Rescheduling subway trains by a discrete event model considering service balance performance

Xiaoming Xu, Keping Li, Lixing Yang

State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China
Beijing Laboratory of Urban Rail Transit, Beijing Jiaotong University, Beijing 100044, China

Article history:
Received 30 October 2013
Revised 20 February 2015
Accepted 28 June 2015
Available online 8 September 2015

Keywords:
Subway line
Train rescheduling
Crossover track
Discrete event model

Abstract

Considering an incident on a track of a double-track subway line, this paper formulates an optimization model to find near-optimal rescheduled timetables with the least total delay time compared to the original one, in which crossover tracks connecting two parallel subway lines are particularly taken into consideration for balancing the service quality under emergent situations. On the basis of a discrete event model where the train position state transitions are characterized as a series of discrete events, an efficient train rescheduling strategy (ETRS) is developed for solving the proposed model, in which a well-developed capacity check algorithm is particularly integrated to prevent the potential deadlocks. By using the infrastructure data of Beijing Yizhuang subway line of China, numerical case studies are implemented to demonstrate the effectiveness and efficiency of the proposed model and algorithm.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

1.1. Motivation

It is widely recognized that subway traffic is an ideal transport mode to relieve the traffic pressure in the large cities due to its inherent features of high-capacity, low-energy consumption and safety. In the real-world applications, however, a variety of factors might lead to the reduction of link capacities owing to the complicate traffic environment, such as the rail deformation, locomotive default and signal system failure. When an incident occurs, the current train timetable will probably be ineffective, and an efficient and rapid rescheduling method is expectedly developed to recover the subway traffic.

To the best of our knowledge, the existing subway system usually consists of two parallel tracks (main tracks) for services in different directions and a number of crossover tracks connecting two main tracks. Without the incident, trains are required to traverse on main tracks according to the pre-specified train schedule, while under the incident (emergency), crossover tracks are allowable to be used as the emergency response in rescheduling involved trains. In reality, according to situations of the line failure, two emergency response methods are usually adopted in the real-world operations. Specifically, (1) when the entire line is paralyzed, the passenger service will be suspended until the line is recovered; and (2) when a part of the line is available, the operation plan will be adjusted in real-time by possibly using the crossover tracks. In general, however, due to lack of real-time and efficient rescheduling methods, dispatchers always employ the first response method to deal with most cases because of its operational simplicity. For instance, on March 26, 2012, the collapsed London Jubilee line, leading to the disruption...
of the train operations, causes more than 350 passengers to walk through the darkened tunnel to safety areas. Typically, if the crossover tracks can be reasonably used to reschedule trains under emergencies, we can be expected to reduce the impacts caused by the incidents, and then recover the subway system as soon as possible.

Note that a few researches in literature focus on the aforementioned train rescheduling problem for subway systems after the occurrence of incidents. This paper will explicitly address this issue. In detail, we particularly investigate a subway train rescheduling problem considering the possible use of crossover tracks. Since the proposed rescheduling problem is essentially an NP-complete mixed inter programming (MIP) problem, we shall also design an efficient heuristic algorithm for the proposed model, which can reschedule large-scale problems within a short computational time.

1.2. Literature review

Train scheduling/rescheduling problems, which are viewed as one of the most crucial problems for train operations, had attracted tremendous attentions from numerous researchers over the last decades. Theoretically, this type of problem, which is often formulated as a job-shop scheduling problem with additional safety and operational constraints [1], is proved to be a NP-complete mixed inter programming (MIP) problem [2]. The earlier work by Amit and Goldfarb [3] first applied mathematical programming technique to the train scheduling problem. Then, some important fundamental works have been further developed by several researches such as Szpigel [4], Higgins et al. [5], Cai et al. [6], etc. In general, the majority of existing researches about train scheduling/rescheduling problem in literature can be roughly classified into two categories, termed as accuracy-first scheduling/rescheduling methods and efficiency-first scheduling/rescheduling methods.

For accuracy-first scheduling, the aim is to formulate a rigorous optimization model for a particular problem with different objectives, such as minimizing total delay, total passenger travel/waiting time, energy consumption, etc. To generate optimal (at least near-optimal) solutions, the solution algorithms always include the branch and bound algorithm, genetic algorithm (GA), LINGO, CPLEX, GAMS optimization softwares, etc. For example, Szpigel [4] firstly formulated the train scheduling problem as a MIP on a single-line railway, and proposed a branch-and-bound algorithm in their research. Recently, Zhou and Zhong [7] formulated a generalized resource-constrained model for single-track train scheduling problem, and proposed a branch and bound algorithm-based solution. Aiming to minimize the train delay time and the total passenger-time, Yang et al. [8] proposed an effective expected value programming model, and solved their model by using a branch and bound algorithm, in which the numbers of passengers loading/unloading trains at stations are assumed to be fuzzy variables. Considering the passenger travel time and energy consumption, Ghoseiri et al. [9] formulated a multi-objective optimization model for train scheduling problem on a network, which is solved by LINGO optimization software to generate a Pareto frontier. Analogously, Zhou and Zhong [10] investigated a double-track train scheduling problem with two considerations, i.e., minimizing expected waiting time for high-speed trains, and minimizing the total travel time of high-speed and medium-speed trains. They generated Pareto solutions based on a developed branch and bound algorithm for the multi-objective model, and applied a beam search algorithm to constructing solutions. For minimizing energy consumption and travel time, Yang et al. [11] formulated an efficient mathematical model, and used GA to search for the optimal train movement strategy through considering the global information of the railway network.

Li et al. [12] studied the robust cruise control scheduling for high speed train on the basis of the sampled-data. To make full use of railway networks, Mu and Dessouky [13] set up two mathematical models where fixed paths and alternative paths for freight trains were taken into account. Based on the proposed models, several heuristic solution approaches are developed. Besides, Cacchiani et al. [14] formulated an integer linear programming formulation to introduce as many new freight trains as possible and keep the prescribed timetables for pre-planned trains, where the problem is solved by a developed Lagrangian heuristic. At the enterprise application level, Mannino and Mascis [15] developed a practical real-time automated traffic control system to operate trains in metro stations. In this system, the train control problem was represented as a job-shop scheduling, where some linear programs were solved by CPLEX software.

For train rescheduling problem, for instance, Cormann [1] formulated the inter-area coordination of train rescheduling problem as a bi-level programming problem, where some constraints may be imposed at the border of each dispatching area by a coordinator, based on which, regional dispatchers then reschedule trains. Moreover, Meng and Zhou [16] studied the single-track train rescheduling problem on the basis of a stochastic programming and a branch and bound-based heuristic. Recently, Yang et al. [17] proposed a two-stage fuzzy optimization model for the train rescheduling problem on a double-track railway line, where a scenario-based fuzzy recovery time was taken into account, and the GAMS software was used to solve their model. With the consideration of the influence that a disruption brings to passenger demand, Cadarso et al. [18] proposed a two-step approach combining an integrated optimization model (for the timetable and rolling stock) with a model for the passengers’ behavior. And they used GAMS/CPLEX to implement their models. For maximizing the service level offered to passengers when unexpected events occur, Louwerse and Huisman [19] developed several models with different considerations, including partial blockade, complete blockade and regularity, to determine disposition timetables, specifying which trains will still be operated during the disruption and determining the timetable of these trains. In both Meng and Zhou [20] and Yang et al. [21], the time–space network is applied to formulating train rescheduling models. Meng and Zhou [20] solved their model by their Lagrangian relaxation based solution, while credibility measure was taken into account in the model by Yang et al. [21], and GAMS software was used to search for optimal solutions. Furthermore, for getting a better understanding of train rescheduling problem, we can refer to

---

Cacchiani et al. [22], where a detailed and systematic review of recovery models and solution approaches for real-time railway disturbance and disruption management is given.

In reality, due to the complexity of train scheduling/rescheduling models, it is generally difficult to find the optimal solution rapidly or within an acceptable computational time. To speed up the solution process, numerous researches aim to generate high-quality timetable through simulation techniques, expert systems and heuristics. In other words, efficiency-first methods focus on creating train timetables with a relatively short computational time at the cost of sacrificing optimization. Typically, however, these methods are practically useful in the real-world applications, e.g., rescheduling trains under the emergency response. In literature, to minimize the average consecutive delay, Sahin [2] considered the real-time conflict resolution problem on a single-track railway as a job-shop scheduling problem, and designed an algorithm based on look-ahead strategies to predict potential consecutive delay and reorder trains. Using the alternative graph model, D’Ariano et al. [23] formulated a variable-speed model to update the train speed profiles and developed two simple rules (i.e., first-in-first-out (FIFO) rule, first-out-first-in (FOFI) rule), a greed heuristic algorithm (i.e., avoid most critical completion (AMCC) time algorithm) and a branch and bound algorithm to solve conflicts. Analogously, these algorithms were also adopted in D’Ariano et al. [24–26]. Albrecht et al. [27] studied the rescheduling problem with considering maintenance disruptions. A problem space search (PSS) meta-heuristic, which can rapidly generate a large number of rescheduled timetables, was developed in their studies. Considering the influence that the disturbances bring to railway transportation, Krasemann [28] designed a greedy algorithm to effectively generate rescheduling timetables.

Moreover, based on the discrete event model (DEM), Dorfman and Medanic [29], Li et al. [30], and Xu et al. [31–33] developed several types of high-efficiency train travel advance strategies. Specifically, as for efficiency-first methods, Dorfman and Medanic [29] proposed a high-efficiency train scheduling algorithm by using a DEM, called greedy travel advance strategy (TAS) method, which can schedule a larger number of trains on a railway network within a short computational time. However, the TAS still does not consider the global information in scheduling process, leading to a relatively large gap between the generated schedule and the optimal schedule. Enlightened by Dorfman and Medanic [29], Li et al. [30] further developed an improved train scheduling algorithm, namely efficient travel advance strategy (ETAS), by considering the global scheduling information. In their algorithm, the trains’ micro operations (i.e., trains’ acceleration and deceleration) are also taken into account. Additionally, aiming to reduce trains coupling effects, Xu et al. [31] developed a genetic algorithm-based travel advance strategy (GA-ITAS) to search for a balanced timetable (i.e., a timetable with the least delay-ratio) on a single-track railway, where the train velocity is specified as a decision variable instead of a constant, and the ITAS method is an improved TAS with the modification of traversing order rules.

1.3. Proposed approach

We note that although the efficiency of aforementioned algorithms developed on the basis of DEM is well verified, the train rescheduling strategies by using DEM-based methods still have not attracted high attention from the researchers in the literature. With this concern, aiming to efficiently reschedule trains, we are particularly interested in developing an efficient train rescheduling strategy (ETRS) for a faulted double-track subway line within the framework of discrete event models. This paper aims to make the following contributions to the study of train rescheduling problems:

(1) A rigorous rescheduling formulation is proposed for the subway operation systems after the occurrence of an incident. In this model, once the incident takes place on a track, the impacted trains are allowed to be rescheduled by using tracks in the opposite direction through crossover tracks, which turns out to be a railway network-based rescheduling problem. The purpose is to generate an optimal rescheduling plan with the least total delay with respect to the original one. As the existing researches usually do not consider the crossover tracks in the rescheduling process, this is actually a new idea in the literature.

(2) An efficient train rescheduling strategy (ETRS) is developed to generate high-quality solutions for the proposed model. Enlightened by Dorfman and Medanic [29], we particularly represent the rescheduling process as a DEM, and design a series of deadlock-free rescheduling strategies for trains in different situations. Furthermore, the dwelling time is also taken into account in defining the dynamic events. Compared to Dorfman and Medanic [29], the algorithmic efficiency is further improved by using a newly designed capacity check procedure. The numerical experiments demonstrate that the proposed approaches can generate high-quality schedules rapidly in comparison with the results created by GAMS.

In order to clarify the contributions of related studies in literature, Table 1 lists some detailed comparisons based on the following four characteristics, i.e., the problem focus, algorithm, algorithm characteristics and solution characteristics.

The rest of this paper is organized as follows. In Section 2, detailed description of the rescheduling process is presented, and a sample example is given to illustrate the process. In Section 3, the formulation of our rescheduling model is described in detail. Furthermore, we present the efficient train rescheduling strategy (ETRS) method in Section 4. With the infrastructure data from Beijing Yizhuang subway line and GAMS optimization software, Section 5 gives some case studies to demonstrate the effectiveness and efficiency of the proposed model and method. A conclusion is made in the last section.

2. Problem description

As shown in Fig. 1, let us consider a double-track subway line consisting of  \( n \) stations, \( n - 1 \) sections and several crossover tracks, where shaded rounded-corner rectangles represent the platforms of individual stations. In this infrastructure, both crossover tracks and main tracks are allowable to be used by trains in both directions under emergency states. For instance,
Table 1
The comparison of different characteristics of closely related methods.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>Algorithm characteristics</th>
<th>Solution characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dorfman and Medanic [29]</td>
<td>Scheduling</td>
<td>TAS</td>
<td>An algorithm developed based on a DEM</td>
</tr>
<tr>
<td>Li et al. [30]</td>
<td>Scheduling</td>
<td>ETAS</td>
<td>Improved TAS with trains' micro operations and global information</td>
</tr>
<tr>
<td>Xu et al. [31]</td>
<td>Scheduling</td>
<td>GA-ITAS</td>
<td>Improved TAS with modification of traversing rules and GA</td>
</tr>
<tr>
<td>This paper</td>
<td>Rescheduling</td>
<td>ETRS</td>
<td>Another algorithm developed based on a DEM, different from TAS</td>
</tr>
</tbody>
</table>

when an incident occurs on an inbound section (e.g., section $k - 1$), inbound trains can be rescheduled to traverse the outbound track through a crossover track (e.g., track $c + 1$), and then return back to the inbound track through another crossover track (e.g., track $c$). By this dispatching strategy, the impacted trains can be expected to continue their remaining trips for keeping the service balance in both directions even if the inbound railway line is broken. To our knowledge, however, the function of crossover tracks is often omitted in major researches of subway rescheduling in literature, in which impacted trains are in general rescheduled right after the incident links are recovered, leading to the inefficiency and unbalance of the transportation services. Next, an example is given to explicitly illustrate the importance of crossover tracks in rescheduling process.

Take a simplified subway line shown in Fig. 2 as an example, in which there are one island platform in stations 1, 2, 3 and 6, and two side platforms in stations 4 and 5. Ten trains, donated by $1, 2, \ldots, 10$, will be scheduled on this subway system. In detail, trains $1, 3, \ldots, 9$ are pre-planned to traverse in inbound direction, while trains $2, 4, \ldots, 10$ head for outbound direction. The favorite timetable, which specifies trains arrival/departure time at each station, is depicted in Fig. 3.
Suppose that Section 3 along the inbound direction is broken (see the shaded rectangle in Fig. 2) at time 600 s and recovered at time 1200 s. Obviously, due to the incident duration and locations of crossover tracks on the subway line, both Sections 3 and 4 in inbound direction will be invalid during the incident period. To recover the traffic service as soon as possible, we here consider two rescheduled timetables with different rescheduling strategies (Here, timetables are actually dispatching solutions. See Remark 2.1 for more details), shown in Fig. 4, in which timetable (I) is generated by using two crossover tracks while timetable (II) is simply created without changing the pre-planned route for each train. To show the roles of crossover tracks, we give four indices, including delay of inbound trains ($d_I$), delay of outbound trains ($d_O$), delay of total trains ($d_T$) and service balance ($\kappa$) to evaluate the performance of two timetables, listed in Table 2. The service balance index $\kappa$ of subway system is calculated by the following equation:

\[
\kappa = \frac{2}{N} \sum_{i=1}^{N/2} |a_{2i,1} - a_{2i-1,K,1}|
\]

(2.1)

where $N$ is the scale number of trains, $a_{2i,1}$ denotes the arrival time for inbound train $2i$ to its terminal station $1$, and $a_{2i-1,K,1}$ represents the arrival time for outbound train $2i - 1$ to its terminal station $K$.

**Remark 2.1.** Typically, to guarantee the safe operations of the subway system, the dispatchers are often required to give a dispatching solution as a response to a disruption. This dispatching solution is usually represented by a train timetable to indicate the arrival and departure times of involved trains at each station. For simplicity, we still use the terms “timetables” to represent the dispatching solutions generated by dispatchers when a disruption occurs. That is, “timetables” actually refer to “dispatching solutions” in this research.

Note that the service balance is a crucial factor to evaluate the quality of a subway system. It is easy to see from Table 2 that if there are no crossover tracks, the rescheduled timetable can correspond to a high unbalance indicator, leading to more delay in one direction. However, this indicator can be decreased to a relative small level by introducing crossover tracks. Practically, we need to mention that when crossover tracks are considered in rescheduling process, less total delay in the subway system might be reduced when compared with timetable (II). However, the more service balance can still be achieved at the cost of merely extra ten percent total delay compared to the operation pattern of disuse of crossover tracks, and what is more, this unbalance performance will be more serious with the increase of incident duration.

The following discussion aims to propose a rigorous formulation for subway train rescheduling with the aid of crossover tracks. Some assumptions are first made below.

(A1) Without loss of generality, an incident is supposed to occur on a track in inbound direction, and only incident link cannot be used during the incident period.

(A2) Each track (station) is allowable to be occupied by trains from both directions.

(A3) The dwelling capacity of each platform in individual direction is only one. That is, no two or more trains in the same direction can occupy a platform simultaneously.

(A4) The communication capability of the railway system is enough for trains traveling from inbound tracks to outbound tracks and vice versa.
3. Model formulation

To characterize the rescheduling process mathematically, this section will set up an optimization model for the problem of interest, which is typically a mixed 0–1 integer programming model.

3.1. Parameters and notations

To describe and understand the problem more conveniently, we hereinafter first list some closely related subscripts and parameters in Table 3.

3.2. Decision variables

\[ a_{ik} : \text{arrival time for train } i \text{ at station } k; \]
\[ d_{ik} : \text{departure time for train } i \text{ from station } k; \]
\[ q_{ijk} : \text{traversing order for trains in the same direction, } i, j \in O \text{ or } i, j \in l, = 1 \text{ if train } i \text{ traverses on section } k \text{ before train } j; = 0 \text{ otherwise}; \]
\[ r_{ijk} : \text{traversing order for trains in different directions, } i \in O \text{ and } j \in O, \text{ or } i \in l \text{ and } j \in O, = 1 \text{ if train } i \text{ traverses on section } k \text{ before train } j; = 0 \text{ otherwise.} \]

3.3. System constraints

Due to the distribution of crossover tracks along the subway line, if an incident occurs on a track section (or a platform) along the inbound direction, some inbound sections and platforms will become invalid. Obviously, because of the limited capacity and capability of the subway line, rescheduling trains on a line with capacity reductions is required to subject to some pre-specified constraints. In this section, some system constraints will be formulated for our model to guarantee the feasible and safe operations of the rescheduled train timetables.

Close-to-original-schedule constraints: Clearly, if the arrival/departure time of a train is earlier than the incident occurrence time, there is no any disturbance on this train. That is, this train can still be dispatched according to the pre-trip favorite timetable. For depicting this requirement, some constraints can be formulated as follows:

\[ a_{ik} = \bar{a}_{ik}, \quad \text{if } \bar{a}_{ik} \leq t_s, \quad i \in O, \quad k \in \{2, 3, \ldots, k^*\}, \quad (3.1) \]
\[ d_{ik} = \bar{d}_{ik}, \quad \text{if } \bar{d}_{ik} \leq t_s, \quad i \in O, \quad k \in \{1, 2, 3, \ldots, k^* - 1\}, \quad (3.2) \]
\[ a_{ik} = \bar{a}_{ik}, \quad \text{if } \bar{a}_{ik} \leq t_s, \quad i \in l, \quad k \in \{l^* + 1, \ldots, m - 1\}, \quad (3.3) \]
\[ d_{ik} = \bar{d}_{ik}, \quad \text{if } \bar{d}_{ik} \leq t_s, \quad i \in l, \quad k \in \{l^* + 1, \ldots, m\}. \quad (3.4) \]

Headway constraints: For guaranteeing the safety of rescheduled operations, i.e., keeping a safe distance between two adjacent trains or two face-to-face trains on each track (or station), we formulate some relevant headway constraints when modeling the rescheduling problem. In this paper, we use different constant times to denote different headway for simplicity. In what follows, we present the detailed analysis of headway constraints.

(i) Headway constraints for outbound trains

For keeping the minimal headway \( h_0 \) between two adjacent trains \( i \) and \( j \) when they leave section \( k \), we formulate the constraints below,

\[ a_{i,k+1} + h_0 \leq a_{j,k+1} + M(1 - q_{ijk}), \quad i, j \in O, \quad k \in E. \quad (3.5) \]
\begin{equation}
a_{ik} + h_0 \leq a_{ik+1} + M(1 - q_{ijk}), \quad i, j \in O, k \in E. \tag{3.6}
\end{equation}

Similarly, to keep the minimal headway \( h_0 \) between adjacent trains \( i \) and \( j \) when they enter section \( k \), we present the following two constraints:
\begin{equation}
d_{ik} + h_0 \leq d_{jk} + M(1 - q_{ijk}), \quad i, j \in O, k \in E \tag{3.7}
\end{equation}
\begin{equation}
d_{jk} + h_0 \leq d_{ik} + M(1 - q_{ijk}), \quad i, j \in O, k \in E \tag{3.8}
\end{equation}

(ii) Headway constraints for inbound trains

Similar conditions are imposed for inbound trains, and we have,
\begin{equation}
a_{ik} + h_0 \leq a_{ik} + M(1 - q_{ijk}), \quad i, j \in I, k \in E, \tag{3.9}
\end{equation}
\begin{equation}
a_{jk} + h_0 \leq a_{ik} + M(1 - q_{ijk}), \quad i, j \in I, k \in E, \tag{3.10}
\end{equation}
\begin{equation}
d_{i,k+1} + h_0 \leq d_{j,k+1} + M(1 - q_{ijk}), \quad i, j \in I, k \in E, \tag{3.11}
\end{equation}
\begin{equation}
d_{j,k+1} + h_0 \leq d_{i,k+1} + M(1 - q_{ijk}), \quad i, j \in I, k \in E, \tag{3.12}
\end{equation}

where the first two constraints and the last two ones, respectively, are used to guarantee the safety between trains \( i \) and \( j \) when they leave from and arrive at the same section \( k \).

(iii) Headway constraints for trains in different directions

Recall the example in Section 2. If the platform of station 3 is occupied by an inbound train, the outbound train which will arrive at station 3, has to wait at station 2 for the inbound trains to leave station 3. For the safety between these two trains, a time headway \( h_{ik}^{ad} \) is considered. The situation for station 5 is different, where inbound trains have to wait at station 5 for outbound train to enter station 5. In this sense, a time headway \( h_{ij}^{ad} \) is taken into account. On the basis of the above analysis, we have,
\begin{equation}
d_{jk}^{ad} + h_{ik}^{ad} \leq a_{ik} + M(1 - p_{ijk}), \quad i \in O \cap R, j \in I \cap R, \tag{3.13}
\end{equation}
\begin{equation}
a_{il+1}^{ad} + h_{ij}^{ad} \leq d_{j,l+1} + M(1 - p_{ij,l+1}), \quad i \in O \cap R, j \in I \cap R. \tag{3.14}
\end{equation}

Dwelling time constraint: In pre-specified timetable, each train is arranged a time interval to dwell at each station for passenger service. And this period is called dwelling time which includes alighting time, boarding time and other operational time such as time for opening and closing doors. Considering the dwelling time, we require a minimum time \( \bar{t}_{ik} \) for train \( i \) at station \( k \) in rescheduled process, and we have,
\begin{equation}
d_{ik} - a_{ik} \geq \bar{t}_{ik}, \quad i \in O \cup I, k \in V. \tag{3.15}
\end{equation}

Arrival time constraints: Clearly, because of the incident, it is impossible that the arrival time for trains to their destinations in the rescheduled timetable is earlier than that in the pre-specified favorite timetable. Corresponding constraints can be set up as,
\begin{equation}
a_{in} \geq \bar{a}_{in}, \quad i \in O, \tag{3.16}
\end{equation}
\begin{equation}
a_{ij} \geq \bar{a}_{ij}, \quad j \in I. \tag{3.17}
\end{equation}

Initial rescheduling time constraints: In the rescheduled timetable, the arrival time of each inbound train at recovered station is not earlier than the recovery time, and we have,
\begin{equation}
a_{ik} \geq t_i, \quad i \in B \cap I, k \in W. \tag{3.18}
\end{equation}

Hereinafter the recovery time is defined as a specific timestamp after which the affected sections and stations can be reused. Besides, since there may be some trapped trains at incident stations or on incident sections, for the safety between the trapped train \( i \) (e.g., train 4 in Fig. 4(a)) and train \( j \) which enters the recovered stations (e.g., train 6 in Fig. 4(a)), the following constraints are concluded:
\begin{equation}
d_{ik} + h_0 \leq d_{jk} + M(1 - q_{ijk}), \quad i \in I \cap G, j \in I \cap B, \tag{3.19}
\end{equation}
\begin{equation}
d_{jk} + h_0 \leq d_{ik} + M(1 - q_{ijk}), \quad i \in I \cap G, j \in I \cap B. \tag{3.20}
\end{equation}

Crossover track constraints: When a train switches its track from inbound line to outbound line, it should travel through a crossover track (as shown in Fig. 2). Then, some additional operational time (i.e., \( t_c \)) for train traveling on the crossover track is practically needed. That is, compared with the original one, section trip time for the train in rescheduled timetable has an additional time \( t_c \). We then formulate two constraints below,
\begin{equation}
\bar{a}_{ik-1} - \bar{d}_{ik} + t_c \leq a_{ik-1} - d_{ik}, \quad i \in I \cap R. \tag{3.21}
\end{equation}
\[ \bar{a}_{ik} - d_{i,k+1} + t_c \leq a_{ik} - d_{i,k+1}, \quad i \in I \cap R. \]  

(3.22)

In fact, except for the trapped train (e.g., train 3 in Fig. 4(a)), notation \( q_{ijk} \in \{0, 1\} \) is pre-determined for trains \((i, j) \in O - G \) or \( (i, j) \in I - G \). That is, if train \( i \) is the lead train of train \( j \), i.e., \( i < j \). \( q_{ijk} \) is determined as 1 since overtaking operations are prohibited due to the platform capacities, leading to constraints below,

\[ q_{ijk} = 1, \quad \text{if} \quad i < j, \quad (i, j) \in O - G \quad \text{or} \quad (i, j) \in I - G. \]  

(3.23)

Service balance constraint: As discussed in Sections 1 and 2, service balance is also a crucial factor for subway system. Thus, the following constraint is given to guarantee the service balance between inbound and outbound trains, formulated by,

\[ \kappa < \Gamma^*, \]  

(3.24)

where \( \Gamma^* \) is the input data which can be provided by experts according to their operating experiences.

Notation \( p_{ijk} \in \{0, 1\} \) here is applied to denoting a rescheduling sequence on shared sections and stations for trains in different directions. And the determination of \( p_{ijk} \) (\( i \in I \) and \( j \in O \), or \( i \in O \) and \( j \in I \)), is the key point in our model.

### 3.4. Objective function

To make a rescheduled timetable close to the original timetable, we use the total delay \( T_d \) as the evaluation criterion to evaluate the rescheduled timetable, and \( T_d \) can be calculated as,

\[ T_d = \sum_{i \in O} (a_{in} - \bar{a}_{in}) + \sum_{i \in I} (a_{i1} - \bar{a}_{i1}). \]  

(3.25)

With the aforementioned constraints, we formulate the rescheduling problem with the least total delay as below,

\[
\begin{align*}
\min & \quad \sum_{i \in O} (a_{in} - \bar{a}_{in}) + \sum_{i \in I} (a_{i1} - \bar{a}_{i1}) \\
\text{s.t.} \quad & q_{ijk} + q_{jik} = 1, \quad i, j \in R \cap I, \quad \text{or} \quad i \in R \cap O, \; k \in V \\
& p_{ijk} + p_{jik} = 1, \quad i \in R \cap I \quad \text{and} \quad j \in R \cap O, \quad \text{or} \quad i \in R \cap O \quad \text{and} \quad j \in R \cap I, \; k \in V \\
& \text{Constraints (3.1) – (3.24)}
\end{align*}
\]  

(3.26)

### 4. Efficient train rescheduling strategies

Based on the discrete event model, this section aims to design an efficient rescheduling strategy to seek the high-quality updated timetables. The following discussion focuses on introduction of the detailed techniques proposed in this solution methodology.

#### 4.1. Determination of train position states

In the scheduling strategy, three train position states (TPS) are defined to describe the occurrence of discrete events. As shown in Fig. 5, indices 1, 2, and 3, respectively, are employed to denote different TPSs with the concrete meanings listed below.

1. **TPS** = 1: A train arrives at a station and its current dwelling time is less than the pre-scheduled dwelling time at this station. In this case, the arrival time of this train can be easily identified, but the departure time cannot be determined at the current state.
(2) **TPS = 2**: A train is dwelling at a station and its current dwelling time is greater than or equal to the pre-scheduled dwelling time. It can be dispatched to depart from the current station once the rescheduling strategy gives the permission to travel toward next station. As shown in Fig. 5, in fact, points $s_{1}'$ and $s_{1}''$ maybe overlap in some cases. That is, there is no delay time for the focus train at the current station.

(3) **TPS = 3**: When a train departs from a station, its TPS will be changed from 2 to 3. Moreover, the updated TPS remains unchanged before the train arrives at its next station. Once the train arrives at its next station, its TPS will be changed to state 1.

### 4.2. Capacity check algorithm

Obviously, when a train’s TPS is 1, it is safe if the train continues its current status until its TPS becomes 2. Analogously, since trains cannot stop on sections, the train can also continue its trip if its TPS is 3. Then what should be considered in the process of rescheduling is that whether or not a train can depart from its current station when its TPS is 2. Therefore, in the rescheduling strategy, the core is capacity check algorithms for trains whose TPSs are the same to 2, to avoid the potential conflicts and deadlocks. Hereinafter a segment is said to be in a state of deadlock when no train on this segment of railway line can advance without causing a collision (see [29]).

Before introducing the detailed capacity check algorithm, we give some illustrative examples to demonstrate the different cases for trains with various positions. More specifically, as shown in Fig. 6(a), since train $i$ is at station $l^* - 1$ and train $j$ is on section $l^*$, for avoiding the potential conflict, train $i$ is required to wait at its station for train $j$ to pass through station $l^*$ (see Fig. 6(b)). Similarly, in Fig. 6(c) and (d), train $j$ is required to stop at station $k^* + 1$ before train $i$ leaves section $k^*$. For train $i$ in Fig. 6(a) and train $j$ in Fig. 6(c), we develop Algorithm 4.1 in ETRS method. In addition, Fig. 7(a) shows the movements of four trains, in Fig. 7(a) since there are no trains in the route between section $l^* - 2$ and station $l^* - 1$, it is a safe operation if train $i$ keeps move forward. On the other hand, since train $j^f$ waits at station $k^* + 1$ for train $i^f$’s pass, there will be a conflict if train $j$ moves to its next station $k^* + 1$. For this case, the safe subsequent movements are illustrated in Fig. 7(b) and (c), corresponding to Algorithm 4.2.
In what follows, the notations used in algorithms are given for descriptive convenience, and then rescheduling algorithms for different cases are presented in detail. Take an outbound train \( i \) whose TPS is 2 as an example. Suppose that train \( i \) is at station \( r \), and the next section and station are \( s \) and \( r' \), respectively. Let \( b_i \) be a binary indicator to identify whether train \( i \) passes the capacity check algorithm, i.e., \( b_i = 1 \): pass, \( b_i = 0 \): no pass.

Variables:
- \( N^o \): the scalar number of outbound trains traveling on the specified route, including the focal train;
- \( N^i \): the scalar number of inbound trains traveling on the specified route;
- \( O_{\text{first}} \): ID of the first outbound train among the aforementioned \( N^o \) trains;
- \( I_{\text{first}} \): ID of the first inbound train among the aforementioned \( N^i \) trains;
- \( lid \): ID of a specified inbound train;
- \( S^o \): the scalar number of stations at which outbound trains can dwell;
- \( OptOI \): the priority of trains \( O_{\text{first}} \) and \( I_{\text{first}} \) = 1. Train \( O_{\text{first}} \) has the priority; = 0, otherwise.

Algorithm 4.1. Capacity check algorithm 1 (CCA1).

Step 1. (Initialization) \( N^i = 0 \). \( lid = -1 \);

Step 2. Search for all the outbound stations and sections between stations \( l^* \) (included) and \( k^* + 1 \) (not included), if there exist \( m_1 \) inbound trains, let \( N^l ← N^l + m_1 \); if there is an inbound train at station \( k^* + 1 \), update \( lid \) as the index of this inbound train;

Step 3. Capacity check procedures:
- Step 3.1. if \( N^o > 0 \), train \( i \) cannot pass the capacity check; otherwise, go to Step 3.2;
- Step 3.2. if \( lid = -1 \), train \( i \) passes the capacity check; otherwise, go to Step 3.3;
- Step 3.3. if \( a_i(l_{i-1}) < a_{lid(k^*+1)} \), train \( i \) passes the capacity check, otherwise, train \( i \) cannot pass the capacity check.

As discussed above, Algorithm 4.1 is applied to determining the succeeding operations of trains which dwell at stations \( l^* - 1 \) and \( k^* + 1 \) (e.g. trains \( i \) in Fig. 6(a) and \( j \) in Fig. 6(c)). Take train \( i \) as an example. Due to the accident, some inbound trains (the number \( N^i \) is recorded in Step 2) may share the same outbound sections and stations (from stations \( l^* \) (included) to \( k^* + 1 \) (not included)). There exist potential conflicts between train \( i \) and those inbound trains. Step 3.1 states that if there are some inbound trains on the shared sections and stations (i.e. \( N^i > 0 \)), train \( i \) cannot continue its travel in this situation for avoiding conflicts. Otherwise, if \( N^o = 0 \), train \( i \)'s succeeding operation is determined by the following steps. If there is no inbound train dwelling at station \( k^* + 1 \), train \( i \) can depart from \( l^* - 1 \), as shown in Step 3.2. On the contrary, if an inbound train dwelling at station \( k^* + 1 \) and its arrival time at station \( k^* + 1 \) (i.e., \( a_{lid(k^*+1)} \)) is later than train \( i \)'s arrival time at station \( l^* - 1 \) (i.e., \( a_{l_{i-1}} \)), train \( i \) can also continue to head for section \( l^* - 1 \); otherwise, train \( i \) should dwell at station \( l^* - 1 \). See Step 3.3. We here point out that Step 3.3 essentially ensures the service balance between inbound and outbound trains.

Algorithm 4.2. Capacity check algorithm 2 (CCA2).

Step 1. (Initialization) \( N^o = 1 \) (include the focal train), \( N^i = 0 \), \( S^o = 0 \), and \( OptOI = -1 \);

Step 2. Search for all the outbound stations and sections between stations \( r \) (not included) and \( l^* - 1 \) (included). If there are \( m_2 \) inbound trains, let \( N^l ← N^l + m_1 \);

Step 3. Search for all the stations and sections between stations \( l^* \) (included) and \( k^* + 1 \) (included). If there are \( m_2 \) inbound trains, let \( N^l ← N^l + m_2 \);

Step 4. Determine the scalar number of the stations between stations \( r \) (not included) and \( l^* \) (not included), update \( S^o \);

Step 5. Capacity check procedures:
- Step 5.1. if \( N^o > 1 \) and \( N^i > 0 \), go to Step 5.2; otherwise, go to Step 5.3;
- Step 5.2. if train \( I_{\text{first}} \) on the route between stations \( l^* \) (included) and \( k^* + 1 \) (not included), or the train in the front of the focus train has been dispatched to wait at a station, or \( a_{l_{\text{first}}(k^*+1)} < a_{l_{\text{first}}(l^*-1)} \), then \( OptOI = 0 \); otherwise, \( OptOI = 1 \).
  - Item 1: \( OptOI = 1 \) and \( N^o = 0 \) ≤ \( S^o \); Item 2: \( OptOI = 0 \) and \( N^o \) ≤ \( S^o \);
  - If Item 1 or Item 2 is true, train \( i \) passes the capacity check, otherwise, train \( i \) cannot pass the capacity check;
- Step 5.3. if \( N^o = 0 \), train \( i \) passes the capacity check; otherwise, go to Step 5.4;
- Step 5.4. Item 1: \( N^o = 0 \) ≤ \( S^o \) and the lead train of the focal one passes the capacity check; Item 2: there are no any outbound trains on/at section \( s \)/station \( r' \);
  - If Item 1 or Item 2 is true, train \( i \) passes the capacity check, otherwise, train \( i \) cannot pass the capacity check.

CCA2 here is applied to determining the binary indicator of an inbound (outbound) train which dwells at stations before (after) station \( l^* - 1 \) (\( k^* + 1 \)), such as \( l^* - 2 \) (\( k^* + 2 \)). Take train \( i \) in Fig. 7(a) as an example. Detailed explanations of CCA2 are given as follows. In Step 1, some parameters are initialized. Steps 2 and 3 respectively record the numbers of outbound and inbound trains on (at) the specified sections (stations). Step 4 is used to count the number of outbound stations from stations \( r' \) to \( l^* - 1 \). After the preparatory work of Steps 1–4, Step 5 determines whether or not the considered train can depart from its current station (i.e., train \( i \) passes the check algorithm).

Specifically, if there are both inbound and outbound trains traveling on the corresponding sections and stations (see Steps 2 and 3), Step 5.2 will be activated. In Step 5.2 three cases are taken into account, including (1) the first inbound train is on the route between stations \( l^* \) (included) and \( k^* + 1 \) (not included); (2) the train in the front of the focus train has been dispatched to wait at a station; and (3) train \( I_{\text{first}} \) dwells at station \( k^* + 1 \) and the arrival time of train \( I_{\text{first}} \) at station \( k^* + 1 \) is earlier than that
of train $O_{\text{first}}$ at station $l^* - 1$. If any one of above three cases is true, train $O_{\text{first}}$ is given the priority to use the shared sections and stations, then $OptOI = 0$; otherwise, train $O_{\text{first}}$ has the priority, thus $OptOI = 1$. Based on the aforementioned information, the check indicator of the focus trains can be determined: if $OptOI = 1 & NO_{l^* - 1} \leq SO$ or $OptOI = 0 & NO \leq SO$ is true, there is enough station capacity for train $i$, and it is safe if train $i$ continues to head for its designated station. Similar to Step 3.3 of CCA1, Step 5.2 here also implicitly ensures the service balance between inbound and outbound trains. Step 5.3 points out that if there is only train $i$ traveling on the route between stations $r$ and $l^* - 1$, train $i$ can pass the capacity check. In Step 5.4, two other situations that train $i$ can pass the capacity check are listed, including (1) $NO_{l^* - 1} \leq SO$ and the lead train of the focal one passes the capacity check; and (2) there are no any outbound trains on/at section $s$/station $r'$.

Algorithm 4.3. Capacity check algorithm 3 (CCA3). This algorithm is applied to rescheduling trains after the faulted sections can be reused. In this algorithm, with the adaption of first-in-first-out (FIFO) rule, if some inbound trains simultaneously exist on the recovered inbound tracks and opposite outbound tracks, we then specify that the train in inbound direction has the priority to continue its trip.

Fig. 8 displays the flowchart for capacity check algorithms, where applications of the aforementioned algorithms CCA1, CCA2 and CCA3 are clearly depicted.
4.3. Determination of the next discrete event

As introduced in Section 4.1, train position states follow the change laws that \(1 \rightarrow 2 \rightarrow 3 \rightarrow 1\). That is to say, when train position state is 1, its next state will be 2, and by this analogy, it is easy to deduce the next states of other train position states. For simplicity, we hereinafter define such state transitions as discrete events. Actually, there is essentially a little difference between our definition and that in Dorfman and Medanic [29], where they assume that an event occurs once a train reaches a station. Let \(t\) be the current system time, \(r_i\) the time interval that train \(i\) needs to reach its next state. Suppose that train \(i\) is at station \(k\). If TPS of train \(i\) is 1, we then deduce \(b_i = 1\) according to Section 4.2. As shown in Fig. 5, \(r_i\) can be calculated by,

\[
    r_i = \bar{t}_{ik} - (t - a_{ik}).
\]

When TPS = 2, train \(i\) should be checked by the capacity check algorithm. If train \(i\) passes the capacity check, i.e., \(b_i = 1\), we can decide to change its state from 2 to 3. Thus, \(r_i = 0\), and then the departure time \(d_{ik}\) can be determined. Otherwise, if train \(i\) cannot pass the capacity check, it is required to wait at its current station for the next discrete event which is triggered by other trains. So, we obtain,

\[
    r_i = M(1 - b_i).
\]

If train \(i\) traverses on a section, e.g., section \(k\), its TPS corresponds to state 3. For this case, we have \(b_i = 1\), which is similar to that in the case TPS = 1, and \(r_i\) can be calculated according to,

\[
    r_i = \begin{cases} 
        (\bar{a}_{ik+1} - \bar{d}_{ik}) - (t - d_{ik}), & \text{if train } i \text{ does not use the cross track,} \\
        (\bar{a}_{ik+1} - \bar{d}_{ik} + t_c) - (t - d_{ik}), & \text{if train } i \text{ uses the cross track.} 
    \end{cases}
\]

Considering the above-mentioned discussions, we finally produce a formal equation to determine parameter \(r_i\), given below,

\[
    r_i = \begin{cases} 
        \bar{t}_{ik} - (t - a_{ik}), & \text{if train } i \text{‘s TPS } = 1, \\
        M(1 - b_i), & \text{if train } i \text{‘s TPS } = 2, \\
        (\bar{a}_{ik+1} - \bar{d}_{ik}) - (t - d_{ik}), & \text{if train } i \text{‘s TPS } = 3 \text{ and does not use the cross track,} \\
        (\bar{a}_{ik+1} - \bar{d}_{ik} + t_c) - (t - d_{ik}), & \text{if train } i \text{‘s TPS } = 3 \text{ and use the cross track.} 
    \end{cases}
\]

Given the vector \(r = \{r_1, r_2, \ldots, r_N\}\), the arguments,

\[
    \{r_{min}, i_{min}\} = \min (r),
\]

denote the time interval that train \(i_{min}\) needs to reach its next state is the least one among the elements of \(r\), i.e., \(r_{min} = r_{min}\). Note that \(i_{min}\) may represent more than one train. Let \(d_{next}\) be a time interval and \(d_{next} = r_{min}\). Obviously, for the entire system, at time \(t + d_{next}\), i.e., the time at which train \(i_{min}\) reaches its next state, the next discrete event of the entire traffic system occurs.

4.4. System information update

Obviously, trains in the traffic system can be categorized into two classes based on the relationships between their time interval \(r_i\) and \(d_{next}\), i.e., (1) \(r_i > d_{next}\) and (2) \(r_i = d_{next}\). For trains in the first class, since all of them cannot reach their next states, their states remain unchanged. For the second class, train states will be updated and the arrival/departure times can be determined. Similar to the analysis in Section 4.3, we shall present the detailed description of trains’ information updating according to their TPS:

1. If TPS = 1, at time \(t + d_{next}\), its TPS is changed as 2;
2. If TPS = 2, at time \(t + d_{next}\), its TPS is changed from 2 to 3 immediately, and the train’s departure time from its current station can be determined as \(t + d_{next}\);
3. If TPS = 3, at time \(t + d_{next}\), its TPS is changed from 3 to 1, and the train’s arrival time for its next station can be determined as \(t + d_{next}\).

It is worth mentioning that we use the state changes and the determination of trains arrival/departure times to characterize the system dynamics model, which is in essence different from that in Dorfman and Medanic [29]. In summary, the procedure of the ETRS method can be referred to Fig. 9 for clearly understanding the entire algorithmic framework.

4.5. Performance criteria

For measuring the performance of the TAS method, Dorfman and Medanic [29] introduced three evaluation criteria, including (a) the time to clear line, (b) the total delay of all trains, and (c) the maximal delay. For the first one, it indicates the time that all trains occupy the railway line, which can be formulated as,

\[
    J_1 = d_{last.terr} - t_0,
\]

(4.6)
where $t_0$ and $d_{\text{last, ter}}$, respectively, are the time of the earliest train arriving at its first station and the time of the latest train departing from its terminal station. In addition, to show the time-efficiency of the timetable obtained by the TAS method, Dorfman and Medanic [29] proposed a ratio formulated by,

$$
\eta = \frac{d_{\text{last, ter}} - t_0}{d_{\text{last, ter}} - t_0}.
$$

(4.7)

Considering the difference between the scheduling method (TAS) and our rescheduling method (ETRS), we especially propose a ratio to measure the delay of $J_1$. This ratio is named as clear time delay ratio and defined by,

$$
\varphi = \frac{d_{\text{last, ter}} - \bar{a}_{\text{last, ter}}}{d_{\text{last, ter}} - t_0}.
$$

(4.8)

The total delay of all trains is similar to our total delay function (see Eq. (3.25)), and we have,

$$
J_2 = T_d.
$$

(4.9)

Besides, the maximal delay is defined by

$$
J_3 = \max_{i} \{\max_{l \in O} \{a_{in} - \bar{a}_{in}\}, \max_{l \in I} \{a_{i1} - \bar{a}_{i1}\}\}.
$$

(4.10)

For understanding convenience, we particularly adopt $M_d$ to denote $J_3$ in the following numerical experiments.

5. Numerical examples

In this section, two examples will be implemented to show the efficiency and effectiveness of the proposed approaches. In the first experiment, we take the sample subway line shown in Fig. 2 as an experimental example, and respectively use the GAMS software and ETRS method to generate the rescheduled train timetables, where the solution with the least relative gap produced by GAMS software is regarded as a near-optimal solution. Moreover, for demonstrating the efficiency of ETRS method for large-scale problems, we implement the second set of experiments adapting the practical operation data from Beijing Yizhuang subway line in the second example. We also conduct extensive case studies to investigate the performance criteria of the rescheduled timetables.

5.1. Computational results

Example 5.1. Let us consider the sample example in Section 2 (see Fig. 2). Suppose that an incident takes place on the inbound Section 3 at time $t_s = 120$ s, and the faulted sections will be recovered at time $t_r = 20,000$ s. For experimental convenience, we set the travel time on crossover track as $t_c = 30$ s. There are 5 inbound trains and 5 outbound trains on the subway line, and the duration of departure interval is assumed to be 180 s. We assume that station dwelling time and inter-station trip time are 30 s and 120 s, respectively, as listed in Table 4. In order to guarantee the service balance of the rescheduled timetable, in this example we particularly assume that trains in two bounds are allowed to occupy the common tracks in turn.

By performing the COINCBC solver in the GAMS software, a near-optimal solution and a feasible solution are finally obtained in this set of experiments, in which a gap parameter (i.e., OPTCR) can be used to return such solutions within the given relative gaps. Specifically, for instance, if OPTCR is set as 0.5, the encountered feasible solution will be outputted once the relative gap between its objective and the estimated optimal objective is less than 50%. In this sense, a larger gap parameter can be expected to return a feasible solution quickly once it satisfies the gap requirements. Then, the first encountered feasible solution can be expected to be outputted if the gap parameter is large enough. We perform this experiment on a personal computer with four
Table 5
Statistical information for the rescheduled timetables on sample line.

<table>
<thead>
<tr>
<th>Item</th>
<th>Statistical information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_d$ [s]</td>
</tr>
<tr>
<td>ETRS</td>
<td>13,800</td>
</tr>
<tr>
<td>GAMS(fea)</td>
<td>15,300</td>
</tr>
<tr>
<td>GAMS(n-opt)</td>
<td>12,780</td>
</tr>
<tr>
<td>$\epsilon$ (fea)</td>
<td>$-9.80%$</td>
</tr>
<tr>
<td>$\epsilon$ (n-opt)</td>
<td>$7.98%$</td>
</tr>
</tbody>
</table>

Table 6
The practical operation timetable for Beijing Yizhuang subway line (unit: s).

<table>
<thead>
<tr>
<th>Station</th>
<th>Songjiazhuang</th>
<th>Xiaocun</th>
<th>Xiaohongmen</th>
<th>Jiugong</th>
<th>Yizhuangqiao</th>
<th>Wenhuayuan</th>
<th>Wanyuan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival</td>
<td>0</td>
<td>220</td>
<td>358</td>
<td>545</td>
<td>710</td>
<td>835</td>
<td>979</td>
</tr>
<tr>
<td>Dwelling</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>35</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Departure</td>
<td>30</td>
<td>250</td>
<td>388</td>
<td>575</td>
<td>745</td>
<td>865</td>
<td>1009</td>
</tr>
<tr>
<td></td>
<td>Rongji</td>
<td>Rongchang</td>
<td>Tongjinan</td>
<td>Jinghai</td>
<td>Ciquan</td>
<td>Ciqu</td>
<td>Yizhuang</td>
</tr>
<tr>
<td>Arrival</td>
<td>1112</td>
<td>1246</td>
<td>1440</td>
<td>1620</td>
<td>1790</td>
<td>1927</td>
<td>2077</td>
</tr>
<tr>
<td>Dwelling</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>35</td>
<td>45</td>
<td>30</td>
</tr>
<tr>
<td>Departure</td>
<td>1142</td>
<td>1276</td>
<td>1470</td>
<td>1650</td>
<td>1825</td>
<td>1972</td>
<td>2107</td>
</tr>
</tbody>
</table>

Example 5.2. To show the efficiency of the proposed approach for large-scale problems, we herein consider the Beijing Yizhuang subway line as an experimental environment, which consists of 14 stations, 13 sections and a series of crossover tracks shown in Fig. 11. All experiments in this example are performed on an Intel(R) Core(TM) 2 Duo PC with 2.33 GHz CPU and 1.96 GB memory, and all the algorithms are implemented in Microsoft Visual Studio 2010 C# on the Windows XP platform.
Table 6 lists the practical operation time for Yizhuang subway line at each station, where $t_c$ is set as 10 s. For understanding convenience, Fig. 12 further depicts this practical timetable without any incidents. In this experiment, a total of 40 trains (20 inbound trains and 20 outbound trains) will be taken into account, which share the same duration $T_{int}$ of departure interval.

Suppose that an incident takes place on the inbound track between stations Xiaohongmen and Jiugong (see Fig. 11). The proposed ETRS method will be implemented to test its efficiency with different cases of $t_s$, $t_r$ and $T_{int}$. Table 7 gives the performance criteria of different rescheduled timetables and the CPU time consumed by the ETRS method. Obviously, the proposed
Table 7
Computational results produced by different methods.

<table>
<thead>
<tr>
<th>$t_s$ [s]</th>
<th>$t_r$ [s]</th>
<th>$T_{in}$ [s]</th>
<th>ETRS method</th>
<th>Dispatcher method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_d$ [h]</td>
<td>$M_d$ [s]</td>
<td>$\varphi$ [s]</td>
<td>$\kappa$ [s]</td>
<td>CT [ms]</td>
</tr>
<tr>
<td>1000</td>
<td>2500</td>
<td>120</td>
<td>10.028</td>
<td>1725</td>
</tr>
<tr>
<td>160</td>
<td>8.026</td>
<td>965</td>
<td>0.187</td>
<td>411</td>
</tr>
<tr>
<td>200</td>
<td>6.668</td>
<td>907</td>
<td>0.149</td>
<td>458</td>
</tr>
<tr>
<td>1500</td>
<td>2800</td>
<td>120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>8.943</td>
<td>1242</td>
<td>0.213</td>
<td>526</td>
</tr>
<tr>
<td>200</td>
<td>7.403</td>
<td>1242</td>
<td>0.165</td>
<td>532</td>
</tr>
<tr>
<td>1500</td>
<td>2800</td>
<td>120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>5.676</td>
<td>962</td>
<td>0.159</td>
<td>431</td>
</tr>
<tr>
<td>200</td>
<td>3.694</td>
<td>884</td>
<td>0.097</td>
<td>408</td>
</tr>
</tbody>
</table>

Note: Symbol “–” represents that there are some trains failed in being rescheduled because of the capacity of stations (see Fig. 14(b) and corresponding explanations for more details).

Fig. 13. Rescheduling timetables with $T_{in} = 160$ (unit: s).
Fig. 14. Rescheduling timetables with $t_s = 2000$, $t_r = 3000$ (unit: s).

ETRS method can reschedule a large number of trains within a very short time. It is also noteworthy that, for each fixed combination of $t_s$ and $t_r$, the larger the $T_{int}$ is, the better performance criteria can be produced. Take the experiment of $t_s = 1000$ s and $t_r = 2500$ s as an example. It is easy to see that the performance criteria $T_d$, $M_d$, and $\varphi$ begin to decrease gradually with the increase of parameter $T_{int}$. As a comparison, results produced by dispatchers (see the discussion in the paragraph 2 in the Introduction Section) are also presented in Table 7. Obviously, all of the service balance $\kappa$ and maximal delay $M_d$ by ETRS are better than those by dispatchers’s method, which illustrates the effectiveness of service balance of the ETRS method. Moreover, we find that compared to that by the dispatchers’s method, the total delay time generated by ETRS method is increased, which can be explained by the reason that with the consideration of service balance, some tracks will be used by trains in opposite direction, leading to the delay of some trains dispatched to travel on the aforementioned tracks. However, in the real-world operations, it is typically acceptable to guarantee the service balance even at the cost of total delay.
Fig. 15. Statistical information for timetables with different $T_{int}$ (unit: s).

(a) $t_s = 1500, t_r = 2800$

(b) $t_s = 2000, t_r = 4000$
Fig. 16. Statistical information for timetables with different $t_s$ (unit: s).

(a) $t_s = 1500$, $T_{int} = 160$

(b) $t_s = 2000$, $T_{int} = 160$
In what follows, we aim to analyze the detailed characteristics of rescheduled timetables created by the ETRS method. To this end, Fig. 13(a) first gives the rescheduled timetable in the case of \( t_5 = 1000 \text{ s}, t_r = 2500 \text{ s} \) and \( T_{\text{int}} = 160 \text{ s} \). In this timetable, as expected, inbound and outbound trains alternately occupy the shared route from station Xiaohongmen to the section between stations Jiugong and Yizhuangqiao, resulting in a relatively small total delay time. In addition, it is also recognized that the incident occurrence time might probably lead to different rescheduling results. For instance, in the case of \( t_5 = 1500 \text{ s}, t_r = 2800 \text{ s} \) and \( T_{\text{int}} = 160 \text{ s} \), train 2 is trapped at station Jiugong until the faulted sections can be reused, while train 4 continues its travel with crossover tracks and outbound sections during the incident duration (see Fig. 13(b)). This rescheduled mode is obviously different from that in Fig. 13(a).

We note that parameter \( T_{\text{int}} \) is also an important factor to impact the finally created results. We first consider the case of \( t_5 = 2000 \text{ s}, t_r = 3000 \text{ s} \) and \( T_{\text{int}} = 160 \text{ s} \), corresponding to the timetable in Fig. 14(a). In this case, trains 6 and 8, respectively, are trapped at stations Xiaohongmen and Jiugong before the subway line is recovered, while train 10 is rescheduled to continue its trip by using the crossover tracks and outbound sections. However, an unreasonable \( T_{\text{int}} \) might cause the deadlock of the generated new timetable. For example, if the duration of departure interval is too small, some trains may be trapped on the same section because of the limited capacity of the next station when an incident takes place. Fig. 14(b) shows the rescheduled results with \( T_{\text{int}} = 120 \text{ s} \). In this timetable, trains 12, 24 and 32 fail in being rescheduled with any rescheduling methods because of the infeasibility of the station capacity. In such cases, we suggest that the practical duration of departure intervals should not be smaller than the sum of the maximal section trip time and dwelling time, to ensure that only one train is allowed to run on any section for all the time.

5.2. The performance criteria of rescheduled timetables

To investigate the performance criteria of rescheduled timetables from the statistical point of view, we are intended to perform more extensive case studies in the following discussion, where the operation data are the same to those adopted in Example 5.2. First, with two given incident time intervals, i.e., (1) \( t_5 = 1500 \text{ s}, t_r = 2800 \text{ s} \) and (2) \( t_5 = 2000 \text{ s} \) and \( t_r = 4000 \text{ s} \), we adjust the duration \( T_{\text{int}} \) of departure intervals to show the variant tendency of different performance criteria. Results of this set of experiments can serve as the reference indicators for subway operators when they schedule train timetables. To avoid the potential deadlocks mentioned above, we particularly take parameter \( T_{\text{int}} \) between 160 s and 300 s. The statistical performance criteria of rescheduled timetables are given in Fig. 15. Typically, with the increase of \( T_{\text{int}} \), both the total delay and clear time delay ratio will change according to a nearly decreasing tendency. However, no evident trend can be deduced for the maximal delay in the computational results. For instance, although different \( T_{\text{int}} \) are adopted in Fig. 15(a), the generated maximal delays still keep the same over the different numerical experiments. Actually, this is in essence caused by the situation that some trains are trapped at stations (see the trajectory of train 2 in Fig. 13(b)). Analogously, the variation tendencies of service balance associated with different departure interval are not clear. While from Fig. 15(b), an interesting phenomenon can be observed that the change rule of service balance is similar to that of maximal delays to some extent when no trains are trapped caused by the disruption.

Next, we consider the influence that the incident duration brings to the rescheduled timetable, where two sets of experiments are made in what follows. For simplicity, we set \( t_r \) as a constant and \( t_5 \) as a randomly generated integer in each experiment. The results are presented in Fig. 16. Clearly, with the increase of \( t_5 \), all of the performance criteria will increase according to approximately linear functions. This coincides with our intuitive understanding. In addition, it is interesting to mention that the variation tendencies of the same performance criterion are similar to each other in these two sets of experiments.

6. Conclusions

To rapidly reschedule the trains in a subway traffic system, we formulated a new mix-integer programming model on a faulted double-track subway line by simultaneously considering the roles of crossover tracks. To generate a high-quality rescheduling plan, we further proposed a new rescheduling method, called efficient train rescheduling strategy (ETRS), based on a discrete event model. In this solution method, the capacity check algorithm is regarded as the core part, which aims to prevent the potential deadlocks in rescheduling process. Due to the high-efficiency of the discrete event model, the ETRS can reschedule a larger number of trains on a subway line within a relative short computational time (at millisecond level), and this performance had been demonstrated by two examples, where a sample line and a real-world line (i.e., Beijing Yizhuang subway line) were taken into consideration. In the first example, the quality of the solution created by ETRS is proved to be similar to that produced by GAMS optimization software, which illustrated the effectiveness of the proposed approaches. In addition, from the results of a series of numerical experiments in the second example, it follows that two performance criteria, i.e., the total delay and the clear time delay ratio will decrease with the increase of duration of train departure intervals, while the maximal delay and service balance have no evident variation tendencies. Moreover, if the incident duration increases, all of the performance criteria are uniformly increasing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 71271022, 71422002), and the Research Foundation of State Key Laboratory of Rail Traffic Control and Safety, China (Grant no. RCS2014ZZ001).
References


