European Journal of Operational Research 295 (2021) 183–202

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

Timetable coordination in a rail transit network with time-dependent passenger demand

Jiateng Yin[a], Andrea D'Ariano[b], Yihui Wang[a,*], Lixing Yang[a], Tao Tang[a]

[a] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China
[b] Department of Engineering, Università degli Studi Roma Tre, Rome, Italy

A R T I C L E I N F O

Article history:
Received 8 May 2020
Accepted 27 February 2021
Available online 3 March 2021

Keywords:
Transportation
Timetable coordination
Rail transit network
Mixed-integer linear programming
Adaptive large neighborhood search

A B S T R A C T

With the expansion of urban rail networks and the increase of passengers demand, the coordination of strongly connected lines becomes more and more important, because passengers transfer several times during their trips and major transfer stations in the rail network often suffer from over-crowdedness, especially during peak-hours. In this paper, we study the optimization of coordinated train timetables for an urban rail network, which is a tactical timetabling problem and includes several operational constraints and time-dependent passenger-related data. We propose a mathematical formulation with the objective of minimizing the crowdedness of stations during peak hours to synchronously generate the optimal coordinated train timetables. By introducing several sets of passenger flow variables, the timetable coordination problem is formulated as a mixed-integer linear programming problem, that is possible to solve to optimum. To capture the train carrying capacity constraints, we explicitly incorporate the number of in-vehicle passengers in the modelling framework by considering the number of boarding and alighting passengers as passenger flow variables. To improve the computational efficiency of large-scale instances, we develop an Adaptive Large Neighborhood Search (ALNS) algorithm with a set of destroying and repairing operators and a decomposition-based ALNS algorithm. Real-world case studies based on the operational data of Beijing urban rail network are conducted to verify the effectiveness of timetable coordination. The computational results illustrate that the proposed approaches reduce the level of crowdedness of metro stations by around 8% in comparison with the current practical timetable of the investigated Beijing urban rail network.

1. Introduction

With the expansion of residents and traffic congestions of urban road networks, most large cities such as Beijing, Tokyo and London have made great efforts to expand the scale of urban rail networks and improve the service quality. Taking Beijing as an example, there are more than 20 operating lines in the Beijing urban rail network, which carries more than 10 million passengers to their destinations in a day. Due to the increased gap between limited line capacity and unbalanced distribution of passenger demands, many stations are suffering from high levels of pedestrian density. For transfer stations that have a huge amount of transferring passengers, the crowdedness issue is even worse and increases the potential safety risks (Xu, Liu, Li, & Hu, 2014).

Different from urban road networks, trains in urban rail networks are operated according to *pre-planned* timetables that specify the arrival and departure times of trains at stations. If the timetables of different lines are planned well and coordinated carefully, the crowdedness of stations can then be evidently reduced (Wong, Yuen, Fung, & Leung, 2008). Thus, the network capacity and service quality will also be enhanced. In practical operations, the coordination of timetables also makes noticeable influence on the passenger volumes at stations and network capacity. For example, Changping Line and Line 13 in Beijing urban rail network are two connected lines that link the suburban and downtown areas in Beijing, as partially shown in Fig. 1(a). Many residents take these two lines in the morning peak-hours to workplaces around Station 1 of Line 13. Without considering timetable coordination, these two lines will both use the minimum headway time (i.e., 2 to 3 minutes) to depart as many trains as possible, which however causes the enormous overcrowdedness of Station 4 on Line 13 and may even cause safety risks on these transfer stations (Fig. 1(b)). In order to alleviate the pressure at station 4 of Line 13, we can properly increase the headway time of Changping Line (e.g., 4 to

* Corresponding author.
E-mail addresses: jtyin@bjtu.edu.cn (J. Yin), a.dariano@ing.uniroma3.it (A. D'Ariano), yihui.wang@bjtu.edu.cn (Y. Wang), lxyang@bjtu.edu.cn (L. Yang), ttang@bjtu.edu.cn (T. Tang).

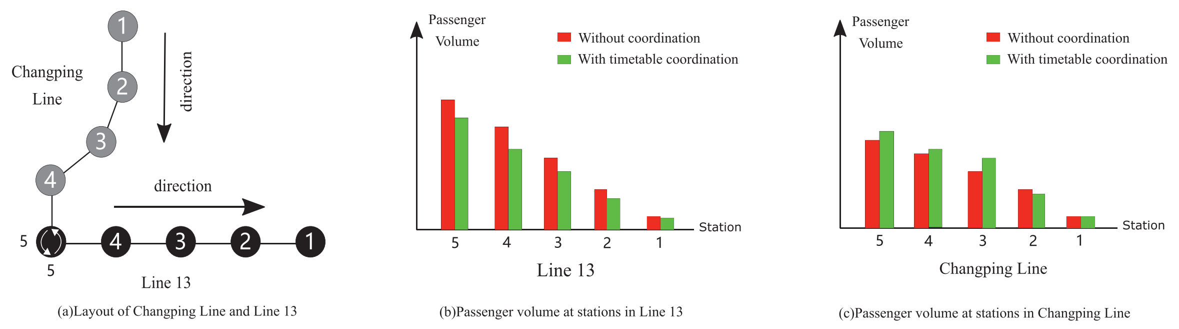


Fig. 1. Illustration of the passenger volume without and with timetable coordination.

5 minutes). This strategy will increase the crowdedness of stations along Changping line, while the overall crowdedness of the network may be eased noticeably (Fig. 1(c)).

Many studies have developed different mathematical models and solution algorithms for train timetabling (Hansen & Pachl, 2014; Louwerse & Huisman, 2014), which is one of the most important problems in railway operations. In particular, the demand-oriented train timetabling on the tactical level is more and more popular for urban rail systems to satisfy the dynamic passenger demands (Barrena, Canca, Coelho, & Laporte, 2014a; Mesa, Ortega, & Pozo, 2014; Niu, Zhou, & Gao, 2015). However, the coordinated train timetabling problem with dynamic passenger demands is hardly addressed for an urban rail network in the literature. The possible reasons may involve: (1) In an urban rail network, there are many possible travelling choices and passengers may transfer several times to arrive at their destinations, which complicate the design of the coordinated timetables. (2) The numbers of boarding, waiting, and alighting passengers are closely coupled via dynamic passenger arriving rates, train timetables, train capacity, etc., which arise highly nonlinear characteristics for the optimization of coordinated timetables. (3) More importantly, existing researches have already shown the computational intensities in solving the passenger-oriented train timetabling for a single metro line, e.g., Barrena et al. (2014a), Yin, Yang, Tang, Gao, and Ran (2017) and Wang et al. (2018); so, the difficulties are intuitive and obvious if we further consider the train timetabling problem on a network with two or more connected lines.

To address the problems mentioned above, this paper presents a coordinated train timetabling framework that considers dynamic passenger demands and passenger transfers in an urban rail network to generate a global-optimal train timetable for multiple connected lines simultaneously. By introducing several sets of passenger flow variables, we formulate the problem as a mixed-integer linear programming (MILP) formulation with the objective of minimizing the crowdedness at the busiest stations, i.e., the highest crowdedness among all the stations in the network. Furthermore, oversaturated conditions, due to the limited train carrying capacity, are also included in the model formulation. To solve large-scale models more efficiently, we also develop two efficient variable neighbor search based algorithms, that can obtain good-quality solutions in an acceptable computation time.

1.1. Literature review

There is a large amount of literature on train timetabling and scheduling problems for railway transportation systems (Caprara, Fischetti, & Toth, 2002; Corman, D'Ariano, Marra, Pacciarelli, & Samà, 2017; D'Ariano, Pacciarelli, & Pranzo, 2007; Fischetti & Monaci, 2017; König & Schön, 2021; Liu & Kozan, 2009; Mesa et al., 2014; Pellegrini, Marliére, & Rodriguez, 2014; Veelenturf, Kroon, & Maróti, 2017; Zhou & Zhong, 2005; Zhang, D'Ariano, He, & Peng, 2019). An overview on different models and solution methodologies is given by Cordeau, Toth, and Vigo (1998), Cacchiani and Toth (2012) and Corman and Meng (2014). Our literature review will focus on research related to train timetabling for urban railways.

1.1.1. Train timetabling for urban rail lines

In urban railways, it is practically desirable to design a passenger-centric timetable to provide high-quality services for commuters. Many researchers have devoted their effort to obtain optimal train timetables for a single line that could provide better service quality to passengers (Espinosa-Aranda, García-Ródenas, del Carmen, López-García, & Angulo, 2015; Vansteenwegen & Oudheusden, 2006). Typically, the research of train timetabling can be classified into two categories. The first category focuses on periodic timetables, where the train schedules are repeated in the planning horizon (Kroon, Peeters, Wagenaar, & Zuidwijk, 2014; Robenek, Azadeh, Maknoon, de Lapparent, & Bierlaire, 2018). As the train departure frequency is constant in a fixed time period, it can only minimize passenger waiting time if passenger demands do not change rapidly. The second category focuses on irregular timetables or demand-oriented timetables that could capture the variation of passenger demands (Barrena, Canca, Coelho, & Laporte, 2014b). The irregular timetables become more and more popular for urban rail systems.

With the consideration of dynamic passenger demands, Barrena et al. (2014a) developed a mixed-integer linear programming (MILP) formulation to optimize the irregular train timetables for an urban rail line. Their aim is to minimize the average waiting time of passengers. The inconvenience costs for the time-varying travelling OD pairs were first introduced in Mesa et al. (2014) and a MILP is then formulated for an urban rail line. A clustering-based heuristic method is developed to solve large-scale problems more efficiently. Niu et al. (2015) considered oversaturated situations of an urban rail line with time-dependent OD matrix and developed a mixed-integer nonlinear programming (MINLP) formulation for minimizing the passenger waiting time. Canca, Barrena, Algaba, and Zarzo (2014) developed a nonlinear integer programming model to determine train arrival and departure times for an urban rail line with dynamic passenger demands, where the objective is to minimize passenger waiting time and train operational cost. Yin et al. (2017) developed two mixed-integer linear programming models based on space-time network formulations to minimize energy consumption and passenger waiting time. Wang et al. (2018) proposed an integrated optimization model for train timetabling and rolling stock circulation planning for an urban rail lines, where the train departure headways between train services are optimized according to time-varying passenger arriving rates. Considering the train timetabling problem integrated with passenger flow control, Shi, Yang, Yang, and Gao (2018) formulated an integer linear programming model to obtain a service-oriented timetable as well as an optimal passenger flow control strategy to avoid congestion at platforms for an oversaturated metro line with time-varying demands.

Overall, the above studies addressed the timetabling problem for a single urban rail line with dynamic passenger demands. However, train timetabling in an urban rail network with multiple connected lines is much more challenging. For the single line train timetabling problem, the passenger arrival rates at the stations are pre-given from historical passenger OD data. For the urban rail network, the passenger arriving rates at each station are determined not only by the historical OD data, but also by the transferring passengers from other lines. Meanwhile, the volume of these transferring passengers is affected by the timetable of former lines that these passengers take.

1.1.2. *Coordinated timetabling in urban transit networks*

Until now, nearly 90% of the urban rail systems all over the world is composed by more than 2 lines that are connected and interacted with each other.[1] As the single-line train timetabling models cannot be directly applied into a rail transit network, some recent studies have begun to concentrate on coordination of train timetables among different lines in order to improve the service quality of passengers.

Based on the assumption that trains have unlimited capacity and passenger flows are evenly distributed within a given time period, Wong et al. (2008) proposed a mixed-integer programming (MIP) model for the synchronization of train timetables of different lines. Their objective is to minimize the interchanging waiting times of all passengers (i.e., the waiting time of passengers for boarding a transferring train). Cadarso and Marin (2012) proposed an integrated planning model to optimize train capacity and system frequencies to satisfy the passengers' demand in a transit network. However, their model ignores the transfer of passengers in the transit network by assuming that the passenger demand on each arc is fixed and not affected by the train schedule. In addition, their study considers the under-saturated situations. It is assumed that there are no remaining passengers at each station when a train departs. Taking the volume of passengers' demand as input, Canca, Barrena, De-Los-Santos, and Andrade-Pineda (2016) presented a mixed-integer nonlinear programming model to determine the optimal line frequency and train capacity rather than the explicit train departure and arrival times. Wang, Tang, Ning, van den, and De Schutter (2015) proposed an event-driven train scheduling model for a transit-network by considering the transfer of passengers. However, their formulation is highly non-convex and can only handle illustrative small-scale cases. Cao, Ceder, Li, and Zhang (2019) studied the optimal synchronization of train timetables under the assumptions of uniform train headways and unlimited train capacity. A new meta-heuristic algorithm based on local search and genetic algorithm was developed and was compared with the performance by CPLEX. Yin, D'Ariano, Wang, Yang, and Tang (2019) studied the optimization of coordinated train timetables in an urban transit network with time-dependent passenger demand and unlimited train boarding capacity. A mixed-integer linear programming was developed and a series of illustrative small-scale examples was conducted with the aid of a commercial MILP solver.

Similar to the coordinated train timetabling problem, a series of studies has been devoted to timetable synchronization in urban transit networks (e.g., bus networks) (Ceder, Golany, & Tal, 2001; Ibarra-Rojas, Delgado, Giesen, & Muñoz, 2016). Recently, Ibarra-Rojas and Ríos-Solís (2012) established an integer programming formulation in order to maximize the number of synchronizations of passenger transfers, where the synchronizations are defined as the arrival of two trips within a given time window. A multi-start iterated local search algorithm was proposed to solve real-world instances by means of their model. To get an exact solution for the above-mentioned model, Fouilhoux, Ibarra-Rojas, and Rós-Solś (2016) proposed four classes of valid inequalities, that enable a commercial solver to solve large-scale instances within a few minutes. Liu and Ceder (2018) addressed an integrated public transit timetable synchronization and vehicle scheduling problem with consideration of users' trip choices. Their problem was formulated as a bi-level bi-objective non-linear integer programming model. By analyzing the characteristics of this model, a sequential search method based on a novel deficit function was proposed by the latter authors to generate a set of Pareto-efficient solutions. Nevertheless, the above studies on timetable synchronization in urban transit networks mainly focus on passenger transfer times, while none of those studies consider the crowdedness of stations, due to the huge amount of transfer passengers in urban rail networks, especially during peak hours.

1.2. *Paper contributions*

In summary, passenger-demand-oriented train timetabling problem is a very hot research topic and many mathematical models and solution methodologies have been well explored to generate an irregular train timetable with different kinds of demand patterns. Nevertheless, most of these approaches are still limited to the timetabling of a single line due to the complexity for modelling passenger transfers in a network (e.g., Barrena et al., 2014a; Wang et al., 2018). Even though some recent studies have begun to consider the timetable coordination in an urban rail network, the passenger demands in these studies are usually considered as invariant values with unlimited train carrying capacity (e.g., Abdolmaleki, Masoud, & Yin, 2020; Cao et al., 2019; Corman et al., 2017; Yin et al., 2019). To the best of our knowledge, there is until now no linear formulation that can model the train timetabling problem with dynamic passenger demands in an urban rail network. Table 1 lists the detailed characteristics of some closely related works to show the contributions of our study, which are detailed as follows.

(1) Our work proposes an integrated mixed-integer linear programming (MILP) formulation for the optimal coordination of passenger-oriented train timetables in an urban railway network during peak hours. Our proposed MILP formulation, extended from Yin et al. (2019), simultaneously considers the time-dependent passenger demands, multiple transfers of passengers, train following headway constraints, train loading capacity constraints, and rolling stock constraints in an urban rail network. Different from the single-line train timetabling problems studied, e.g., in Barrena et al. (2014a) and Yin et al. (2017), we first propose an extended flow conservation formulation for managing traffic flows in a rail transit network with multiple connected lines. In our formulation, the passenger flow conservation constraints at each station are not only influenced by the passenger flows at the belonging line, but these also depend on the passenger flows related to the connected (or indirectly connected) lines. Besides, we define the concept of congestion level (substantially different with the objective function in Yin et al., 2019) as the objective function of our formulation, in order to minimize the passenger crowdedness at non-transfer and transfer stations. The new objective function explicitly captures different loading capacities of involved stations, and thus is more suitable than that in Yin et al. (2019) for practical applications. Furthermore, we demonstrate that the proposed formulation can be reformulated into a novel MILP formulation, that is equivalent with the original formulation but presents with significantly less decision variables and constraints.

[1] http://www.qizuang.com/baike/16017.html.

Table 1
Summary of relevant recent studies on passenger demand oriented train timetabling for urban railways.

Publications	Infrastructure	Passenger demand[b]	Objective function	Train capacity	Model structure	Solution method (model structure)
Barrena et al. (2014a)	Single-line	Dynamic	Average passenger waiting time	Yes	MILP	B&C[a]
Barrena et al. (2014b)	Single-line	Dynamic	Average passenger waiting time	No	MINLP	ALNS
Niu et al. (2015)	Single-line	Dynamic	Total passenger waiting time	Yes	MINLP	GAMS
Mesa et al. (2014)	Single-line	Dynamic	Passenger inconvenience cost[c]	Yes	MILP	CA
Canca et al. (2014)	Single-line	Dynamic	Average passenger waiting time	Yes	MINLP	GAMS
Hassannayebi and Zegordi (2017)	Single-line	Dynamic	Total and maximum passenger waiting time	No	MILP	VNS
Ortega, Pozo, and Puerto (2018)	Single-line	Dynamic	Passenger inconvenience cost[c]	No	MILP	MIP Express
Wang et al. (2018)	Single-line	Dynamic	Variations of load factor and headway	Yes	MINLP	CPLEX
Yin et al. (2017)	Single-line	Dynamic	Passenger waiting time and energy consumption	Yes	MILP	LR
Cao et al. (2019)	Network	Static	Number of synchronized meetings	No	MILP	Local search and genetic algorithm
Abdolmaleki et al. (2020)	Network	Static	Transfer time	No	MILP	Local search
Wong et al. (2008)	Network	Piecewise constant	Transfer waiting time	No	MILP	Heuristics + CPLEX
Cadarso and Marin (2012)	Network	Static	Operating and passenger penalty cost	Yes	MILP	GAMS/CPLEX
Wang et al. (2015)	Network	Piecewise constant	Energy consumption and passenger travelling time	Yes	Non-convex	Genetic algorithm
Canca et al. (2016)	Network	Static	Operational costs and generalized user cost	Yes	MILP	Heuristics
Yin et al. (2019)	Network	Dynamic	Number of waiting passengers	No	MILP	CPLEX
This paper	Network	Dynamic	Crowdedness of stations	Yes	MILP	CPLEX/VLNS

[a] Symbols description in Table 1: branch-and-cut (B&C); adaptive large neighborhood search (ALNS); Lagrangian relaxation (LR);
[b] "Static" indicates that the passenger demand is assumed to be constant during different time periods (e.g., one-hour periods) of the same day; "Dynamic" represents that the passenger demand is time-varying through the whole day.
[c] The passenger inconvenience cost is measured by the summation of passenger waiting time and penalty of trips that are not served;

(2) By analyzing the mathematical properties of our MILP formulation, we propose customized solution methodologies to generate good quality solutions in a suitable computation time, even for large-scale instances. We first show that the proposed MILP formulation can be viewed as separate sub-problems, and we then derive some useful mathematical properties, including feasibility and optimality bounds with respect to each sub-problem. These properties provide key information for the design of advanced optimization algorithms, including an adaptive large neighborhood search (ALNS) algorithm and a Decomposition-based ALNS (DALNS) algorithm. Specifically, ALNS is a metaheuristic based on the systematic change of neighborhoods towards improving the current local optima, while DALNS is a two-level variable neighborhood search algorithm based on the specific problem decomposition (Hassannayebi & Zegordi, 2017; Lazić, Hanafi, Mladenovíc, & Urošević, 2010; Lejeune, 2006; Masson, Lehuédé, & Péton, 2013).

(3) Two sets of complex instances are investigated, involving small-scale realistic instances and real-world medium- and large-scale instances based on the rolling stock and passenger flow data of Beijing urban rail network. We compare the computational results obtained by our algorithms with a commercial MILP solver and the practical timetable used for the Beijing urban rail network. These experiments demonstrate that the proposed approaches compute near-optimal solutions and outperform CPLEX. Moreover, the performance enhancement by our approach is about 8% to 20% in comparison with the current practical (non-coordinated) timetable.

The rest of this paper is organized as follows. In Section 2, we give a description for the timetable coordination problem with time-dependent passenger demands in an urban transit network, and formulate the problem into a MILP model. Then, we propose the solution methodology based on ALNS and DALNS in Section 3. In Section 4, we propose two case studies, i.e., a small case and a real-world case based on the real-world detected data in Beijing metro network, to demonstrate the effectiveness of timetable coordination and efficiency of solution algorithms. Section 5 gives detailed conclusions on the potential of the proposed methodology and future research directions.

2. Problem statement

2.1. Network structure

Our study considers a public transportation network which provides information about the stations and the tracks connecting the stations, denoted by $G = (S, E)$. In railway transportation systems, the vertices S represent train stations and the edges E represent the physical connections between the vertices, i.e., the tracks. A *line* $l \in L$ is composed by a sequence of stations $(s_1, \dots, s_l)$ and edges in the public transportation network G. Let S_l denote the set of stations on line l, and we have the set of stations $S = \cup_{l \in L} S_l$ in the network. The lines are called *connected lines* if there exist one or more transfer stations that connect the lines. If two lines l and l' are connected, there exist two connected transfer stations i on line l and station j on line l'. At transfer stations, the travelling passengers alight the train from station i of line l and walk to station j of line l' to take another train. These two transfer stations i and j, that connect two different lines, are called *connected stations*. An illustrative metro network with two connected lines and seven stations is presented in Fig. 2. In the network, we have $a, b, c, d \in S_1$ and $e, f, g \in S_2$. Stations c and f are the transfer stations, where the passengers from line 2 can transfer to line 1 and the passengers from line 1 can transfer to line 2 as well. In fact, the connected transfer stations (stations c and f) can be the same physical station but with different labels, depending on the line they belong, due to the formulation convenience.

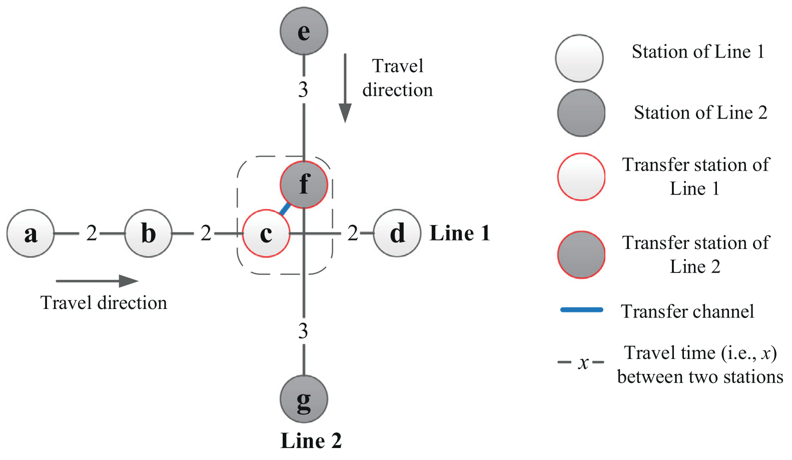


Fig. 2. An illustrative network of two connected lines.

In the network G with $|L|$ lines, a feasible travelling *path* is defined if the passengers can move from one node i to another j without any repetition of nodes. In particular, if p is a *direct* path on line l with no transfer, p can be denoted by (i, j) where $i \in S_l$ and $j \in S_l$ respectively denote the origin and destination of path p. Otherwise, path p involves transfers among different connected lines. We represent path p by a set of origin, transfer and destination stations $S_p = \{i, r_1, \ldots, r_n, j\}$, where $r_1, \ldots, r_n$ represent *the set of transferring stations* denoted by S_p^r. In particular, if a passenger transfers from station i of line l to station i' of line l', we only record station i' as a component of the sequence of transferring stations, since transfer station i is uniquely associated with station i' of line l'.

We define $\mathcal{P}$ as a set of feasible travelling paths of passengers in the graph G. We also define two subsets $\mathcal{P}_i^o \subset \mathcal{P}$ and $\mathcal{P}_j^d \subset \mathcal{P}$ that respectively represent the set of travelling paths with origin i and destination j, where $i, j \in S$, and the subset $\mathcal{P}_\tau^r \subset \mathcal{P}$, that represents the set of travelling paths with transfer station $\tau \in S^r$, i.e., the set of transfer stations. In addition, we denote $\mathcal{P}_i = \mathcal{P}_i^o \cup \mathcal{P}_i^r$ that represents the set of paths that originate from or transfer to station i from other lines. In other words, the path set $\mathcal{P}_i$ defines the set of feasible paths, by which the passengers should wait for boarding the trains at station i. In particular, if a path p involves transfers among different lines, we denote i_p^+ to represent the former node of station i on path p and i_p^- to represent the subsequent node of station i on path p.

Example 2.1. In the illustrative metro network depicted by Fig. 2, the set $\mathcal{P}$ involves 11 paths: six direct paths on line 1, three direct paths on line 2 and three transfer paths. Specifically, the paths on line 2 are $(e, f), (e, g) \in \mathcal{P}_e^o$ and $(f, g) \in \mathcal{P}_f^o$. Three transfer paths are thus identified in the network of Fig. 2 according to the above definitions, i.e., $p_1 = (a, f, g)$, $p_2 = (b, f, g)$ and $p_3 = (e, c, d)$. We also have $p_1, p_2 \in \mathcal{P}_f^r$ and $p_3 \in \mathcal{P}_c^r$. Subsequently, we can see that $f_{p_1}^+ = a$, which indicates the former station of f along path p_1, and $f_{p_2}^+ = b$, which indicates the former station of f along path p_2.

The time horizon of our problem is discretized into a set of timestamps $T = \{0, 1, \ldots, |T|\}$. The travel time of the trains between the connected stations i and $i+1$ is set as t_i^r, where $(i, i+1) \in E$, and the dwelling time at station $i \in S$ is given by t_i^d. Let t_{ij}^s represent the transfer time between two connected stations i and j on different lines (i.e., transferring from one line to another). Given the above parameters (i.e., travel time, dwelling time and transfer time), we can actually obtain the total travel time of passengers between any two connected nodes i_p^+ and i on each path p, denoted by t_i^p. Specifically, if i is the destination of path p, t_i^p represents the travel time between boarding at station i_p^+ and alighting at station i. Otherwise, if i is a transfer station, t_i^p represents the travel time from boarding at station i_p^+, alighting at station i' (i.e., the connected station of i) and arriving to transfer station i. An illustration of the travel time of passengers between two connected nodels is shown in Fig. 3.

For modeling convenience, in the next subsection we will adopt the following three sets, for each $0 \le t \le |T|$:

$$T_t = \{t' \in T | t' \le t\} \tag{1}$$

$$T_{pi} = \{t \in T | t \ge t_i^p\} \tag{2}$$

$$T_{pit} = \left\{t' \in T | t' + t_i^p \le t\right\} \tag{3}$$

The first set T_t defines the discrete time units $t' \in T$ that are not later than time t. The second set T_{pi} is defined on path p of each node i and includes the discrete time units $t \in T$ that are larger than t_i^p. The third set T_{pit} includes the discrete time units $t' \in T$ that are not larger than $t - t_i^p$. The latter set indicates that, if passengers arrives at the former station of i along path p at time t', they can reach transfer station $i \in S_p^r$ before (or at) time t.

Example 2.2. We now consider the link travel times displayed on Fig. 2 and assume that the transfer time of passengers is 2 at station c. We can thus calculate the travelling time from e to station c along path p_3, given by $t_c^{p_3} = 5$. At time unit t, we know that $T_{p_3ct} = \emptyset$ if $t < 5$; otherwise, we have $T_{p_3ct} = \{0, \ldots, t-5\}$.

2.2. Time-dependent passenger demand in a transit network

In metro systems, it is widely recognized that the passenger demand is time-dependent and the passenger arriving rates vary at different time periods. In the single-line train scheduling literature (e.g., Barrena et al., 2014a; Yin et al., 2017), the dynamic passenger demand is captured by using OD matrixes, in which each term (i, j) is a function of time that indicates the number of passengers willing to travel from station $i \in S$ to station $j \in S$. Each number in this matrix is thus time dependent, and the passenger arrival time will thus depend on the timetable and the chosen paths. However, this method cannot be straightforwardly applied to model time-dependent passenger demands in a transit network, since a subset of passengers may enter station s at time t and transfer to other lines to reach their destination station.

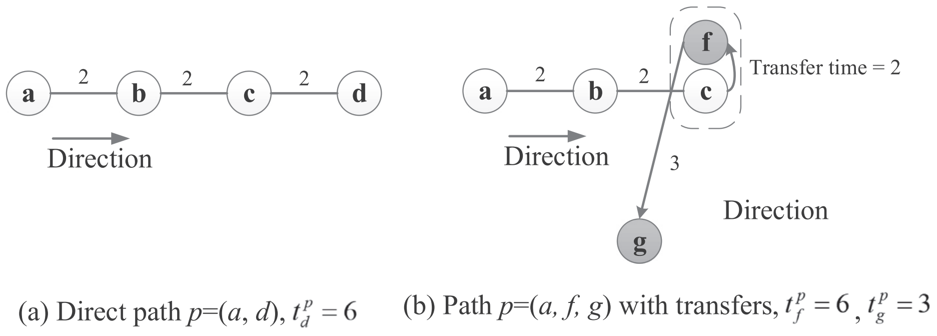

2
2
2
a
b
c
d
Direction
a
b
f
c
g
2
2
3
Transfer time = 2
Direction
Direction
(a) Direct path p=(a, d), $t_d^p = 6$
(b) Path p=(a, f, g) with transfers, $t_f^p = 6$, $t_g^p = 3$

Fig. 3. Travel time definition for two illustrative paths (without and with transfer).

Due to the above mentioned reason, our study employs a *path-based* representation to depict the time-dependent passenger demand on a transit network. Specifically, we denote a time-dependent matrix $\mathcal{D}_t = \{D_p^t | p \in \mathcal{P}, t \in T\}$, where each element D_p^t represents the number of passengers with path p entering the network at time t. Thus, the number of passengers who originate from station i at time t is $d_i^t = \sum_{p \in \mathcal{P}_i^o} D_p^t$.

Remark 2.1. Different from the studies in Shi, Yang, Yang, Zhou, and Gao (2019), our path-based representation enables to consider more than one travelling path with the same origin and destination. Passengers can choose different travelling (feasible) paths in the network to reach their destinations. Meanwhile, we note that the path-based time-dependent matrix can also be obtained from historical AFC (Automatic Fare Collection) data in metro systems by using travel demand assignment methods (see e.g., Domencich & McFadden, 1975; Gangdi, Cantarella, & Vitetta, 2019; Wang, Zhao, D'Ariano, & Peng, 2020).

2.3. Problem definition and assumptions

Let us consider an urban rail transit network, the crowdedness of a station in the network is directly influenced by four elements: the time-dependent passenger OD matrix, the timetables of the involved lines, the train carrying capacity and the station capacity threshold. Since the train capacity and station capacity are usually fixed at the network planning level, the only way to alleviate the crowdedness of stations is to vary the train departure frequency, i.e., adjusting the timetables. For example, the train departure headway is reduced to 2 minutes in some busy lines in Beijing metro. Nevertheless, a new issue raises when increasing the departure frequency of one line. The connected stations will suffer a very large volume of transfer passengers. Reducing the train departure headway of a line can alleviate the crowdedness of stations along this line, but would evidently increase the crowdedness of stations at other lines. In other words, the crowdedness of stations in an urban metro network are closely coupled with each other. However, determining the optimal train timetables in a rail transit network synchronously is very challenging in practice due to the complex interactions of transferring passengers of different lines. For this reason, in practice the rail managers still construct train timetables of different lines separately, with slightly adjustments by Beijing metro operators based on their former experiences.

In summary, the aim of this paper is to provide a unified mathematical formulation for the Coordinated Train Timetabling (CTT) problem, which minimizes the crowdedness of most overcrowded stations in an urban rail transit network. Our formulation also explicitly captures the time-dependent passengers demand, the transfer of passengers among different lines, and the train/station capacity. The mathematical formulation is based on the following four main assumptions.

Assumption 1. We assume that the time-dependent passenger demand in the railway network is denoted by the number of passengers on each travelling path, which fixes the origin/destination station, the departure time from the origin station and the list of transfer stations among the different lines. The travelling paths of passengers are pre-given by the historical data and we thus do not consider the passengers' route choice. This assumption is valid in our study as our focus is to manage traffic flows during peak hours, where the train frequency is significantly high.

Assumption 2. Since the number of passengers on board of the trains is a large integer value, the approximation error, caused by treating this number as a real-valued variable, is small. This is a similar treatment as in Wang et al. (2018).

Assumption 3. We assume in this paper that trains depart from the start terminal in a first-in-first-out manner. Each station can only accommodate one train at a time, where overtaking and crossing operations are prohibited at any positions, which is consistent with the practical requirement in urban railway systems (Yin et al., 2017).

Assumption 4. As we consider the coordinated train timetabling problem on the tactical planning level, we assume that the train running times at each segment and the dwell times at each station are not affected by the volume of boarding/alighting passengers, but are pre-given according to the technical rolling stock characteristics and to the known passenger demand by rail managers. Similar assumptions have been widely adopted in recent previous studies (Barrena et al., 2014a; Wang et al., 2018).

3. Mathematical formulation

3.1. Initial formulation

We present a mixed-integer linear programming (MILP) formulation for the CTT. In particular, we will first ignore the train capacity constraints and assume that all the passengers can board the next coming train. Based on this simplification, we define the following three sets of decision variables:

$$x_{lt} = 1 \text{ if a train departs from line } l \in L \text{ at time } t \in T; = 0 \text{ otherwise,} \tag{4}$$

$$n_i^t = \text{ number of waiting passengers at station } i \in S \text{ and time } t \in T, \tag{5}$$

$$b_{pi}^t = \text{number of boarding passengers with travel path } p \in \mathcal{P} \text{ at station } i \text{ and time } t \in T. \tag{6}$$

In the above sets of variables, x_{lt} determines the departure times of trains on each line l. According to Assumption 4, this set of variables actually defines a coordinated train schedule (i.e., train arrival and departure times of each station) in the network. Variables n_i^t and b_{pi}^t are associated with the movement of passenger flows in the network. Here, we notice that b_{pi}^t is huge, since it is defined on each $p \in \mathcal{P}$, $i \in S$ and $t \in T$. If the scale of the network is large, there will be a very large number of feasible paths. In practice, this value can be greatly reduced according to the properties associated with our problem. First, if $p \in \mathcal{P}$ is a direct path with no transfer, only one station i (i.e., the origin station) is associated with path p, since the destination data is already included in the path p. Otherwise, p has several transfer stations $r \in S_p^r$, where the number of variables b_{pi}^t is still very limited due to the fact that not so many transfer stations exist on a single line. Second, we notice that the passengers are actually unable to board any train until the first train arrives into the station. For example, we assume that the first train from the depot takes $\bar{t}_i$ time units to arrive at station i. Then, b_{pi}^t is always equal to zeros if $t \le \bar{t}_i$. It is reasonable to define b_{pi}^t on a restricted set of time units $T_{pi} = \{t : t > t_i^p : t \in T_t\}$. The detailed list of sets, parameters and decision variables used in the mathematical formulation is presented in Table 7 of Appendix A.

Remark 3.1. Theoretically, there can be numerous paths between any pair of graph nodes. Therefore, finding the resulting k-shortest paths is a combinatorial problem (Meng et al., 2005). In our study, we do not enumerate all possible origin-destination paths, while we only focus a limited number of paths between any pair of graph nodes, which is derived from (off-line) processing of the historical AFC data.

The full mathematical formulation for CTT is provided as follows.

$$\min_{x} \quad \max_{t,i} \frac{n_i^t}{SC_i}, \tag{7}$$

$$\text{s.t. } n_i^t = \sum_{\tau \in T_t} d_i^\tau + \sum_{p \in \mathcal{P}_i^r} \sum_{\tau \in T_{pit}} b_{pi_p^+}^\tau - \sum_{\tau \in T_t} \sum_{p \in \mathcal{P}_i^o \cup \mathcal{P}_i^r} b_{pi}^\tau, \qquad \forall i \in S, t \in T \tag{8}$$

$$\sum_{p \in \mathcal{P}_i} b_{pi}^t \le C \cdot x_{l(t-t_i)}, \qquad \forall i \in S_l, l \in L, t \in T : t \ge t_i \tag{9}$$

$$\sum_{\tau \in T_t} b_{pi}^\tau \le \sum_{\tau \in T_t} D_p^\tau, \qquad \forall i \in S_l, p \in \mathcal{P}_i^o, t \in T_{pi} \tag{10}$$

$$\sum_{\tau \in T_t} b_{pi}^\tau \le \sum_{\tau \in T_{pit}} b_{pi_p^+}^\tau, \qquad \forall i \in S_p^r, p \in \mathcal{P}_i^r, t \in T_{pi} \tag{11}$$

$$\sum_{t \le t' \le t + T_{min}^l} x_{lt'} \le 1, \qquad \forall l \in L, t \in T \tag{12}$$

$$\sum_{t \le t' \le t + T_{cycle}^l} x_{lt'} \le K_l, \qquad \forall l \in L, t \in T \tag{13}$$

$$x_{lt} \in \{0, 1\}, \qquad \forall l \in L, t \in T \tag{14}$$

$$b_{pi}^t \ge 0, \qquad \forall p \in \mathcal{P}, i \in S, t \in T \tag{15}$$

$$n_i^t \ge 0, \qquad \forall i \in S, t \in T \tag{16}$$

We define *congestion level* as the total number of waiting passengers (i.e., n_i^t) at station i and time t divided by the station capacity SC_i, where SC_i indicates the maximum total number of passengers that station i can accommodate. The value of SC_i is pre-determined by the layout of each station (in the metro infrastructure construction phase). Thus, the objective function (7) minimizes the highest congestion level through all the stations in S along time horizon T.

Constraints (8) fix the flows of waiting passengers, arriving passengers, transferring passengers and boarding passengers at each stations $i \in S$ during the considered time horizon T. In particular, the number of waiting passengers at station $i \in S$ and time unit $t \in T$ is calculated by considering the cumulative arriving passengers and by subtracting all the boarding passengers for time units $\{0, 1, \dots, t\}$. Here, the cumulative arriving passengers involve two types of passengers. The first type refers to the newly arrived passengers with origin station i, denoted by $\sum_{\tau \in T_t} d_i^\tau$. The second type is the transferred passengers from other lines. If station i is a transfer station, we need to find out the number of boarding passengers from any former node of i along each path p. The boarding time is also restricted to the set T_{pit}, to guarantee enough time for passengers travelling from node i_p^+ to i. Thus, the number of transferring passengers at station i and time τ is given by $\sum_{p \in \mathcal{P}_i^r} \sum_{\tau \in T_{pit}} b_{pi_p^+}^\tau$. The number of boarding passengers at each time unit τ is also calculated by considering the boarding passengers with different paths p that originate or transfer at station i.

Constraints (9) are the passenger boarding constraints, which guarantee that the passengers at station i are able to board when the next train arrives at station i with path p. t_i is a pre-given parameter, which denotes the travel time of any train from the origin station of line l to station i. In particular, if no train departs from the origin station of line l at time $t - t_i$ (i.e., $x_{l(t-t_i)} = 0$), the number of boarding passengers at station i and time t is zero; otherwise, the right hand side of constraints (9) imposes an upper bound C on the number of passengers that can board each train. In the initial formulation, the passenger boarding constraints do not consider the passengers that are already on the train, before starting the current boarding process. In Section 3.2, we will extend this set of constraints to better model train capacity constraints with consideration of the readily on-board passengers.

Constraints (10) and (11) set the thresholds for the number of boarding passengers at each time unit t, which should be less or equal to the maximum number of available passengers for each p. Specifically, constraints (10) limit the number of boarding passengers with respect to the newly arriving passengers, while constraints (11) aim to limit the number of boarding passengers transferring from other lines. In particular, for the origin station $i = p^o$ of path p, constraints (10) ensure that the number of boarding passengers with origin station i cannot exceed the total number of arriving passengers until time t; for each transfer station $i \in S_p^r$ of path p, constraints (11) guarantee that the number of boarding passengers at station i cannot exceed the total number of transferring passengers from previous stations.

Constraints (12) are the train headway constraints, which ensure that no more than one train can depart from the depot within the time window $[t, t + T_{\min}^l]$ on line l. Constraints (13) are the rolling stock circulation constraints, which ensure that, given a set of K_l physical trains on line l, at most K_l train services can depart from the depot within each time window $[t, t + T_{cycle}^l]$ on line l.

3.2. Extended formulation with train capacity constraints

In model (7)–(16), the passenger boarding constraints (9) limit the number of boarding passengers with a fixed value, i.e., train capacity C. As long as a train arrives at the current station, passengers (the number of passengers should be no more than C) are allowed to board the train. However, constraints (9) do not include overcrowded situations when the arrival train is partially

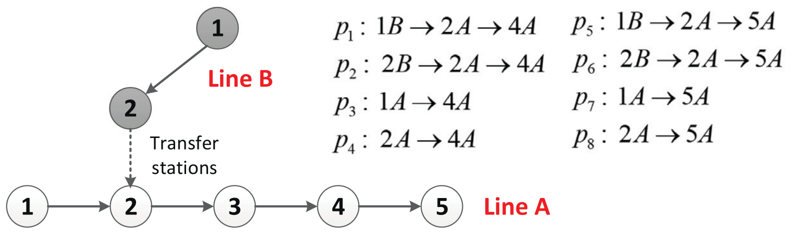


Fig. 4. Illustrative example for capacity constraints.

occupied by some passengers and its available capacity is less than C (Wang et al., 2018). Considering the oversaturated metro networks, such as Beijing or Tokyo, some passengers may not be able to board the next arrival train, since this train has too small available capacity. Thus, these passengers have to wait for the following train. In such cases, train capacity constraints are very important to model more realistic passenger flows in an oversaturated metro network. In addition, the objective function in model (7)–(16) is a non-linear *min-max* function. In what follows, we show how to embed the explicit train capacity in our model and how to reformulate CTT model into a mixed integer linear programming (MILP) formulation.

We first consider the explicit train capacity constraints at station i on line l. Define $S_{il} = \{s_1, s_2 \dots, s_{i-1}\}$ as a set of stations before station i along line l. The above set indicates that there are passengers who start from (or transfer at) station $i' \in S_{il}$ and end their journey at station i. Therefore, we generate a path set $\mathcal{P}_{i'i} \subseteq \mathcal{P}^o_{i'} \cup \mathcal{P}^r_{i'}$. Each path in $\mathcal{P}_{i'i}$ originates from (or transfers at) node i' and ends at stations that are located after station i. In other words, each path in $\mathcal{P}_{i'i}$ crosses over station i. The aim of denoting this path set is to calculate the total number of in-vehicle passengers when the next train arrives at station i. The path set $\mathcal{P}_{i'i}$ is formally represented as follows:

$$\mathcal{P}_{i'i} = (\mathcal{P}^o_{i'} \cup \mathcal{P}^r_{i'}) \cap (\mathcal{P}(s_{i+1}) \cup \dots \cup \mathcal{P}(s_I)). \tag{17}$$

In Eq. (17), $\mathcal{P}^o_{i'} \cup \mathcal{P}^r_{i'}$ indicates the paths that start from (or transferred at) station i' on line l. $\mathcal{P}(s_{i+1}) = \mathcal{P}^d_{i+1} \cup \mathcal{P}^r_{i+1}$ indicates the set of paths that end at station s_{i+1} or transfer at station s_{i+1}. Specifically, $\mathcal{P}^d_{i+1}$ represents the paths that end at station s_{i+1}, while $\mathcal{P}^r_{i+1}$ represents the paths that transfer at station s_{i+1}. An example of Eq. (17) is given in Example 3.1.

According to the above definitions, we know the fact that the passengers with path $p \in \mathcal{P}_{i'i}$ are still on the corresponding train when this train departs from station i. We consider that the number of passengers with path $p \in \mathcal{P}_{i'i}$ is n_p. Then, the available capacity is $C - \sum_{p\in\mathcal{P}_{i'i}} n_p$. In other words, at most $C - \sum_{p\in\mathcal{P}_{i'i}} n_p$ passengers can board the current train when this train departs from station i. Therefore, we can formally derive the following train capacity constraints.

$$\sum_{p\in\mathcal{P}_i} b^t_{pi} \le C - \sum_{i'\in S_{il}} \sum_{p'\in\mathcal{P}_{i'i}} b^{t(i',i)}_{pi'}, \qquad \forall i \in S_l, l \in L, t \in T_{pi} \tag{18}$$

where $t(i', i) = t - t^l_{i'i}$ indicates the passengers' boarding time at the former stations $i' \in S_{il}$, where $t^l_{i'i}$ is a parameter that represents the travel time from station i' to i on line l. In Eq. (18), the left term indicates the number of boarding passengers,while the right term indicates the available boarding capacity.

Example 3.1. An illustrative example is given in Fig. 4. Line A involves 5 stations and line B involves 2 stations. Station 2 of line A and Station 2 of Line B are transfer stations. We use symbols "*NA*" and "*NB*" (for instance, stations $2B$, $4A$) to represent the index of stations of line A and line B, respectively. For illustrative purposes, we only consider the train capacity constraints for station $3A$ at time $t = 10$. As can be seen from Fig. 4, there are two stations of Line A that are located before station $3A$, i.e., $S_{3A} = \{1A, 2A\}$. There are a total of 8 paths that cross over station $1A$ (or $2A$) and end at stations after $3A$. Then, according to Eq. (17), we have $\mathcal{P}_{1A,3A} = \{p_3, p_7\}$ and $\mathcal{P}_{2A,3A} = \{p_1, p_2, p_5, p_6, p_4, p_8\}$.

In order to illustrate the purpose of Eq. (18), we also set the following parameters in the studied example:

The time from passenger boarding at 1B till arriving at station $3A$ is 7;
The time from passenger boarding at 2B till arriving at station $3A$ is 5;
The time from passenger boarding at 1A till arriving at station $3A$ is 4;
The time from passenger boarding at 2A till arriving at station $3A$ is 2;

Train capacity is 15.

With the above parameters, we can derive that the in-vehicle passengers when the train dwells at station 3 at time $t = 10$ is composed of passengers who board the train at stations $1B$, $2B$, $1A$, $2A$ at time $10 - 7 = 3$, $10 - 5 = 5$, $10 - 4 = 6$ and $10 - 2 = 8$, respectively. In addition, the destinations of these passengers should be $4A$ and $5A$, which are located after station $3A$. We consider that the numbers of boarding passengers with origins $1B$, $2B$, $1A$ and $2A$ and destination $4A$ are 1, 2, 3 and 2. The numbers of boarding passengers with origins $1B$, $2B$, $1A$ and $2A$ and destination $5A$ are 1, 1, 1 and 1. We can thus calculate the available train capacity when the train dwells at station 3 at time $t = 10$: $15 - (1 + 2 + 3 + 2) - (1 + 1 + 1 + 1) = 3$. Therefore, only three additional passengers can board the train at station 3 and $t = 10$.

The illustrative example 3.1 shows how the train capacity works to limit the number of boarding passengers.

Next, we note that the mathematical formulation defined by (7)–(16) is essentially a *max-min* problem with several sets of linear constraints. We next introduce the new continuous variable $\xi \in R$ that is used to represent the maximum level of crowdedness in the studied rail transit network. Therefore, the original formulation can be viewed as the following equivalent linear problem.

Lemma 3.1. *The CTT model can be equivalently linearized as a MILP model with additional continuous variable ξ, given as follows:*

$$\text{(L-CTT)} \qquad \min_{x,\xi} \ \xi \tag{19}$$

$$\text{s.t.} \quad \xi \ge c_i \left(\sum_{\tau\in T_t} d^\tau_i + \sum_{p\in\mathcal{P}^r_i} \sum_{\tau\in T_{pit}} b^\tau_{pi^+_p} - \sum_{p\in\mathcal{P}^o_i\cup\mathcal{P}^r_i} \sum_{\tau\in T_{pit}} b^\tau_{pi} \right) \quad \forall i \in S, t \in T \tag{20}$$

$$\text{Constraints } (9)-(15), (18)$$

$$\xi \ge 0 \tag{21}$$

where $c_i = \frac{1}{SC_i}$.

Proof. See Appendix B. □

Remark 3.2. It is worth to mention that, as the aim of our paper is to reduce the congestion of stations with large passenger volumes, we denote continuous variable ξ (in the objective function) in L-CTT model, which actually represents the maximum value of congestion level among all involved stations in the considered time horizon. To prevent the potential spread of COVID-19 in urban rail networks, reducing the congestion level at crowded stations (especially transfer stations) can be considered as a primary requirement for rail managers. This is especially the case for Beijing metro. Meanwhile, we observe that our methodology can be extended to incorporate other relevant practical aspects of the investigated problem (as we have done in Appendix C in terms of the minimization of passenger waiting time), according to the further specific requirements of subway managers.

4. Theoretical properties of the L-CTT model

The L-CTT model is actually a 0–1 Mixed Integer Programming (0–1 MIP) formulation consisting of minimizing a linear function ξ subject to a series of inequality constraints (9)–(16), (18) and (20). Theoretically, L-CTT can be solved to optimality with branch-and-bound or cutting plane methods, since this is a linear programming model with integer variables. However, the developed model involves a large number of coupling constraints and 0–1 binary variables. Moreover, the number of variables and constraints grows exponentially with the number of involved lines and time units, which results in a significant computational intensity for solving even medium-scale experiments. For example, CPLEX takes more than 72 hours to generate a feasible solution (with less than 5% optimality) for medium-scale experiments (see the numerical experiments in Section 5). The reason is that CPLEX branches in each iteration on an arbitrary node and solves the relaxed LP problem to obtain a bound to the original problem. The bound can be quite weak, since the integer variables associated with the corresponding line are closely coupled with each other. Fixing one integer variable makes little sense to the improvement of the overall bound. However, the solution time plays an important role and the acceptable computation time is usually less than one hour in practice. The reason is that the urban rail train timetabling is an iterative procedure and managers need to adjust it several times to include new service intentions or business rules. Thus, the solver has to compute a solution quickly and is eventually recalled several times during the timetabling process. Due to these reasons, this section aims to explore the underlining theoretical properties of the L-CTT model, which motivate us to develop efficient solution methodologies and obtain high-quality solutions in a much shorter computation time compared with CPLEX.

For clarity reasons, we next express the L-CTT model as follows.

$$(P)\quad \min\{\xi \,|\, f(\xi,b)\geq 0, g(b,x)\geq 0, x\in X, b\in Y\} \tag{22}$$

where $X=\{x\in\{0,1\}|\text{Eqs. }(12),(13)\}, Y=\{b\in\mathbb{R}^{|\mathcal{P}||S||T|}|\text{Eqs. }(10),(11)\ b\geq 0\}$. $f(\xi,b)$ and $g(b,x)$ are the sets of affine functions defined by Eqs. (20) and (9), respectively. It is clear that the problem (P) involves binary variables x and continuous variables b and ξ. Eqs. (9) couple the x and b variables, while Eqs. (20) couple the x and ξ variables.

To formally analyze some key theoretical properties of the L-CTT model, we first derive the following two definitions associated with the problem (P).

Definition 4.1. Let $\bar{x}$ be an arbitrary vector of binary values that satisfies Eqs. (12) and (13). The reduced problem associated with $\bar{x}$ is defined as follows.

$$P(\bar{x})\quad \min\{\xi \,|\, f(\xi,b)\geq 0, g(b,\bar{x})\geq 0, b\in Y\} \tag{23}$$

As binary variables $\bar{x}$ are fixed, $P(\bar{x})$ is actually a Linear Programming (LP) model that assigns passengers on each train service by considering the movement of passenger flows in the rail transit network.

Definition 4.2. Define $\delta(x)$ as a vector of binary variables with the same dimension of x. If $\delta(x)_{lt}\leq x_{lt}$ for any $l\in L$ and $t\in T$, we define $\delta(x)\preceq x$ as a componentwise inequality between vectors x and $\delta(x)$.

From the above definition, we observe that vector $\delta(x)$ is essentially an operator that changes some values of x from 1 to 0. In our problem, $\delta(x)$ indicates that some train services are canceled from an original solution x.

Example 4.1. We consider an example in which vector x is given by $\{1,0,0,1,0,0\}$. This indicates that two train services depart from their origin at time unit 0 and time unit 3, respectively. From Definition 4.2, if we remove the first train service, we have $\delta(x)=\{0,0,0,1,0,0\}$ and $\delta(x)\preceq x$.

Based on these definitions, we will denote the following properties with respect to problem (P).

Lemma 4.1. *$P(\bar{x})$ is always feasible by giving any $\bar{x}\in X$ that satisfies Eqs. (12) to (13).*

Proof. The feasibility of $P(\bar{x})$ can be proved by finding one set of solutions ξ and b, which are feasible to $P(\bar{x})$. We can consider a special case when $b^t_{pi}=0$ for $p\in\mathcal{P}$, $i\in S$ and $t\in T$. In this case, Eqs. (9)–(11) are straightforwardly satisfied. We note that constraints (20) are fulfilled if and only if

$$\sum_{\tau\in T_t} d^\tau_i + \sum_{p\in\mathcal{P}^r_i}\sum_{\tau\in T_{pit}} b^\tau_{pi^+_p} - \sum_{p\in\mathcal{P}^o_i\cup\mathcal{P}^r_i}\sum_{\tau\in T_{pit}} b^\tau_{pi}\geq 0$$

for any $i\in S$ and any $t\in T$. As d^τ_i is an input parameter, which specifies the number of arriving passengers at time τ and station i, this is a nonnegative value. We thus know that all the above inequalities are also fulfilled. Given any $\bar{x}\in X$, there is thus at least one set of solutions that is always feasible to $P(\bar{x})$, which completes the proof. □

Lemma 4.2. *If vector $x\in X$, (13), then $\delta(x)\in X$.*

Proof. If $x\in X$, we have $\sum_{t\leq t'\leq t+T^l_{\min}} x_{lt'}\leq 1$ and $\sum_{t\leq t'\leq t+T^l_{cycle}} x_{lt'}\leq K_l$ for $l\in L$ and $t\in T$. Meanwhile, according to Definition 3.2, we have $\delta(x)_{lt}\leq x_{lt}$ for any $l\in L$ and $t\in T$. Thus, the following inequalities can be derived:

$$\sum_{t\leq t'\leq t+T^l_{min}}\delta(x)_{lt'}\leq\sum_{t\leq t'\leq t+T^l_{min}} x_{lt'}\leq 1, \forall l\in L, t\in T$$

$$\sum_{t\leq t'\leq t+T^l_{cycle}}\delta(x)_{lt}\leq\sum_{t\leq t'\leq t+T^l_{cycle}} x_{lt}\leq K_l, \forall l\in L, t\in T.$$

We deduce that $\sum_{t\leq t'\leq t+T^l_{\min}}\delta(x)_{lt'}\leq 1$ and $\sum_{t\leq t'\leq t+T^l_{cycle}}\delta(x)_{lt'}\leq 1$ for any $l\in L$ and $t\in T$. These two inequalities conclude the proof. □

Lemma 4.3. *If vector $x\in X$, then $L(x)\leq L(\delta(x))$; where $L(x)$ and $L(\delta(x))$ are the optimal values of $P(x)$ and $P(\delta(x))$, respectively.*

Proof. First, according Lemmas 4.1 and 4.2, when $x\in X$ both $P(x)$ and $P(\delta(x))$ are feasible. Let $\hat{b}$ and $\hat{\xi}$ denote an optimal solution of $P(\delta(x))$, indicating that $L(\delta(x))$ is achieved with $\hat{b}$ and $\hat{\xi}$. We have the following inequality:

$$g(\hat{b},\delta)\geq 0.$$

Since $\delta(x)\preceq x$, we can deduce that $\hat{b}$ and $\hat{\xi}$ are also feasible to problem $P(x)$. Thus, we have

$$L(\bar{x})\leq P(\bar{x})=L(\delta(\bar{x})),$$

which derives the proof. □

Lemma 4.4. *Given any vector $x \in X$, problem $P(x)$ is always bounded.*

Proof. According to Lemma 4.1, $P(x)$ has a feasible solution. In addition, since $\xi \geq 0$, it is straightforward to show that a lower bound of $P(x)$ is 0. In order to analyze an upper bound of $P(x)$, we first look at Lemma 4.3 and we have that $L(x) \leq L(\delta(x))$ if $\delta(x) \preceq x$. Let x_0 denote a vector of zeros. It is easy to see that $P(x_0)$ serves as an upper bound of $P(x)$. We take x_0 to the original problem $P(x)$. Then, an upper bound of $P(x)$ is denoted by:

$$\max_{i \in S} \frac{\sum_{\tau \in T_T} d_i^\tau}{SC_i}.$$

We conclude that an upper bound and an lower bound always exist for $P(x)$. This lemma thus is proved. □

5. Solution methodologies

According to the studied properties of the L-CTT model, we observe that the original problem (19)–(21) can be divided into two subproblems. The first subproblem determines the binary variable set x (i.e., the train schedule for involved lines), while the second level optimizes the passenger flows on the network when x is given as input. The second subproblem (i.e., problem $P(x)$) is actually an LP model that can be solved to optimality efficiently given any value of x, according to Lemma 4.1. It is thus reasonable to adopt some strategies to search for a good quality solution of x and then use a LP solver to solve the second level problem for the evaluation of a candidate solution. Specifically, we can heuristically compute a candidate timetable (i.e., solution x) to schedule as many train services as possible, and solve the resultant LP (i.e., $P(x)$) to obtain the optimal objective value. Even though the computational efficiency of solving $P(x)$ is very high, the resulting optimality gap might be relatively large. Therefore, an efficient approach is to use a local search algorithm to search the possible neighbors of candidate solution x and then use a suitable LP solver to evaluate the solution in its neighborhood. Unfortunately, our primarily experiments (see computational results in Section 6 for details) indicate that the local search algorithm can only find a local optimal solution.

To escape from local optimal solutions during the searching process, our study considers a large-scale neighbor search (LNS), which is an efficient method for solving discrete combinational optimization problems. LNS gradually improves an initial feasible solution by alternately destroying and repairing the solution. In addition, we also consider the Adaptive Large-scale neighbour search (ALNS), i.e., an extension of LNS by allowing multiple destroy and repair methods to be used within the same search (Pisinger & Ropke, 2010). A destroy method destructs part of the current solution, while a repair method rebuilds the destroyed solution. Each destroy/repair method has a weight that controls how often the particular method is attempted during the search. The weights are adjusted dynamically as the search progresses in such a way that the metaheuristic adapts the solving process to the instance at hand. ALNS has been successfully applied in large-scale discrete optimization problems, for example the vehicle routing problems with pickup and delivery (Ropke & Pisinger, 2006; Yuan, Chen, & Prins, 2016), pollution routing problem (Demir, Bektas, & Laporte, 2012), order batching problem (Žulj, Kramer, & Schneider, 2018), production routing problem (Adulyasak, Cordeau, & Jans, 2012) and open shop scheduling problem (Mejía & Yuraszeck, 2020), which demonstrate a much better performance compared with state-of-art commercial solvers.

On the basis of our LNS and ALNS frameworks, we apply the mathematical properties from Lemmas 4.1 to 4.4 for the improvement of the proposed algorithms when solving the L-CTT model: First, we introduce destroy and repair operators according to the studied mathematical properties of the model. For example, we recover solution feasibility by means of Lemmas 4.1 and 4.2. We also propose repair operators that might potentially improve the current best-known solution according to Lemma 4.3. Lemma 4.4 is adopted to guarantee that the objective function value of L-CTT is always bounded, given any $x \in X$. The latter property enables to use an arbitrary combination of repair and destroy operators (as the objective value function always exists and is bounded in a certain range), thus improving the problem-solving customization of our meta-heuristics. Second, we further introduce a decomposition based ALNS (termed as DALNS), which decomposes the network-wide train scheduling problem into smaller train scheduling problems for separated lines. This section will present the ad-hoc characteristics of ALNS and DNLS algorithms, with the aim to find high quality solutions in an acceptable computation time for subway timetabling planners.

5.1. Initial solution generation

The first step to design an efficient ANLS algorithm is to find a good initial solution quickly as the starting point of the search. We use an iterative heuristic method to generate an initial feasible solution, which is commonly used by the metro managers (in practice) to build timetables with constant headways. For line l with a total of K_l trains, we first denote a constant headway by h_c. The value of h_c is initialized as the minimum headway T^l_{min}, i.e., $h_c \leftarrow T^l_{\text{min}}$. Then, a timetable (maybe infeasible) is generated by defining a list of train departure times $\{0, h_c, 2h_c, \ldots, Nh_c\}$, where $Nh_c \leq T$ and N represents the last train service in the considered time horizon. Next, we check the feasibility of this timetable with respect to the rolling stock utilization constraints (13). If this solution violates some constraints, we update the train departure headway by $h_c \leftarrow h_c + 1$ and repeat this procedure until a timetable that satisfies all the constraints is found. We use the same method to generate the timetable for each line $l \in L$ in the rail transit network, respectively.

Actually, the above timetable generation procedure is essentially equivalent to solving the following optimization model:

$$\max \sum_{l \in L} \sum_{t \in T} x_{lt} \tag{24}$$

$$\text{s.t.} \quad \sum_{t \leq t' \leq t + T^l_{min}} x_{lt'} \leq 1, \qquad \forall l \in L, t \in T \tag{25}$$

$$\sum_{t \leq t' \leq t + T^l_{cycle}} x_{lt'} \leq K_l, \qquad \forall l \in L, t \in T \tag{26}$$

$$x_{lt} \in \{0, 1\} \tag{27}$$

which is only related to decision variables x_{lt} $(\forall l, t)$ that maximize the number of trains services without considering the passengers' convenience. The developed model is a classical knapsack problem and can be solved efficiently e.g. by dynamic programming algorithms.

5.2. Destroy operators

In ALNS, a new solution is obtained based on the implementation of a series of destroy and repair operators. These operators determine the possible moves of candidate solutions in their neighborhood. The choice of how to define destroy operators and repair operators (in other words, the neighborhood of a solution) is of great importance for improving the solution quality. In this study, we develop the following destroy operators for each line $l \in L$.

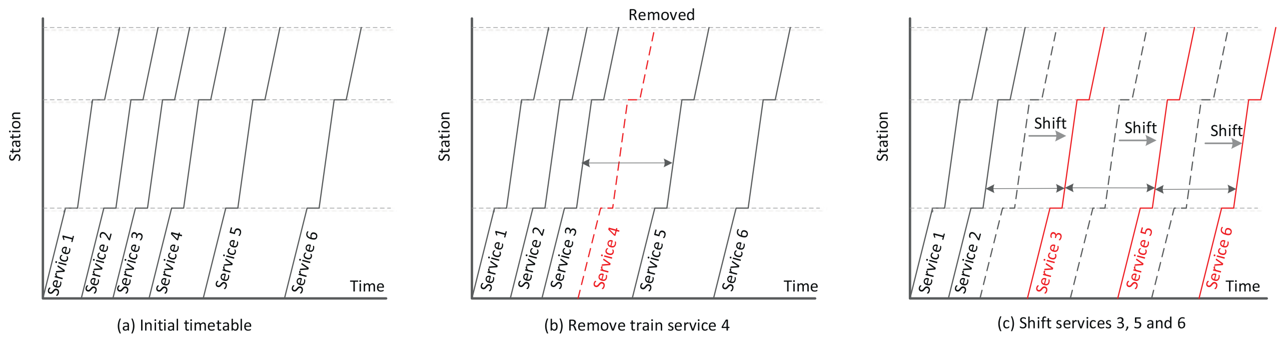


Fig. 5. Illustration of destroy operators.

Destroy_1. *Randomly select and remove s_1 train services from line l.*

This operator randomly removes s_l train services of line l. We note that: (1) according to Lemma 4.2, if a solution x is feasible, the generated solution with Destroy_1. is also a feasible solution; (2) according to Lemma 4.3, the objective function value will not be improved with this operator. In our experiments, we find that removing some train services does not have a noticeable impact on the objective function value, but this actually leaves enough space for alternating the coordination of trains. Combined with other operators introduced below, the objective value can be reduced evidently. The reason is that the neighbor of candidate solution is enlarged after conducting this destroy operators, which is beneficial to escape from the local optimal solutions.

Destroy_2. *Randomly select a time unit $t_l \in T$ and shift the subsequent services.*

This operator randomly selects a time unit t_l from the set of discrete time units T and then shifts the train services that depart after t_l. Specifically, we initialize an empty list $\hat{x}_l^t$ for each $t \in T$ and $l \in L$, and we begin to iterate from $t = 0$ till $t = |T| - t_s$. Here, t_s is the length of shift operations, which is set as $t_s = 1$ in our experiments. In this process, if $x_l^t = 1$ and $t > t_l$, we let $\hat{x}_l^{(t+t_s)} = 1$; otherwise, we let $\hat{x}_l^t = x_l^t$. Until $t = |T| - t_s$, and we output $\hat{x}_l^t$ as the destroy neighbor solution of $\hat{x}_l^t$. We can see that the Destroy_2 operator is essentially a local search process, where the solutions in the neighbor of x_l^t are searched by randomly selecting different t_l.

An illustration of destroy operators is shown in Fig. 5, where a total of six train services are planned in the initial timetable. By using Destroy_1, train service 4 is selected and removed from the timetable (Fig. 5b). Then, Destroy_2 selects a time unit between train services 2 and 3, and moves the afterwards services (i.e., services 3, 5, and 6). In other words, the departure times of train services 3, 5, and 6 in the adjusted solution are increased by Destroy_1 and Destroy_2. The headway time of the new solution is more regular and may be more beneficial under the situation of invariant passenger demand.

5.3. Repair operators

Repair operators are very important for ALNS, since these operators rebuild the destroyed solution, resulting into a new set of candidate solutions. In our problem, an intuitive repair operator is to insert new train services in the largest interval or intervals with the highest demand, as indicated in Barrena et al. (2014b). Nevertheless, simply inserting train services in the largest intervals may lead to an infeasibility e.g. due to non-satisfied headway constraints (12) and/or rolling stock constraints (13). This issue is particularly evident when the train departure frequency is high and the set of feasible solutions is very limited. Therefore, we denote the following three repair operators for each line l to generate a set of feasible neighbor solutions during each iteration.

Repair_1. *Insert s_2 train services into the largest time intervals.*

This operator sorts all the candidate time intervals between each pair of adjacent train services and picks up the largest time interval $T_{\max}$ between two consecutive train services s and s'. If $T_{\max} < 2 * T_{\min}^l$, then no service can be inserted to guarantee feasibility; otherwise, we insert a new train service between s and s'. The process is repeated until s_2 train services are inserted into the reconstructed solution, or no service can be inserted anymore.

Repair_2. *Insert s_3 train services into the time period with the highest passenger demand.*

This operator is similar to Repair_1. The only difference is that we first find time periods when the passenger demand is very high. The insert operation is thus conducted only in the periods with high passenger demand.

Repair_3. *Repair the violation of headway and/or rolling stock constraints.*

The neighborhood solution with Destroy_2 may violate Eqs. (12) and (13), leading to an infeasibility of the generated solution. We here propose another repair operator to convert an infeasibility to a feasible solution. The operator is conducted as follows: Algorithm 1 checks the feasibility of solution x_{lt} $(\forall l)$ from

Algorithm 1 The procedure Repair_3.

```
Input: For each line $l$, input solution vector $x_{lt}$ for $t \in T$.
Step 1. Do for each $t = 0, 1, 2, \ldots, T$:
    Step 1.1 If Eqs. (12) and (13) are satisfied, set $t \leftarrow t + 1$
      and go to Step 2;
    Step 1.2 Otherwise, conduce operator Destroy_2(t), set
      $t \leftarrow t + 1$ and go to Step 2;
Step 2. Output the revised solution $\hat{x}$.
```

$t = 0$ to $t = |T|$. In this process, if constraints (12) or (13) are violated at any time unit t, we use Destroy_2 to move the train services that are not checked (i.e., train services that depart after t). The aim is to separate the trains that are too close to guarantee the satisfaction of headway and rolling stock constraints. The process is terminated after that all the time units are checked. An illustration of Repair_3 is shown in Fig. 6, where five train services are planned. In the input solution (see Fig. 6a), the headway constraints are violated between service 1 and service 2, and between service 3 and service 4. In the implementation of Algorithm 1, the violation between service 1 and service 2 is first checked. We move the afterwards services (i.e., services 2 to 5) in order to satisfy the headway constraints (see Fig. 6b). Next, we continue to check the following time units until a violation between service 3 and service 4 is found. Then, we fine-tune services 4 and 5 (i.e., increase the departure times of services 4 and 5) to satisfy the headway constraints

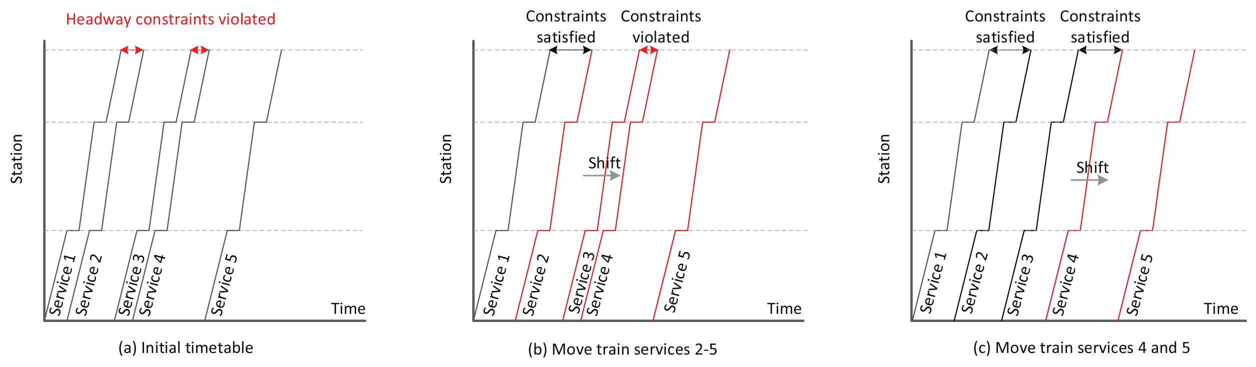


Fig. 6. Illustration of Repair_3.

(see Fig. 6c). The final timetable is converted into a feasible solution when all the time units are checked. Note that this simple heuristic can always generate feasible solutions in our study, since (i) Constraints (12) and (13) are defined for each line $l \in L$ separately; (ii) this operator does not create new train services but only adjust the departure times of existing train services. An extreme case is that no train service is departed for line l, i.e., $x_{lt} = 0$ for $t \in T$. This solution still satisfies constraints (12) and (13).

5.4. Adaptive searching strategy

In each iteration, a set of destroy and repair operators is chosen to generate new solutions. Here, we design an adaptive search strategy, which is a standard adaptive searching strategy in ALNS(Ropke & Pisinger, 2006), to select the most effective operators. Two sets of parameters π_i^d and π_i^r are denoted as the *score* of the destroy and repair operators (each operator indexed by i), respectively; another two sets of parameters w_i^d and w_i^r are denoted as the *weight* of the destroy and repair operators. The score is used to evaluate the effectiveness of each operator, which is measured as its potential contribution to the solving process. The weight defines the importance of each operator and is updated according to its current score.

In an initial phase of our algorithm, all weights are set to one and all scores are set to zero. The operators are selected randomly. At each iteration, the scores of selected operators are updated according to the performance of the best generated solution. The updating rules of scores are set as follows: π_i is increased by δ_1 if a new best solution is reached; π_i is increased by δ_2 if the new solution is better than the current solution but worse than the best known solution; π_i is increased by δ_3 if the new solution is not as good as the current solution. Then, the weights of each operator in both the destroy neighborhood and the repair neighborhood are updated according to the following formulae:

$$w_i^d = (1-\lambda)w_i^d + \lambda\pi_i^d / \sum_{i=1}^{|O_d|}\pi_i^d, \tag{28}$$

$$w_i^r = (1-\lambda)w_i^r + \lambda\pi_i^r / \sum_{i=1}^{|O_r|}\pi_i^r, \tag{29}$$

where $|O_d|$ and $|O_r|$ represent the number of destroy and repair operators, respectively. $\lambda \in [0, 1]$ is a *reaction* factor, which controls how sensitive the weights are to the changes in the operators' performance.

Adaptive selection of the operators ensures high-efficiency in the algorithm searching process (see Gendreau & Potvin, 2018). In order to select the most effective pair of destroy and repair operators in the iterations of ALNS, we adopt two roulette wheel mechanisms, one for selecting destroy operators and the other one for selecting repair operators. The probability of selecting a destroy operator is calculated by $w_i^d / \sum_{i=1}^{|O_d|} w_i^d$, and the probability of selecting a repair operator is calculated by $w_i^r / \sum_{i=1}^{|O_r|} w_i^r$. The process is iterated until one of the following stopping criteria has been met: 1) after a fixed amount of CPU time (i.e., T_{limit}); 2) after a given number of iterations N_{limit} without any improvement in the objective function.

5.5. Decomposition based ALNS

We recall the L-CTT model from Eq. (22):

$$(P) \quad \min\{\xi \,|\, f(\xi, b) \geq 0, g(b, x) \geq 0, x \in X, b \in Y\}$$

where $X = \{x \in \{0,1\} |$ Eqs. (12), (13)$\}$, and $Y = \{b \in \mathbb{R}^{|\mathcal{P}||S||T|} |$ Eqs. (10), (11) $, b \geq 0\}$. In the implementation of ALNS, we choose different operators to search in the domain of X. Then, a selected candidate solution x from X is evaluated by optimizing the reduced problem of P, i.e., $P(x)$. We observe from formulation (22) that the domain of X is directly determined by Eqs. (12) and (13), while the variables $\{X | x_{lt}, t \in T\}$ of each line $l \in L$ are independent with each other. This important property motivates us to further decompose the considered problem into separate smaller subproblems, in which each subproblem optimizes the train schedule of a single line l. Therefore, we adopt a decomposition-based variable neighborhood search.

Decomposition-based variable neighborhood search is a two-level metaheuristic based upon the decomposition of the problem (Lazić et al., 2010). Instead of searching all the neighbors of a candidate solution, this kind of algorithm successively fixes a given number of variables from the candidate solution. In this way, a set of small-scale subproblems are generated and solved exactly (or within a given computation time limit). If better solutions are found in this process, the candidate solution is updated as a better solution in order to perform a neighbor search again. The process is terminated when one of the stopping criteria (the same as ALNS) is reached.

For clarity reasons, we begin the description of algorithm by the following definition.

Definition 5.1. Let $\bar{x}$ be a feasible solution to problem (P). (P) has a total of $|L|$ subproblems, and the $\bar{l}$th subproblem corresponding to (P) is denoted by

$$P(\bar{x}, \bar{l}) \quad \min\{\xi \,|\, f(\xi, n) \geq 0, g(b, x) \geq 0, (b, n) \in Y, \\ x_{lt} = \bar{x}_{lt}, \forall l \in L \backslash \bar{l}, t \in T\} \tag{30}$$

Our DALNS method combines ALNS with a suitable MIP solver in order to solve the L-CTT model more efficiently. Specifically, our developed DALNS is composed by two iteration loops, i.e., a outer-loop and an inner-loop, as shown in Fig. 7. In the implementation

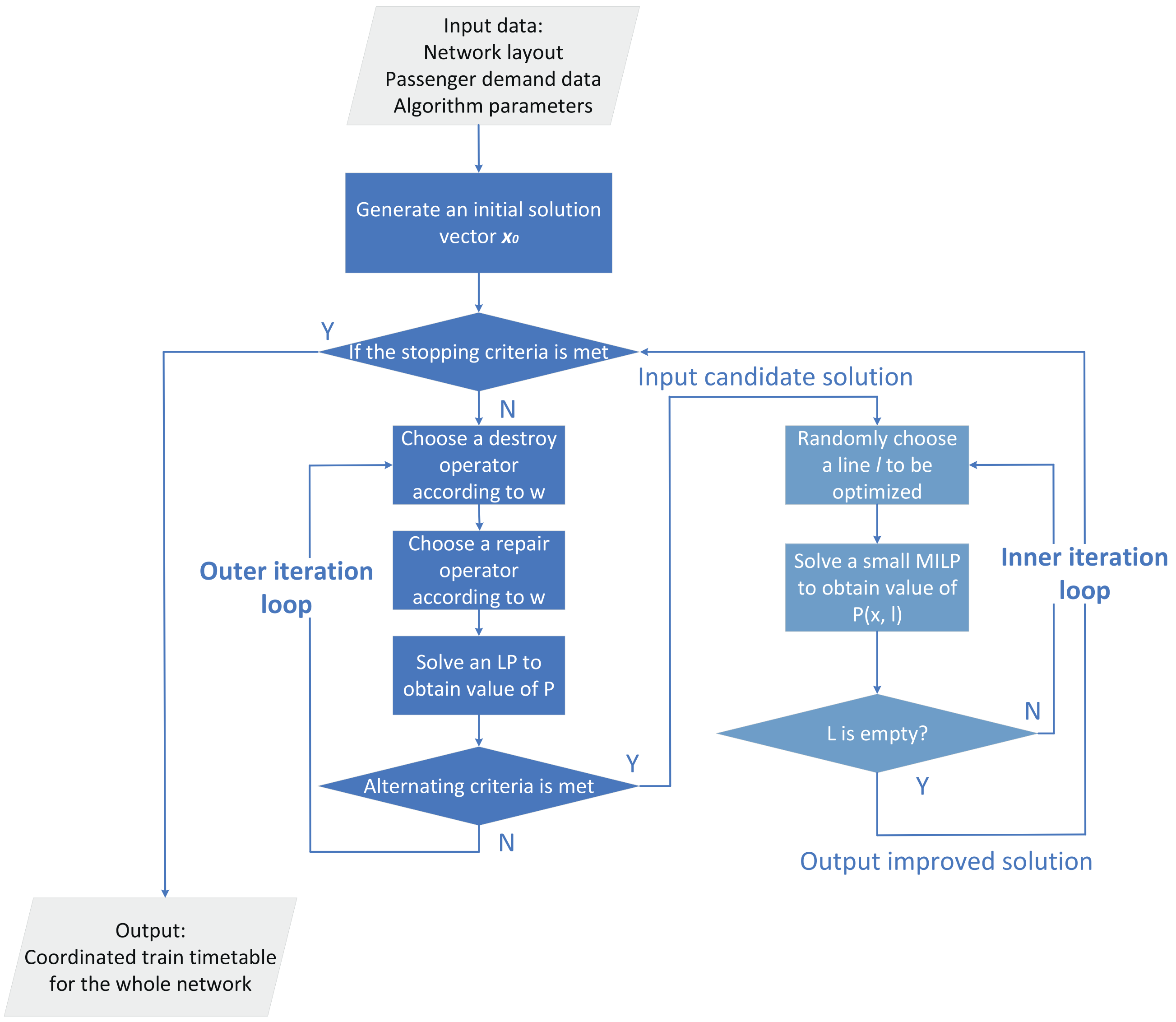

Input data:
Network layout
Passenger demand data
Algorithm parameters
Generate an initial solution vector x_0
Y
If the stopping criteria is met
N
Input candidate solution
Choose a destroy operator according to w
Choose a repair operator according to w
Solve an LP to obtain value of P
Alternating criteria is met
Y
N
Outer iteration loop
Randomly choose a line l to be optimized
Solve a small MILP to obtain value of P(x, l)
Inner iteration loop
L is empty?
N
Y
Output improved solution
Output:
Coordinated train timetable for the whole network

Fig. 7. Overall flowchart of DALNS.

of DALNS, we use ALNS to generate an initial solution. The outer-loop of DALNS adopts the proposed variable neighborhood search and a LP solver to evaluate candidate solutions. The main difference of DALNS with ALNS is that we define the following two *alternating criteria* in DALNS:

(i) The neighborhood search time exceeds a time limit $TS_{\max}$;

(ii) The number of searched neighborhood solutions exceeds the limit NUM_{sn}.

Remark 5.1. The alternating criteria differ from the stopping criteria as the settings of their values are different. In the alternating criteria, the values of $TS_{\max}$ and NUM_{sn} are smaller than T_{limit} and N_{limit} in the stopping criteria.

In the neighborhood search process of a candidate solution x, if the solution quality is not improved and one of the above alternating criteria is met, we use the current best solution x as an input candidate solution to generate a series of small-scale subproblems, according to Definition 5.1. Then, we begin a new process of inner-loop iterations. During each iteration, we randomly choose a line $\bar{l} \in L$, fix x_{lt} as the input solution x for $l \in L \backslash \bar{l}$ and set $x_{\bar{l}t}$ for $t \in T$ as the set of decision variables. Then, we use a MIP solver to solve the resulting small-scale integer linear programming model $P(x_{\bar{l}t}, \bar{l})$. If the obtained solution is a better solution, we use this newly produced solution as the next candidate solution and adopt ALNS to search the neighbors of this newly produced solution; otherwise, we update set L by removing $\bar{l}$ from L, i.e.,

$$L \leftarrow L - \bar{l}.$$

At the end of each iteration, we choose another set $x_{l't}$, where $l' \in L$, to repeat this process. This search process is terminated if L is empty and a new candidate solution is given to the outer-loop to begin a new search iteration. The general algorithmic procedure is summarized in Algorithm 2 (see Appendix D).

Remark 5.2. We need to mention here that, even though the overall procedures of ALNS and DALNS are similar, the underline difference between ALNS and DALNS is that DALNS explores a larger variable neighborhood starting from each candidate solution. In ALNS we use destroy and repair operators to explore the neighbors of a candidate solution x, while in DALNS we use not only destroy and repair operators, but also a series of "exact" solutions generated from the subproblems (see Lazić et al., 2010) to better explore possible improvements of the candidate solution. As we fix the train timetable of a part of the involved railway lines in DALNS, this algorithm can be focused on searching for better qual-

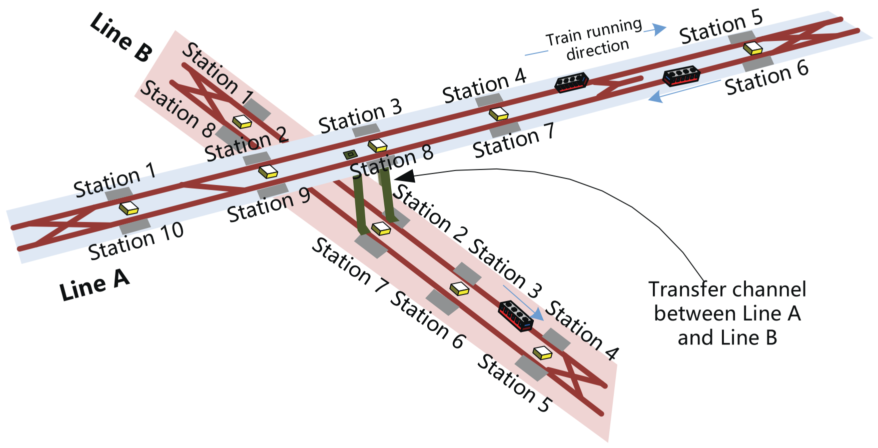


Fig. 8. Illustrative metro network with two lines.

ity solutions by coordinating the timetable of other lines, leading to potential improvements of the overall train schedule at the cost of an additional computation time (by MIP solver) to solve the subproblems for each single line.

6. Numerical experiments

In this section, we present the computational results on a small realistic case study and a real-world case study to illustrate the potential application and advantages of the proposed models and algorithms. These are coded by C++ on a workstation with a total of 32 Intel Xeon 6140 processors and 48GB RAM. The LP problems are solved by IBM ILOG CPLEX 12.1.

6.1. Small case study

The small realistic case study considers the transit rail network described in Fig. 8 as the experimental environment, which involves two bi-directional urban rail lines, termed as Line A and Line B. Since the lines are bi-directional, the station numbers in brackets represent stations in the down-direction for each line. For example, the trains of Line A depart from station 1, turn around at station 5 and finally go back to station 10. To avoid misunderstanding, we let S-A to represent station S of Line A. Thus, the connected stations of these two lines (i.e., transfer stations) are denoted by stations A3 (or A8 in the opposite direction) and B2 (or B7 in the opposite direction). For simplicity, we assume in this example that the capacity of each station is the same, and we can thus set the station capacity as 1 for illustrating convenience. In addition, the minimum time interval (i.e., the time between two consecutive time units t and $t+1$) is set as 30s. In order to simulate the peak-hour and off-peak hour, the time-dependent passenger demand is randomly generated as follows. We first divide the time horizon $|T|$ into two periods. For the first time period, we use a monotonically increasing parameter to indicate the passenger arriving rate; in the latter time period period, we generate a monotonically decreasing passenger arriving rate. Thus, the passenger demand distribution can be similar to realistic situations from peak-hour to off-peak-hour period. Then, we randomly generate OD pairs according to passenger arriving rate. The other parameters associated with this set of numerical experiments are summarized in Table 2.

For this case study, we assume that there is only one depot on each line and it is located at station 1. All the trains depart from station 1, turn around at the end of station 5 by using a given turn-around time and then return to the origin station 1 for the next cycle. The passenger transfer time is assumed to be constant along the different stations.

Table 2
Parameters of the rail network and traffic flows for the small case study.

Parameters	Line A	Line B
Dwelling time	30 s	30 s
Running time	120 s	120 s
Turn-around time	120 s	120 s
Minimum headway	150 s	150 s
Number of rolling stocks	7	6
Passenger transfer time	60 s	60 s
Maximum train capacity	3000	3000

After defining the above parameter setting, we next present two sets of numerical experiments. The first set of experiments aims to demonstrate the performance improvement that can be potentially achieved by coordinating train timetables. The second set of experiments compares the developed ALNS and DALNS algorithms with the benchmark solver CPLEX.

6.1.1. Comparison among different train timetables

We first consider the following experimental setting to illustrate the performance improvement that can be potentially achieved by coordinating train timetables. The considered time horizon is set as $|T| = 100$. As the scale is relatively small, we can use CPLEX to solve the model to optimality. For comparison, we generate three different train timetables for the rail network of Fig. 8, which are (i) a constant time headway timetable (i.e., Timetable with Average Headway, termed as TAH); (ii) a separatively optimized train timetable (i.e., Non-Coordinated Timetable, termed as NCT); (iii) the coordinated train timetable obtained by solving L-CTT to optimum. NCT is obtained by solving the L-CTT model for each line, respectively, without considering the passenger transfers. CPLEX takes about 2 minutes to obtain the NCT solution of Line A and Line B. For the coordinated train timetable, the computational intensity is much higher compared to cases (i) and (ii); and the CPLEX MIP solver takes nearly 30 minutes to generate a solution within a given small tolerance gap (i.e., less than 10^{-9}).

Remark 6.1. Here, we need to mention that both TAH and NCT are not optimally coordinated timetables. And TAH is generated by Algorithm 1, which is actually the initial solution of ALNS. The aim for comparing coordinated train timetable with TAH and NCT is to qualitatively analyze the value of performance improvement by adopting timetable coordinations.

The generated TAH, NCT and coordinated train timetable solutions for these two lines are presented in Figure 13 of Appendix E.

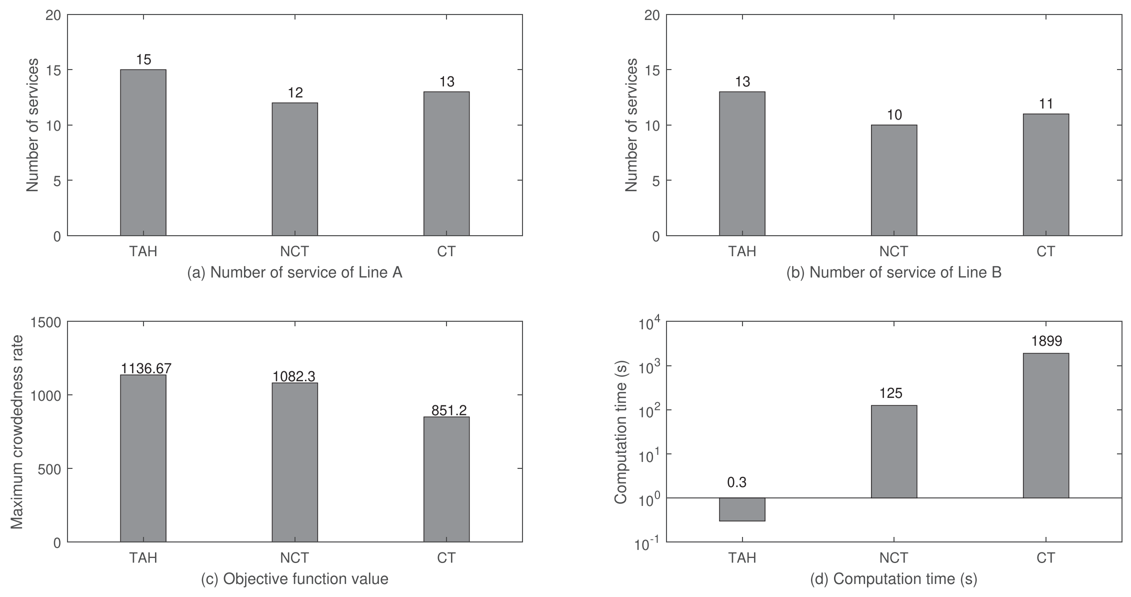


Fig. 9. Performance comparison of the three studied timetables.

We can see from Figures 13a-13b that TAH generates a timetable in which the train departure times for each line are constant. Meanwhile, from Figures 13c-13f, we visualize that the train headway by NCT is actually inconstant, since the latter solutions have to be consistent with the time-varying passenger arrival rates. In addition, the coordinated train timetable is different from NCT, indicating that the train departure times are adjusted, evidently for the purpose of train timetable coordination. Table 8 of Appendix F presents the detailed train service departure time (unit: each 30 seconds) by TAH, NCT and the coordinated train timetable for Line A and Line B.

Fig. 9 presents the performance of the three above mentioned timetables, which involves the number of train services (i.e., the number of train services that depart from the origin) for Line A and Line B, the Maximum Crowdedness Rate (i.e., the objective function value ξ of L-CTT, termed as MCR in the following content) and the computation time required by CPLEX to certify solution optimality. We observe that CPLEX takes more than half hour to obtain the optimal coordinated train timetable, which is much longer than solving TAH and NCT. Even though solving the coordinated train timetable takes a long time, the reduction of MCR is very evident. The coordinate train timetable can reduce the MCR by about 20% compared with the non-coordinate train timetables computed by TAH and NCT. This indicates the importance of coordinating train timetables of different lines in a busy metro network to reduce the crowdedness at transfer stations.

6.1.2. Comparison among various solution methods

In Fig. 9, the CPLEX MIP solver takes more than half hour to solve the L-CTT model. For instances with a longer time horizon, we first try to use CPLEX to solve the L-CTT model. We then use the developed TAH, ALNS, and DALNS to solve the L-CTT model and compare their performance with the results obtained by CPLEX. We set the maximum computation time as one hour (i.e., 3600 seconds), since this is the longest time that practitioners might accept to make and adjust train timetables during operational planning.

We generate a total of seven instances by varying a key parameter to test the performance of the various solution methods. In these instances, we gradually increase the number of time units, i.e., $|T|$, from 100 to 400, while the other parameters are set as in Section 6.1.1.

Table 3 provides a detailed performance comparison for the investigated instances. We report the number of services for each line, the objective function value and the total computation time. We also report the optimality gap obtained according to the best-known bounds. As can be seen in the results of Table 3, the CPLEX performance is far from optimum for the instances with a long time horizon (i.e. for instances $|T| = 250$, $|T| = 300$, $|T| = 350$, and $|T| = 400$). We remark that CPLEX is not able to compute the optimal solution after one hour of computation time. In order to provide a more convincing benchmark, we again use CPLEX to solve each of these instances to near-optimum through three days (i.e. 72 hours) of computation, until a solution with less than 0.1% gap optimality is found. The objective value of the generated near-optimal solutions clearly corresponds to the best-known objective value found for each instance. Then, we use the following formula:

$$\begin{aligned}&(\text{Objective value} - \text{Best-known objective value})\\&/(\text{Objective value}) \times 100\end{aligned}$$

to calculate an optimality gap for ALNS and DALNS, as reported in the last column of Table 3.

From the experimental results of Table 3, we conclude that the timetable with constant headways always yields the worst objective function value. This again demonstrates that, in a metro network with transfer stations, a well-designed timetable must carefully consider time-dependent passenger demands and the coordination of train schedules for the different involved lines. In addition, we observe that the CPLEX performance is the best for the smallest instance (i.e. when $|T| = 100$). In the latter case, the optimal solution is returned after half hour. However, as we gradually increase the number of time units (i.e., $|T|$), the CPLEX performance degenerates quickly. For more than 200 time units, the optimality gap is always larger than 20% after the 1-hour time limit of computation. Meanwhile, our developed ALNS and DALNS are much more time-efficient than CPLEX. For the relatively small-scale instances (e.g., $|T| = 100$), both ALNS and DALNS take much a shorter computation time and generate good quality solutions. For medium and large-scale instances (e.g., when $|T| \geq 200$), a noticeable performance improvement is achieved when using the ALNS and DALNS algorithms, because they can are able to compute high-quality solutions for the largest instances within a reasonable com-

Table 3
Computational results for TAH, CPLEX, ALNS, and DALNS.

Time units	Best known objective value[a]	Solution method[b]	Number of services for lines A and B	Objective function value	Computation time (in: seconds)	Opt Gap (%)
$\|T\| = 100$	851.2	TAH	15, 13	1136.7	-	-
		CPLEX	13, 11	851.2	1899.0	0.0
		ALNS	13, 11	878.3	89.8	3.1
		DALNS	13, 11	873.6	261.3	2.5
$\|T\| = 150$	940.8	TAH	17, 14	1308.7	-	-
		CPLEX	16, 14	1033.3	3600.2	24.1
		ALNS	16, 14	1054.2	134.8	10.8
		DALNS	16, 14	940.8	665.8	0.0
$\|T\| = 200$	1030.4	TAH	29, 25	1306.1	-	-
		CPLEX	27, 23	1119.4	3600.2	26.2
		ALNS	27, 23	1204.7	253.0	14.5
		DALNS	26, 21	1098.0	856.4	6.1
$\|T\| = 250$	1065.4	TAH	36, 32	1384.2	-	-
		CPLEX	35, 30	1140.5	3600.3	32.1
		ALNS	34, 20	1225.7	102.4	13.1
		DALNS	32, 26	1094.4	301.2	2.6
$\|T\| = 300$	1108.3	TAH	43, 38	1395.6	-	-
		CPLEX	40, 36	1200.0	3600.5	36.2
		ALNS	38, 33	1254.8	587.3	11.7
		DALNS	37, 33	1140.0	2338.7	2.8
$\|T\| = 350$	1080.0	TAH	50, 44	1409.3	-	-
		CPLEX	47, 41	1302.0	3600.9	49.3
		ALNS	47, 42	1301.5	121.3	17.0
		DALNS	47, 42	1134.0	223.5	4.8
$\|T\| = 400$	1130.2	TAH	58, 50	1442.0	-	-
		CPLEX	53, 43	1310.4	3600.9	42.2
		ALNS	52, 46	1270.8	189.8	11.1
		DALNS	55, 48	1168.0	262.4	3.2

[a] The best-known objective function value is obtained by CPLEX through 48-hour computation time and the returned optimality gap is less than 0.1%.
[b] In this column, “CPLEX” indicates that we use CPLEX to solve the L-CTT model within one-hour time limit of computation.

putation time, whereas the CPLEX performance becomes extremely poor. Furthermore, ALNS is much faster than DALNS, while the ALNS solution quality is much worse than the one of DALNS for most of the test instances. The possible reason is that ALNS gets “stuck” into a local optimum and the algorithm terminates. For example, in the instance with $|T| = 200$, ALNS consumes a total of 188 sec and obtains a solution with 14.5% optimality gap, while DALNS takes a longer computation time (856.4 seconds) but returns a much better solution. We observe that the longest computation time of DALNS for these instances is 2338.66 seconds (when $|T| = 300$), which is still less than one hour. To summarize, we conclude that DALNS outperforms the other approaches for solving the L-CTT model in terms of solution quality, while requiring a reasonable computation time.

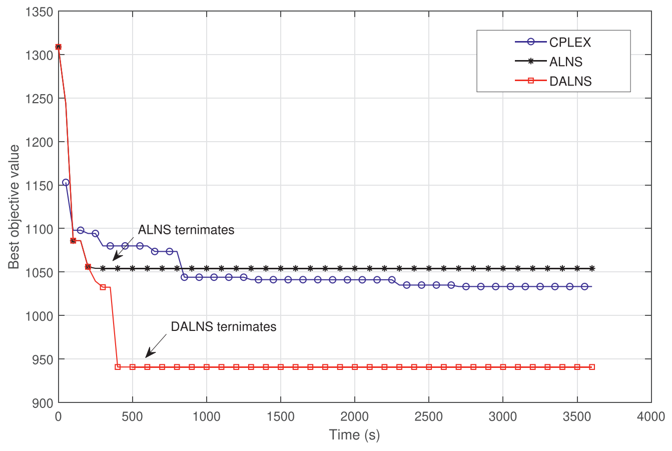
We next present the convergence tendency of the objective function values of CPLEX, ALNS and DALNS within one-hour computation time. Since the convergence tendency is similar, we only present the results of the instances with $|T| = 150$ and $|T| = 200$, as shown in Fig. 10. From these results, we can derive the following observations. (1) The initial solutions of ALNS and DALNS are both generated by TAH, while the first feasible solution that CPLEX finds is better than TAH. (2) The convergency rate of both ALNS and DALNS is very fast. In particular, even though the initial solution of DALNS is worse than that of CPLEX, the objective function value of DALNS is reduced quickly and outperforms CPLEX after around 150 to 300 seconds. (3) ALNS requires the shortest computation time, but the quality of the final solution computed by ALNS is the worst among these three methods. The possible reason is that ALNS finds a local optimal solution and terminates without improving this local optimum. (4) Within one-hour computation time, the best objective function value of CPLEX is gradually reduced, but its convergency rate is much slower than DALNS. Considering both computation time efficiency and best objective function value, we conclude that our developed DALNS outperforms the other methods.

6.2. Real-world case study

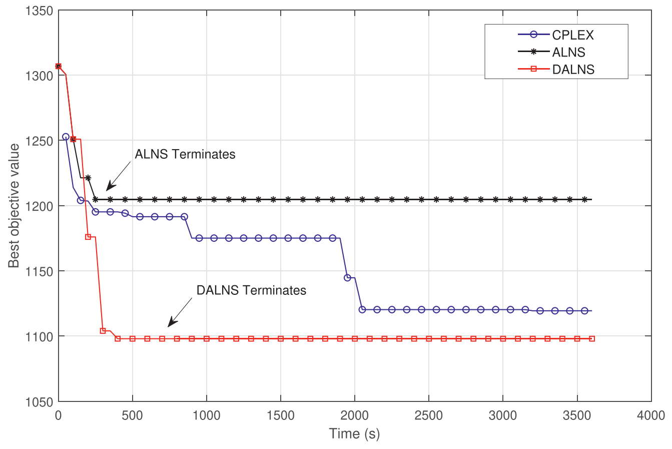
In this section, we consider a real-world case study on the Beijing metro network. In the studied instances, the dynamic parameters (i.e., time-dependent origin-destination passenger demands) are all taken from historically detected AFC data. We consider 11 practical instances to verify the effectiveness of the proposed approaches (i.e., ALNS and DALNS) for solving the L-CTT model. We do not use CPLEX for these instances, due to its very poor performance (i.e., since these instances cannot be solved by CPLEX after several hours of computation). For these instances, we have a new benchmark, i.e., the practical train timetable of Beijing metro system, given from metro managers.

6.2.1. Description of the experiments and parameter settings

As reported in Fig. 11, our experiments consider a total of three lines, i.e., Line No. 8, Line No. 13 and Line Changping, in Beijing metro network. The reason that we choose these lines is that the three lines constitute the northern part of Beijing metro and are relatively separate with other lines in the network. In morning peak hours, the three lines transport a large amount of commuters from suburban areas in the north of Beijing to downtown working areas. These lines contain a total of three transfer stations. Station Xierqi (termed as XEQ) connects line Changping and line 13. Station Zhuxinzhuang (ZXZ) connects line Changping and line 8. Station Huoying (termed as HY) connects line 13 and line 8. In peak hours, these stations suffer from a great amount of passengers, composed of both arriving passengers and transfer passengers from the other two lines. The operational parameters associated with these three lines (involving the train running time on each



(a) Instance with $|T| = 150$



(b) Instance with $|T| = 200$

Fig. 10. Convergence tendency of the objective function values within one-hour computation time (Black arrows indicates when the corresponding algorithm satisfies its stopping criteria and thus terminates.).

segment, the train dwell time at each station, the planned transfer time at each transfer station) are given in Appendix G. In these experiments, the minimum time unit is set as 10 sec, in order to be consistent with the operational parameters of the three lines.

To test the effectiveness of our approaches, we design a total of 11 instances under different scenarios and passenger demand distributions. The basic characteristics of these instances are presented in Table 4. In the latter table, the last column presents the total number of arriving passengers in the network for each investigated instance. Instances S1 and S2 represent a short time horizon (i.e., one hour). The time horizon considered in instances D1 to D6 is two hours. Instances L1 to L3 are the large-scale instances, and the corresponding time horizon is three hours.

6.2.2. Computational results

We now report the computational results on the real-world instances introduced in Section 6.2.1, involving the objective function value, the total computation time of each algorithm (T_{total}), and the computation time required by each algorithm to find its best solution (T_{best}). As ALNS and DALNS are adaptive metaheuristics, their results may vary due to the dynamic probability of choosing the operators. To conduct a fair comparison, we re-run each instance for multiple times (we set 10 runs in the experiments) by ALNS and DNLS, respectively; and then take the average values (and best values) for the reported quantitative indicators (i.e., the objective function value, T_{total}, and T_{best}). For this set of real-world instances, we use the practical timetable of Line No. 13, Line No. 8, and Line Changping as our benchmark.

Table 5 shows the computational results for the practical timetable plus the coordinated timetables obtained by ALNS and DALNS. Among all these instances, the coordinated timetables obtained by ALNS and DALNS outperform the current practical timetable (in 2019) for most of the instances. This is consistent with our conclusions in Section 6.1 that optimized timetable coordination (according to the dynamic passenger distribution) can reduce the crowdedness of metro stations. From the results of Table 5, we also observe as aother interesting feature that the objective function values for instance D5 obtained by the practical timetable and the coordinated timetables are exactly the same. The reason might be that the passenger demand is quite low in instance D5 (with the time horizon from 14:00 pm to 16:00 pm). From this practical instance, we conclude that the coordination of timetables may not always be required during off-peak hours, since the stations can be in some cases not significantly crowded. However, for the majority of the tested instances this is not the case, and the coordinated timetables offer very good results in terms of reducing station crowdedness.

When comparing the two proposed metaheuristics, the results of Table 5 show that DALNS reduces the average objective function value by about 8% to 20% compared to ALNS; however, DALNS takes a significantly longer computation time. This is consistent with the previous observations that DALNS requires a longer computation time to search for high-quality solutions. This is a meaningful observation for the practitioners, and they can adopt the proposed algorithms according to the practical requirements.

To analyze the influence of timetable coordination on the waiting time of passengers, we next report on a new set of experiments on the instances of Table 4. In these experiments, we use the solutions (i.e., train timetables) of Table 5 (i.e., the practical timetable and the timetables generated by ALNS and DALNS) as input. To perform the experiments in Fig. 12, we adopt formulation (37)–(44) that is reported in Appendix C. The latter formulation is used to minimize the total passenger waiting time for each timetable of Table 5. From the results in Fig. 12, we derive the following two observations: (1) The practical timetable yields the worst performance in most instances (except for instance D3), with respect to the total waiting time of passengers, compared with the results obtained by adopting our CTT-based timetables. The main reasons are that the practical timetable adopts a constant-headway timetable and the service frequency is not optimized with respect to the time-variant passenger demands. (2) In most cases, DALNS presents the smallest passenger waiting time compared with both the practical timetable and ALNS. Since DALNS is also the best algorithm when minimizing the maximum station crowdedness, we conclude that this algorithm and the proposed objective function can also indirectly reduce the total waiting time of passengers. More in general, these computational results let us conclude that minimizing the maximum station crowdedness might not necessarily cause a significant increase of the total passenger waiting time.

We next report on the unsatisfied passenger demand regarding the experiments in Fig. 12. Table 6 presents the average percentage of passengers that are still waiting at platforms at the end of the time period for each instance of Table 4. This information is computed by solving the LP model of Appendix C for the practical timetable and for the timetables generated by ALNS and DALNS. Specifically, we compute the following information for each station: $\sum_{i\in S} n_i^{|T|} / \sum_{i\in S}\sum_{t\in T} d_i^t$, in which $n_i^{|T|}$ is the number of untraveled passengers at station i at the end of the time

Table 4
Characteristics of the real-world instances.

Instance	Fleet size of Lines 1, 2 and 3	$\lvert T \rvert$ (unit: $\times$ 10 seconds)	Start time	End time	Total passenger demand (unit: $\times 10^3$)
S1	15, 28, 28	360	9:00	10:00	57.9
S2	15, 28, 28	360	8:00	9:00	63.4
D1	15, 28, 28	720	10:00	12.00	103.5
D2	15, 28, 28	720	12:00	14.00	110.0
D3	15, 28, 28	720	8:00	10:00	121.3
D4	15, 28, 28	720	6:00	8:00	108.9
D5	15, 28, 28	720	14:00	16:00	102.1
D6	15, 28, 28	720	16:00	18:00	111.8
L1	15, 28, 28	1080	12:00	15:00	132.3
L2	15, 28, 28	1080	16:00	19:00	147.4
L3	15, 28, 28	1080	6:00	9:00	172.3

Table 5
Computational results for ALNS and DALNS on the real-world instances.

Instance	Practical	ALNS			DALNS		
Index	timetable[a]	Objective function value	T_{total} (seconds)	T_{best} (seconds)	Objective function value	T_{total} (seconds)	T_{best} (seconds)
S1	698	688 (598)[b]	218.92	145.2	541 (533.2)[b]	1312.5	1179.3
S2	688	676 (602.5)	188.3	89.3	608 (591.4)	1263.9	117.99
D1	817	783 (734)	331.8	121.7	712 (709)	3234.4	318.18
D2	837	824.4 (819)	581.3	198.1	762 (759)	4379.1	1318.4
D3	1050.8	930 (925)	1231.2	313.2	924.2 (918.2)	6325.7	924.16
D4	586	521 (498.2)	1013.1	134.1	488 (479)	4771.5	893.2
D5	576	576 (576)	213.2	10.9	576 (576)	2133.2	10.9
D6	902.7	900.2 (854)	632.3	100.8	766 (759)	6917.2	6821.1
L1	1000.3	987.3 (933.5)	321.4	129.0	910 (903.2)	12127.9	3212.3
L2	1290.7	1250.3 (1235.2)	439.3	204.2	1200.7 (1192.3)	15527.5	12153.3
L3	1422.3	1420.4 (1401)	1293.2	211.3	1337.2 (1316.4)	8791.2	4512.9

[a] The timetable used during year of 2019 in the Beijing metro system does not consider any coordination strategy among the connected lines.
[b] The values in the brackets of this column report the best objective function value.

period T while $\sum_{i\in S}\sum_{t\in T} d_i^t$ is the total passenger demand. The $\sum_{i\in S} n_i^{|T|}/\sum_{i\in S}\sum_{t\in T} d_i^t$ values are shown in percentage. As can be observed from Table 6, only a small percentage of passengers is not served for each station at the end of time period T with respect to the total number of passengers. Moreover, the DALNS timetable outperforms the practical timetables for all the considered instances, while ALNS is better than the practical timetable in all the instances but one (D2). In Appendix H, Figures 15 - Figure 18 show the number of waiting passengers at various stations and for two interesting timetables of Table 5 (i.e., for instances D6 and L3, for which Fig. 12 shows relatively large gaps, in terms of passenger waiting time, among the different timetables).

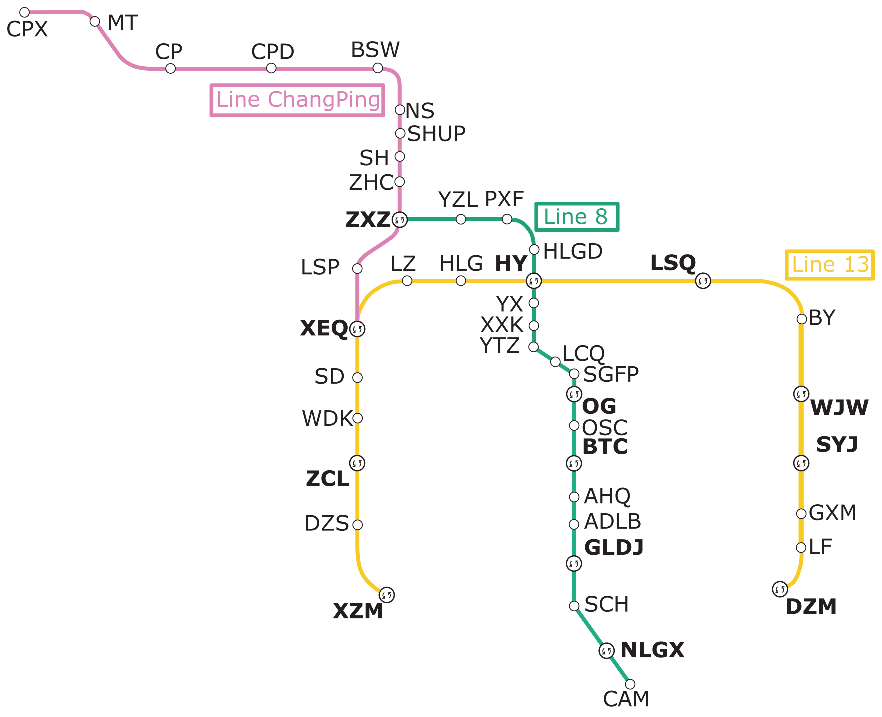


Fig. 11. Line No. 8, line No. 13, and line ChangPing of the Beijing metro network.

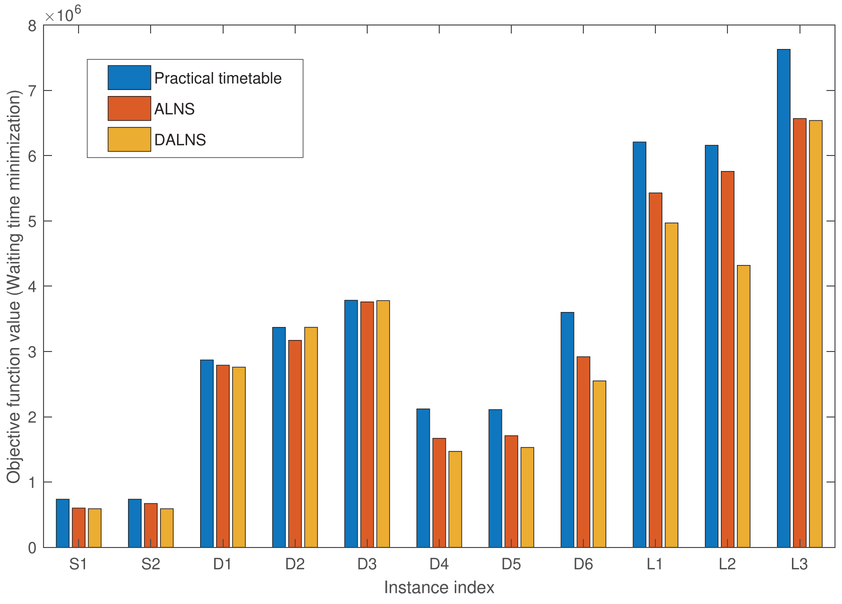


Fig. 12. Comparison among timetables in terms of total passenger waiting time.

Table 6
The average percentage of untraveled passengers for each station at the end of the studied time period.

Instance	S1 (%)	S2 (%)	D1 (%)	D2 (%)	D3 (%)	D4 (%)	D5 (%)	D6 (%)	L1 (%)	L2 (%)	L3 (%)
Practical timetable	2.92	4.11	1.56	1.18	1.65	2.56	1.37	2.15	1.04	1.36	1.22
ALNS	2.88	3.74	1.49	1.18	1.50	2.51	1.26	2.05	1.00	1.31	1.19
DALNS	2.81	3.69	1.51	1.11	1.42	2.53	1.26	1.95	1.02	1.33	1.01

7. Conclusions and future research

This paper addressed the optimal coordination of train timetables in an urban rail network under the consideration of time-dependent passenger demands. We have proposed a new mathematical formulation to generate coordinated train timetables for all the involved lines in a rail transit network synchronously (i.e., at a network-wide level), in order to minimize the maximal level of crowdedness at transfer stations. We have introduced the required sets of passenger flow variables to microscopically formulate this complex railway scheduling problem as a mixed integer linear programming formulation. To obtain high-quality solutions for large-scale problems, we have also developed two problem customized metaheuristic algorithms.

To verify the effectiveness of the proposed coordination-based train timetabling approaches, two sets of numerical experiments have been proposed, i.e., a small realistic case study and a real-world case study on the Beijing metro network. The computational results show that, for medium-scale and large-scale instances, the developed ALNS and DALNS algorithms outperform CPLEX evidently. Furthermore, these two algorithms can obtain good quality solutions in a reasonable computation time for operational planning/timetabling. Based on the real-world case study on the Beijing metro network, the computational results show that the crowdedness of metro stations can be reduced by about 8% to 20% by the coordinated timetables when compared with the practical (non-coordinated) timetable.

Future research can be focused on the following aspects. (1) This study assumes that the travelling path of passengers is predetermined by historical data. Nevertheless, the passengers in practice may also choose different travelling paths according to the actual train timetable. Thus, analyzing the passenger travelling behavior and consequently extending our work to a bi-level optimization model can be an interesting research direction. (2) The transfer of passengers between different lines is a very complex stochastic process affected by many uncertain factors, such as the layout and structure of transfer stations, the composition of passengers with different movement speed (for example, male/female, disabled people, old people, and children). Future research can address the coordination of train timetables with the consideration of these uncertainty factors. (3) The study of integrated and connected multimodal transport systems is also a research trend for the development of future urban mobility ecosystems. The methodology proposed in this study can also be extended to coordinate a multimodal (for example, a bus-metro network) system to transport users seamless and to implement better connected mobility services. (4) Another interesting direction for future research is to investigate efficient algorithms for even larger timetabling problems (e.g., the metro traffic control at regional level). From the computational results, we can conclude that the proposed methodology can efficiently solve the CTT problem for multiple lines, while we acknowledge that further research is required to deal with even more complex networks or to consider additional practical requirements of the rail managers into the optimization model (e.g., the minimization of further passenger-related and/or cost-related aspects). The latter aspects would require to deal with new challenging issues, and future research could also be, e.g., focused on the study of advanced machine learning and artificial intelligence techniques in combination of our optimization-based approaches to solve extended CTT problems.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (No. 71901016, 72071016, and 71825004), the Beijing Natural Science Foundation (No. L191015) and the State Key Laboratory of Rail Traffic Control and Safety (No. RCS2020ZZ004).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2021.02.059.

References

Abdolmaleki, M., Masoud, N., & Yin, Y. (2020). Transit timetable synchronization for transfer time minimization. *Transportation Research Part B, 131*, 143–159.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2012). Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science, 48*(1), 20–45.

Barrena, E., Canca, D., Coelho, L., & Laporte, G. (2014a). Exact formulations and algorithm for the train timetabling problem with dynamic demand. *Computers & Operations Research, 44*, 66–74.

Barrena, E., Canca, D., Coelho, L., & Laporte, G. (2014b). Single-line rail rapid transit timetabling under dynamic passenger demand. *Transportation Research Part B, 70*, 134–150.

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

Cadarso, L., & Marin, A. (2012). Integration of timetable planning and rolling stock in rapid transit networks. *Annals of Operations Research, 199*(1), 113–135.

Canca, D., Barrena, E., Algaba, E., & Zarzo, A. (2014). Design and analysis of demand-adapted railway timetables. *Journal of Advanced Transportation, 48*(2), 119–137.

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

Cao, Z., Ceder, A., Li, D., & Zhang, S. (2019). Optimal synchronization and coordination of actual passenger-rail timetables. *Journal of Intelligent Transportation Systems*, 1–19.

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

Ceder, A., Golany, B., & Tal, O. (2001). Creating bus timetable with maximal synchronization. *Transportation Research Part A, 35*(10), 913–928.

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

Corman, F., D'Ariano, A., Marra, A., Pacciarelli, D., & Samà, M. (2017). Integrating train scheduling and delay management in real-time railway traffic control. *Transportation Research Part E, 105*, 213–239.

Corman, F., & Meng, L. (2014). A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems, 16*(3), 1274–1284.

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

Demir, E., Bektas, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research, 223*(2), 346–359.

Domencich, T., & McFadden, D. (1975). *Urban travel demand: a behavioral analysis*. North-Holland: North-Holland Publishing Company.

Espinosa-Aranda, J., García-Ródenas, R., del Carmen, R.-F. M., López-García, M., & Angulo, E. (2015). High-speed railway scheduling based on user preferences. *European Journal of Operational Research, 246*(3), 772–786.

Fischetti, M., & Monaci, M. (2017). Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research, 263*(1), 258–264.

Fouilhoux, P., Ibarra-Rojas, O. J., & Rós-Solś, Y. A. (2016). Valid inequalities for the synchronization bus timetabling problem. *European Journal of Operational Research, 251*(2), 442–450.

Gangdi, M., Cantarella, G., & Vitetta, A. (2019). Solving stochastic frequency-based assignment to transit networks with pre-trip en-route path choice. *EURO Journal on Transportation and Logistics, 8*, 661–681.

Gendreau, M., & Potvin, J. (2018). Handbook of metaheuristics. *International Series in Operations Research & Management Science*, 272.

Hansen, I., & Pachl, J. (2014). *Railway timetabling & operations*. PMCMedia/Eurailpress.

Hassannayebi, E., & Zegordi, S. (2017). Variable and adaptive neighbourhood search algorithms for rail rapid transit timetabling problem. *Computers & Operations Research, 78*, 439–453.

Ibarra-Rojas, O. J., Delgado, F., Giesen, R., & Muñoz, J. C. (2016). Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B, 77*, 38–75.

Ibarra-Rojas, O. J., & Ríos-Solís, Y. A. (2012). Synchronization of bus timetabling. *Transportation Research Part B, 46*(5), 599–614.

König, E., & Schön, C. (2021). Railway delay management with passenger rerouting considering train capacity constraints. *European Journal of Operational Research, 288*(2), 450–465.

Kroon, L. G., Peeters, L. W., Wagenaar, J. C., & Zuidwijk, R. A. (2014). Flexible connections in PESP models for cyclic passenger railway timetabling. *Transportation Science, 48*(1), 136–154.

Lazić, J., Hanafi, S., Mladenovíc, N., & Urošević, D. (2010). Variable neighbourhood decomposition search for 0–1 mixed integer programs. *Computers & Operations Research, 37*, 1055–1067.

Lejeune, M. (2006). A variable neighborhood decomposition search method for supply chain management planning problems. *European Journal of Operational Research, 175*, 959–976.

Liu, S. Q., & Kozan, E. (2009). Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Operations Research, 36*(10), 2840–2852.

Liu, T., & Ceder, A. (2018). Integrated public transport timetable synchronization and vehicle scheduling with demand assignment: A bi-objective bi-level model using deficit function approach. *European Journal of Operational Research, 117*, 935–955.

Louwerse, I., & Huisman, D. (2014). Adjusting a railway timetable in case of partial or complete blockades. *European Journal of Operational Research, 235*(3), 583–593.

Masson, R., Lehuédé, F., & Péton, O. (2013). An adaptive large neighbourhood search for the pickup and delivery problem with transfers. *Transportation Science, 47*(3), 344–355.

Mejía, G., & Yuraszeck, F. (2020). A self-turning variable neighborhood search algorithm and an effective decoding scheme for open shop scheduling problems with travel/setup times. *European Journal of Operational Research, 285*(2), 484–496.

Mesa, J. A., Ortega, F. A., & Pozo (2014). Locating optimal timetables and vehicle schedules in a transit line. *Annals of Operations Research, 222*, 439–455.

Niu, H., Zhou, X., & Gao, R. (2015). 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, 76*, 117–135.

Ortega, F. A., Pozo, M. A., & Puerto, J. (2018). On-line timetable rescheduling in a transit line. *Transportation Science, 52*(5), 1106–1121.

Pellegrini, P., Marliére, G., & Rodriguez, J. (2014). Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B, 59*, 58–80.

Pisinger, D., & Ropke, S. (2010). Large neighborhood search. In M. Gendreau (Ed.), *Handbook of metaheuristics* ((2 ed.)). Springer.

Robenek, T., Azadeh, S., Maknoon, Y., de Lapparent, M., & Bierlaire, m. (2018). Train timetable design under elastic passenger demand. *Transportation Research Part B, 111*, 19–38.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455–472.

Shi, J., Yang, L., Yang, J., & Gao, Z. (2018). Service-oriented train timetabling with collaborative passenger flow control on an oversaturated metro line: An integer linear optimization approach. *Transportation Research Part B, 110*, 26–59.

Shi, J., Yang, L., Yang, J., Zhou, F., & Gao, Z. (2019). Cooperative passenger flow control in an oversaturated metro network with operational risk thresholds. *Transportation Research Part C, 107*, 301–336.

Vansteenwegen, P., & Oudheusden, D. V. (2006). Developing railway timetables which guarantee a better service. *European Journal of Operational Research, 173*(1), 337–350.

Veelenturf, L. P., Kroon, L. G., & Maróti, G. (2017). Passenger oriented railway disruption management by adapting timetables and rolling stock schedules. *Transportation Research Part C, 80*, 133–147.

Wang, D., Zhao, J., D'Ariano, A., & Peng, Q. (2020). Simultaneous node and link districting in transportation networks: Model, algorithms and railway application. *European Journal of Operational Research, 292*, 73–94.

Wang, Y., D'Ariano, A., Yin, J., Meng, L., Tang, T., & Ning, B. (2018). Passenger demand oriented train scheduling and rolling stock circulation planning for an urban rail transit line. *Transportation Research Part B, 118*, 193–227.

Wang, Y., Tang, T., Ning, B., van den, B. T., & De Schutter, B. (2015). Passenger-demands-oriented train scheduling for an urban rail transit network. *Transportation Research Part C, 60*, 1–23.

Wong, R., Yuen, T., Fung, K., & Leung, J. (2008). Optimizing timetable synchronization for rail mass transit. *Transportation Science, 42*(1), 57–69.

Xu, X. Y., Liu, J., Li, H. Y., & Hu, J. Q. (2014). Analysis of subway station capacity with the use of queueing theory. *Transportation Research Part C, 38*, 28–43.

Yin, J., D'Ariano, A., Wang, Y., Yang, L., & Tang, T. (2019). Mixed-integer linear programming models for coordinated train timetabling with dynamic demand. In *Proceedings of the ieee international conference on intelligent transportation systems, Auckland, NZ*.

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

Yuan, L., Chen, H., & Prins, C. (2016). Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research, 252*(1), 27–38.

Zhang, Y., D'Ariano, A., He, B., & Peng, Q. (2019). Microscopic optimization model and algorithm for integrating train timetabling and track maintenance task scheduling. *Transportation Research, Part B, 127*(1), 237–278.

Zhou, X., & Zhong, M. (2005). Bicriteria train scheduling for high-speed passenger railroad planning applications. *European Journal of Operational Research, 167*(3), 752–771.

Žulj, I., Kramer, S., & Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research, 264*(2), 653–664.