$ for $j = 1, \ldots, n$, as is the case in our application, by defining r_j and s^k so that there is no advantage in covering a trip with two TUs, e.g., $r_j := 2^{\mu(j)} - 1$ for $j = 1, \ldots, n$ and $s^k := 2^k - 1$ for $k = 1, \ldots, p$ (the rest of the reduction is identical and the fairly easy details

about its correctness are left out). However, the following simpler reduction shows that, when $u_j = 2$ for $j = 1, \ldots, n$, the problem is strongly NP-hard even if all trips are simultaneous, due to its numerical nature. The proof is very similar to the one provided in [2] for the specific problem considered in that context.

Proposition 2 *TUAP is strongly NP-hard in the special case in which* $u_j = 2$ *for* $j = 1, \ldots, n$ *and* $A^k = \emptyset$ *for* $k = 1, \ldots, p$.

Proof. The reduction is from the following numerical 3-dimensional matching problem (see, e.g., [28]): given disjoint sets X, Y, Z, each containing m elements, a positive integer target value t, and a weight $w_j \in \{0, \ldots, t\}$ for each $j \in X \cup Y \cup Z$, decide if it is possible to partition $X \cup Y \cup Z$ into m disjoint triples $T_1, \ldots, T_m$, such that $|T_i \cap X| = |T_i \cap Y| = |T_i \cap Z| = 1$ and $\sum_{j \in T_i} w_j = t$ for $i = 1, \ldots, m$.

The reduction is the following: $n := m$, $r_j := 7t - w_j$ and $u_j := 2$ for $j \in X$, $p := 2m$, $d^k := 1$ for $k = 1, \ldots, p$, $s^k := 2t + w_k$ for $k \in Y$, $s^{m+k} := 4t + w_k$ for $k \in Z$. Moreover, we let $A^k = \emptyset$ for $k = 1, \ldots, p$, representing, e.g., the fact that all trips have the same timetable. In order to prove the correctness of the reduction, we observe that there is a one-to-one correspondence between solutions. Given a solution of numerical 3-dimensional matching, a feasible solution of our problem is obtained by assigning, for each triple, the two TUs corresponding to two items in the triple to the trip corresponding to an item in the triple. Viceversa, given a solution of our problem, note that each TU performs at most one trip, as all trips are simultaneous, and each trip is covered by exactly two TUs, as otherwise its request cannot be satisfied. Moreover, since the sum of the TU capacities is equal to the sum of the trip requests, and the sum of the capacities of two TUs corresponding to items in Y is smaller than the request of any trip, we have that each trip, corresponding to an item in X, is covered by one TU corresponding to an item in Y and one TU corresponding to an item in Z. The associated triples define a solution of numerical 3-dimensional matching. □

4.4 ILP Formulations

The two ILP formulations that we use for our problem, one with variables associated with arcs of G and the other with variables associated with paths in G, are standard, being analogous to others that have been widely used both in the context of TU assignment and for other optimization problems in transportation, see, e.g., the survey in [24].

4.4.1 Arc formulation

Let us introduce an integer variable x_a, for each $k = 1, \ldots, p$ and $a = (i, j)^k \in A^k$, that indicates the number of arcs $a \in A^k$ selected in the solution, i.e., the number of TUs of type k that execute trip i before trip j in the associated sequence. The arc-variables ILP model is the following:

$$\min \sum_{k=1}^{p} \sum_{a \in \delta_+^k(0)} c^k x_a, \tag{4.1}$$

$$\sum_{a \in \delta_-^k(j)} x_a = \sum_{a \in \delta_+^k(j)} x_a, \qquad k = 1, \ldots, p,\ j = 1, \ldots, n, \tag{4.2}$$

$$\sum_{a \in \delta_+^k(0)} x_a \leq d^k, \qquad k = 1, \ldots, p, \tag{4.3}$$

$$\sum_{k=1}^{p} \sum_{a \in \delta_-^k(j)} s^k x_a \geq r_j, \qquad j = 1, \ldots, n, \tag{4.4}$$

$$\sum_{k=1}^{p} \sum_{a \in \delta_-^k(j)} x_a \leq u_j, \qquad j = 1, \ldots, n, \tag{4.5}$$

$$x_a \geq 0,\ \text{integer}, \qquad k = 1, \ldots, p,\ a \in A^k. \tag{4.6}$$

Flow conservation constraints (4.2) guarantee that the solution contains a number of paths in (V, A^k) from 0 to $n + 1$ equal to the number of arcs in A^k leaving node 0. Accordingly, constraints (4.3) ensure that the solution contains at most d^k such paths, i.e., no more than d^k TUs of type k are used. Moreover, as each of these paths has cost c^k, the objective function (4.1) calls for the minimization of the total cost of the paths. Finally, constraints (4.4) and (4.5) guarantee that each trip j is visited by at most u_j paths, having overall capacity at least r_j.

In the general context of multicommodity flow, it is well known that the ILP formulation based on path variables, illustrated in the next section, is to be preferred to the one above when approaches based on the solution of the LP relaxation are used, see, e.g., [19]. This will also be shown by the experiments performed for our case study.

4.4.2 Path formulation

Let $\mathcal{P}^k$ denote the collection of paths from 0 to $n + 1$ in (V, A^k), and introduce an integer variable x_P, for each $k = 1, \ldots, p$ and $P \in \mathcal{P}^k$, that indicates the number of times that path P is selected in the solution, i.e., the number of TUs of type k that execute the trips sequence corresponding to P. Moreover, for each $k = 1, \ldots, p$ and $j = 1, \ldots, n$, let $\mathcal{P}_j^k \subseteq \mathcal{P}^k$ denote the

subcollection of paths in $\mathcal{P}^k$ that visit trip j. The path-variables ILP model is the following:

$$\min \sum_{k=1}^{p} \sum_{P \in \mathcal{P}^k} c^k x_P, \tag{4.7}$$

$$\sum_{P \in \mathcal{P}^k} x_P \leq d^k, \qquad k = 1, \ldots, p, \tag{4.8}$$

$$\sum_{k=1}^{p} \sum_{P \in \mathcal{P}_j^k} s^k x_P \geq r_j, \qquad j = 1, \ldots, n, \tag{4.9}$$

$$\sum_{k=1}^{p} \sum_{P \in \mathcal{P}_j^k} x_P \leq u_j, \qquad j = 1, \ldots, n, \tag{4.10}$$

$$x_P \geq 0, \text{ integer }, \qquad k = 1, \ldots, p, \ P \in \mathcal{P}^k. \tag{4.11}$$

The interpretation and verification of correctness of the model is analogous (and in fact simpler) than the one of model (4.1)–(4.6). The fact that the LP relaxations of the two models presented are equivalent is a well known fact; see, e.g., [19].

Observation 2 *To each solution of the LP relaxation of (4.1)–(4.6) there corresponds a solution of the LP relaxation of (4.7)–(4.11) of the same value, and viceversa.*

Although model (4.7)–(4.11) has, in general, an exponential number of variables, as opposed to model (4.1)–(4.6), the LP relaxation of the former is faster to solve in practice by column generation techniques. Letting $J_P \subseteq \{1, \ldots, n\}$ be the set of trips visited by a path $P \in \mathcal{P}^k$, the dual of the LP relaxation of model (4.7)–(4.11) reads:

$$\begin{aligned} \max & -\sum_{k=1}^{p} d^k \alpha^k + \sum_{j=1}^{n} r_j \beta_j - \sum_{j=1}^{n} u_j \gamma_j, \\ & -\alpha^k + \sum_{j \in J_P} s^k \beta_j - \sum_{j \in J_P} \gamma_j \leq c^k, \qquad k = 1, \ldots, p, \ P \in \mathcal{P}^k, \\ & \alpha^k, \beta_j, \gamma_j \geq 0, \qquad k = 1, \ldots, p, \ j = 1, \ldots, n, \end{aligned} \tag{4.12}$$

and hence the column generation problem, which is the separation problem for constraints (4.12), given a dual solution $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ calls for $k \in \{1, \ldots, p\}$ and $P \in \mathcal{P}^k$ such that

$$\sum_{j \in J_P} (s^k \overline{\beta}_j - \overline{\gamma}_j) > c^k + \overline{\alpha}^k,$$

and can be solved as a maximum-profit path from 0 to $n+1$ in (V, A^k) with node profits $s^k \overline{\beta}_j - \overline{\gamma}_j$ for each $j \in V \setminus \{0, n+1\}$.

Not only the LP relaxation of (4.7)–(4.11) is much faster to solve in practice by column generation techniques than the LP relaxation of (4.1)–(4.6), but also heuristic methods based on this LP relaxation, that proceed by fixing variables x_P, i.e., entire sequences for TUs in the solution, tend to perform better in practice. However, as already mentioned, in order to get useful results for our case study we had to replace constraints (4.9) by stronger constraints, as illustrated in the next section.

4.5 Strengthening the Capacity Constraints for the Case Study

In all natural ILP models for the problem, including those of the previous section, letting w_j^k be an integer variable indicating the number of TUs of type k assigned to a trip j ($k = 1, \ldots, p,\ j = 1, \ldots, n$), the following constraints are imposed:

$$\sum_{k=1}^{p} s^k w_j^k \geq r_j, \quad j = 1, ..., n, \tag{4.13}$$

$$\sum_{k=1}^{p} w_j^k \leq u_j, \quad j = 1, ..., n. \tag{4.14}$$

(In particular, variables w_j^k would be defined by equations $w_j^k = \sum_{a \in \delta_-^k(j)} x_a$ in model (4.1)–(4.6), and by equations $w_j^k = \sum_{P \in \mathcal{P}_j^k} x_P$ in model (4.7)–(4.11).)

It is well known that the constraints (4.13) can be very weak for the LP relaxation. Moreover, since in our case study we have $u_j = 2$ for $j = 1, \ldots, n$, the dominant of the convex hull of the nonnegative integer vectors satisfying (4.13) and (4.14) is defined by $O(p)$ simple constraints, that we will use to replace (4.13) in our models. In order to simplify the notation, we will remove the index j and study the following polytope:

$$P := \text{conv}\left\{ w \in \mathbb{Z}_+^p : \sum_{k=1}^{p} s^k w^k \geq r, \sum_{k=1}^{p} w^k \leq 2 \right\}, \tag{4.15}$$

assuming $s^1 \geq s^2 \geq \ldots \geq s^p$. Its *dominant* $\overline{P}$ is defined as follows:

$$\overline{P} := \left\{ w \in \mathbb{R}^p : \text{ there exists } \overline{w} \in P \text{ such that } w \geq \overline{w} \right\}. \tag{4.16}$$

All the inequalities in "$\geq$" form with nonnegative coefficients that are valid for P are also valid for $\overline{P}$ and viceversa, so the description of $\overline{P}$ yields a set of stronger inequalities to replace the "weak" inequality $\sum_{k=1}^{p} s^k w^k \geq r$.

The following theorem provides a simple description of $\overline{P}$ by $O(p)$ linear inequalities.

Theorem 1 *If $2s^1 < r$, then $\overline{P} = \emptyset$. Otherwise, letting u be such that $s^u \geq r$ and $s^{u+1} < r$ (with $u := 0$ if $s^1 < r$ and $u := p$ if $s^p \geq r$), t be such that $2s^t \geq r$ and $2s^{t+1} < r$ (with $t := p$ if $2s^p \geq r$), and, for each $k = u+1, \ldots, t$, $f(k)$ be such that $s^k + s^{f(k)} \geq r$ and $s^k + s^{f(k)+1} < r$ (with $f(k) := p$ if $s^k + s^p \geq r$ and $f(t+1) := t$):*

$$\overline{P} = \left\{ w \in \mathbb{R}^p_+ : \sum_{\ell=1}^{k-1} 2w^\ell + \sum_{\ell=k}^{f(k)} w^\ell \geq 2, \quad k = u+1, \ldots, t+1 \right\}. \tag{4.17}$$

Proof. If $2s^1 < r$, we have $P = \emptyset$ and therefore $\overline{P} = \emptyset$. In the reminder of the proof we will assume $2s^1 \geq r$ and let $u, t, f(\cdot)$ be as defined in the statement. We have that the vertices of P are the vectors in $w \in \mathbb{Z}^p_+$ with either one component $w^k > 0$ for some $k \leq u$ and sum of the components not larger than 2, or with one component $w^k > 0$ for some $u < k \leq t$, and either $w^k = 2$ or $w^k = 1$ and $w^\ell = 1$ for some $\ell \leq f(k)$. In order to show that (4.17) holds, first note that the $t - u + 1$ inequalities listed are clearly satisfied by all vertices of P, and therefore valid for P and $\overline{P}$, and violated by all vectors in $w \in \mathbb{Z}^p_+$ that are not vertices of P nor dominate a vertex of P, namely, by all vectors that are not in $\overline{P}$. This implies:

$$\overline{P} = \text{conv} \left\{ w \in \mathbb{Z}^p_+ : \sum_{\ell=1}^{k-1} 2w^\ell + \sum_{\ell=k}^{f(k)} w^\ell \geq 2, \quad k = u+1, \ldots, t+1 \right\}. \tag{4.18}$$

Accordingly, what remains to be shown is that all vertices of the polyhedron on the left-hand-side of (4.17) are integer. Since the polyhedron is defined by inequalities of "$\geq$" type with nonnegative coefficients, this holds if and only if, for each $c \in \mathbb{R}^p_+$, the minimization of $\sum_{k=1}^p c^k w^k$ over the polyhedron has an integer optimal solution (see, e.g., [45]). Accordingly, consider a generic $c \in \mathbb{R}^p_+$ and the LP:

$$\min \left\{ \sum_{k=1}^{p} c^k w^k : \sum_{\ell=1}^{k-1} 2w^\ell + \sum_{\ell=k}^{f(k)} w^\ell \geq 2 \ (k = u+1, \ldots, t+1), \quad w^k \geq 0 \ (k = 1, \ldots, p) \right\}. \tag{4.19}$$

In order to show that (4.19) has an optimal integer solution, we first perform a simple pre-processing phase.

First of all, note that we can assume that $c^1 > c^2 > \ldots > c^p$. Indeed, if we have $c^k \leq c^{k+1}$ for some $k \in \{1, \ldots, p-1\}$, given any solution $\overline{w}$ with $\overline{w}^{k+1} > 0$, the solution $\tilde{w}$ defined by $\tilde{w}^k := \overline{w}^k + \overline{w}^{k+1}$, $\tilde{w}^{k+1} := 0$, $\tilde{w}^\ell := \overline{w}^\ell$ for $\ell \neq k, k+1$ is also feasible, as the coefficient of w^k in all constraints is not smaller than the coefficient of w^{k+1}, and not worse than that of $\overline{w}$ in terms of objective value. This implies that we can remove variable w^{k+1}, iterating the process until $c^1 > c^2 > \ldots > c^p$ (after having scaled the indices).

Moreover, we can assume that, for each $\ell = t+1, \ldots, p$, there exists $k \in \{u+1, \ldots, t\}$ such that $f(k) = \ell$. Indeed, if there is an ℓ without this property, given any solution $\overline{w}$ with $\overline{w}^\ell > 0$, the solution $\tilde{w}$ defined by $\tilde{w}^{\ell+1} := \overline{w}^\ell + \overline{w}^{\ell+1}$, $\tilde{w}^\ell := 0$, $\tilde{w}^k := \overline{w}^k$ for $k \neq \ell, \ell+1$ is also feasible, as w^ℓ and $w^{\ell+1}$ have the same coefficients in all constraints, and not worse than $\overline{w}$. Again, we can remove w^ℓ and iterate the process. At the end of this process, we have $f(k) = p - k + u + 1$ for $k = u+1, \ldots, t$, with either $p = u + 2t$ and $f(t) = t+1$ or $p = u + 2t - 1$ and $f(t) = t$. As the proof is identical in the two cases, we consider only the first one in the following.

Taking into account the two assumptions above, we next show that there is an optimal solution $\overline{w}$ such that $\overline{w}^k = \overline{w}^{f(k)}$ for $k = u+1, \ldots, t$. This is proved by induction on k. To prove the basis of the induction, namely $k = u+1$ and $f(k) = p$, consider a solution $\overline{w}$ in which $\overline{w}^{u+1} \neq \overline{w}^p$. If $\overline{w}^{u+1} < \overline{w}^p$, we have that the solution $\tilde{w}$ defined by $\tilde{w}^p := \overline{w}^{u+1}$ and $\tilde{w}^\ell := \overline{w}^\ell$ for $\ell \neq p$ has clearly a value not worse than that of $\overline{w}$ and is feasible, given that w^p has a positive coefficient only in the constraint

$$\sum_{\ell=1}^{u} 2w^\ell + \sum_{\ell=u+1}^{p} w^\ell \geq 2, \tag{4.20}$$

and the constraint

$$\sum_{\ell=1}^{u+1} 2w^\ell + \sum_{\ell=u+2}^{p-1} w^\ell \geq 2 \tag{4.21}$$

is satisfied by $\overline{w}$ and hence by $\tilde{w}$. Otherwise, i.e., if $\overline{w}^{u+1} > \overline{w}^p$, we have that the solution $\tilde{w}$ defined by $\tilde{w}^{u+1} := \overline{w}^p$, $\tilde{w}^{u+2} := \overline{w}^{u+2} + (\overline{w}^{u+1} - \overline{w}^p)$ and $\tilde{w}^\ell := \overline{w}^\ell$ for $\ell \neq u+1, u+2$ has clearly a value not worse than that of $\overline{w}$ and is feasible, given that w^{u+1} and w^{u+2} have different coefficients only in (4.21), and (4.20) is satisfied by $\overline{w}$ and hence by $\tilde{w}$. Now, assuming by the induction hypothesis $\overline{w}^\ell = \overline{w}^{f(\ell)}$ for $\ell = u+1, \ldots, k-1$, we show that also $\overline{w}^k = \overline{w}^{f(k)}$, using arguments totally analogous to those used for the basis of the induction. As before, assume $\overline{w}^k \neq \overline{w}^{f(k)}$. If $\overline{w}^k < \overline{w}^{f(k)}$, we have that the solution $\tilde{w}$ defined by $\tilde{w}^{f(k)} := \overline{w}^k$ and $\tilde{w}^\ell := \overline{w}^\ell$ for $\ell \neq f(k)$ has clearly a value not worse than that of $\overline{w}$ and is feasible, as we show next. Consider the constraints

$$\sum_{\ell=1}^{k} 2w^\ell + \sum_{\ell=k+1}^{f(k)-1} w^\ell \geq 2 \tag{4.22}$$

and

$$\sum_{\ell=1}^{k-1} 2w^\ell + \sum_{\ell=k}^{f(k)} w^\ell \geq 2 \tag{4.23}$$

Even if $\tilde{w}^{f(k)} < \overline{w}^{f(k)}$, (4.23) is satisfied by $\tilde{w}$ since (4.22) is. Moreover, since (4.23) is satisfied by $\overline{w}$, also all other constraints in which $w^{f(k)}$ has a positive coefficient are satisfied by $\overline{w}$, by the induction hypothesis. Otherwise, i.e., if $\overline{w}^k > \overline{w}^{f(k)}$, we have to distinguish the two cases $k < t$ and $k = t$. If $k < t$ (and hence $f(k) > k+1$), we have that the solution $\tilde{w}$ defined by $\tilde{w}^k := \overline{w}^{f(k)}$, $\tilde{w}^{k+1} := \overline{w}^{k+1} + (\overline{w}^k - \overline{w}^{f(k)})$ and $\tilde{w}^\ell := \overline{w}^\ell$ for $\ell \neq k, k+1$ has clearly a value not worse than that of $\overline{w}$ and is feasible, given that w^k and w^{k+1} have different coefficients only in (4.22), and (4.23) is satisfied by $\overline{w}$ and hence by $\tilde{w}$. Finally, if $k = t$ (and hence $f(k) = t+1$), we have that the solution $\tilde{w}$ defined by $\tilde{w}^t := \tilde{w}^{t+1} := (\overline{w}^t + \overline{w}^{t+1})/2$, $\tilde{w}^\ell := \overline{w}^\ell$ for $\ell \neq t, t+1$ has clearly a value not worse than that of $\overline{w}$ and is feasible, given that w^t and w^{t+1} have different coefficients only in (4.22), and (4.23) is satisfied by $\overline{w}$ and hence by $\tilde{w}$. This completes the proof of the fact that there is an optimal solution $\overline{w}$ such that $\overline{w}^k = \overline{w}^{f(k)}$ for $k = u+1, \ldots, t$. Note also that such a solution satisfies all the constraints with equality, in particular $\sum_{\ell=1}^p \overline{w}^\ell = 2$, since, in case $\sum_{\ell=1}^p \overline{w}^\ell > 2$, the solution $\tilde{w}$ obtained by multiplying all the components of $\overline{w}$ by $2/(\sum_{\ell=1}^p \overline{w}^\ell)$ would be feasible and have a better objective value.

Now consider an optimal solution $\overline{w}$ with the form above. Recalling the structure of the vertices of P discussed at the beginning of the proof, it is evident that $\overline{w}$ is a convex combination of these vertices, and therefore another optimal solution is given by the best solution between the one defined by $\tilde{w}^k := 1$ for $k = \arg\min\{c^\ell : \ell \in \{1, \ldots, u\}\}$, $\tilde{w}^\ell := 0$ for $\ell \neq k$, and the one defined by $\tilde{w}^k := \tilde{w}^{f(k)} := 1$ for $k = \arg\min\{c^\ell + c^{f(\ell)} : \ell = u+1, \ldots, t\}$, $\tilde{w}^\ell := 0$ for $\ell \neq k, f(k)$. This proves that (4.19) has an optimal integer solution.

Since we have shown that, for each $c \in \mathbb{R}^p_+$, the minimization of $\sum_{k=1}^p c^k w^k$ over (4.17) has an integer optimal solution, the proof of the theorem is complete. □

Example

In order to illustrate the above result, let us consider the numerical example, taken from our case study, in which $p = 8$, $r_=$1302 and $s = (1150, 1044, 786, 702, 543, 516, 495, 360)$. In this case we have $u = 0$, $t = 4$, $f(1) = f(2) = 8$, $f(3) = 6$, $f(4) = 4$, leading to the following constraints:

$$\begin{aligned}
w_j^1 + w_j^2 + w_j^3 + w_j^4 + w_j^5 + w_j^6 + w_j^7 + w_j^8 &\geq 2 \\
2w_j^1 + w_j^2 + w_j^3 + w_j^4 + w_j^5 + w_j^6 + w_j^7 + w_j^8 &\geq 2 \\
2w_j^1 + 2w_j^2 + w_j^3 + w_j^4 + w_j^5 + w_j^6 &\geq 2 \\
2w_j^1 + 2w_j^2 + 2w_j^3 + w_j^4 &\geq 2 \\
2w_j^1 + 2w_j^2 + 2w_j^3 + 2w_j^4 &\geq 2
\end{aligned}$$

out of which the second is dominated by the first and the last is dominated by the last but

one.

□

According to the above discussion, the two ILP models of the previous section can be strengthened by letting $u_j, t_j, f_j(\cdot)$ be defined from r_j as $u, t, f(\cdot)$ were defined from r in the statement of Theorem 1, and replace (4.13) by the following constraints:

$$\sum_{\ell=1}^{k-1} 2w_j^\ell + \sum_{\ell=k}^{f_j(k)} w_j^\ell \geq 2, \quad j = 1, ..., n, \ k = u_j + 1, \ldots, t_j + 1, \tag{4.24}$$

noting that some of the constraints in the list may be dominated by others and therefore not imposed in practice. As the results on our case study will show, the use of (4.24) causes a sharp improvement in the results that we achieve.

For instance, in model (4.7)–(4.11) constraints (4.9) can be replaced by:

$$\sum_{\ell=1}^{k-1} \sum_{P \in \mathcal{P}_j^\ell} 2x_P + \sum_{\ell=k}^{f_j(k)} \sum_{P \in \mathcal{P}_j^\ell} x_P \geq 2, \quad j = 1, ..., n, \ k = u_j + 1, \ldots, t_j + 1, \tag{4.25}$$

observing that this replacement does not affect the structure of the column generation problem discussed in the previous section.

4.6 An LP-Based Heuristic Method

We next illustrate the heuristic method, based on the LP relaxation of model (4.7)–(4.11) with (4.9) replaced by (4.25), that eventually allowed us to improve the practitioners' solution for our case study. Besides the (customary) column-generation based procedure to solve the LP relaxation, the heuristic method has three main components: (1) a diving rule to fix the value of some of the variables following the current LP optimal solution, reoptimizing the LP after the addition of these fixing constraints, (2) a simple constructive heuristic procedure based on the current dual LP solution that is applied at each iteration of the column-generation based procedure, and (3) a refinement procedure that is applied to improve each solution produced by the constructive heuristic procedure in (2). The general structure of the method is outlined in Figure 4.1.

4.6.1 Fixing phase

Each time we have obtained the optimal solution $\overline{x}$ of the current LP with the fixing constraints, i.e., there are no dual constraints violated, we change the bounds of the variables

TUAP_heur
begin

initialize the current LP as a reduced version of LP (4.7)–(4.11), with (4.9) replaced by (4.24), with only a subset of the variables;

repeat

solve the current LP by a general-purpose LP solver, letting $\overline{x}$ be the optimal primal solution, $(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$ the optimal dual solution, and $\overline{z}$ the corresponding value;

apply the constructive heuristic procedure based on $(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$;

refine the solution found by the constructive heuristic procedure, possibly updating the incumbent solution;

if there are dual constraints violated by $(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$ **then**

add some of the corresponding primal variables to the current LP;

else

fix the value of some of the primal variables by changing the associated bounds;

until the current LP is infeasible **or** $\overline{z} \geq$ value of the incumbent solution;

end.

Figure 4.1: General structure of the LP-based heuristic method.

as follows. We consider all variables x_P such that $\overline{x}_P$ is integer, setting the associated lower bound to $\overline{x}_P$, i.e., imposing at least $\overline{x}_P$ paths P in the solution. Moreover, we consider the variable x_P whose value $\overline{x}_P$ is the largest among the fractional ones, and set the associated lower bound to $\lceil \overline{x}_P \rceil$. Note that, in this way, we may, e.g., fix the lower bound of a variable to 1, and then find values of these variables that are strictly larger than 1 in subsequent LP solutions.

We observed that, after the fixing phase, it may happen that the current LP becomes infeasible, and then become feasible again after some iterations of the column generation procedure. In order to avoid dealing with LPs that are infeasible due to the fact that we are only considering a subset of the variables, we introduce explicit slack variables for constraints (4.25), adding them to the objective function with a high penalty. This simplifies also the initialization of the current LP at the beginning of the procedure. Note that the "the current LP is infeasible" condition to be checked at the end is then equivalent to having some of the slack variables strictly positive in the solution.

4.6.2 Constructive heuristic procedure

The constructive heuristic procedure that we apply at each iteration considers the TU types one at the time, according to increasing values of c^k/s^k. For each TU type k, we define up to d^k paths to be added to the solution. In addition to the paths that possibly were already fixed in the solution by the fixing phase, the remaining paths are found by computing maximum-profit paths in (V, A^k), analogously to the column generation procedure, with node profits defined in a more complex way. For the trips that are not covered by the previously-defined paths, the profit takes into account (a) the associated dual variables, and (b) how well the capacity of the current TU type matches the residual request of the trip, i.e., by assigning a TU of this type to the trip, will it be possible to satisfy *at equality* the trip request? Moreover, we assign in any case a small positive profit to the trips already covered.

One of the main ideas is to try to follow the dual profits for the trips that still have to be covered, but also to try to satisfy at equality the request of these trips and to over-cover trips that have already been covered, in the hope of being able to achieve larger improvements with the subsequent refinement procedure. To this aim, we do not consider explicitly constraints (4.10) on the maximum number of TUs that can be assigned to a trip in the construction.

The constructive procedure terminates either when we have used all the available TUs, or when the paths constructed so far cover all the trips. Note that in the latter case we have saved some TUs of the last type (largest c^k/s^k ratio), and, in case all of them were saved, some TUs of the last but one type, and so on. On the other hand, in the former case, some of the trips are not covered. Moreover, in both cases we have that constraints (4.10) may be violated. The following refinement procedure tries to take care of these infeasibilities.

Concerning the fact that we are trying to satisfy at equality the trip requests, note that the input instance can always be preprocessed so that this is possible, by redefining the request r_j of each trip $j \in \{1, \dots, n\}$ as:

$$r_j := \min \left\{ \sum_{k=1}^{p} s^k w_j^k : \sum_{k=1}^{p} s^k w_j^k \geq r_j, \quad \sum_{k=1}^{p} w_j^k \leq u_j, \quad w_j^k \in \{0, \dots, d^k\} \ (k = 1, \dots, p) \right\}.$$

The associated optimization problem, which is a cardinality constrained bounded subset sum problem [41], can easily be solved by enumeration given the small values of p in practical cases.

4.6.3 Refinement

This is a key step in our framework. We consider the solution produced by the constructive heuristic procedure by taking into account only the information about the number of times $\overline{w}_j^k$ that each trip $j \in \{1, \dots, n\}$ is assigned to a TU of type $k \in \{1, \dots, p\}$, *without* considering

the specific sequences (paths) defined. In other words, we take care only of the information that would be given by variables w_j^k as defined in Section 4.5.

First of all, we guarantee that constraints (4.10) are satisfied, and that no trip is over-covered, as follows. For each trip j which is covered and would remain such even by decreasing some $\overline{w}_j^k$ value, we redefine the $\overline{w}_j^k$ values as the optimal solution of the problem:

$$\min \left\{ \sum_{k=1}^{p} s^k w_j^k : \sum_{k=1}^{p} s^k w_j^k \geq r_j, \quad \sum_{k=1}^{p} w_j^k \leq u_j, \quad w_j^k \in \{0, \ldots, \overline{w}_j^k\} \ (k = 1, \ldots, p) \right\}.$$

(Again, this problem is solved quickly by enumeration.)

On the other hand, for each trip j which is not covered but such that $\sum_{k=1}^{p} \overline{w}_j^k > u_j$, we redefine the $\overline{w}_j^k$ values as the optimal solution of the problem:

$$\max \left\{ \sum_{k=1}^{p} s^k w_j^k : \sum_{k=1}^{p} w_j^k \leq u_j, \quad w_j^k \in \{0, \ldots, \overline{w}_j^k\} \ (k = 1, \ldots, p) \right\}.$$

(This problem is trivially solvable in $O(p)$ time.)

After the above preprocessing phase, we define the new solution by noting that Observation 1 may be rephrased as follows:

Observation 3 *The variant of TUAP in which the number of times $\overline{w}_j^k$ that each trip $j \in \{1, \ldots, n\}$ is assigned to a TU of type $k \in \{1, \ldots, p\}$ is specified on input is solvable efficiently.*

If all trips are covered after the preprocessing phase above, the refinement procedure returns the solution found by applying Observation 3. Otherwise, we iterate the procedure as follows. First, we consider each trip j that is not covered and, for each TU type k, we check if by increasing $\overline{w}_j^k$ by 1 the number of TUs of type k that are required according to Observation 3 is smaller than d^k: if this is the case we increase $\overline{w}_j^k$ by 1, otherwise we leave it unchanged. Then, we apply the preprocessing phase above to the increased $\overline{w}$, followed by Observation 3, and so on, until either a feasible solution is found, which is returned on output, or the set of uncovered trips along with their number of missing seats do not change in two consecutive iterations, in which case no solution is returned on output.

4.7 An Extension of the Problem

4.7.1 Problem Description

In the problem described in Section 4.2, given that there is an overnight break, every trip i can have any other trip j as successor in a sequence if i and j are performed in two different days. Basically, we allow to move the train units from any station to any other station, during

the night without any constraint. This is not always the case. In particular, the train units need a conductor to be moved and consequently not the whole overnight break can be used to move train units from a station to a different one.

In this Section, we consider only the allowed connection between stations, during the overnight break. For this formulation, we use a slightly different graph from the one described in Section 4.2. Let $\overline{G} = (V \setminus \{0, n+1\}, \overline{A})$ be the graph for representing this problem, where arc set $\overline{A}$ is partitioned into p subsets $\overline{A}^1, \ldots, \overline{A}^p$, where $\overline{A}^k$ is associated with TUs of type k for $k = 1, \ldots, p$. Given two distinct trips $i, j \in V \setminus \{0, n+1\}$, arc $(i,j)^k \in \overline{A}^k$ exists if and only if a TU of type k can be assigned to i and then to j (not necessarily within the same day).

In the following we call *roster* for a train unit type k an assignment of trips to train units of type $k \in \{1, \ldots, p\}$.

There is a one-to-one correspondence between trips assigned to a TU of type k and a circuit in $\overline{G}$ formed by arcs in $\overline{A}^k$. The problem calls for the determination, for each TU type $k \in \{1, \ldots, p\}$, of circuits formed by arcs in $\overline{A}^k$. Each trip $j \in \{1, \ldots, n\}$ must be visited by at most u_j circuits whose overall capacity, given by s^k for a circuit of a TU of type k, is at least r_j. We can have on one hand up to d^k circuits for a train unit of type $k \in \{1, \ldots, p\}$, or on the other hand (equivalently) one circuit of length at most d^k days, or all the possibilities in between.

The objective is to minimize the overall cost of the circuits, where we impose $c_{ij}^k := +\infty$ if $(i,j)^k \notin \overline{A}$ and we let c_{ij}^k be given by the time (in minutes) that elapses between the departure of trip i and the departure of trip j if $(i,j)^k \in \overline{A}$. Noting that the length of each circuit of $\overline{G}$ is given by an integer number of days and the number of days in a roster coincides with the number of train units that are needed to execute such a roster, minimizing the overall cost of the circuits is equivalent to minimizing the number of train units globally used.

4.7.2 Arc formulation

Let us introduce an integer variable x_{ij}^k, for each $k = 1, \ldots, p$, $i = 1, \ldots, n$ and $j = 1, \ldots, n$, that indicates the number of times that arc $(i,j)^k$ is selected in the solution, i.e., the number of TUs of type k that execute trip i before trip j in the associated sequence.

The arc-variables ILP model is the following:

$$\min \sum_{k=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^k x_{ij}^k, \tag{4.26}$$

$$\sum_{i=1}^{n} x_{ij}^k = \sum_{i=1}^{n} x_{ji}^k, \qquad k = 1, \ldots, p,\ j = 1, \ldots, n, \tag{4.27}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij}^k \leq 1440\ d^k, \qquad k = 1, \ldots, p, \tag{4.28}$$

$$\sum_{k=1}^{p}\sum_{i=1}^{n} s^k x_{ij}^k \geq r_j, \qquad j = 1, \ldots, n, \tag{4.29}$$

$$\sum_{k=1}^{p}\sum_{i=1}^{n} x_{ij}^k \leq u_j, \qquad j = 1, \ldots, n, \tag{4.30}$$

$$x_{ij}^k \geq 0,\ \text{integer}\ , \qquad k = 1, \ldots, p, i = 1, \ldots, n, j = 1, \ldots, n. \tag{4.31}$$

The objective is to minimize the total duration of the rosters, i.e. the number of train units globally used.

Constraints (4.27) express the flow conservation. Constraints (4.28) forbid to use more than the available train units for each type (we use the constant 1440 which is the number of minutes in a day). Constraints (4.29) impose to cover each trip with the corresponding seat request. Constraints (4.30) impose a bound on the number of train units that can be used to cover each trip.

Given the relatively large size of the ILP in our case study, it is natural to consider the Lagrangian relaxation of the above formulation, obtained by relaxing constraints (4.28) and (4.29) in a Lagrangian way.

4.7.3 Lagrangian Relaxation

We consider a first relaxation of the model for obtaining some properties on the corresponding Lagrangian Relaxation, which will be described in this section. In particular, we replace constraints (4.30) with the following constraints, which impose that each train unit of type k can cover each trip j at most u_j^k times:

$$\sum_{i=1}^{n} x_{ij}^k \leq u_j^k, k = 1, \ldots, p, j = 1, \ldots, n, \tag{4.32}$$

In particular, we consider the maximum number of times that is useful for a train unit of type k to cover a trip j.

We relax in a Lagrangian way constraints (4.28) and (4.29) in the obtained problem, by using nonnegative Lagrangian multipliers $\lambda_j, j = 1, \ldots, n$ and $\sigma_k, k = 1, \ldots, p$, respectively.

The model that we obtain is the following:

$$\sum_{j=1}^{n} \lambda_j r_j - \sum_{k=1}^{p} \sigma_k\ 1440\ d^k + \min \sum_{k=1}^{p}\sum_{i=1}^{n}\sum_{j=1}^{n} (c_{ij}^k - \lambda_j s^k + \sigma_k c_{ij}^k) x_{ij}^k, \tag{4.33}$$

$$\sum_{i=1}^{n} x_{ij}^k = \sum_{i=1}^{n} x_{ji}^k, \quad k = 1, ..., p, j = 1, ..., n, \tag{4.34}$$

$$\sum_{i=1}^{n} x_{ij}^k \leq u_j^k, \quad k = 1, ..., p, j = 1, ..., n, \tag{4.35}$$

$$x_{ij}^k \geq 0, \quad \text{integer}\ , \quad k = 1, ..., p, i = 1, ..., n, j = 1, ..., n. \tag{4.36}$$

For each $\lambda, \sigma \geq 0$, the optimal solution value of (4.33) - (4.36) is a valid lower bound on the optimal solution value of (4.26) - (4.31) and therefore on the optimal solution value of the original problem. Lagrangian relaxed problem (4.33) - (4.36) decomposes onto p independent subproblems, one for each train unit. For each train unit type k, let $\tilde{c}_{ij}^k := (c_{ij}^k - \lambda_j s^k + \sigma_k c_{ij}^k)$ ($\tilde{c}_{ij}^k := +\infty$ if $(i,j)^k \notin \overline{A}$) be the Lagrangian cost for each arc $(i,j)^k \in \overline{A}$. We impose a null Lagrangian cost ($\tilde{c}_{ii}^k := 0$) for loops $(i,i)^k$ for each vertex $i \in V \setminus \{0, n+1\}$ (that had originally infinity cost). The subproblem associated with train unit type k reads:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \tilde{c}_{ij}^k x_{ij}^k, \tag{4.37}$$

$$\sum_{i=1}^{n} x_{ij}^k \quad = \quad u_j^k, \quad j = 1, ..., n, \tag{4.38}$$

$$\sum_{i=1}^{n} x_{ji}^k \quad = \quad u_j^k, \quad j = 1, ..., n, \tag{4.39}$$

$$x_{ij}^k \geq 0, \quad \text{integer} \quad , \quad i = 1, ..., n, j = 1, ..., n. \tag{4.40}$$

In particular, there is an immediate one-to-one correspondence between solutions $\overline{x}^k$ of (4.34)-(4.35) and solutions $\tilde{x}^k$ of (4.38)-(4.39), given by $\tilde{x}_{jj}^k := u_j^k - \sum_{i \neq j} \overline{x}_{ij}^k$ for $j \in V$. It is well known that all vertices of the feasible region (4.38)-(4.40) are integer for u_j integer and the associated constraint matrix is totally unimodular.

Let us now consider the particular case, arising in our case study, where u_j^k does not depend on j. We then replace u_j^k with u^k for $k = 1, \ldots, p$ and $j = 1, \ldots, n$. Accordingly, problem (4.37) - (4.40) is equivalent to the following assignment problem (AP):

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \tilde{c}_{ij}^k y_{ij}, \tag{4.41}$$

$$\sum_{i=1}^{n} y_{ij} \quad = \quad 1, \quad j = 1, ..., n, \tag{4.42}$$

$$\sum_{l=1}^{n} y_{jl} \quad = \quad 1, \quad j = 1, ..., n, \tag{4.43}$$

$$y_{ij} \geq 0, \quad \text{integer} \quad , \quad i = 1, ..., n, j = 1, ..., n. \tag{4.44}$$

The correspondence between solutions $\tilde{x}$ of (4.37)-(4.40) and $\overline{y}$ of (4.41)-(4.44) is given by $\tilde{x} = u^k \overline{y}$.

As well known, the assignment problem can be solved in $O(n^3)$ time, using, e.g., the Hungarian Algorithm.

In order to find good Lagrangian multipliers we apply a standard iterative subgradient procedure. At each iteration of the subgradient procedure we solve assignment subproblems as described above, finding the optimal solution $\overline{x}$ of (4.33) - (4.36) corresponding to the current Lagrangian multipliers. We check if $\overline{x}$ violates some of the relaxed constraints and update the multipliers by using the well-known formula:

$\lambda_j = \max\left\{\lambda_j + \delta \frac{UB-LB}{||\mu||^2}\mu_j, 0\right\}$, $\sigma_k = \max\left\{\sigma_k + \delta \frac{UB-LB}{||\tau||^2}\tau_k, 0\right\}$,

where δ is a step parameter, LB is the best lower bound value found so far, UB is the best upper bound, i.e., value of a feasible solution of (4.26)-(4.31), determined by a heuristic procedure to be illustrated later and μ, τ are the subgradient vectors associated with $\overline{x}$, namely $\mu_j := r_j - \sum_{k=1}^{p}\sum_{i=1}^{n} s^k \overline{x}_{ij}^k$, for $j = 1, \ldots, n$, $\tau_k := \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k \overline{x}_{ij}^k - 1440 d^k$ for $k = 1, \ldots, p$. The procedure is applied for 1000 iterations. Besides yielding a valid lower bound on the optimal solution value, the best Lagrangian multipliers λ^*, σ^* found throughout the iterations and the corresponding reduced costs $\overline{c}_{ij}^k := c_{ij}^k - \lambda_j^* s^k + \sigma_k^* c_{ij}^k - \overline{w}_i^k - \overline{v}_j^k$, where $\overline{w}^k$ and $\overline{v}^k$ are the optimal AP dual variables associated with constraints (4.42) and (4.43) for train unit type k, are used to derive a constructive Lagrangian heuristic algorithm.

4.7.4 A Heuristic based on Lagrangian Relaxation

At each iteration of the subgradient procedure, we compute the reduced cost of each arc corresponding to the current set of Lagrangian multipliers, and we use this information to decide how to sequence the trips. The rosters are built separately for each type of train unit. We stop the procedure when we have covered all the trips with the corresponding requests or no more train units are available.

The algorithm consists of the following steps:

1. **Ordering the train unit types**: we order the different types of train units in decreasing order of capacity (i.e., number of available seats).

2. **Initial trip**: we choose as initial trip one which leaves its first station early in the morning, as we want to have busy daily assignments. Moreover, we prefer a trip which

easily allows to create a circuit, i.e. a trip which can be easily reached by other trips, because we need to go back to the first trip in order to form a roster.

3. **Next trip**: we assign a score to each trip that has not been completely covered. The score takes into account different elements:

   a. the reduced cost of the arc connecting the node corresponding to the last chosen trip and the current trip;

   b. the original cost of the arc connecting the node corresponding to the last chosen trip and the current trip: the reduced cost has a higher weight in the score, but if two arcs have the same reduced cost we choose the one with smaller original cost;

   c. how well the capacity of the current train unit matches with the request of the trip (i.e., we don't want to use a train unit with a big capacity to cover a trip with a small request);

   d. a score for forcing maintenance, when necessary.

4. **Updating**: we decrease the number of seats required by the trip and the reduced cost of each arc entering the vertex corresponding to the current trip.

5. **Ending the roster**: we compute a score which gives an evaluation of how "good" is to end the roster with the current trip. This is done by considering only the residual train units and trips, and solving the Lagrangian relaxation of this reduced problem. The solution of the reduced relaxed problem gives a lower bound on the global number of train units that are needed to cover all the residual trips. If the remaining train units are enough, then the current trip can be used as the last trip and we store the value of the Lagrangian solution in order to compare it with other possible ways to end the roster. In any case, we continue the construction of the roster until we reach the maximum number of available train units of the current type. The trip which produces the smallest value of the Lagrangian solution on the reduced problem is then chosen as last trip.

As shown, the resulting Lagrangian relaxed problem is easy to solve, recalling also Observation 1, as it amounts to finding optimal circuits in graphs $(V \setminus \{0, n+1\}, \overline{A}^k)$ for $k = 1, \ldots, p$. However, despite completely analogous approaches are the best ones in practice in many similar cases, our implementation of a customary heuristic method based on this Lagrangian relaxation performed extremely poorly in practice for our case study, in terms of both lower bound produced and solution found (in fact, it was never able to find a solution respecting all constraints, always requiring more TUs than those available). Given that they were so poor, we will not even present these results in the experimental section.

4.7.5 Circuit formulation

Let $\mathcal{C}^k$ denote the collection of circuits in $(V, \overline{A}^k)$, and let us introduce an integer variable x_C, for each $k = 1, \ldots, p$ and $C \in \mathcal{C}^k$, that indicates the number of times that circuit C is selected in the solution, i.e., the number of TUs of type k that execute the trips sequence corresponding to C. Let c_C be the cost of the circuit $C \in \mathcal{C}^k$, for $k = 1 \ldots, p$, given by the sum of the costs of the arcs belonging to the circuit. Moreover, for each $k = 1, \ldots, p$ and $j = 1, \ldots, n$, let $\mathcal{C}^k_j \subseteq \mathcal{C}^k$ denote the subcollection of circuits in $\mathcal{C}^k$ that visit trip j. The circuit-variables ILP model is the following:

$$\min \sum_{k=1}^{p} \sum_{C \in \mathcal{C}^k} c_C x_C, \tag{4.45}$$

$$\sum_{C \in \mathcal{C}^k} c_C x_C \leq 1440 d^k, \qquad k = 1, \ldots, p, \tag{4.46}$$

$$\sum_{k=1}^{p} \sum_{C \in \mathcal{C}^k_j} s^k x_C \geq r_j, \qquad j = 1, \ldots, n, \tag{4.47}$$

$$\sum_{k=1}^{p} \sum_{C \in \mathcal{C}^k_j} x_C \leq u_j, \qquad j = 1, \ldots, n, \tag{4.48}$$

$$x_C \geq 0, \text{ integer }, \qquad k = 1, \ldots, p, \ C \in \mathcal{C}^k. \tag{4.49}$$

The interpretation and verification of correctness of the model is analogous to the one of model (4.26)–(4.31). The fact that the LP relaxations of the two models presented are equivalent is a well known fact; see, e.g., [19].

Observation 4 *To each solution of the LP relaxation of (4.26)–(4.31) there corresponds a solution of the LP relaxation of (4.45)–(4.49) of the same value, and viceversa.*

Although model (4.45)–(4.49) has, in general, an exponential number of variables, as opposed to model (4.26)–(4.31), the LP relaxation of the former is faster to solve in practice by column generation techniques. Letting $J_C \subseteq \{1, \ldots, n\}$ be the set of trips visited by a circuit $C \in \mathcal{C}^k$, the dual of the LP relaxation of model (4.45)–(4.49) reads:

$$\begin{aligned} \max & -\sum_{k=1}^{p} 1440 d^k \alpha^k + \sum_{j=1}^{n} r_j \beta_j - \sum_{j=1}^{n} u_j \gamma_j, \\ & -c_C k \alpha^k + \sum_{j \in J_C} s^k \beta_j - \sum_{j \in J_C} \gamma_j \leq c_C, \qquad k = 1, \ldots, p, \ C \in \mathcal{C}^k, \\ & \alpha^k, \beta_j, \gamma_j \geq 0, \qquad k = 1, \ldots, p, \ j = 1, \ldots, n, \end{aligned} \tag{4.50}$$

and hence the column generation problem, which is the separation problem for constraints (4.50), given a dual solution $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ calls for $k \in \{1, \ldots, p\}$ and $C \in \mathcal{C}^k$ such that

$$\sum_{j \in J_C} (s^k \overline{\beta}_j - \overline{\gamma}_j) > c_C(1 + \overline{\alpha}^k),$$

and can be solved as an assignment problem in $(V, \overline{A}^k)$ with arc costs $c^k_{ij}(1 + \overline{\alpha}^k) + \overline{\gamma}_j/2 + \overline{\gamma}_i/2 - s^k(\overline{\beta}_i + \overline{\beta}_j)/2$ for each arc $(i, j)^k \in \overline{A}^k$.

The LP relaxation of (4.45)–(4.49) is much faster to solve in practice by column generation techniques than the LP relaxation of (4.26)–(4.31).

We have implemented an LP-based heuristic algorithm, which follows the structure of the fixing procedure described in Section 4.6. However this LP-based heuristic algorithm doesn't obtain solutions which satisfy all the required constraints and for this reason we don't present the corresponding results in the experimental section.

4.8 An Alternative Lower Bound

We next illustrate a relaxation of TUAP that can be used to derive lower bounds, and is generally faster to solve than the LP relaxations of the ILP models illustrated in the previous sections. The main idea is to find a set of trips that are pairwise "incompatible", i.e., they cannot be assigned to the same TU (of whatever type), and then optimally solve the subinstance restricted to these trips: clearly the resulting optimal value is a lower bound on the optimal TUAP value.

The set $I \subseteq \{1, \ldots, n\}$ of incompatible trips that we compute is the maximum-weight stable set in the auxiliary (undirected) graph $(\{1, \ldots, n\}, E)$ in which the weight of each node j is equal to r_j, and edges are defined as follows. We first consider the (directed) graph $(\{1, \ldots, n\}, B)$ for which $(i, j) \in B$ whenever there exists $k \in \{1, \ldots, p\}$ such that $(i, j)^k \in A^k$. Then, we compute the transitive closure of this graph (in fact, in our case study, the graph is already transitive), and let E be the set of edges obtained by neglecting the orientation of the arcs in this transitive closure. In this way, the auxiliary graph is a *comparability graph* for which a maximum-weight stable set can be computed efficiently by flow techniques (see, e.g., [29]).

Given the set I, and recalling that TUAP, even restricted to the case in which each TU can perform only one trip, is NP-hard by Proposition 2, we determine the optimal assignment of TUs to the trips in I by solving the following ILP, in which variables w^k_j represent, as

before, the number of TUs of type k assigned to trip j:

$$\min \sum_{k=1}^{p} \sum_{j \in I} c^k w_j^k, \tag{4.51}$$

$$\sum_{j \in I} w_j^k \leq d^k, \qquad k = 1, \ldots, p, \tag{4.52}$$

$$\sum_{k=1}^{p} s^k w_j^k \geq r_j, \qquad j \in I, \tag{4.53}$$

$$\sum_{k=1}^{p} w_j^k \leq u_j, \qquad j \in I, \tag{4.54}$$

$$w_j^k \geq 0, \text{ integer }, \qquad k = 1, \ldots, p, \ j \in I. \tag{4.55}$$

In our case study, the solution of this ILP takes much shorter than the solution of the LP relaxation of (4.7)–(4.11) and provides the same lower bound value, see Section 4.10. On the other hand, it is not clear (at least to us) how to use this alternative relaxation to drive a heuristic method.

4.9 Maintenance Constraints

A key constraint that is imposed in our case study, and that we did not discuss in detail so far to keep the presentation simple, is the one imposing that each TU of type k ($k = 1, \ldots, p$) has to undergo a maintenance operation every m^k days. Generally speaking, this operation requires a transfer to a maintenance point (by deadheading), a certain amount of time at the maintenance point, and then a transfer from the maintenance point.

Given the very flexible representation of the sequencing constraints via graph G, we can model the maintenance constraints by specifying, for each $k \in \{1, \ldots, p\}$, a subset of arcs $M^k \subseteq A^k$ corresponding to sequences of two trips that allow a maintenance in between for a TU of type k. Possibly, we have that M^k contains arcs of the form $(0, j)^k$, $(j, n+1)^k$ (e.g., if the maintenance can be performed overnight). Recalling the cyclic nature of the daily assignments to TUs of type k illustrated at the beginning of Section 4.2, letting $e^k \leq d^k$ be the number of paths in (V, A^k) selected in the solution, the maintenance constraints impose that at least $\lceil e^k / m^k \rceil$ of these paths contain at least one arc in M^k.

Within ILP model (4.7)–(4.11), letting $\mathcal{Q}^k \subseteq \mathcal{P}^k$ denote the subcollection of paths in $\mathcal{P}^k$ that contain at least one arc in M^k, the maintenance constraints can be represented by adding the integer variables y^k, indicating the number of paths in $\mathcal{Q}^k$ selected for TUs of type k, along

with the constraints:

$$\sum_{P \in \mathcal{Q}^k} x_P \geq y^k, \qquad k = 1, \ldots, p, \tag{4.56}$$

$$\sum_{P \in \mathcal{P}^k} x_P \leq m^k y^k, \qquad k = 1, \ldots, p, \tag{4.57}$$

$$y^k \geq 0, \text{ integer }, \qquad k = 1, \ldots, p. \tag{4.58}$$

(Note that the addition of maintenance constraints to ILP model (4.1)–(4.6) is not as easy.) The presence of maintenance constraints complicates slightly the column generation procedure, that now calls both for the path of maximum profit in $\mathcal{P}^k$ as well as the path of maximum profit in $\mathcal{Q}^k$. On the other hand, given that the paths have to be found in an acyclic directed graph, their determination simply requires, in the canonical dynamic programming procedure, to store for each node not only the maximum-profit path from 0 to that node, but also the maximum-profit path from 0 to that node containing at least one arc in M^k (if any).

The presence of maintenance constraints must also be carefully taken into account in the heuristic method described in Section 4.6, since these constraints are systematically violated, at least in our case study, if they are not imposed explicitly. In particular, in the fixing phase, when searching for the fractional variable of maximum value to fix, we exclude variables x_P for which the addition of $\lceil \overline{x}_P \rceil$ paths to the other paths in $\mathcal{P}^k$ already imposed by previous fixing phases leads to a collection $\overline{P}^k$ of paths such that $|\overline{P}^k \cap \mathcal{Q}^k| < \lceil |\overline{P}^k|/m^k \rceil$ (in other words, these paths would violate the maintenance constraint for the TUs of type k). The same is done in the constructive heuristic procedure: we do not add a path to those already created for a TU of type k if this violates the maintenance constraint – this simply means that in some cases we add the maximum-profit path in $\mathcal{Q}^k$. Finally, in the refinement procedure, rather than applying Observation 3 literally, namely finding for each TU type the minimum number of paths that cover a given set of vertices, we find, still efficiently, a minimum-cost set of paths that cover all the vertices, where the costs are set to 1 for the arcs leaving the dummy source node 0, to $-\varepsilon$ for the arcs associated with maintenances, and to 0 for the remaining arcs – for sufficiently small ε this guarantees to find the minimum number of paths, but also that the maintenance constraints are satisfied also after refining if they were before.

4.10 Experimental Results on the Case Study

In this section we illustrate the preliminary results obtained by the methods discussed in the chapter on a real-world case study.

In our case study we have $n = 528$ trips, for all of which $u_j = 2$, and whose seat requests are distributed as shown in Table 4.1, left. Moreover, we have $p = 8$ TU types, for all of

which $c^k = 1$, i.e., we wish to minimize the overall number of TUs used, and with capacities and number of available units as shown in Table 4.1, right. Moreover, we have that A^k does not depend on k, since two trips i and j can be assigned to the same TU if and only if the time elapsing between the arrival of i and the departure of j is at least equal to the travel time from the arrival station of i to the departure station of j (and this travel time does not depend on the TU type). Finally, the maintenance constraints require a maintenance every at most $m^k = 5$ days ($k = 1, \ldots, p$), and a maintenance requires a period of at least 6 hours between 5AM and 10PM at a specific maintenance point – the time to travel to and from this maintenance point must be taken into account to establish if a given arc is in $\mathcal{Q}^k$.

#trips	r_j
12	1404
14	1329
6	1302
17	1086
20	1044
1	1032
34	990
23	855
34	786
205	702
19	543
27	516
78	495
38	360

TU type k	s^k	d^k
1	1150	2
2	1044	4
3	786	5
4	702	18
5	543	11
6	516	5
7	495	24
8	360	3

Table 4.1: Characteristics of the trips and TU types in our case study.

Our methods were implemented in C, the computational tests were executed on a Pentium 4, 3.2 GHz, 1 Gb Ram, and the LP-solver used was ILOG-CPLEX 9.0.

The solution found be the practitioners uses all the 72 available TUs, implying that finding a feasible solution is the same as finding a solution which is at least as good. Table 4.2 provides a comparison of the lower bounds on the optimum obtained by the models illustrated in the chapter, reporting the associated computing times in seconds. In our case study, these lower bound values are in fact not affected by the presence of the maintenance constraints. The table shows that, as already known for the case without maintenance constraints, the lower bounds provided by the LP relaxations of (4.1)–(4.6) and (4.7)–(4.11) coincide, but also, as expected, that the time required to compute the latter is much smaller (namely, by one order of magnitude) than that required by the former. Moreover, the table also shows that there is an improvement by almost 10% in the lower bound value if the stronger capacity constraints described in Section 4.5 are used, also with a decrease of the associated computing time. Finally, the lower bound obtained by the relaxation described in Section 4.8, provided an

integer optimal solution of ILP (4.51)–(4.55) is found, is the same as the bound obtained by the other models and can be computed within much smaller time.

model	LB	time
(4.1)–(4.6) (LP rel.)	57	6596
(4.1)–(4.6) + (4.24) (LP rel.)	62	4607
(4.7)–(4.11) (LP rel.)	57	414
(4.7)–(4.11) + (4.25) (LP rel.)	62	125
(4.51)–(4.55) (LP rel.)	57	8
(4.51)–(4.55)	62	8

Table 4.2: Lower bound values.

Given the above lower bounds, we have a gap of $72 - 62 = 10$ between the practitioners' solution and the best bound. Our main achievement was to obtain, with the heuristic method described in Section 4.6, that took 1027 seconds, a solution of value 63, that saves 1 TU of type 5, 5 TUs of type 7 and 3 TUs of type 8 with respect to the available ones.

We do not report a table for the heuristic solutions, since, together with many other alternative approaches not even mentioned in this chapter, none of the following variants of the method described in Section 4.6 finds a feasible solution (i.e., a solution of value ≤ 72), not even for the case in which maintenance constraints are neglected (in which case our heuristic method finds an optimal solution of value 62):

- the one in which constraints (4.25) are not used;
- the one in which the fixing phase is not used, terminating the procedure when there are no violated dual constraints;
- the one in which the refinement procedure is not used;
- the one in which the constructive heuristic procedure is not used, and refinement is applied only to the final solution found by the fixing phase.

As already mentioned, the fact that there is a wide margin of improvement over the practitioners' solution and that such an improvement is indeed achieved by the best approach we could design is apparently in contrast with the fact that, as soon as any of the parts of this approach are deactivated, no improvement is obtained any more. This is certainly an intriguing aspect of our case study that we plan to investigate further in the future.

Bibliography

[1] E.W.J. Abbink, B.W.V. van den Berg, L.G. Kroon, and M. Salomon, "Allocation of Railway Rolling Stock for Passenger Trains", *Transportation Science* 38 (2004), 33–41.

[2] A. Alfieri, R. Groot, L.G. Kroon, and A. Schrijver, "Efficient Circulation of Railway Rolling Stock", ERIM Research Report, ERS-2002-110-LIS, Erasmus Universiteit Rotterdam, The Netherlands, 2002.

[3] F. Barber, M. A. Salido, L. P. Ingolotti, M. Abril, A. L. Lova, M. P. Tormos, "An Interactive Train Scheduling Tool for Solving and Plotting Running Maps", *Lecture Notes in Computer Science*, 3040/2004, 646–655 (2004).

[4] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, "Branch-and-Price: Column Generation for Solving Huge Integer Programs", *Operations Research* 46, 316–329 (1998)

[5] N. Ben-Khedher, J. Kintanar, C. Queille, and W. Stripling, "Schedule Optimization at SNCF: From Conception to Day of Departure", *Interfaces* 28 (1998), 6–23.

[6] F. Bonomo, G. Durán, and J. Marenco, "Exploring the Complexity Boundary between Coloring and List-Coloring", *Electronic Notes in Discrete Mathematics* 25 (2006), 41–47.

[7] R. Borndörfer, M. Grötschel, S. Lukac, K. Mitusch, T. Schlechte, S. Schultz, A. Tanner, "An Auctioning Approach to Railway Slot Allocation", ZIB Technical Report ZR-05-45, 2005.

[8] U. Brännlund, P.O. Lindberg, A. Nöu and J.E. Nilsson, "Allocation of Scarce Track Capacity Using Lagrangian Relaxation", *Transportation Science* 32, 358–369 (1998).

[9] J. Brucker, J.L. Hurink, and T. Rolfes, "Routing of Railway Carriages: A Case Study", *Osnabrücker Schriften zur Mathematik*, Reihe P, Heft 205, 1998.

[10] M.R. Bussieck, T. Winter, and U.T. Zimmermann. "Discrete Optimization in Public Rail Transport", *Mathematical Programming* 79 (1997), 415–444.

[11] V. Cacchiani, A. Caprara and P. Toth, "A Column Generation Approach to Train Timetabling on a Corridor", *4OR*, to appear.

[12] V. Cacchiani, A. Caprara and P. Toth, "Freight Transportation in Railway Networks", *Technical Report OR-06-11, DEIS, University of Bologna*, accepted at EURO Winter Institute 2007, Portugal.

[13] V. Cacchiani, A. Caprara and P. Toth, "Solving a Real-World Train Unit Assignment Problem", *Technical Report OR-06-10, DEIS, University of Bologna*, presented at Informs 2006, Pittsburgh (USA).

[14] X. Cai and C.J. Goh, "A Fast Heuristic for the Train Scheduling Problem", *Computers and Operations Research* 21, 499–510 (1994).

[15] A. Caprara, M. Fischetti and P. Toth, "Modeling and Solving the Train Timetabling Problem", *Operations Research* 50, 851–861 (2002).

[16] A. Caprara, L. Kroon, M. Monaci, M. Peeters and P. Toth, "Passenger Railway Optimization", in C. Barnhart, G. Laporte (eds.), *Transportation*, Handbooks in Operations Research and Management Science 12, Elsevier, Amsterdam, 2005.

[17] A. Caprara, M. Monaci, P. Toth and P.L. Guida, "A Lagrangian Heuristic Approach to Real-World Train Timetabling Problems", *Discrete Applied Mathematics* 154, 738–753 (2006).

[18] M. Carey and D. Lockwood, "A Model, Algorithms and Strategy for Train Pathing", *Journal of the Operational Research Society* 46, 988–1005 (1995).

[19] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver *Combinatorial Optimization*, John Wiley and Sons, 1998.

[20] J.-F. Cordeau, P. Toth, and D. Vigo, "A Survey of Optimization Models for Train Routing and Scheduling", *Transportation Science*, 32 (1998), 380–404.

[21] J.-F. Cordeau, F. Soumis, and J. Desrosiers, "A Benders Decomposition Approach for the Locomotive and Car Assignment Problem", *Transportation Science* 34 (2000), 133–149.

[22] J.-F. Cordeau, F. Soumis, and J. Desrosiers, "Simultaneous Assignment of Locomotives and Cars to Passenger Trains", *Operations Research* 49 (2001), 531–548.

[23] J.-F. Cordeau, G. Desaulniers, N. Lingaya, F. Soumis, and J. Desrosiers, "Simultaneous Locomotive and Car Assignment at VIA Rail Canada", *Transportation Research* 35 (2002), 767–787.

[24] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis, "Time Constrained Routing and Scheduling", in M.O. Ball et al. (eds.), *Handbooks in OR & MS*, Vol. 8, Elsevier Science, 1995, 35–139.

[25] L.F. Escudero, M. Guignard and K. Malik, "A Lagrangian Relax and Cut Approach for the Sequential Ordering with Precedence Constraints", *Annals of Operations Research* 50, 219–237 (1994).

[26] P.-J. Fioole, L.G. Kroon, G. Maróti, and A. Schrijver, "A Rolling Stock Circulation Model for Combining and Splitting of Passenger Trains", *European Journal of Operational Research* 174 (2006), 1281–1297 .

[27] M. Fisher, "Optimal Solution of Vehicle Routing Problems Using Minimum k-Trees", *Operations Research* 42, 626–642 (1994).

[28] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[29] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag (1988).

[30] M. Held and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II", *Mathematical Programming* 1, 6–25 (1971).

[31] A. Higgings, E. Kozan, L. Ferreira, "Heuristic Techniques for Single Line Train Scheduling", *Journal of Heuristics* 3, 43–62 (1997).

[32] D. Huisman, L.G. Kroon, R.M. Lentink, and M.J.C.M. Vromans, "Operations Research in Passenger Railway Transportation", *Statistica Neerlandica* 59 (2005), 467–497.

[33] D. Jovanovic and P.T. Harker, "Tactical Scheduling of Rail Operations: the SCAN I System", *Transportation Science* 25, 46–64 (1991).

[34] L.G. Kroon, R. Dekker, M.J.C.M. Vromans, "Cyclic Railway Timetabling: a Stochastic Optimization Approach", Technical Report ERS-2005-051-LIS, Erasmus University Rotterdam, (2005).

[35] L. G. Kroon, L. W. P. Peeters, "A variable trip time Model for Cyclic Railway Timetabling",*Transportation Science*, 37, 198–212 (2003).

[36] C. Liebchen, M. Proksch and F.H. Wagner "Performance of Algorithms for Periodic Timetable Optimization", TU Preprint 021/2004, TU Berlin, 2004, to appear in

Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin.

[37] T. Lindner, "Train Schedule Optimization in Public Rail Transport", Ph.D. Thesis, University of Technology, Braunschweig, 2000.

[38] T. Lindner and U.T. Zimmermann, "Cost-Oriented Train Scheduling", *8th International Conference on Computer Aided Scheduling of Public Transport (CASPT 2000)*, Berlin, June 2000.

[39] T. Lindner and U.T. Zimmermann, "Cost Optimal Periodic Train Scheduling", *Mathematical Methods of Operations Research* 62, 281–295 (2005).

[40] N. Lingaya, J.-F. Cordeau, G. Desaulniers, J. Desrosiers, and F. Soumis, "Operational Car Assignment at VIA Rail Canada", *Transportation Research* 36 (2002), 755–778.

[41] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons (1990).

[42] P. Mistry, R. S. K. Kwan, "Generation and Optimization of Train Timetables using coevolution", *Lecture Notes in Computer Science*, 2723/2003, 693–694 (2003).

[43] K. Nachtigall, "Periodic Network Optimization and Fixed Interval Timetables", Habilitation Thesis, Deutsches Zentrum für Luft-und Raumfahrt, Braunschweig, 1999.

[44] K. Nachtigall and S. Voget. "A Genetic Algorithm Approach to Periodic Railway Synchronization. *Computers and Operations Research* 23 (1996), 453–463.

[45] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley and Sons (1988).

[46] M. Odijk. "A Constraint Generation Algorithm for the Construction of Periodic Railway Timetables", *Transportation Research* 30 (1996), 455–464.

[47] E. Oliveira and B.M. Smith, "A Job-Shop Scheduling Model for the Single-Track Railway Scheduling Problem ", Technical Report 2000.21, School of Computing Research Report, University of Leeds, 2000.

[48] L.W.P. Peeters, *Cyclic Railway Timetable Optimization*, Ph.D Thesis, Erasmus Research Institute of Management, Erasmus University Rotterdam, 2003.

[49] L.W.P. Peeters and L.G. Kroon, "A Cycle Based Optimization Model for the Cyclic Railway Timetabling Problem", in J. Daduna, S. Voss (eds.), *Computer-Aided Transit*

Scheduling, Lecture Notes in Economics and Mathematical Systems 505, Springer-Verlag, Berlin, 275–296, 2001.

[50] M. Peeters and L.G. Kroon,. "Circulation of Railway Rolling Stock: a Branch-and-Price Approach", ERIM Research Report, ERS-2003-055-LIS, Erasmus Universiteit Rotterdam, The Netherlands, 2003.

[51] S. Rouillon, G. Desaulniers, and F. Soumis, "An Extended Branch-and-Bound Method for Locomotive Assignment", *Transportation Research* 40 (2006), 404-423.

[52] P. Serafini and W. Ukovich, "A Mathematical Model for Periodic Event Scheduling Problems", *SIAM Journal on Discrete Mathematics* 2 (1989), 550–581.

[53] A. Schrijver, "Minimum Circulation of Railway Stock", *CWI Quarterly* 6 (1993), 205–217.

[54] A. Schrijver and A. Steenbeek, "Timetable Construction for Railned", Technical Report, CWI, Amsterdam, 1994 (in Dutch).

[55] Y. Semet, M. Schoenauer, "An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling", *Evolutionary Computation*, 3, 2752–2759 (2005).

[56] B. Szpigel, "Optimal Train Scheduling on a Single Track Railway", in M. Ross (editor) *OR'72*, North-Holland, Amsterdam, 343–351, 1973.

[57] P. Vansteenwegen, D. Van Oudheusden, "Developing railway timetables which guarantee a better service", *European Journal of Operational Research*, 173, 337–350 (2006).