Train timetabling by skip-stop planning in highly congested lines

Train Timetabling by Skip-Stop Planning in Highly Congested Lines

Feng Jiang[1], Valentina Cacchiani[2], Paolo Toth[2]

[1]School of Transportation and Logistics, Southwest Jiaotong University,
Erhuan Road 111, North section1, Chengdu, China
feng.jiang@my.swjtu.edu.cn

[2]DEI, Università di Bologna,
Viale Risorgimento 2, 40136 Bologna, Italy
valentina.cacchiani@unibo.it, paolo.toth@unibo.it

Abstract

We study the problem of scheduling passenger trains in a highly congested railway double-track line with the aim of increasing the number of scheduled trains. A feasible timetable of the trains currently scheduled in the network is given. Additional trains should be scheduled to meet the increasing passenger demand. To achieve this goal, we are allowed to increase the dwelling time of some trains at some stations, to let them stop at some additional stations and even to skip a few stops. Thereby, we need to take explicitly into account the deceleration and acceleration times that are needed by the train when it stops at a station. This problem integrates the choice of the train schedule with the choice of the train stops, the latter being usually made in the Line Planning process. To solve this problem, we propose a heuristic algorithm, extended from a previous method to include the new features of the studied application, and show its performance on real-world instances of the Chinese high-speed JingHu corridor (between Beijing and Shanghai) involving up to 387 trains.

Keywords: Train Timetabling, Skip-stop planning, Heuristic algorithm, Real-world case study.

1 Introduction

Railway networks are more and more utilized due to the increase of passenger demand. Therefore, it is very important to achieve a good utilization of the infrastructure capacity. Train scheduling (or Train Timetabling) is a fundamental step to obtain an efficient use of the railway networks. In this paper, we focus on a real-world application arising in the high-speed JingHu double-track line (corridor) between Beijing and Shanghai. Since it was put into operation in 2011, the average increase of passenger volume has been of about 30% every year. In 2014, more than one hundred million of passengers travelled along this corridor. With the construction of new high-speed lines linked to the JingHu corridor, the number of passengers is expected to go up rapidly. In 2015, more than 300 trains ran every day along the corridor between 6 a.m. and midnight. However, due to the increasing passenger demand, especially coming from the linked lines, additional trains need to be scheduled in this highly congested line. For these additional trains a desired schedule is given by the operator, even though it can be changed (e.g. by changing the desired departure time from the origin station). When planning the new train schedule, the existing timetable can be changed by increasing the dwelling time of some trains at some stations, stopping them at some additional stations and even skipping a few stops, but for each train, a maximum number of stops that can be cancelled is imposed. The goal is to maximize the number of scheduled trains, while changing the desired and the current schedules as little as possible. Important elements that must be taken into account consist of the deceleration and acceleration times that are needed by the train when it stops at a station. Since we are dealing with a high-speed line in which trains run with a speed of 250 km/h or 300 km/h, these times cannot be neglected. Furthermore, preliminary computational experiments showed that simply adding these times to the travel time, even if the train does not stop at the station, leads to solutions of poor quality.

The studied problem falls in the category of the so-called *Non-Periodic* (or non-cyclic) Train Timetabling Problem (TTP), in which the train schedules can be different in different periods of the day, although they are repeated every day. We consider a macroscopic level of detail of the railway infrastructure, i.e., we take into account the stations and the tracks connecting them, but neglect the details on train routing inside stations or at junctions. Several real-world constraints are imposed in our application: headway times must be respected between consecutive trains along the same track, station capacity (computed as maximum number of trains that can be present at the same time in a station) must be satisfied, overtaking can

only take place at stations (since the trains are assumed to travel on one track from Beijing to Shanghai, and on the other track from Shanghai to Beijing), minimum travel times, based on the train speeds, and minimum dwelling times must be respected. In addition, no train can run between midnight and 6 a.m. due to maintenance operations that completely block the corridor. Furthermore, we need to consider the constraints related to the changes that we can apply to the existing timetable and to the desired schedules: there is a maximum global amount of additional dwelling time (*maximum stretch*) that can be used by each train, and the departure time of a train from its origin station can be changed earlier or later up to a given *maximum shift*.

1.1 Related Works

A huge amount of research has been done on the TTP. In the following, we describe the works that are more related to the studied problem, and refer the interested reader to the surveys by Bussieck et al. [1997], Cordeau et al. [1998], Törnquist [2006], Caprara et al. [2007], Lusby et al. [2011], Caprara et al. [2011], Cacchiani and Toth [2012], Cacchiani et al. [2014], and to the tutorials by Harrod [2012] and Cacchiani et al. [2015] for a more comprehensive overview on timetable planning, robust planning and rescheduling.

Most of the works that study which stops can be skipped and which should be kept (stop-skipping pattern) concern the Line Planning Problem (Schöbel [2012]) that determines, based on the passenger origin-destination demand, the train stopping pattern, the frequency, and the scheduling of the passenger train lines, but does not determine the train timetables (see e.g. Jamili and Aghaee [2015], Chang et al. [2000] Fu et al. [2015]). However, some recent works deal with the possibility of skipping stops while scheduling trains. In Niu et al. [2015], the goal is to minimize the passenger waiting times by adjusting the existing train timetables for a rail corridor with given time-varying origin-to-destination passenger demand matrices. A quadratic integer programming model is proposed: it takes as input the skip-stop patterns and determines the train schedule, in order to minimize the total waiting passenger times at the stations, while taking into account constraints on train capacity, headway times, departure and dwelling times. The model is reformulated to be expressed in GAMS and tested on an instance of the Shanghai-Hangzhou line in China, which comprises 9 stations and 73 trains. In Yang et al. [2016], the combination of optimizing the train scheduling and the train stop planning is considered. The goal is to minimize the total dwelling time and the total time difference between

the actual and the preferred departure times from the origin station for all the trains. A Mixed Integer Linear Programming model is proposed: it uses time variables to express the departure and arrival times of the trains from/at the stations, binary variables to represent the train sequence and additional binary variables to allow for the choice of stopping or not at a station. Constraints are imposed to satisfy time windows at the departure stations, headway and dwelling times, passenger demand at each station over the entire planning horizon, and minimum number of stops at each station. The times spent by the train for accelerations and decelerations are not considered, i.e., the speed of trains of the same type is a constant on each railway section. Every train is assumed to travel from the same origin station to the same destination station. The model is formulated in GAMS and tested on a real-world instance of the year 2014 with 38 trains traveling on the Beijing-Shanghai high-speed corridor, and on additional instances with up to 96 trains, by considering fixed or free order of the trains from the initial station. In Yue et al. [2016], Integer Linear Programming models and a column-generation-based algorithm are proposed to simultaneously consider the train service plan and the train schedule. The authors assume that the travel time includes the acceleration and deceleration times. The decision variables correspond to the stopping patterns, specifying at which stations each train stops, and the stopping times, specifying how long each train stops at the intermediate stations. The goal is to maximize the total profit of all the trains, where the profit of each train takes into account a penalty for the global stopping time and a penalty for the number of stops of the train. The constraints impose a minimum number of trains travelling between given pairs of stations, require to respect minimum and maximum dwelling times at the stations (if the train stops), headway times, station capacity and overtaking constraints (the latter turn out to be not relevant for the considered case study). A column-generation based algorithm is proposed, in which each variable corresponds to a train trajectory. The models and the algorithm are tested on the Beijing-Shanghai high-speed corridor with 220 trains. The results show that the profit increases by 30%, with a reduction of the average number of stops from 7.1 to 2.8.

We show in Table 1 a comparison of the problem studied in this work with those studied in the literature. With respect to the described works that integrate timetabling and stop planning, we identify some relevant differences that appear in our application. First of all, Niu et al. [2015] and Yang et al. [2016] consider different objective functions, while the objective of Yue et al. [2016] is more similar to ours, even though not the same: indeed, our main goal is to maximize the number of scheduled trains while

paper	goal	constraints	acc/dec	inst. size
Niu et al. [2015]	min passenger waiting times	passenger demand train capacity min headway min/max dwelling dep time windows	yes	73 trains
Yang et al. [2016]	min total dwelling time and total difference between timetables	passenger demand min headway min dwelling min #stops station dep time windows	no	96 trains
Yue et al. [2016]	max profit: penalize stop time and #stops	min #trains station min headway min/max dwelling station capacity overtaking	no	220 trains
This work	max profit: max #trains penalize shift, stretch, skip-stop	max canc stops min headway min/max dwelling station capacity overtaking dep time windows maintenance	yes	304 +83 trains

Table 1: Comparison with works that integrate timetabling and stop planning.

avoiding stop cancellations (and other changes to the timetable), while in Yue et al. [2016] the goal is to maximize the total profit reduced by penalties for the stopping times and for the number of stops of each train. Both Niu et al. [2015] and Yang et al. [2016] consider instances smaller than those considered in our real-world case study. The instance used in Yue et al. [2016] contains 220 trains: this is a large instance, but we deal with a more congested setting, since we start with a timetable that already contains more than 300 trains on the same corridor. In addition, acceleration and deceleration times are directly summed to the travel times in Yang et al. [2016] and in Yue et al. [2016], while taking them explicitly into account significantly complicates the problem tackled in our application. While both Niu et al. [2015] and Yang et al. [2016] take explicitly passenger demand into account, in Yue et al. [2016] it is imposed that a minimum number of trains must serve the passenger flow from an origin station to a destination station, according to the train service plan. In our work, we do not explicitly consider passenger demand but impose, for each train, a maximum number of stops that can be cancelled: this number can be different for different trains and can also be set to 0, according to the passenger demand at each station. In addition, stop skipping is penalized. A discussion on passenger demand for

the JingHu line is presented in Section 2.

The problem of inserting additional trains given an existing timetable has been considered in several works. In Ingolotti et al. [2004], the goal is to add new trains on a heterogeneous heavily loaded railway network, minimizing the travel times of the new trains. Constraints to respect departure time windows, minimum stopping times at stations, headway times, precedences between trains, and station capacity are imposed. The authors propose a sequential heuristic algorithm to schedule the new trains, and test it on instances of the Spanish Railways. In Flier et al. [2009], the risk of train delays when adding a new train to an existing timetable is predicted, by using a series of linear regression models on the basis of extensive real-world delay data of trains. These models are integrated into a combinatorial shortest path model to compute a set of Pareto optimal train schedules with respect to risk and travel time. They test the proposed model on instances of the Swiss Federal Railways. In Burdett and Kozan [2009], the problem of scheduling additional trains is represented as a hybrid job shop scheduling problem with time window constraints, and formulated using a disjunctive graph model. A constructive algorithm and a simulated annealing procedure are proposed for solving the considered problem. The test instances were selected to be indicative of real life applications. First the construction of an existing timetable from scratch is tested, then an existing schedule is considered and additional train services are inserted, with the aim of minimizing the changes with respect to the existing schedule. Finally, the insertion of the additional services is tested subject to the fixing of the existing services. Cacchiani et al. [2010] consider a European network with alternative routes and model the TTP by using a time-space graph. A heuristic algorithm based on the Lagrangian relaxation of a set of complex constraints is proposed. Additional freight trains have to be scheduled while keeping the timetable of the passenger trains as fixed. In this paper, we build upon this study, and extend the algorithm proposed in Cacchiani et al. [2010] to additionally consider the possibility of skipping stops and to take into account acceleration and deceleration times (instead of having fixed travel times between consecutive stations). As we will explain in Section 3, these additional features make the problem solving more difficult, and several changes need to be performed. In Tan [2015], the problem of adding trains to a cyclic timetable is studied. The initial timetable is cyclic, but the possibility to change it to a non-cyclic timetable is considered. Additional trains need to be scheduled according to a given frequency, but a tolerance to deviate from the periodicity is allowed. The author proposes an event-activity network to model the problem. Many real-world constraints

are taken into account: minimum and maximum travel and dwelling times, connections, headway times, departure time windows, acceleration and deceleration times, and station capacity. Various objectives are considered, such as the minimization of the travel times of the additional trains, the minimization of the adjustments to the initial train schedules, and the maximization of the robustness of the new timetable. In addition, the study is extended to deal with the integration of the train-unit circulation problem with the problem of scheduling additional trains. The proposed models are tested on real-world instances of the Shanghai-Hangzhou line, which includes 9 stations and 159 trains. The results show that up to 20 additional trains can be scheduled. This work shows some similarities with our application; however, train stop planning is not considered, and the initial timetable is cyclic. A preliminary version of this paper has been presented in Cacchiani et al. [2016b]. In that work, we considered one of the real-world instances used in this paper, but we neglected both the station capacity and the acceleration/deceleration times, i.e., we dealt with (minimum) fixed travel times between consecutive stations. An iterative heuristic algorithm, based on a two-phase approach, was proposed: the first phase was devoted to schedule additional trains, while the second phase was used to improve the regularity of the timetables, i.e., to define a schedule with regularity in the train frequency at the main stations. We here focus only on the first objective, as we want to keep the focus on the main goal of maximizing the number of scheduled trains.

Other approaches originally developed for the TTP could also be adapted to tackle the problem of scheduling additional trains. Among the most successful and recent ones we mention D'Ariano et al. [2007], Liebchen [2008], Kroon et al. [2009], Bešinović et al. [2016], Lamorgese et al. [to appear].

1.2 Contributions

The main contributions of this paper are the following ones:

1. a real-world application is studied, that incorporates difficult aspects that are studied separately in the existing literature (see Section 1.1): indeed, the studied application combines train scheduling (and the insertion of additional trains) with the possibility of adding and/or skipping stops, while taking explicitly into account acceleration/deceleration times;

2. a Lagrangian-based heuristic algorithm, extended from the method

presented in Cacchiani et al. [2010] to deal with the additional real-world features, is proposed;

3. real-world instances of the JingHu line are tested in different settings, showing the effectiveness of the proposed algorithm both in improving the existing timetable and in scheduling additional trains.

To highlight the contributions of this work, we provide the differences of the objectives (in Table 2) and the constraints (in Table 3) of the problems studied in this work and in the previous works we build upon. In Table 2, we show that the goal of Caprara et al. [2002] was to change the desired timetables of a given set of trains as little as possible: this was obtained by penalizing shift and stretch changes. In Cacchiani et al. [2010], the goal was to maximize the number of scheduled trains, while minimizing the changes to the desired timetables. However, none of the two works considered the possibility of skipping stops. In this work, we investigate this possibility: thus, an additional goal is to penalize the stop cancellations.

paper	max #trains	shift pen	stretch pen	skip-stop pen
Caprara et al. [2002]	no	yes	yes	no
Cacchiani et al. [2010]	yes	yes	yes	no
This work	yes	yes	yes	yes

Table 2: Comparison on the objectives with previous works.

All the constraints considered in this work are listed in Tables 3: a departure time window (dep), minimum headway times (hw), minimum dwelling times (dw), minimum travel times (tr), maximum shift (sh) and maximum stretch (str) must be respected; the maximum station capacity and the maintenance schedule (maint) must be satisfied; the maximum number of stops (canc) that can be cancelled is taken into account for each train. As it can be seen, both Caprara et al. [2002] and Cacchiani et al. [2010] neglect some of the considered constraints.

paper	dep	hw	dw	tr	ov	sh	str	cap	maint	canc
Caprara et al. [2002]	yes	yes	yes	yes	yes	yes	yes	no	no	no
Cacchiani et al. [2010]	yes	yes	yes	yes	yes	yes	yes	no	no	no
This work	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

Table 3: Comparison on the constraints with previous works.

Finally, in Table 4, we report the features considered in each of the works: the possibility of scheduling additional trains in a network where some trains are already planned, of rerouting trains, of skipping stops and

of taking into account acceleration and deceleration times. As it can be seen, the two latter features are not present in Caprara et al. [2002] and Cacchiani et al. [2010]. Note that we do not consider rerouting in this work, as we do not deal with a network but with a double-track line.

paper	add trains	rerouting	skip-stop	acc/dec
Caprara et al. [2002]	no	no	no	no
Cacchiani et al. [2010]	yes	yes	no	no
This work	yes	no	yes	yes

Table 4: Comparison on the considered features with previous works.

The main difference lies in considering skip-stop planning in the construction of the timetables. Indeed, this feature was not considered in Caprara et al. [2002] nor in Cacchiani et al. [2010]. It affects the objective function (that takes into account a penalty for skipping stops) and the constraints. By allowing the possibility of skipping stops, acceleration and deceleration times must also be taken into account, while fixed travel times were considered in Caprara et al. [2002] and Cacchiani et al. [2010]. To model the problem including these two additional features, the graph representation has been modified, as will be explained in Section 3.1, and the overtaking constraints become more complex (see Section 3.2). The latter constraints need to be relaxed in two steps to deal with the increased complexity (see Section 3.3). Finally, in order to embed skip-stop planning together with acceleration and deceleration times in the Lagrangian-based heuristic, a dynamic programming algorithm has been developed, that includes an additional condition to decide whether or not to stop at each station, as will be illustrated in Section 3.4. Compared to the works we build upon, we have combined skip-stop planning with timetabling, by simultaneously considering acceleration and deceleration times, and have extended the model and the methodology to tackle these new aspects.

The paper is organized as follows: in Section 2 we report a formal description of the studied problem; Section 3 first describes a graph representation for the problem and explains how to handle some classes of constraints when acceleration and deceleration times must be considered, and then presents a heuristic algorithm to solve the problem. In Section 4, we report computational results on real-world instances. We conclude summarizing our work in Section 5. Finally, in the Appendix, we present an example to illustrate the main steps of the proposed algorithm.

2 Problem Description

We consider a double-track line represented as a sequence of stations or important points where trains can stop. Each train travels along a track in one direction, and trains can interact at stations. Let $S = \{1, \ldots, n_s\}$ be the set of stations. Each station is characterized by a capacity c_s, which corresponds to the number of platforms available at station $s \in S$, and by the minimum departure and arrival headway times d_s and a_s, respectively, that two consecutive trains departing from or arriving at station s in the same direction must respect for safety reasons. We consider a planning horizon H of one day, discretized in time units (e.g. in minutes). More precisely, since no train can run between midnight and 6 a.m. due to maintenance operations that completely block the corridor, we consider a time horizon $\bar{H} \subset H$ as the planning horizon and avoid trains to be scheduled outside $\bar{H}$. A set $T = \{1, \ldots, n_t\}$ of *existing* trains are scheduled to travel along the line, in both directions, according to the current timetable. For each train $t \in T$, a timetable, containing the departure time of train t from its origin station, the arrival time at its destination station, and the arrival and departure times at each intermediate station visited by train t, is given. This is the current timetable that has been defined by the operator. From this timetable, we know the subset of stations at which train t stops and the subset of stations through which the train travels without stopping. A minimum dwelling time dw_{ts} at each station $s \in S$ (equal to 0 if train t does not stop at s) is also known. Note that the stopping time of train t at station s in the current timetable can be larger than dw_{ts}. In this case, we say that the train has undergone a *stretch*, i.e., it has a stop longer than the minimum required one. This can happen, for example, if the operator has imposed a longer stop to allow an overtaking. In addition, for each train $t \in T$, we are given its minimum travel time $l_t^{s_1,s_2}$ between each pair (s_1, s_2) of consecutive stations ($s_1, s_2 \in S$) visited by t, and the acceleration acc_t and deceleration dec_t times that are needed if the train stops at a station. Every train must accelerate from its origin station and must decelerate at its destination station. At every station at which the train stops, it has both to decelerate before the stop and to accelerate after the stop. When the train does not stop at a station, we consider that it travels at its maximum speed, i.e., we do not allow travel times longer than the minimum ones.

Beside the existing trains in the current timetable, we are also given a set T_{new} of *new* trains that we would like to schedule along the double-track line. For each train $t \in T_{new}$, we are given its origin station, its destination station, the set of stations at which the train should stop, with the

corresponding minimum dwelling times, the minimum travel times between each pair of consecutive stations visited by the train, the deceleration and acceleration times that are needed if the train stops at a station, and a desired departure time from its origin station. Note that, since we know, for each train in T_{new}, a desired departure time from the origin station, and the minimum dwelling and travel times, we also know its desired schedule.

The main goal of the studied application is to schedule as many trains as possible, but we also want to take into account the changes that are applied to the current timetable and to the desired schedules, and perform as few changes as possible. For each train, the allowed changes are the following ones: (i) to increase the dwelling time of the train at one or more visited stations, (ii) to stop the train at one or more additional stations (where it was not planned to stop), (iii) to skip one or more stops of the train; (iv) to change the departure time of the train from its origin station within a given departure time window. Note that, by increasing the dwelling time of a train at an intermediate station or by skipping the associated stop, the departure time of the train from the station is also changed. These changes are subject to limits, so as not to change the existing schedules too much, and possibly follow the desired schedules for the new trains. In particular, for each train $t \in T \cup T_{new}$, we are given the global maximum stretch $maxstr_t$ that the train can undergo, i.e., the maximum global increase of the stopping time throughout the train path: this limits the increase in the dwelling times, and is imposed to avoid "very long" stops at the stations, which make the global passenger travel time longer. Note that, as mentioned above, existing trains $t \in T$ can already be subject to a stretch in the current schedule: in this case, the maximum stretch $maxstr_t$ is referred to the allowed additional stretch that we can apply. Stop skipping is seen as a significant change: therefore, it is highly penalized and, for each train $t \in T \cup T_{new}$, a maximum non negative number $maxcanc_t$ of stops that can be cancelled is imposed. Finally, we are allowed to *shift* (i.e. to anticipate or postpone) the departure time of a train from its origin station, but a maximum change $maxsh_t$, for each train $t \in T \cup T_{new}$, is allowed.

In order to apply as few changes as possible to the existing and to the new schedules, each change is assigned a penalty, and each train is assigned a profit that is decreased if a change occurs in its schedule. For each train $t \in T \cup T_{new}$, a profit p_t is given, where p_t is a large positive number so that it is always favorable to schedule train t, if a feasible schedule exists, as our main goal is to maximize the number of scheduled trains. In particular, we consider a higher profit value for the existing trains and a lower value for the new trains, so as to reduce the changes to the current schedule. The

following penalties are considered for each train $t \in T \cup T_{new}$: a penalty $pstr_t$ for each time unit of stretch, a penalty $pskip_t$ for skipping a stop and a penalty psh_t for each time unit of shift. The goal is to maximize the sum of the actual profits of the scheduled trains, which take into account the listed penalties, so as to schedule as many trains as possible, and to minimize the changes to the existing and to the desired schedules.

The volume of passengers traveling along the JingHu line is very large and is increasing from 2011, especially due to the newly built linked lines. Since the passenger demand is very large, we do not explicitly take it into account in the problem, as the main goal is to schedule as many additional trains as possible. Recall that we are allowed to cancel stops in order to schedule additional trains, but a limit $maxcanc_t$ is imposed on the maximum number of stops that can be cancelled for each train t, and stop cancellation is highly penalized (through the parameter $pskip_t$) in the objective function. In this way, we can control the trade-off between stop-skipping and scheduling of additional trains, even though the passenger demand is not explicitly considered. Note that $maxcanc_t$ can be set to 0 for selected trains $t \in T \cup T_{new}$, if stop-skipping is not allowed for them.

We refer the reader to the following recent works that consider passenger demand oriented metro or train scheduling and rescheduling: Canca et al. [2016], Peer et al. [2016], Sels et al. [2016], Robenek et al. [2016], Zhou and Teng [2016], Gao et al. [2016], Yin et al. [2016], Li et al. [2017], Yin et al. [2017].

3 Solution Method

We modify the method proposed in Cacchiani et al. [2010] to deal with the possibility of adding or removing stops and to take explicitly into account acceleration and deceleration times. In this section, we present the main changes that have been performed to extend this method.

3.1 Graph Representation

To represent the problem described in Section 2 we adopt a time-space directed multigraph $G = (V, A)$ that is very commonly used for non-periodic TTP: see e.g. Caprara et al. [2002], Caprara et al. [2006], Cacchiani et al. [2010] and Cacchiani et al. [2016a]. In these works, however, for each train, the travel times between consecutive stations are considered as fixed, and it is not allowed to skip stops. We here describe the graph structure and

how it is extended to deal with acceleration and deceleration times as well as stop skipping.

Set V contains *departure nodes* and *arrival nodes*: each node represents, respectively, a time instant of departure or arrival of a train from/at a station. Let R be the set of a mono-directional tracks between two consecutive stations along the line. Each track is identified by $r = (h, i) \in R$, where h and i are two consecutive stations in S. The track can be used by trains travelling from h to i. The set of nodes V is defined as:

$$V = \{\sigma, \tau\} \cup \bigcup_{r=(h,i)\in R} (U(i, r) \cup W(h, r)),$$

where σ and τ are fictitious source and sink nodes, respectively, $U(i, r)$ is the set of arrival nodes at station i along track r and $W(h, r)$ is the set of departure nodes from station h along track r. The set of arcs A is partitioned into arc sets $A^1, \ldots, A^{|T\cup T_{new}|}$, where arc set A^t contains the *starting arcs*, the *travel arcs*, the *station arcs* and the *ending arcs* of train $t \in T \cup T_{new}$. In particular, a starting arc connects the source node σ to each departure node of train t from its origin station, a travel arc (that connects a departure node to an arrival node) represents the travel of a train between two consecutive stations, a station arc (that connects an arrival node to a departure node) represents the stop or the passing-through of a train at a station, and an ending arc connects each arrival node of train t at its destination station to the sink node τ. For each train $t \in T \cup T_{new}$, set V^t contains the set of nodes that can be visited by train t, based on the values of $maxsh_t$ and $maxstr_t$ that define the corresponding time windows in each station visited by the train. In particular, for each train $t \in T \cup T_{new}$, set V^t contains σ, the departure nodes from the origin station of t, the arrival nodes at the destination station of t, the arrival and departure nodes at/from each intermediate station visited by t, and τ. Sets V^t and A^t define a subgraph $G^t = (V^t, A^t)$ for each train $t \in T \cup T_{new}$. We refer the reader to Cacchiani et al. [2015] for more details on time-space multigraphs for the non-periodic TTP.

When the travel times are fixed, a single travel arc exists from a departure node at a station to an arrival node at the consecutive station: in Caprara et al. [2002], Caprara et al. [2006] and Cacchiani et al. [2010], this arc corresponds to the minimum travel time arc. Beside these arcs, we additionally need travel arcs that correspond to the minimum travel time plus the acceleration time, those that correspond to the minimum travel time plus the deceleration time, and those that correspond to the sum of all the

three times. Let $a \in A^t$ be a travel arc representing the travel of train $t \in T \cup T_{new}$ between the two consecutive intermediate stations s_1 and s_2. Four cases can occur: (i) train t neither stops at s_1 nor at s_2: in this case, a represents the minimum travel time $l_t^{s_1,s_2}$; (ii) train t stops at s_1 but does not stop at s_2: in this case, a represents the travel time $l_t^{s_1,s_2} + acc_t$; (iii) train t stops at s_2 but does not stop at s_1: in this case, a represents the travel time $l_t^{s_1,s_2} + dec_t$; (iv) train t stops both at s_1 and at s_2: in this case, a represents the travel time $l_t^{s_1,s_2} + acc_t + dec_t$. Note that, when s_1 is the origin station of train t we only consider a travel arc with time $l_t^{s_1,s_2} + acc_t$, while, when s_2 is the destination station of train t we only consider a travel arc with time $l_t^{s_1,s_2} + dec_t$.

Standard station arcs, used in Caprara et al. [2002], Caprara et al. [2006] and Cacchiani et al. [2010] when stop skipping is not allowed, represent the different stopping time durations at a station: a train t can stop at station s for the minimum dwelling time dw_{ts} or for at most $dw_{ts}+maxstr_t$ time units. When stop skipping is allowed, we need to consider, for each intermediate station s and each train t visiting s, an additional arc that represents the passing-through of train t at s.

For each train $t \in T \cup T_{new}$, train profits and penalties are associated with arcs of G^t. Let p_{ta} be the profit of arc $a \in A^t$, $t \in T \cup T_{new}$. In particular, the train profit minus the shift penalty (psh_t for each time unit of shift) is associated with each starting arc from σ to a departure node w from the origin station of train t: $p_{t(\sigma,w)} := p_t - psh_t * \nu_{t(w)}$, where $\nu_{t(w)}$ is the shift of train t when it departs from node w. The stretch penalty ($pstr_t$ for each time unit of stretch) is associated with each station arc of station $s \in S$ of duration larger than dw_{ts}. The penalty $pskip_t$ for skipping a stop is associated with the station arcs corresponding to a passing-through of train t without stopping. Therefore, the profit of a station arc $(u, w) \in A^t$ corresponding to a stop of train t at station s longer than the minimum dwelling time dw_{ts} is defined as: $p_{t(u,w)} := -pstr_t * \mu_{t(u,w)}$, where $\mu_{t(u,w)}$ is the stretch of train t along arc (u, w) (i.e., the number of time units above dw_{ts}. The profit of a station arc $(u, w) \in A^t$ corresponding to skipping the stop at station s is defined as: $p_{t(u,w)} := -pskip_t$. No profit or penalties are associated with the travel arcs and with the ending arcs.

Minimum travel time constraints, minimum dwelling time constraints, and maximum shift constraints are imposed directly by the graph definition. On the contrary, maximum stretch constraints and the acceleration and deceleration times are handled in the dynamic programming algorithm, described in Section 3.3. The constraints on the maximum number of stops that can be cancelled are also handled in the dynamic programming algo-

rithm, but in a heuristic way: the dynamic programming algorithm checks that every computed maximum profit path for train $t \in T \cup T_{new}$ has at most $maxcanc_t$ cancelled stops and, if this is not the case, computes the maximum profit path without allowing stop skipping. Each feasible path of a train t in G^t that satisfies all these constraints corresponds to a feasible timetable for t. When stop skipping is allowed, the minimum dwelling time constraints become soft constraints, and skipping a stop is penalized in the objective function.

3.2 Modelling the Problem

We model the TTP as an Integer Linear Programming (ILP) model, by using graph G and a binary variable x_{ta} for each train $t \in T \cup T_{new}$ and each arc $a \in A^t$ that assumes value 1 if arc a is selected in the solution for train t (and 0 otherwise), as in Caprara et al. [2002] and Cacchiani et al. [2010]. The main difference with respect to the ILP models proposed in these works derives from the different time-space graph, described in the previous section, due to the acceleration and deceleration times. Consequently, the constraints related to the train travel times, i.e., the overtaking constraints, need to be modified.

In this section, we first briefly describe the constraints that are modelled as in the previous papers (and refer the reader to Caprara et al. [2002] and Cacchiani et al. [2010] for further information), and then explain in detail the new way to handle the overtaking constraints. Finally, we present an ILP model for the studied problem.

The headway departure (arrival, resp.) constraints are modelled as follows: for any time interval with duration shorter than the minimum departure (arrival, resp.) headway time, we impose to select in the solution at most one travel arc among all the arcs (of any train) leaving (entering, resp.) a node that belongs to such time interval. In this way, two consecutive departures (arrivals, resp.) of trains from a station along a track will respect the minimum departure (arrival, resp.) headway time. To make the formulation of the departure and arrival constraints clear, an auxiliary binary variable y_v is introduced for each node $v \in V$ which assumes value 1 if node v is visited by any train (and 0 otherwise). The capacity constraints are expressed by imposing, for each time instant of the planning horizon, to select in the solution at most a number of station arcs (of any train) equal to the capacity of the station. Note that we also have station arcs corresponding to the passing-through of a train at a station. The maintenance constraints can be simply imposed by considering a shorter planning horizon, as ex-

plained in Section 2. When stop skipping is allowed, the constraints on the maximum number of stops that can be cancelled are expressed by imposing an upper bound on the number of stops for which the minimum dwelling time is not respected. Finally, the constraints on the minimum travel times, minimum dwelling times and maximum shift are satisfied, for each train, by its graph definition, while the constraints on the maximum stretch and the acceleration and deceleration times are directly handled by the dynamic programming algorithm (see Section 3.3).

The novelty in the problem modelling consists of the overtaking constraints. We first show an example to illustrate the overtaking constraints in the case of fixed travel times, and then we explain how these constraints have to be changed to take into account the acceleration and deceleration times. In Figure 1, time grows from left to right. The two horizontal lines correspond to two consecutive stations s_1 and s_2 along a track. Departure nodes from s_1 and arrival nodes at s_2 are reported on these lines. The arrows correspond to travel arcs with minimum fixed travel times. Arc (w_1, v_1) represents the travel of train j from s_1 to s_2, while arc (w_2, v_2) represents the travel of train k between the same two stations. Train j is the "slow" train, while train k is the "fast" one. Since the two arcs cross, a forbidden overtaking is occurring between k and j. To avoid it, at most one of the two crossing arcs can be selected in a feasible solution. In Caprara et al. [2002] and Cacchiani et al. [2010], the overtaking constraints were strengthened by inserting, in the same constraint, additional travel arcs corresponding to further overtakings of trains j and k, and imposing to select at most one among all these arcs. In particular, let w_3 be the first departure time of train j from station s_1 that is feasible with the departure of train k from s_1 at time w_2, i.e., the minimum departure headway time between w_2 and w_3 is respected. Let w_4 be the first departure time of train k such that the corresponding arrival time v_4 at station s_2 is feasible with the arrival at time v_1 of train j, i.e., the minimum arrival headway time between v_1 and v_4 is respected. The strengthened overtaking constraint imposes to select at most one arc among all the travel arcs of train j from time w_1 up to time w_3 excluded, and of train k from time w_2 up to time w_4 excluded. Since the travel times between consecutive stations are fixed, overtaking constraints are expressed by imposing to select at most one node among the departure nodes of train j in $W_j := [w_1, w_3)$, and the departure nodes of train k in $W_k := [w_2, w_4)$. In particular, an auxiliary binary variable z_{tv} is introduced for each train $t \in T \cup T_{new}$ and each departure node $v \in V^t$ which assumes value 1 if train t visits node v. These variables are defined by equations as the sum of the corresponding arc variables.

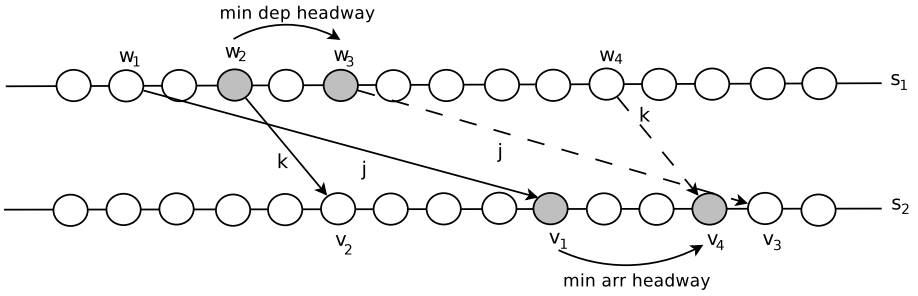

Figure 1: Example of overtaking constraints in the case of fixed travel times.

When we take into account acceleration and deceleration times, we have to consider not only the departure node but also the arrival node of each arc, since the travel times are not fixed anymore. In particular, we must insert, in the overtaking constraints, any arc of the slow train j that leaves from a node before time w_3 and arrives at or after time v_1, and any arc of the fast train k that leaves from a node after or at time w_2 and arrives before time v_4, and select at most one among all these arcs. In the next section, we will explain how we deal with these constraints in the proposed solution method. Stop skipping, which was not considered in the ILP models of Caprara et al. [2002] and Cacchiani et al. [2010], is easily handled by associating a penalty with the station arcs that correspond to a passing-through and by minimizing the sum of these penalties in the objective function.

We conclude this section by presenting an ILP model for the studied problem. Recall that the minimum travel time constraints, minimum dwelling time constraints, and maximum shift constraints are imposed directly by the graph definition. Therefore, all these constraints do not appear explicitly in the formulation. Let $\delta_t^+(v)$ and $\delta_t^-(v)$ denote the sets of arcs in A^t leaving and entering node v, respectively. Let $\theta(v)$ be the time instant associated with a given node $v \in V$ and $\Delta(u,v) := \theta(v) - \theta(u)$. We say that node u *precedes* node v (i.e., $u \preceq v$) if $\Delta(v,u) \geq \Delta(u,v)$. Analogously, we will use the notation $u \prec v$, $u \succeq v$, $u \succ v$. Let $R^t \subseteq R$ be the set of tracks visited by train $t \in T \cup T_{new}$.

An ILP model for the considered problem reads as follows.

$$\max \sum_{t \in T \cup T_{new}} \sum_{a \in A^t} p_{ta} x_{ta} \tag{1}$$

$$\sum_{a \in \delta_t^+(\sigma)} x_{ta} \leq 1, \qquad t \in T \cup T_{new}, \tag{2}$$

$$\sum_{a\in\delta_t^-(v)} x_{ta} = \sum_{a\in\delta_t^+(v)} x_{ta}, \qquad t\in T\cup T_{new}, v\in V^t\setminus\{\sigma,\tau\}, \tag{3}$$

$$z_{tv} = \sum_{a\in\delta_t^-(v)} x_{ta}, \qquad t\in T\cup T_{new}, v\in V^t\setminus\{\sigma,\tau\}, \tag{4}$$

$$y_v = \sum_{t\in T\cup T_{new}: v\in V^t} z_{tv}, \qquad v\in V\setminus\{\sigma,\tau\}, \tag{5}$$

$$\sum_{v\in W(i,r): v\succeq w, \Delta(v,w)<d_i} y_v \le 1, \qquad i\in S, r=(i,h)\in R, w\in W(i,r) \tag{6}$$

$$\sum_{v\in U(h,r): v\succeq u, \Delta(v,u)<a_h} y_v \le 1, \qquad h\in S, r=(i,h)\in R, u\in U(h,r) \tag{7}$$

$$\sum_{t\in T\cup T_{new}} \sum_{r\in R} \sum_{\substack{a=(u,v):\\ u\in U(i,r)\cap V^t,\\ v\in W(i,r)\cap V^t,\\ \theta(u)\le\bar{q},\ \theta(v)\ge\bar{q}}} x_{ta} \le c_i, \qquad i\in S, \bar{q}\in\bar{H} \tag{8}$$

$$\sum_{r\in R}\sum_{i\in S} \sum_{\substack{a=(u,v):\\ u\in U(i,r)\cap V^t,\\ v\in W(i,r)\cap V^t,\\ \Delta(v,u)>dw_{ti}}} (\Delta(v,u)-dw_{ti})x_{ta} \le maxstr_t, \qquad t\in T\cup T_{new} \tag{9}$$

$$\sum_{r\in R}\sum_{i\in S} \sum_{\substack{a=(u,v):\\ u\in U(i,r)\cap V^t,\\ v\in W(i,r)\cap V^t,\\ \Delta(v,u)<dw_{ti}}} x_{ta} \le maxcanc_t, \qquad t\in T\cup T_{new} \tag{10}$$

$$\sum_{\substack{a=(w,v)\in A^t:\\ w\in W(h,r)\cap V^t,\\ v\in U(i,r)\cap V^t:\\ w\prec w_3, v\succeq v_1}} x_{ta} + \sum_{\substack{a=(w,v)\in A^p:\\ w\in W(h,r)\cap V^p,\\ v\in U(i,r)\cap V^p:\\ w\succeq w_2, v\prec v_4}} x_{pa} \le 1, \qquad t,p\in T\cup T_{new},$$

$$\begin{aligned} r=(h,i)\in R^t\cap R^p,\\ w_1,w_3\in W(h,r)\cap V^t, w_2\in W(h,r)\cap V^p,\\ v_1\in U(i,r)\cap V^t, v_2,v_4\in U(i,r)\cap V^p:\\ w_1\preceq w_2, v_2\preceq v_1,\\ (w_1,v_1)\in A^t, (w_2,v_2)\in A^p,\\ \Delta(w_3,w_2)=d_h, \Delta(v_4,v_1)=a_i \end{aligned} \tag{11}$$

$$x_{ta}, z_{tv}\in\{0,1\}, \qquad t\in T, a\in A^t, v\in V^t, \tag{12}$$

$$y_v \in \{0,1\}, \qquad v \in V. \tag{13}$$

The objective (1) requires to maximize the total profit of all the selected arcs. Recall the definition of the arc profits given in Section 3.1: the profit of the starting arcs corresponds to the train profit p_t and is decreased if shift is applied; the profit of the station arcs is 0 if stretch or stop skipping are not executed along the arc, and is decreased if they are applied. The profit p_t is chosen as a large positive number for every train $t \in T \cup T_{new}$: in this way, the objective aims at maximizing the number of scheduled trains. In addition, since the profit is decreased if shift, stretch and/or stop-skipping are applied, the goal is also to minimize the changes to the existing train timetables and to the desired timetables for the additional trains.

Constraints (2) ensure to select at most one timetable for each train t, by imposing to choose at most one outgoing arc from the source σ. We do not use equality constraints since it might be infeasible to schedule all the trains. Constrains (3) are flow conservation constraints used to guarantee that a path is chosen in the time-space graph G^t for train $t \in T \cup T_{new}$. Constraints (4) and (5) are used to define the auxiliary variables. In constraints (6) we define the headway departure constraints. In particular, for every station $i \in S$ and every track r departing from the station, we define one headway departure constraint for every departure node $w \in W(i,r)$, by imposing to choose at most one departure node v that succeeds w and whose time distance from w is smaller than the minimum required headway time d_i. The headway arrival constraints (7) are defined in a similar way. Constraints (8) ensure to have at most c_i trains simultaneously present at station $i \in S$, where c_i is the capacity of the station. There is one constraint for every station and every time instant $\bar{q}$ in the time horizon $\bar{H}$. The constraint requires to select at most c_i station arcs (u,v), such that the arrival time $\theta(u)$ at station i is before or equal to $\bar{q}$ and the departure time $\theta(v)$ from station i is after or equal to $\bar{q}$, among all arcs reaching station i from every track. In constraints (9) and (10) we impose, for each train $t \in T \cup T_{new}$, a maximum global amount of stretch and a maximum number of cancelled stops, respectively. In constraints (9) we consider all the station arcs that correspond to a stop longer than the minimum dwelling time, while in constraints (10) we consider all the station arcs that correspond to a stop shorter than the minimum dwelling time (i.e. to a stop skipping). Finally, constraints (11) are the constraints to avoid overtaking between a pair of trains travelling between consecutive stations h and i along track r, when acceleration and deceleration times are taken into account. We consider two trains t and p travelling between stations h and i along track r. Train t

departs from node w_1 and arrives at node v_1, while train p departs from w_2 and arrives at v_2, with $w_1 \preceq w_2$. Since $w_1 \preceq w_2, v_2 \preceq v_1$, train t overtakes train p, but this is not allowed, as we are considering one track r between two consecutive stations. Therefore, we can select at most one travel arc between (w_1, v_1) and (w_2, v_2). To strengthen the overtaking constraints, we extend them by including additional incompatible arcs. In particular, we consider w_3 as the first departure time of train t from station h that is feasible with the departure of train p at time w_2 ($\Delta(w_3, w_2) = d_h$), and v_4 as the first arrival time of train p at station i that is feasible with the arrival at time v_1 of train t ($\Delta(v_4, v_1) = a_i$). Then, we include in the constraints all the travel arcs of train t (between stations h and i), such that the train departs before w_3 and arrives at or after v_1, and all the travel arcs of train p (between stations h and i), such that the train departs after or at w_2 and arrives before v_4. The variable domains are specified by constraints (12) and (13).

3.3 Heuristic Algorithm

We generalize the heuristic algorithm proposed in Cacchiani et al. [2010] to deal with the possibility of skipping stops and to manage acceleration and deceleration times. The main changes concern the type of relaxation considered, and the dynamic programming algorithm that is the key element of the solution method, as it is used for determining both the relaxed and the feasible solutions.

The heuristic algorithm is based on relaxing the ILP model as follows. Headway constraints are relaxed in a Lagrangian way: the corresponding Lagrangian multipliers are associated with departure and arrival nodes. Overtaking constraints concerning a slow train j and a fast train k are relaxed in two steps. As observed in Section 3.2, these constraints cannot be expressed by using only departure nodes, as in the case of fixed travel times, but rather by using arcs. However, having expressed the overtaking constraints simply by using the (departure) node variables, considerably speeds up the solution process based on Lagrangian relaxation, as observed in Caprara et al. [2002], since the Lagrangian multipliers can be associated with the departure nodes. Therefore, we first relax these constraints by expressing them only on the departure nodes: similar to the fixed travel time case, we use two sets of departure nodes, say W'_j for the departure nodes of train j and W'_k for those of train k. We must ensure that *any type of travel arc* leaving from a node in W'_j is incompatible with *any type of travel arc* leaving from a node in W'_k, i.e. at most one of these arcs can be selected in a feasible

solution. To this aim we consider for the slow train j the minimum travel time (i.e. the fastest arc) and for the fast train k the minimum travel time plus the acceleration and deceleration times (i.e. the slowest arc), and define the strengthened overtaking constraint based on these two arcs. In Figure 2, we show an example: dotted arrows represent two travel arcs: (w_1, v_1) of the slow train j, and (w_2, v_2) of the fast train k, respectively, while solid arrows correspond to the fastest travel time arc (w_1, v_1') for the slow train j and to the slowest travel time arc (w_2, v_2') for the fast train k. Based on the latter travel arcs, we define nodes w_3 and w_4' (see the dashed arrows). The overtaking constraint impose to select at most one departure node among all the nodes of train j in $W_j' := [w_1, w_3)$ (as before) and all the nodes of train k in $W_k' := [w_2, w_4')$. Clearly, these constraints are less strong than the corresponding ones when no acceleration or deceleration is taken into account. However, they are still expressed by using only departure nodes.

Recall that $l_t^{h,i}$ and $l_p^{h,i}$ represent the minimum travel time of trains t and p, respectively, between stations h and i. In addition, recall that acc_p and dec_p are the acceleration and deceleration times of train p. The overtaking constraints expressed by using only departure nodes read as follows.

$$\begin{aligned}
\sum_{w \in W(h,r) \cap V^t : w_1 \preceq w \prec w_3} z_{tw} + \sum_{w \in W(h,r) \cap V^p : w_2 \preceq w \prec w_4'} z_{pw} \leq 1, \qquad & t, p \in T \cup T_{new}, \\
& r = (h,i) \in R^t \cap R^p, \\
& w_1, w_3 \in W(h,r) \cap V^t, w_2, w_4' \in W(h,r) \cap V^p, \\
& v_1' \in U(i,r) \cap V^t, v_2', v_4 \in U(i,r) \cap V^p : \\
& w_1 \preceq w_2, v_2' \preceq v_1', \\
& (w_1, v_1') \in A^t, (w_2, v_2') \in A^p, \\
& \Delta(v_1', w_1) = l_t^{h,i}, \\
& \Delta(v_2', w_2) = l_p^{h,i} + acc_p + dec_p, \\
& \Delta(v_4, w_4') = l_p^{h,i} + acc_p + dec_p, \\
& \Delta(w_3, w_2) = d_h, \Delta(v_4, v_1') = a_i
\end{aligned} \tag{14}$$

We have one constraint for every pair of trains t and p travelling between consecutive stations h and i along track r. Train t departs from station h at time w_1 and train p at time w_2. As explained above and shown in Figure 2, we consider for train t the fastest travel arc ($\Delta(v_1', w_1) = l_t^{h,i}$) and for train p the slowest one ($\Delta(v_2', w_2) = l_p^{h,i} + acc_p + dec_p$). Since $w_1 \preceq w_2, v_2' \preceq v_1'$, train

t overtakes train p. To strengthen the constraints, we consider w_3 as the first departure time of train t from station h that is feasible with the departure of train p from h at time w_2 ($\Delta(w_3, w_2) = d_h$). In addition, we consider w'_4 as the first departure time of train p such that the corresponding arrival time v_4 at station i is feasible with the arrival at time v'_1 of train t ($\Delta(v_4, v'_1) = a_i$), by considering for train p the slowest travel arc ($\Delta(v_4, w'_4) = l_p^{h,i} + acc_p + dec_p$). The strengthened overtaking constraint imposes to select at most one arc among all the travel arcs of train t from time w_1 up to time w_3 excluded, and of train p from time w_2 up to time w'_4 excluded.

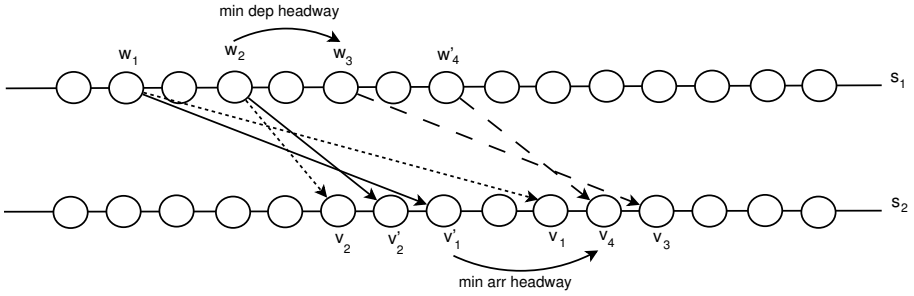

Figure 2: Example of overtaking constraints in the case of acceleration and deceleration times.

In a second step, the overtaking constraints are further relaxed in a Lagrangian way and the corresponding Lagrangian multipliers are associated with the departure nodes of the trains involved in these constraints. Constraints imposing to respect the maximum station capacity are relaxed by elimination: indeed, handling these constraints in a Lagrangian way would lead to have Lagrangian multipliers on the arcs, and we prefer to keep multipliers on the nodes only. The constraints on the maximum number of stops that can be cancelled for each train are also relaxed by elimination, as preliminary computational results showed that they are usually satisfied in the considered real-world instances, and thus do not significantly affect the value of the relaxed problem solution. Finally, recall that the constraints on the maximum stretch and on the acceleration and deceleration times are taken into account in the dynamic programming algorithm, as explained in Section 3.4.

By solving this relaxation, we can compute an upper bound on the value of the objective function. The relaxed problem calls for finding, for each train $t \in T \cup T_{new}$, the maximum Lagrangian profit path in subgraph $G^t = (V^t, A^t)$. In particular, the Lagrangian profit of a path takes into

account the initial profits and the penalties associated with nodes and arcs (as described in Section 3.1) as well as the Lagrangian multipliers associated with the nodes for the headway/overtaking constraint violations. For each train, the maximum Lagrangian profit path is computed by a dynamic programming algorithm, as explained in Section 3.4. By executing, for each train, the dynamic programming algorithm, the optimal solution of the relaxed problem is computed.

Since the number of relaxed constraints is very large, we dynamically generate constraints when they are violated by the current Lagrangian solution, and collect them in a pool. To check if, in the current Lagrangian solution, overtakings occur, we consider every pair of consecutive stations along the track (in each direction) and every pair of trains visiting these stations. We identify the slow train and the fast one of the considered pair. Then, we take for the slow train its fastest arc (i.e., the arc corresponding to the minimum travel time) and for the fast train its slowest arc (i.e., the arc corresponding to the minimum travel time plus the acceleration and deceleration times). If these two arcs still cross, we create a constraint to avoid this overtaking, strengthen it as explained in Section 3.2, and add it to the pool. In order to determine good Lagrangian multipliers, a subgradient optimization procedure is iteratively applied. Lagrangian multipliers are initialized to zero. At each iteration, a new Lagrangian solution is computed, the violated constraints are added to the pool, and the Lagrangian multipliers are updated, taking into account whether the current Lagrangian solution violates or not the constraints in the pool.

At each iteration of the subgradient optimization procedure, we also compute a heuristic solution for the problem. The trains are ordered according to a given criterion: in particular, preliminary experiments suggested that, in the studied application, the best order corresponds to increasing values of the maximum global stretch allowed for the considered trains. Indeed, in our application, the trains have very different values of the maximum global stretch, and using, as in Cacchiani et al. [2010], the order based on decreasing Lagrangian profits leads to schedule a significantly smaller number of trains. After ordering the trains, for each train $t \in T \cup T_{new}$ in the given order, the maximum Lagrangian profit path in G^t is computed (see Section 3.4) by taking into account the nodes occupied by the trains already scheduled. After scheduling a train in the heuristic solution computation, we set as occupied all the nodes in G visited by the train and all the "nearby" nodes that, if visited by a train, would cause violation of one of the imposed (headway, overtaking or station capacity) constraints. Lagrangian profits, instead of the initial arc profits, are considered in the heuristic computation

in order to take into account the effect of scheduling the current train t on the trains, conflicting with t, that still need to be scheduled.

At each iteration of the subgradient optimization procedure, a refinement procedure is applied to the obtained heuristic solution: it consists of a local search procedure that tries to improve, in turn, the path of a train t, which received shift and/or stretch changes or was not scheduled. All the train paths, except that of t, are kept as fixed and the maximum profit path is recomputed, this time considering the initial arc profits (instead of the Lagrangian ones). If a better path for train t is found, it replaces the previous one in the current solution, otherwise the next train is considered.

3.4 Dynamic Programming Algorithm

The maximum profit paths of the trains (both considering the Lagrangian or the initial profits) are computed by a dynamic programming algorithm. When the dynamic programming algorithm is used to determine a heuristic solution, we check that the constraints relaxed by elimination, namely the station capacity constraints and the constraints on the maximum number of cancelled stops, are satisfied. If the station capacity constraints are not satisfied, the computed path is neglected and the corresponding train is cancelled. If the constraint on the maximum number of cancelled stops is not respected, we recompute a path for the same train without allowing stop skipping. When the dynamic programming algorithm is used to solve the relaxed problem, the optimal solution is determined, as we take into account in the computation both the maximum stretch $maxstr_t$, for every train $t \in T \cup T_{new}$, and the possibility of choosing whether to stop or not at a station. Note that, in a standard dynamic programming procedure to compute a maximum-profit path from σ to τ in G^t, it is enough to store, for each node v, a label with the maximum profit of a path from σ to v, along with the predecessor of v along this path to be able to reconstruct it. When a maximum stretch is imposed for every train, it is necessary to know the stretch accumulated at a node v by a path from σ to v. To this aim, in Cacchiani et al. [2010], for each train t, each node $v \in V^t$ is associated with an array of labels $lab(v, str)$, one for each value str of the stretch between 0 and $maxstr_t$. Each label $lab(v, str)$ of node $v \in V^t$ stores the maximum profit of a path reaching node v with global stretch str, along with the predecessor of node v in the path (used to reconstruct the path). One label for each value of the stretch str (with $0 \leq str \leq maxstr_t$) is needed, as the constraint on the maximum stretch is referred to the global amount of stretch accumulated by the train along its path. On the contrary, the maximum shift

constraint is managed by considering only the subset of departure nodes from the origin station of the train that respect the maximum shift. Note that in order to consider, in the dynamic programming algorithm, the constraints on the maximum number of stops that can be cancelled, we would need to know the number of cancelled stops at a node v in a path from σ to τ (as it happens for the maximum stretch). Therefore, an additional level of labels would be required for each value of the number of cancelled stops between 0 and $maxcanc_t$. Since preliminary computational results showed that these constraints are usually satisfied for the considered real-world instances, we decided to relax them by elimination in the relaxed problem and handle them in a heuristic way in the computation of the heuristic solution (as explained above).

When we consider the possibility of choosing whether to stop or not at a station and we take into account acceleration and deceleration times, an additional level of labels has to be included in the dynamic programming algorithm (*stop* or *not stop*). More precisely, we need to have, for each node $v \in V^t$ of train t, an array of labels (one for each value of the stretch between 0 and $maxstr_t$) *both* for the choice of stopping and for the choice of not stopping at the station associated with node v. Therefore, a label $lab(v, str, status)$ is characterized by the node v, the global amount of stretch str accumulated by the train along its path from σ to v and the $status$, i.e. stop or not stop at the station corresponding to node v. The label $lab(v, str, stop)$ ($lab(v, str, notstop)$, resp.) stores the maximum profit of a path reaching node v with global stretch str by stopping (not stopping, resp.) at node v, along with the predecessor of node v in the path (used to reconstruct the path).

Clearly, this addition not only complicates the dynamic programming algorithm, but also requires a longer computing time. However, it is important to consider this additional level of labels, as shown in Figure 3. Two alternative paths of the same train, departing from station s_1, visiting station s_2 and arriving at station s_3, are shown. The dashed path corresponds to a stop at station s_2, and the solid one corresponds to not stop there. Suppose that both paths have zero stretch. The gray node of station s_2 belongs to both paths and can be labelled by both its predecessor nodes. When stopping, deceleration is required to reach the gray node and acceleration is required to leave it. Let us suppose that the profit of the fastest (solid) path is higher than that of the slowest (dashed) one. Let us also suppose that the node of station s_3 shown in black cannot be used: for example, because this node is occupied by a previously scheduled train (this can happen in a heuristic solution computation) or has a very large Lagrangian penalty (this

can happen in a Lagrangian solution computation). If we do not consider the additional level of labels "to stop or not to stop", only the fastest (solid) path can be stored at the gray node, since it has a higher profit, but actually this path turns out to be infeasible, since the black node cannot be used. Therefore, we need to store both paths. Note that, in general, several intermediate stations are visited by the train, and the choice of stopping or not has to be done at each of them. These choices can lead to feasible or infeasible paths and to paths with different profits. Thus, in each node, we take into account two arrays of labels, one for the condition of stopping and the other one for not stopping at the station corresponding to the node. Each array has, as in Cacchiani et al. [2010], one label for each possible value of the stretch.

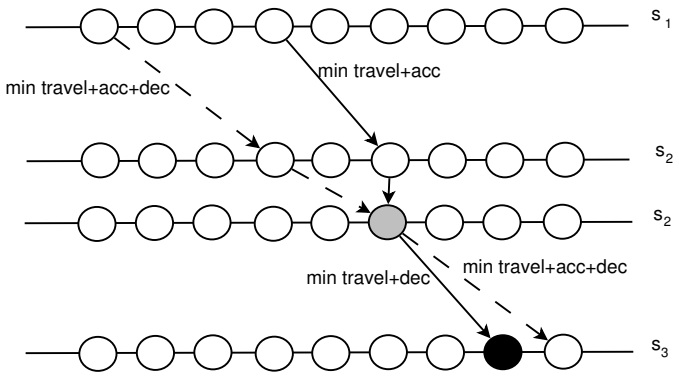

Figure 3: Example showing that it is important to have the additional level of labels.

Given these arrays of labels, we next explain how the labelling operation is executed for the travel arcs and for the station arcs. Consider the computation of the maximum profit path for a given train t. When labelling from a departure node v one of its successors u, we now have four types of travel arcs (v, u). We need to verify which labels of v can be extended to which labels of u and by using which types of travel arcs, since each label is associated with a stop or to a not stop. Four examples, one for each type of travel arc, are illustrated in Figure 4. Let s_1 and s_2 be two consecutive stations and (v, u) a travel arc between these stations. We have two arrays of labels (with values of the stretch from 0 to $maxstr_t$) corresponding to the condition of "stop" or "not stop" both for the departure node v and for the arrival node u. We show in gray the condition stop/not stop of the array of labels of v that we want to extend, and also in gray the condition

stop/not stop of the array of labels of u that can be labelled according to the considered travel arc. When we consider the travel arc with minimum travel time (upper left part of Figure 4), we can only extend a label of v corresponding to a not stop to a label of u also corresponding to a not stop: indeed, no acceleration or deceleration times are taken into account. When the travel arc with minimum travel time plus acceleration time (upper right part of Figure 4) is considered, we can only extend a label of v corresponding to a stop to a label of u corresponding to a not stop. The opposite happens when we consider a travel arc with minimum travel time plus deceleration time (lower left part of Figure 4): indeed, we can only extend a label of v corresponding to a not stop to a label of u corresponding to a stop. Finally, when the travel arc with minimum travel time plus acceleration and deceleration times is considered (lower right part of Figure 4), we can extend a label of v corresponding to a stop to a label of u also corresponding to a stop.

In other words, along travel arcs, labels can be extended by maintaining the status of stop (or not stop, resp.), if the arc corresponds to the minimum travel time (or to the minimum travel time plus acceleration and deceleration times, resp.), while, for the other arcs, it can be extended by changing the status (from stop to not stop if the arc corresponds to the minimum travel time plus acceleration time, or vice versa if the arc corresponds to the minimum travel time plus deceleration time).

The labelling related to the station arcs is relatively easier. Two examples, one for stopping and one for stop skipping, are shown in Figure 5. The figure shows station s_1 and a station arc (v, u). Both for the arrival node v and for the departure node u we show two arrays of labels as before, by indicating in gray how the status is extended. When we consider the station arc with minimum stopping time (left part of Figure 5), we can only extend a label of v corresponding to a stop to a label of u also corresponding to a stop. The same holds for station arcs corresponding to longer stops: for these arcs, the value of the stretch, say str', consumed at this station will be summed to the current value of the stretch, say str, and a label $lab(v, str)$ (corresponding to a stop) will be extended to a label $lab(u, str + str')$ (corresponding to a stop). On the contrary, when we consider the station arc corresponding to stop skipping (right part of Figure 5), we can only extend a label of v corresponding to a not stop to a label of u also corresponding to a not stop. In other words, along station arcs, the labels can be extended only maintaining the status of stop or not stop.

In Figure 6, we report the pseudocode of the dynamic programming algorithm for computing the maximum profit path for train $t \in T \cup T_{new}$

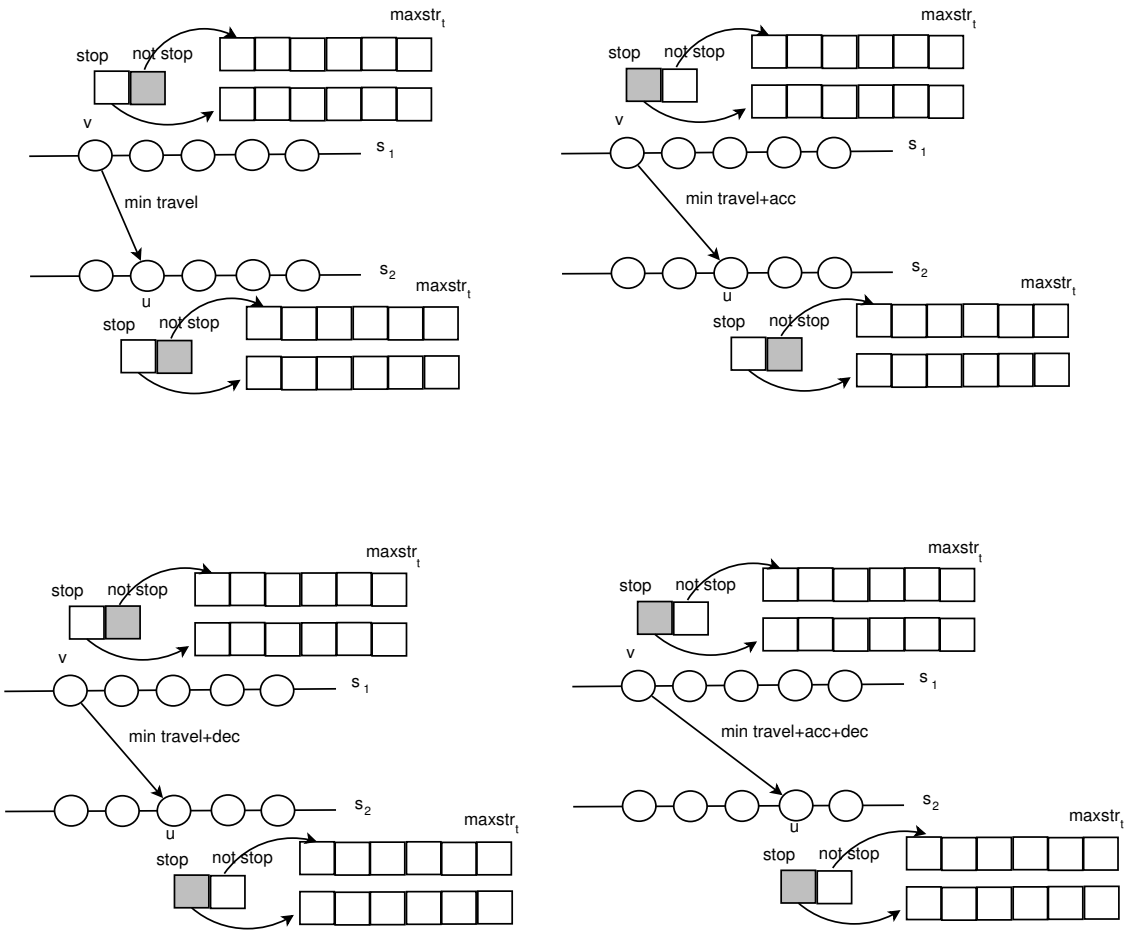

Figure 4: Example of labelling for travel arcs.

in order to solve the Lagrangian relaxed problem. In Figures 7 and 8, we show the pseudocodes of the procedures used by the dynamic programming algorithm for initializing the labels and for labelling a node, respectively.

Let us consider a train $t \in T \cup T_{new}$. Recall that each node is associated with two arrays of labels (one for the stop case and one of for the not stop case), each with length equal to $maxstr_t$. In addition, recall the definition of $lab(v, str, status)$. The first step of the dynamic programming algorithm is to initialize all the labels for all the nodes in $V^t \cup \{\tau\}$. This is done by executing the procedure INITIALIZE LABELS, shown in Figure 7. This procedure scans all the nodes $v \in V^t \cup \{\tau\}$ and sets the profit of each label in the two arrays equal to 0, unless v is the endpoint of a starting arc and

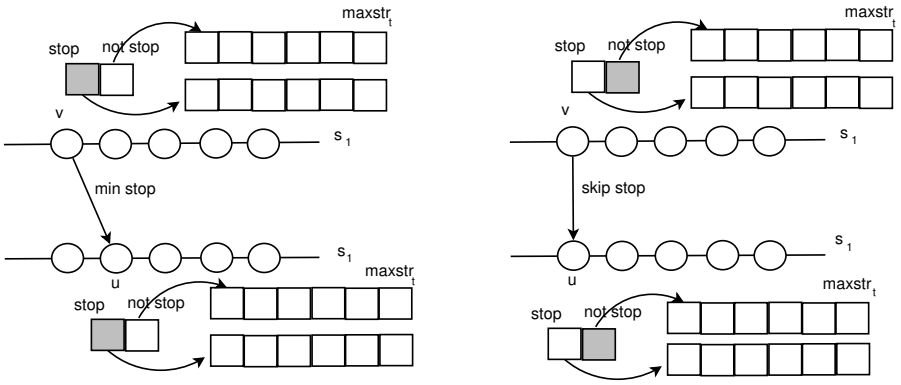

Figure 5: Example of labelling for station arcs.

the stretch associated with the label is 0. In this case, the profit is equal to the train profit decreased by the shift of node v. The predecessor of the label is set equal to σ (or to NONE if the profit is 0).

Once that the labels have been initialized, the nodes $v \in V^t$ are scanned in chronological order. For each node v, for each value of stretch str up to $maxstr_t$, and for the two cases of the status (stop or not stop), all the successor $succ$ nodes of node v are considered (in chronological order). Let $lab_v = (v, str, status)$ be the label of node v with associated stretch str and status $status$. Then, three cases can occur: the first one is that $(v, succ)$ is a travel arc, the second one is that $(v, succ)$ is a station arc, and the third case is that $(v, succ)$ is an ending arc. Let h and i denote, respectively, the stations associated with nodes v and $succ$.

If $(v, succ)$ is a travel arc, one of the four cases shown in Figure 4 occurs, and the labelling is applied accordingly. For example (see the upper left part of Figure 4), if the time difference $\Delta(succ, v)$ between node $succ$ and node v is equal to the minimum travel time $l_t^{h,i}$ of train t between station h and station i, and the $status$ is not stop, then the label $lab_{succ} := (succ, str, notstop)$ of node $succ$ has to be labelled. A similar behavior happens for the other cases shown in Figure 4.

If $(v, succ)$ is a station arc, then the train can stop for its minimum dwelling time dw_{th} (see the left part of Figure 5), can skip the stop (see the right part of Figure 5), or it can stop for a longer time str' provided that $str + str' \leq maxstr_t$. For example, if the train stops for a longer time str' at station h, i.e. if the time difference $\Delta(succ, v)$ between node $succ$ and node v is equal to the minimum dwelling time dw_{th} of train t at station h plus str', then the label $lab_{succ} := (succ, str + str', stop)$ of node $succ$ has to

be labelled. A similar behavior happens for the other cases shown in Figure 5.

If $(v, succ)$ is an ending arc, then node τ has to be labelled. The labelling is done by executing procedure LABEL(lab_v,lab_{succ}), shown in Figure 8. This procedure updates the profit of lab_{succ} and its predecessor $pred(lab_{succ})$, if the profit of lab_v is greater than the profit of lab_{succ} minus the cost of $(v, succ)$ and $pen(v)$, i.e. the Lagrangian penalty of node v. The cost is 0, unless $(v, succ)$ is a station arc that corresponds to skipping the stop (and in this case the cost is equal to the stop skipping penalty) or it is a station arc that corresponds to a longer stop than the minimum dwelling time (and in this case the cost is equal to the stretch penalty times the stretch str' of arc $(v, succ)$). The penalty $pen(v)$ takes into account the violation or satisfaction of the constraints, relaxed in a Lagrangian way, that involve node v.

Once that all the loops have been executed, the dynamic programming algorithm finds the best label among all the labels of node τ. If all the labels of node τ have profit 0, it means that no path has been found for train t and the train cannot be scheduled. Otherwise, the maximum profit path for train t is reconstructed by following the predecessor of each label until node σ is reached. The station capacity is checked at each station visited by the path and if it is not respected in at least one station, the path is neglected and the train is not scheduled. Otherwise, the number of cancelled stops in the path is counted and, if it is smaller or equal to $maxcanc_t$, the maximum profit path is returned by the dynamic programming algorithm. If it is larger than $maxcanc_t$, the path is recomputed without allowing stop skipping. Note that if the algorithm is used for computing a heuristic solution, after computing a path for a train t, the graphs of the remaining trains need to be updated by removing all the nodes and the arcs that cannot be used, as they would cause constraint violation. Finally, in the refinement procedure, since the initial profits are considered instead of the Lagrangian ones, $pen(v)$ is set to 0 for all nodes $v \in V^t$.

4 Computational Experiments on the JingHu High-speed Corridor

To show the performance of the proposed method, we consider real-world instances of the double-track JingHu line between Beijing and Shanghai. The timetable data are provided by the National Railway Train Diagram Research and Training Center (Southwest Jiaotong University, Chengdu,

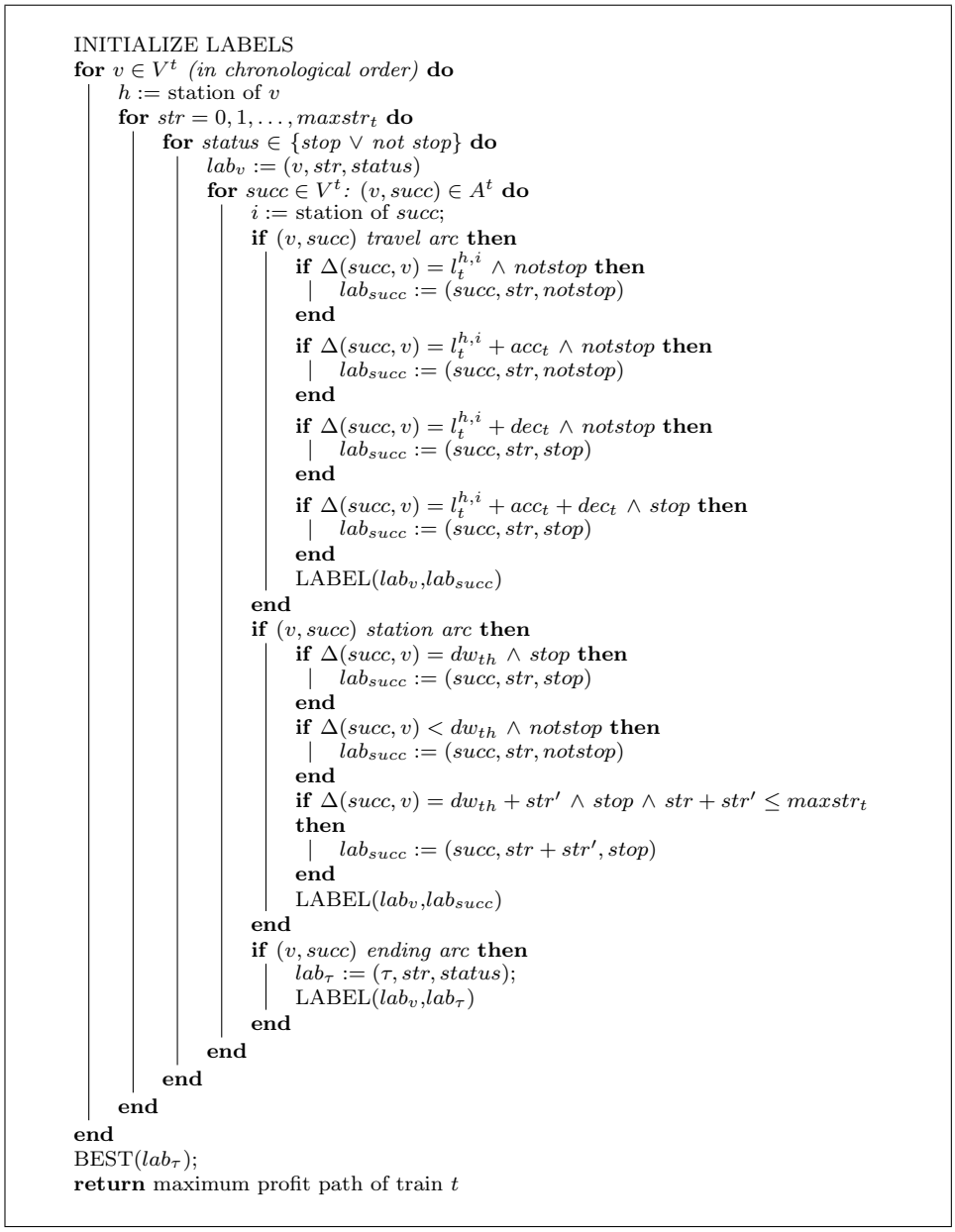

Figure 6: Pseudocode for the dynamic programming algorithm for train t ($t \in T \cup T_{new}$).

```
for v ∈ V^t ∪ {τ} do
    for str = 0, 1, ..., maxstr_t do
        for status ∈ {stop ∨ not stop} do
            if (σ, v) ∈ A^t ∧ str = 0 then
                lab_v := (v, 0, status);
                profit(lab_v):=p_t − psh_t ∗ ν_t(v)
                pred(lab_v):=σ
            end
            else
                profit(lab_v):=0
                pred(lab_v):=NONE
            end
        end
    end
end
```

Figure 7: Pseudocode for the INITIALIZE LABELS procedure.

```
//lab_v := (v, str, status)
//lab_succ := (succ, str', status')
cost:= 0;
if (v, succ) station arc then
    h := station of v
    if Δ(succ, v) < dw_th then
        cost:=pskip_t
    end
    if Δ(succ, v) = dw_th + str' then
        cost:=pstr_t ∗ str'
    end
end
if profit(lab_v)-cost(v, succ)-pen(v)>profit(lab_succ) then
    profit(lab_succ):=profit(lab_v)-cost(v, succ)-pen(v)
    pred(lab_succ):=lab_v
end
```

Figure 8: Pseudocode for the LABEL(lab_v,lab_{succ}) procedure.

China). The high-speed line contains 29 stations, including 6 block sectors connecting the JungHu line to the other lines. In Figure 9 we report a map of the line.

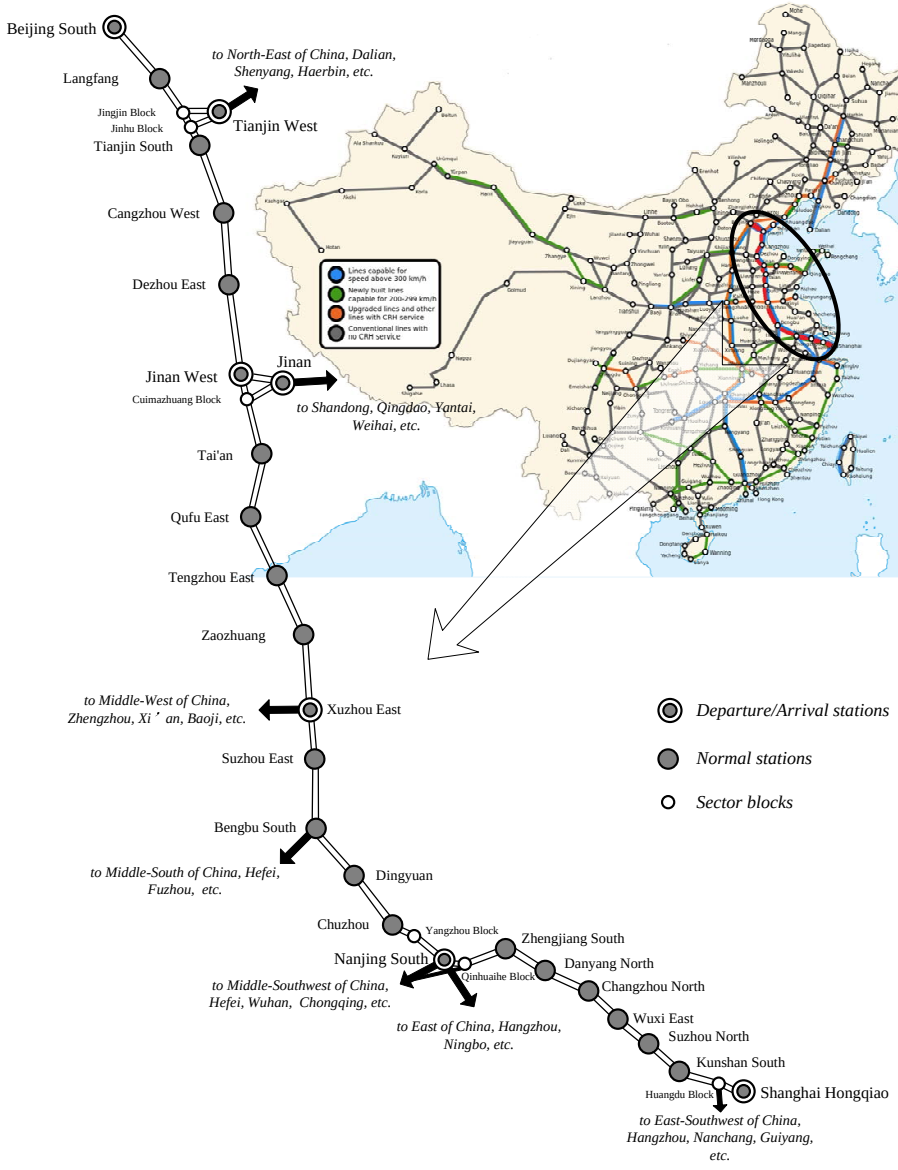

Figure 9: JingHu High-speed Corridor.

We developed the solution method in C and tested it on a personal computer with a i5-2400 3.1 GHz, 16 GB RAM. Time is discretized in minutes. We refer to the timetable of 2015 in which 304 trains run every day in both directions between 6 a.m. and midnight. The station capacities are set equal to the number of existing platforms. The trains are of two different categories: D trains with travel speed of 250 km/h and G trains with travel speed of 300 km/h. For the D trains we consider an acceleration time of 2 minutes and a deceleration time of 2 minutes, while for the G trains the acceleration time is 2 minutes and the deceleration time is 3 minutes. The minimum headway times are different according to the station and can be 2, 3 or 4 minutes. In particular, three different railway bureaus are responsible for the JingHu line: the Beijing railway bureau is responsible for the stations between Beijing Southwest and Dezhou East and requires minimum departure and arrival headways of 3 and 4 minutes, respectively; the Jinan railway bureau is responsible for the stations between Jinan West and Zaozhuang and requires minimum departure and arrival headways of 2 and 3 minutes, respectively; finally, the Shanghai railway bureau is responsible for the stations between Xuzhou East and Shanghai Hongqiao and requires minimum departure and arrival headways of 2 and 3 minutes, respectively.

From the 2015 timetable, we know the minimum travel times of the existing trains. We also know the stopping times, however, they can be larger than the corresponding minimum ones, due to the changes that the operator has done during the scheduling process. We set the minimum dwelling time to 2 minutes unless it was set to 1 minute in the existing timetable.

In Section 4.1 we report the results obtained on the 2015 timetable by considering the parameters suggested by the operator. Section 4.2 presents the results obtained with different parameter settings. Finally, in Section 4.3, we show the performance of the proposed algorithm on a larger instance.

4.1 Results with the Operator Parameters

To meet the suggestions of the operator, for all the trains, the shift penalty is set to 1 for each minute of shift, while the stretch penalty is set to 10 for each minute of stretch. Thus, we largely penalize longer stops at stations, since these changes more significantly affect the passenger trip. In addition, the penalty for stop skipping is set to 200, so as to have a good trade-off between allowing to skip stops and avoiding that too many stops are cancelled. After parameter tuning, the number of subgradient iterations is set to 50 so as to have a good trade-off between solution quality and

computing time. We emphasize that the shift, stretch and stop skipping penalties are set based on the requirements of the operator. However the operator does not need to decide the specific values of the penalties but rather to choose which type of change (shift, stretch or stop skipping) is less or more desirable. The operator suggested us that shift, stretch and stop skipping should be penalized according to increasing penalties. The reason for penalizing stretch more than shift is that stretch causes an increase of the passenger travel time (due to the additional dwelling times) that is usually perceived in a negative way by the passengers. On the contrary, shift consists of changing the train departure time, while respecting the departure time windows of the trains, and thus it only slightly influences passenger satisfaction. Finally, the highest penalty is associated with stop skipping, since this is a major change that should be used only if it allows to schedule additional trains. Following these directions, we set shift, stretch and stop skipping penalties as described above. In Section 4.2, we report the results obtained with different numbers of iterations and different values for the penalties.

Our first experiment concerns the improvement of the current timetable to reduce the global stretch of all the trains. We want to determine an optimized timetable with the proposed method. The schedule of 114 trains (fixed trains) out of the given 304 trains cannot be changed. Therefore, we impose their schedules as in the current timetable. We allow, for each other existing train, a maximum stretch equal to the stretch it has undergone in the existing timetable. The maximum stretch ranges between 1 and 64 minutes, and the average maximum stretch is 5.3 minutes. In addition, we impose a maximum shift of 0 minutes, as we want to keep the computed schedule as close as possible to the existing one. Each train profit of the non-fixed trains is set to 5,000 in order to have a high possibility of scheduling all the trains. The fixed trains are assigned a profit of value 0, as they are always scheduled. In Table 5 we evaluate the quality of the existing schedule and compare it with the results obtained by the proposed method in two cases: in the first one, stop skipping is not allowed, while it is allowed in the second one. In the latter case, for each of the 190 non-fixed trains, $maxcanc_t$ is set equal to 1. In the table, the first column indicates the considered instance (where "exist" means the 2015 timetable), the second column is used to define if stop skipping is allowed or not, the third column reports the number of scheduled trains. Then, we report the total travel time (expressed in minutes, and corresponding to the sum, over all the non-fixed trains, of the differences between the arrival time at destination and the departure time from the origin), the total stretch (i.e. the total additional stopping time

of the non-fixed trains expressed in minutes) and the total profit. Finally, we show the percentage gap computed as $(UB - LB)/UB * 100$ (where UB corresponds to the upper bound computed by solving the Lagrangian relaxation, and LB is the profit of the best heuristic solution found), the number of skipped stops and the computing time in seconds.

#trains	skip	#sched	travel	stretch	profit	gap%	#skip	time
304 exist	no	304	46019	1012	939880	-	0	-
304	no	304	45797	709	942910	0.36	0	62
304	yes	304	45716	638	943020	0.37	3	190

Table 5: Comparison of the 2015 timetable with the computed timetable without or with the possibility of skipping stops.

In all the cases, all the given 304 trains are scheduled. Both the total travel time and the total stretch are reduced by the proposed heuristic algorithm with respect to the corresponding values of the existing timetable. While the total travel time decreases of less than 0.5% when stop skipping is not allowed, and 0.66% when it is allowed, the total stretch is significantly reduced of about 30% when stop skipping is not allowed and about 37% when it is allowed. The percentage gaps between the upper and the lower bounds are small, which means that the solutions obtained by the proposed method are very close to the optimal ones. The computing times are very short.

The second set of experiments deals with the scheduling of additional trains. We are given by the operator a set of 42 new trains that should be scheduled in the JingHu line. These trains are of type D. For each train, we are given its origin and destination stations, the set of stations to be visited with the corresponding minimum dwelling times, and the desired departure time from the origin station. In addition, for each train we know the minimum travel time between each pair of consecutive stations visited by the train in the corridor. Each of these trains is assigned a profit equal to 2,000 so that it is always better to schedule one of the existing trains than one of the new trains. The maximum global stretch of each train depends on the total distance from the origin station to the destination station. The maximum allowed global stretch is between 27 and 131 minutes, and its average is 70.8 minutes, i.e., a much larger change is allowed for the additional trains than for the existing ones. We consider different possibilities for the maximum shift allowed for each of the new trains: ± 10, ± 20 or ± 30 minutes. The parameters for the existing trains are set as in the previous experiment (and 114 trains are kept as fixed). In this way, we are still allowed to improve

the schedule of the existing trains, while trying to schedule the additional ones. In Table 6, we report the results of the timetables computed by the proposed heuristic algorithm when stop skipping is not allowed. The first column shows the three considered settings for the maximum shift allowed. Column "shift" reports the total shift used in the solution, and column "#sched" the number of scheduled trains, where in brackets we report the number of existing trains cancelled. In column "travel" we report the sum of the travel times for the non-fixed existing trains, so that we can compare it with the results reported in Table 5. In column "stretch", we report the total stretch of all the trains in the computed solution, and in brackets the total stretch only for the non-fixed existing trains (to compare it with the results in Table 5). Then, we show the total profit, the percentage gap between the upper and the lower bounds and the computing time in seconds.

#trains	shift	#sched	travel	stretch	profit	gap%	time
346 sh±10	109	328(0)	45829	1132(737)	986571	3.72	3857
346 sh±20	294	333(0)	45827	1142(740)	996286	2.98	6153
346 sh±30	415	336(1)	45681	1161(689)	998975	2.95	9732

Table 6: Results for the scheduling of the additional trains, when stop skipping is not allowed.

From Table 6 we can observe that solving the problem becomes more complex when additional trains need to be scheduled in an already very congested line. In addition, the new trains are allowed to have more flexibility for what concerns both the shift and the stretch operations. Therefore, we need to deal with larger time-space graphs, and the dynamic programming algorithm requires a longer computing time. We can see that additional trains can be scheduled in all the cases, and that this number increases as long as we allow to use larger departure time windows. Only for the last instance one existing train was cancelled, but the overall profit is increased. Finally, we can see that the percentage gaps, even though larger than in the previous experiment, are always below 4%. When stop skipping is not allowed, by comparing the total travel time and the total stretch of the existing trains with those reported in Table 5, we can see that larger travel times and stretches are required (except when one existing train is cancelled). Indeed, the insertion of new trains affects the existing timetable and more changes need to be performed.

We next show the results obtained by allowing stop skipping. We set the parameters for the existing and new trains as for the results presented in Table 6. The results are reported in Table 7, where the meaning of the

columns is the same as in Table 6.

#trains	shift	#sched	travel	stretch	profit	gap%	#skip	time
346 sh±10	115	329(0)	45756	1096(662)	988525	3.66	2	4969
346 sh±20	279	334(0)	45731	1113(648)	997991	2.86	3	7510
346 sh±30	415	337(0)	45752	1192(664)	1003265	2.55	2	11112

Table 7: Results for the scheduling of the additional trains, when stop skipping is allowed (with stop skipping penalty equal to 200).

As shown in Table 7, the computing time increases as additional flexibility is available, because we are allowed to skip stops. The number of cancelled stops is very small (at most 3), even though the stop skipping enables to schedule additional trains. The overall profits increase with respect to those reported in Table 6, and the percentage gaps are still below 4%. We can also see that not only the number of scheduled trains is larger than that shown in Table 6, but also no existing train has been cancelled.

4.2 Results with Different Parameter Values

In this section, we report the results obtained with different parameter settings. In particular, we evaluate the performance of the proposed heuristic algorithm by considering the following cases: (i) different number of iterations, (ii) different values for the shift and stretch penalties, and (iii) different values for the stop skipping penalty.

We performed a set of experiments by considering different numbers of iterations of the proposed algorithm when stop cancellation is forbidden or allowed. The results are shown in Table 8, where we report the maximum shift allowed, the number of iterations considered, and, both when stop cancellation is forbidden or allowed, the number of scheduled trains (with in brackets the number of existing trains cancelled), the profit, the percentage gap and the computing time (in seconds). In addition, we report the number of cancelled stops when stop skipping is allowed. As expected, the gap reduces while the computing time increases with the number of iterations. When 200 iterations are executed, the percentage gap are rather small, slightly larger for the case in which stop skipping is permitted. Since the computing time is proportional to the number of iterations, one can choose the appropriate number based on the available computing time.

In Table 9, we report the results obtained with different values for the shift and stretch penalties, when stop skipping is not allowed, and compare them with those reported in Table 6. In particular, we consider $psh_t = 0$ and

		no skip				skip				
#trains	#iter	#sch	profit	gap	time	#sch	profit	gap	#sk	time
346 sh±10	10	326(1)	979998	4.81	821	327(1)	982100	4.62	3	891
	50	328(0)	986571	3.72	3857	329(0)	988525	3.66	2	4969
	100	328(0)	986763	3.46	7977	329(0)	988525	3.51	2	10287
	150	329(0)	988265	2.87	12107	330(0)	988742	3.59	13	15429
	200	329(0)	988265	2.57	16169	331(0)	990911	3.11	11	20722
346 sh±20	10	332(2)	987505	4.15	1226	334(2)	991708	3.73	4	1146
	50	333(0)	996286	2.98	6153	334(0)	997991	2.86	3	7510
	100	334(0)	997624	2.71	12847	334(0)	997991	2.76	3	16361
	150	334(0)	997624	2.42	19422	334(0)	997991	2.69	3	25246
	200	334(0)	997624	2.11	25971	334(0)	997991	2.65	3	34165
346 sh±30	10	334(1)	994924	3.60	1702	334(1)	995156	3.58	3	1727
	50	336(1)	998975	2.95	9732	337(0)	1003265	2.55	2	11112
	100	337(0)	1002884	2.44	19310	337(0)	1003265	2.46	2	23167
	150	337(0)	1002884	2.14	29272	337(0)	1003265	2.41	2	35245
	200	338(0)	1004868	1.90	39009	337(0)	1003265	2.39	2	47794

Table 8: Results after different numbers of iterations.

$pstr_t = 0$ ($t \in T \cup T_{new}$), that corresponds to allow shift and stretch without any penalty, $psh_t = 5$ and $pstr_t = 5$ ($t \in T \cup T_{new}$), that corresponds to give the same importance to both changes, and the case $psh_t = 10$ and $pstr_t = 1$ ($t \in T \cup T_{new}$), that corresponds to consider shift as more penalized than stretch. Note that the profits shown in Table 9 can only be compared when the same values of the shift and stretch penalties are used, since the values of the penalties affect the value of the obtained profit.

The results show that small percentage gaps between upper and lower bounds are obtained for all these cases. When no penalty is associated with shift and stretch ($psh_t = 0$, $pstr_t = 0$), larger shift and stretch changes are applied to the timetables, and a larger number of trains is scheduled. In addition, the computing times are shorter, since the problem becomes easier. When the same penalty values are used for shift and stretch ($psh_t = 5$, $pstr_t = 5$), the number of scheduled trains and the computing times are comparable to the case reported in Table 6, while smaller shift and larger stretch changes are applied to the timetables. Finally, when a large penalty is associated with the shift and a small one with the stretch ($psh_t = 10$, $pstr_t = 1$), we can see that the shift is significantly reduced, while the stretch becomes larger. In this case, the number of scheduled trains is larger than that in Table 6, meaning that if we accept to have a larger stretch in the timetable, additional trains can be scheduled. Thus, depending on the specific suggestions of the operator, different penalties can be associated with the shift and stretch changes, and the proposed algorithm is able to effectively adapt to the different settings.

$psh_t = 0$, $pstr_t = 0$							
#trains	shift	#sched	travel	stretch	profit	gap%	time
346 sh±10	143	329(0)	45854	1298(776)	1000000	3.08	2099
346 sh±20	392	338(0)	45877	1636(800)	1018000	1.36	3368
346 sh±30	521	340(0)	45855	1796(783)	1022000	1.16	5143
$psh_t = 5$, $pstr_t = 5$							
#trains	shift	#sched	travel	stretch	profit	gap%	time
346 sh±10	103	328(1)	45548	1094(709)	989015	3.70	3902
346 sh±20	253	335(1)	45691	1222(703)	1001625	2.57	6306
346 sh±30	304	337(1)	45588	1390(724)	1004530	2.46	9615
$psh_t = 10$, $pstr_t = 1$							
#trains	shift	#sched	travel	stretch	profit	gap%	time
346 sh±10	75	329(0)	45805	1201(718)	998049	3.06	4035
346 sh±20	163	335(1)	45554	1431(702)	1005939	2.33	6498
346 sh±30	227	338(1)	45590	1559(687)	1011171	1.97	9654

Table 9: Results for different shift and stretch penalties, when stop skipping is not allowed.

In Table 10, we report the results obtained with different values for the stop skipping penalty, and compare them with those reported in Table 7. In particular, we consider $pskip_t = 150$, $t \in T \cup T_{new}$, and $pskip_t = 250$, $t \in T \cup T_{new}$. Note that the profits shown in Table 10 can only be compared when the same value of stop skipping penalty is used, since the values of the penalty affects the value of the obtained profit.

As we can see, the total shift, the total stretch and the number of cancelled stops are comparable for all the considered values of $pskip_t$. By comparing the results obtained with $pskip_t = 150$ with those reported in Table 7 (where $pskip_t = 200$), we can observe that the global number of scheduled trains is increased by one when the maximum shift is ±10, but an existing train is cancelled. When the maximum shift is ±20, the same global number of trains is scheduled, but an existing train is cancelled, and when the maximum shift is ±30, a smaller number of trains is scheduled (335 trains instead of 337). By comparing the results obtained with $pskip_t = 250$ with those reported in Table 7, we can see that, when the maximum shift is ±10, one train less is scheduled, while, with maximum shift ±20 or ±30, the same number of trains is scheduled. In all the considered cases, the obtained percentage gaps are rather small and the computing times are similar to those shown in Table 7. The results reported in Table 10 suggest that the best parameter choice for the considered real-world application is $pskip_t = 200$ ($t \in T \cup T_{new}$), since it corresponds to the best performance by considering different values of the maximum shift.

$pskip_t = 150$								
#trains	shift	#sched	travel	stretch	profit	gap%	#skip	time
346 sh±10	129	330(1)	45482	1082(629)	987151	3.83	6	5678
346 sh±20	298	334(1)	45387	1139(649)	994712	3.20	4	9291
346 sh±30	361	335(0)	45740	1142(673)	999619	2.94	4	12439
$pskip_t = 250$								
#trains	shift	#sched	travel	stretch	profit	gap%	#skip	time
346 sh±10	114	328(0)	45763	1069(674)	985946	3.88	5	5026
346 sh±20	283	334(0)	45768	1145(693)	997767	2.89	2	7673
346 sh±30	397	337(0)	45764	1172(689)	1003383	2.53	2	11623

Table 10: Results for different stop skipping penalties.

By considering the presented analysis of the performance of the algorithm with different values of the parameters, we can conclude that, based on the available computing time and on the choice of the operator on the inconvenience caused to the passengers if shift, stretch or stop skipping are performed, we can set a higher or lower number of iterations, and higher or lower penalty values for psh_t, $pstr_t$ and $pskip_t$ ($t \in T \cup T_{new}$), and the algorithm will effectively compute a heuristic solution, while taking into account the given parameter values.

4.3 A larger instance

In order to further evaluate the performance of the proposed algorithm, we generated a larger instance and tested it when stop skipping is forbidden or allowed. This instance contains the 304 existing trains, the 42 additional trains given by the operator, and other 42 trains with the same desired timetable as the 42 additional trains but shifted of one hour later. Since one of these trains cannot be scheduled because its timetable is during the maintenance period, we end up with an instance containing 387 trains. We consider the same parameters used for the results shown in Table 6, and set $pskip_t = 100$ and $maxcanc_t = 4$, since this is a larger instance and more flexibility is needed in order to schedule additional trains. The obtained results are reported in Table 11. As we can see, the computing times increase, as we are dealing with a larger number of trains, but are still acceptable for planning purposes. Although the average percentage gaps are larger than those reported in Tables 6 and 7, we can see that several additional trains can be scheduled, and that stop skipping is very effective, as it allows the scheduling of more trains and the cancellation of a smaller number of existing trains. Therefore, we can conclude that the proposed algorithm can

be used for timetable planning in practical applications even when a large number of trains is involved.

no skip								
#trains	shift	#sched	travel	stretch	profit	gap%	time	
387 sh±10	256	352(1)	45692	1546(708)	1027284	6.68	6556	
387 sh±20	546	361(3)	45405	1706(669)	1037394	6.18	11618	
387 sh±30	848	361(2)	45442	1610(686)	1041052	6.16	18506	
skip								
#trains	shift	#sched	travel	stretch	profit	gap%	#skip	time
387 sh±10	296	356(2)	45070	1491(620)	1029994	6.68	28	8763
387 sh±20	551	361(1)	45283	1568(623)	1041969	5.65	28	15308
387 sh±30	795	362(1)	45591	1574(632)	1045265	5.83	12	24307

Table 11: Results for the scheduling of the additional trains with a larger instance (83 additional trains).

5 Conclusions

We have studied a real-world application of Train Timetabling in which we are allowed to stop at additional stations or to skip stops (Stop Planning) in order to schedule additional trains along a double-track line. In addition, we have explicitly taken into account the deceleration and acceleration times, associated with the stop of a train at a station. We have extended an existing heuristic method, based on Lagrangian relaxation, to deal with these additional features. In particular, several changes were performed to model the overtaking constraints, and the solution method was modified both in the way to relax these constraints and in the dynamic programming algorithm. The proposed method has been tested on a real-world instance of the JingHu corridor connecting Beijing and Shanghai. The obtained results show that the proposed method is able to improve the current timetable within very short computing times, obtaining solutions with an optimality gap smaller than 0.4%. Furthermore, additional trains can be scheduled with a small impact on the existing timetable and with optimality gaps below 4%. When stop skipping is allowed, more trains can be scheduled by cancelling only a few stops. We evaluated the performance of the algorithm with different parameter values: the results show that, based on the available computing time and on the choice of the operator, different parameters can be selected and the algorithm will effectively compute a heuristic solution. Finally, we tested a larger instance containing 387 trains, and showed that the algorithm is able to schedule additional trains and is especially effective

when stop skipping is allowed.

Future research will be devoted to improve the regularity of the timetables, i.e., to define a schedule with regularity in the train frequency at the main stations. Although this is a secondary goal, it can significantly improve the perception that the passengers have of the service. Another direction of research consists of including additional constraints related to rolling stock circulation, so as to define schedules that more likely will be easily managed in the rolling stock planning phase.

Acknowledgments

Feng Jiang would like to thank the support of National Natural Science Foundation of China (Project No. 61403317, 61273242) and gratefully acknowledges the financial support from the China Scholarship Council during his research visit at DEI, University of Bologna. The authors would like to thank the anonymous Reviewers for their insightful comments.

In order to fully satisfy the rules for the assignment of the above mentioned grant and of the PhD title from the Southwest Jiaotong University, the other two authors have agreed to have Feng Jiang as first author of the paper.

Appendix

In this Appendix we illustrate the main steps of the proposed algorithm by using a simple example consisting of a line with 4 stations where 3 trains should be scheduled. We consider $psh_t = 1$, $pstr_t = 10$, $pskip_t = 100$, $maxsh_t = \pm 2$, $maxstr_t = 5$, $maxcanc_t = 1$, $p_t = 2000$, $t \in \{$train 1, train 2, train 3$\}$. For each station $i \in \{s1, s2, s3, s4\}$, we consider capacity $c_i = 3$, minimum departure headway time $d_i = 3$, and minimum arrival headway time $a_i = 4$. For each train, the minimum travel times between pairs of consecutive stations, the minimum dwelling time at each station, and the acceleration and deceleration times are reported in Table 12. If the train does not visit a station, we indicate '-' in the table. The desired timetables of the trains are shown in a time-space graph in Figure 10. Recall that acceleration and deceleration times must be considered when the train accelerates (if it starts from its origin station or after it stops at a station) or decelerates (if it arrives at its destination station or before it stops at a station), respectively. There is one line representing the departure nodes from station 1, and one line representing the arrival nodes at station 4. In

addition, there are two lines representing the arrival and departure nodes at station 2 and station 3, respectively. We show only the used nodes (each with the corresponding scheduled time) and use arrows of different types (*continuous* for train 1, *dashed* for train 2, *dotted* for train 3) to represent the timetable for each train (i.e. its path in the time-space graph). As we can see, a conflict occurs between train 1 and train 3 at station 2, as the minimum departure headway is violated. In addition, train 1 overtakes train 3 between station 2 and station 3. A conflict occurs between train 2 and train 1, as they both depart from station 3 at the same time. Finally, a conflict occurs between train 2 and train 3, as train 3 overtakes train 2 between station 3 and station 4.

	$l_t^{s1,s2}$	$l_t^{s2,s3}$	$l_t^{s3,s4}$	dw_{ts2}	dw_{ts3}	acc_t	dec_t
train 1	8	2	3	2	0	2	2
train 2	-	6	9	-	4	2	2
train 3	-	3	3	-	1	2	2

Table 12: Data used in the example.

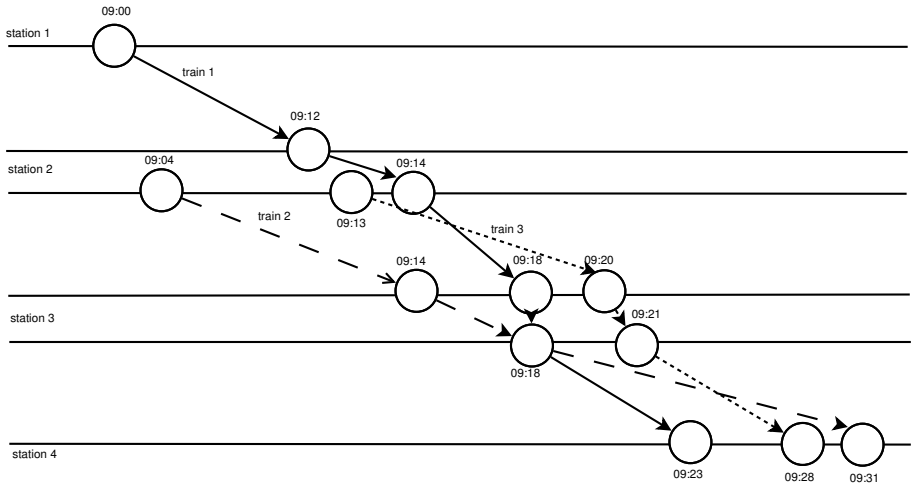

Figure 10: Desired timetables.

The proposed algorithm is iteratively executed for 50 iterations. At iteration 1, it computes the solution of the Lagrangian relaxed problem and obtains a heuristic solution, by applying the dynamic programming algorithm presented in Section 3.4. The obtained solution is shown in Figure 11. The profit of this solution is 3970 and trains 1 and 2 are scheduled.

Train 1 is scheduled according to its desired timetable, while train 2 undergoes a stretch of 3 minutes at station 3 (consequently, the train stops 7 minutes, but the travel times are not changed as train 2 was required to stop at station 3). Several constraints are dynamically added to the constraint pool during the following iterations and the Lagrangian multipliers are updated accordingly by the subgradient optimization procedure. Meanwhile the dynamic programming algorithm is used to compute the solution of the corresponding Lagrangian relaxed problems and heuristic solutions. We do not report the details of this updating as it would require several steps.

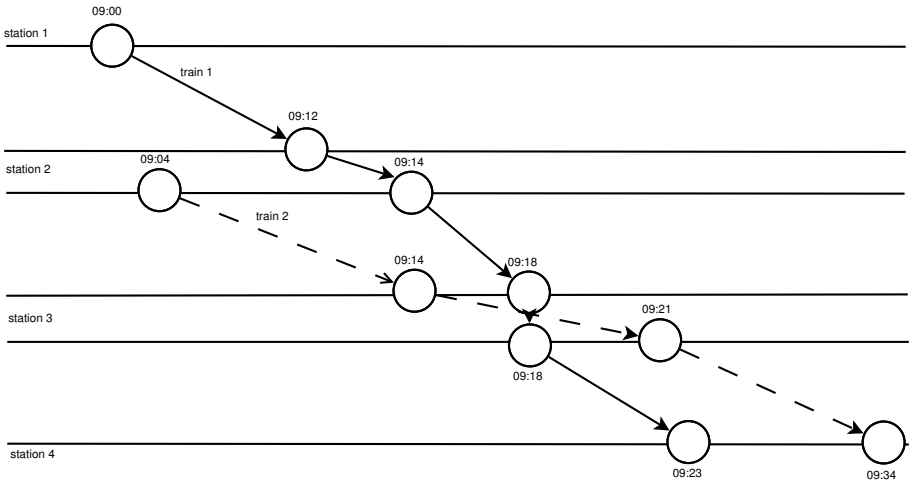

Figure 11: Solution at iteration 1.

The next improvement in the heuristic solution is obtained at iteration 11. The obtained solution is shown in Figure 12. The profit of this solution is 3996 and trains 1 and 3 are scheduled. Train 1 undergoes a shift of 2 minutes in advance, and train 3 a shift of 2 minutes in delay.

The process is executed iteratively by solving, at each iteration, the Lagrangian relaxed problem and computing a heuristic solution, and updating the constraint pool and the Lagrangian multipliers, until the last iteration is reached. In Figure 13, we show the best solution found. The profit of this solution is 5885, and all the three trains are scheduled. Train 1 undergoes a shift of 2 minutes in delay and a stretch of 1 minute at station 3 (consequently, since the stop at station 3 was not planned in its desired timetable, the travel time from station 2 to station 3 is increased to 6 minutes, as the train has to decelerate before stopping at station 3, and is increased to 7 minutes between stations 3 and 4, since the train has to accelerate after

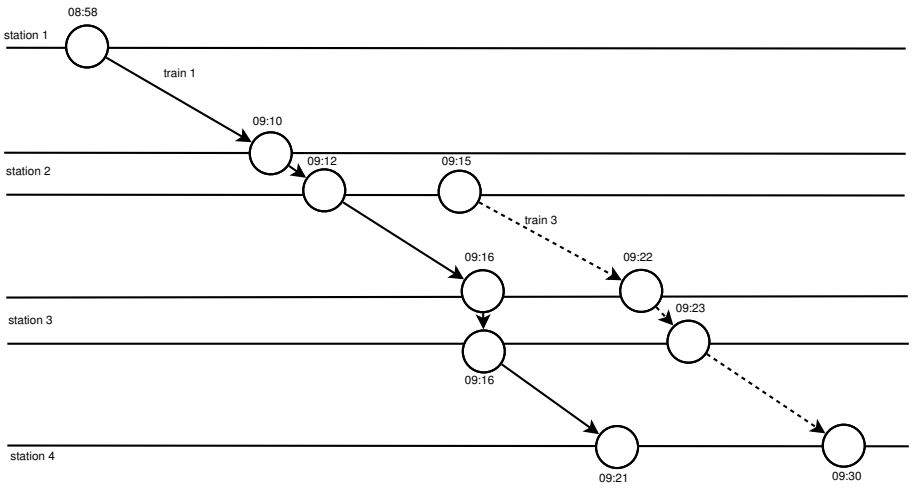

Figure 12: Solution at iteration 11.

stopping at station 3). Train 2 undergoes a shift of 1 minute in advance, while train 3 undergoes a shift of 2 minutes in advance. In addition, the stop of train 2 at station 3 is cancelled: consequently, the travel time between stations 2 and 3 is decreased to 8 minutes, since the train does not need to decelerate before station 3, and is decreased to 11 minutes between stations 3 and 4, since the train does not need to accelerate after station 3.

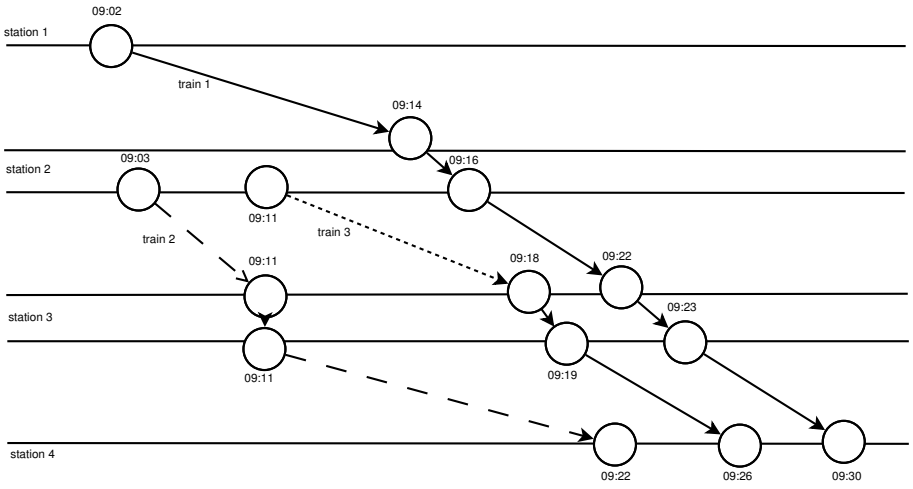

Figure 13: Solution at the last iteration.

References

N. Bešinović, R.M.P. Goverde, E. Quaglietta, and R. Roberti. An integrated micro–macro approach to robust railway timetabling. *Transportation Research Part B: Methodological*, 87:14–32, 2016.

R.L. Burdett and E. Kozan. Techniques for inserting additional trains into existing timetables. *Transportation Research Part B: Methodological*, 43 (8):821–836, 2009.

M.R. Bussieck, T. Winter, and U.T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming*, 79(1-3):415–444, 1997.

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

V. Cacchiani, A. Caprara, and P. Toth. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2): 215–231, 2010.

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

V. Cacchiani, L. Galli, and P. Toth. A tutorial on non-periodic train timetabling and platforming problems. *EURO Journal on Transportation and Logistics*, 4(3):285–320, 2015.

V. Cacchiani, F. Furini, and M.P. Kidd. Approaches to a real-world train timetabling problem in a railway node. *Omega*, 58:97–110, 2016a.

V. Cacchiani, F. Jiang, and P. Toth. Timetable optimization for high-speed trains at chinese railways. *Electronic Notes in Discrete Mathematics*, 55: 29–32, 2016b.

D. Canca, E. Barrena, A. De-Los-Santos, and J.L. Andrade-Pineda. Setting lines frequency and capacity in dense railway rapid transit networks with simultaneous passenger assignment. *Transportation Research Part B: Methodological*, 93:251–267, 2016.

A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.

A. Caprara, M. Monaci, P. Toth, and P.L. Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154(5):738–753, 2006.

A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth. Passenger railway optimization. *Handbooks in Operations Research and Management Science*, 14:129–187, 2007.

A. Caprara, L. Kroon, and P. Toth. Optimization problems in passenger railway systems. *In J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh, and J. Cole Smith, editors, Wiley Encyclopedia of Operations Research and Management Science*, 2011.

Y-H. Chang, C-H. Yeh, and C-C. Shen. A multiobjective model for passenger train services planning: application to taiwan's high-speed rail line. *Transportation Research Part B: Methodological*, 34(2):91–106, 2000.

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

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

H. Flier, T. Graffagnino, and M. Nunkesser. Scheduling additional trains on dense corridors. In *International Symposium on Experimental Algorithms*, pages 149–160. Springer, 2009.

H. Fu, L. Nie, L. Meng, B.R. Sperry, and Z. He. A hierarchical line planning approach for a large-scale high speed rail network: The china case. *Transportation Research Part A: Policy and Practice*, 75:61–83, 2015.

Y. Gao, L. Kroon, M. Schmidt, and L. Yang. Rescheduling a metro line in an over-crowded situation after disruptions. *Transportation Research Part B: Methodological*, 93:425–449, 2016.

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

L. Ingolotti, F. Barber, P. Tormos, A. Lova, M.A. Salido, and M. Abril. An efficient method to schedule new trains on a heavily loaded railway

network. In *Ibero-American Conference on Artificial Intelligence*, pages 164–173. Springer, 2004.

A. Jamili and M.P. Aghaee. Robust stop-skipping patterns in urban railway operations under traffic alteration situation. *Transportation Research Part C: Emerging Technologies*, 61:63–74, 2015.

L. Kroon, D. Huisman, E. Abbink, P-J. Fioole, M. Fischetti, G. Maróti, A. Schrijver, A. Steenbeek, and R. Ybema. The new dutch timetable: The or revolution. *Interfaces*, 39(1):6–17, 2009.

L. Lamorgese, C. Mannino, and E. Natvig. An exact micro-macro approach to cyclic and non-cyclic train timetabling. *Omega*, to appear. doi: http://dx.doi.org/10.1016/j.omega.2016.11.004.

S. Li, M.M. Dessouky, L. Yang, and Z. Gao. Joint optimal train regulation and passenger flow control strategy for high-frequency metro lines. *Transportation Research Part B: Methodological*, 99:113–137, 2017.

C. Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435, 2008.

R.M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR Spectrum*, 33(4):843–883, 2011.

H. Niu, X. Zhou, and R. Gao. Train scheduling for minimizing passenger waiting time with time-dependent demand and skip-stop patterns: Nonlinear integer programming models with linear constraints. *Transportation Research Part B: Methodological*, 76:117–135, 2015.

S. Peer, J. Knockaert, and E.T. Verhoef. Train commuters' scheduling preferences: Evidence from a large-scale peak avoidance experiment. *Transportation Research Part B: Methodological*, 83:314–333, 2016.

T. Robenek, Y. Maknoon, S.S. Azadeh, J. Chen, and M. Bierlaire. Passenger centric train timetabling problem. *Transportation Research Part B: Methodological*, 89:107–126, 2016.

A. Schöbel. Line planning in public transportation: models and methods. *OR Spectrum*, 34(3):491–510, 2012.

P. Sels, T. Dewilde, D. Cattrysse, and P. Vansteenwegen. Reducing the passenger travel time in practice by the automated construction of a robust railway timetable. *Transportation Research Part B: Methodological*, 84: 124–156, 2016.

Y. Tan. *Techniques for Inserting Additional Train Paths into Existing Cyclic Timetables*. PhD thesis, Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2015.

J. Törnquist. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In *Kroon L.G., Möhring R.H., eds. Proc. 5th Workshop on Algorithmic Methods and Models for Optim. Railways, ATMOS 2005*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.

L. Yang, J. Qi, S. Li, and Y. Gao. Collaborative optimization for train scheduling and train stop planning on high-speed railways. *Omega*, 64: 57–76, 2016.

J. Yin, T. Tang, L. Yang, Z. Gao, and B. Ran. Energy-efficient metro train rescheduling with uncertain time-variant passenger demands: An approximate dynamic programming approach. *Transportation Research Part B: Methodological*, 91:178–210, 2016.

J. Yin, L. Yang, T. Tang, Z. Gao, and B. Ran. Dynamic passenger demand oriented metro train scheduling with energy-efficiency and waiting time minimization: Mixed-integer linear programming approaches. *Transportation Research Part B: Methodological*, 97:182–213, 2017.

Y. Yue, S. Wang, L. Zhou, L. Tong, and M.R. Saat. Optimizing train stopping patterns and schedules for high-speed passenger rail corridors. *Transportation Research Part C: Emerging Technologies*, 63:126–146, 2016.

W. Zhou and H. Teng. Simultaneous passenger train routing and timetabling using an efficient train-based lagrangian relaxation decomposition. *Transportation Research Part B: Methodological*, 94:409–439, 2016.