Train Shunting and Routing in High-Speed Railway Depot

Xiaoming Xu*
Maged M. Dessouky†

June 9, 2021

Abstract

The high-quality scheduling of passenger trains in a depot influences the reliability of train timetables during the day. Accordingly, this study examines the problem of train shunting and routing in a depot with consideration of daily maintenance, cleaning operation, and safety operational requirements. To cope with this complex problem, we first construct a two-layer time-space network in which each layer can only be used by trains traveling in the same direction. We then formulate the considered problem as a minimum-cost multi-commodity network flow model with incompatible arc sets and operational constraints. To solve the network flow problem, we present a Lagrangian relaxation heuristic. Finally, several computational experiments with real-life data on depots and randomly generated data on trains’ arrival and departure times at the depot are conducted to confirm the effectiveness of our model and the efficiency of the proposed heuristics.

Keywords: Train shunting; maintenance; schedule; time-space network; Lagrangian relaxation

---

*School of Automotive and Transportation Engineering, Hefei University of Technology, Hefei, China; email: xmxu@hfut.edu.cn. Corresponding author.
†Daniel J. Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, California 90089-0193, U.S.A.; email: maged@usc.edu
1 Introduction

To ensure safety, trains (electric multiple units; EMUs) on China’s high-speed railways are required to undergo maintenance each night. This daily maintenance is performed in a depot, which is usually near a station (see, e.g., Haahr et al. 2017). A train schedule in the depot indicates the assignment of tracks and the arrival and departure times for trains on each assigned track. Such a schedule plays an important role in railway operation because it can directly influence the reliability of the train schedule during the day. A low-quality train shunting schedule may result in delays in the availability of trains during the day, which in turn results in delays in the departures of planned train services. Even worse, low-quality shunting schedules may decrease the use of depot resources, which may result in the rejection of some trains to be checked and/or parked in the current depot for picking up train services, thus leading to cancellations of some planned train services for the next day.

In this study, we examine a train scheduling problem of simultaneously determining train shunting schedules and routes in the depots of high-speed railways such as in China, where some practical requirements, such as daily maintenance, cleaning operations, and safety operational requirements such as minimum headway constraints and track capacity constraints, are taken into consideration. We assume that (i) there are two kinds of train lengths; (ii) each train has planned arrival and departure times at the depot; (iii) train coupling and decoupling are not allowed in the depot; (iv) trains can move bidirectionally on tracks without locomotives; and (v) the depot is empty at both the beginning and ending times of the planning horizon. We then develop a Lagrangian relaxation heuristic to solve the considered problem. Several computational experiments are conducted with practical data on the depot and randomly generated data on train arrival and departure times to confirm the efficiency and effectiveness of the developed heuristic.

Following the definition of Freling et al. (2005) and Kroon et al. (2008) “The process of parking train units, together with several related processes, is called shunting,” the corresponding planning problem is then called the Train Unit Shunting Problem (TUSP) (see, e.g., Lentink et al. 2006). The literature includes a great deal of research on train shunting. Blasum et al. (1999) proved the NP-completeness of the problem of dispatching trams in a depot with the objective of minimizing the number of shunting movements. More special cases of dispatching trams in a depot were addressed by Winter and Zimmermann (2000). Tomii et al. (1999) and Tomii and Zhou (2000) proposed a two-stage algorithm based on the genetic algorithm and a “program evaluation and review technique” for a simple version of TUSP in which a single train unit can park at a shunt track at each time point. Freling et al. (2005) introduced the problem of shunting passenger train units in a railway station. To cope with this complex problem, Freling et al. decomposed their problem into two smaller subproblems, including a matching problem and a track allocation (or parking) problem. The former aims to make an optimal assignment of arriving shunt units to departing shunt units, and the latter aims to make an optimal assignment of tracks to train units. Kroon et al. (2008) later presented a new model for TUSP that could solve the matching and parking subproblems in an integrated manner. Lentink et al. (2006) introduced the routing subproblem of TUSP and presented a four-step algorithm solution approach. Jacobsen and Pisinger (2011) studied the problem of
shunting train units in a railway workshop area in which train unit coupling and decoupling were not considered. Considering the complexity of their problem, Jacobsen and Pisinger proposed three heuristic approaches. Haahr et al. (2017) presented several solution methods for TUSP, including a constraint programming formulation, a column generation approach, and a randomized greedy heuristic. Interested readers are referred to Cordeau et al. (1998) and Lusby et al. (2011a) for comprehensive reviews of TUSPs. Moreover, van den Broek (2016) developed a local search approach for the train shunting and scheduling problem with the consideration of train matching, parking, service tasks scheduling as well as train routing decisions.

Our problem is also related to the problem of routing or dispatching trains through railway stations. Caprara et al. (2007) performed a comprehensive survey of the train routing problem. The train routing problem has three versions. The easiest version requires that the planned train arrival and departure times are fixed and that the paths used by the trains are uniquely determined once the platforms have been selected, see, e.g., De Luca Cardillo and Mione (1998) and Billionnet (2003). A more complex version allows the planned arrival and departure times to be changed, whereas the paths traversed by the trains are uniquely determined by the choice of the platforms, see, e.g., Carey and Carville (2003). The more general version of the problem allows the planned train arrival and departure times to be changed, and the arrival and departure paths are not fixed, see, e.g., Zwaneveld et al. (1996; 2001), Caprara et al. (2011), and Lusby et al. (2011b).

Our work differs from traditional studies of TUSPs in that our model makes train shunting and routing decisions simultaneously. Our model, however, does not include train unit coupling and decoupling decisions (see, e.g., matching decisions in Freling et al. 2005 and Kroon et al. 2008), but it includes minimum headway requirements at track joints, which are not covered in most models of TUSPs but are quite common in train platforming/routing models (see, e.g., Caprara et al. 2011). Our research topic bears some similarities to that of Jacobsen and Pisinger (2011) and van den Broek (2016) because our study also considers the assignment of tracks to trains (e.g., parking decisions) and the routing decisions of trains in the depot, whereas Jacobsen and Pisinger (2011) and van den Broek (2016) did not consider the minimum headway requirements at track joints.

This paper therefore makes several contributions to the literature. First, to the best of our knowledge, our study is the first to examine an integrated optimization approach to train shunting and routing in a depot, that considers both maintenance requirements and safety operational constraints, such as track capacity and minimum headway time at joint points. Second, we design a two-layer time-space network for the considered problem. In this time-space network, several vertices for each repair track (storage track) are elaborately designed to represent the situation in which one long train or two short trains can park at a repair track (storage track) at each time instant. This network is flexible and can be modified or extended for various practical operational requirements. Third, we propose a Lagrangian relaxation heuristic to solve the proposed model in which the feasible solution heuristic schedules trains one by one according to a ranked train order. To obtain the train order, we propose three ranking heuristics: a relaxed solution-based heuristic, a relatedness-based ranking heuristic, and a greedy heuristic.

The rest of this paper is organized as follows. In Section 2, we provide a detailed description of the considered
problem. In Section 3, we introduce the two-layer time-space network representation and a corresponding multi-commodity network flow formulation of the problem. In Section 4, we describe our Lagrangian relaxation heuristic and feasible solution methods. Computational experiments are conducted to confirm the effectiveness of our network flow model and solution methods, and the results are reported in Section 5. Some conclusions are drawn in Section 6.

2 Problem Description

Figure 1: Layout of a typical electric multiple unit depot.

Figure 1 shows the topology of a typical EMU depot in China. It consists of a stabling yard, a cleaning area, a running shed, and a series of shunting track segments. The stabling yard has several parallel storage tracks (see, e.g., the depository track in Jacobsen and Pisinger 2011) on which trains can park to wait for daily maintenance or departure. The running shed has several parallel repair tracks on which trains can park for daily maintenance (i.e., trains are repaired on repair tracks). The cleaning area includes one or more cleaning tracks along which equipment is installed to clean the trains. In this study, all repair track segments are first in last out tracks; that is, they can be approached from one side only.

On China’s high-speed railways, EMUs can be roughly divided into two categories according to whether they contain 8 cars (i.e., short trains) or 16 cars (i.e., long trains). In addition, train coupling and decoupling operations are usually not allowed during daily maintenance. Each storage or repair track has two positions, as shown in Figure 1. Each position can be occupied by a maximum of one 8-car train at each time point. A 16-car train must occupy both position I and position II when on a storage or repair track. Each cleaning track can be occupied by a maximum of one train at any time. These storage tracks, repair tracks, and clean tracks are all called operation tracks because some operations may be implemented for trains on these tracks. In addition to these operation tracks, a series of shunting tracks are used to draw trains among the depot’s various areas. Considering the safety operational constraints (see Section 2.2), the nodes along all of the tracks are also marked in the depot.

The general daily maintenance process of a train in the depot proceeds as follows. After arriving at the depot, a train is first cleaned on a cleaning track. Next, maintenance is implemented on a repair track, and the train finally parks on a storage track to await its planned departure the next day. However, due to the limited number
of cleaning tracks, a train may first park on a storage track before cleaning, or even undergo maintenance on the repair track before being cleaned on a cleaning track. Therefore, the train scheduling problem in the depot studied here is defined as follows. Given a train depot and a set of trains with planned arrival and departure times, the problem aims to determine a shunting schedule and routing solution for these trains that satisfies a series of maintenance requirements and safety operational requirements such as minimum headway constraints and track capacity constraints. The shunting schedule determines the assignment of operation tracks to the trains and train maintenance processes. The routing solution determines the time at which the trains enter and leave their assigned tracks. It is apparent that the considered problem is NP-hard because it covers several NP-hard problems as special cases, such as the train timetabling problem (see, e.g., Caprara et al. 2002) and the train unit shunting problem (see, e.g., Freling et al. 2005 and Haahr et al. 2017).

2.1 Input data

The planning horizon, denoted by \([0, T]\), is discretized, and the time units are expressed as integers (e.g., \(T = 1440\) if the planning horizon is 24 hours and each time unit is 1 minute). Table 1 summarizes the problem’s input parameters, where all time-related parameters are integer-valued.

2.1.1 Depot data

Let \(N = \{n_1, n_2, \ldots, n_{|N|}\}\) be the set of nodes marked in the depot, and let \(N_0 \subseteq N\) be the set of joint nodes in which each joint node connects three or more tracks (see node 3 in Figure 1). Note that one physical joint node between two crossover tracks is represented by two separate joint nodes in \(N_0\); see, for example, nodes 19 and 22 in Figure 1, which differs from the presentation in Caprara et al. (2011, Figure 1). We collect these separate joint nodes in node set \(N_1\) and \(N_1 \subseteq N_0\) and denote \(\phi(i) = j\) and \(\phi(j) = i\) if and only if nodes \(i\) and \(j\)’s physical positions in the depot are the same. According to our representation, tracks \((28,29), (29,33), (31,32),\) and \((32,30)\) are considered, while tracks \((29,30), (29,31), (32,28),\) and \((32,33)\) do not exist in our study (see, Figure 1). Thus, some impossible paths (passing through tracks) in practical operations, such as paths \((31,32) \rightarrow (32,33)\) and \((28,29) \rightarrow (29,30)\), then do not exist in our solution. Let \(S = \{s_1, s_2, \ldots, s_{|S|}\}\) be the set of considered storage tracks in the stabling yard, \(R = \{r_1, r_2, \ldots, r_{|R|}\}\) be the set of considered repair tracks in the running shed, \(W = \{w_1, w_2, \ldots, w_{|W|}\}\) be the set of considered cleaning tracks in the cleaning area, and \(E = \{e_1, e_2, \ldots, e_{|E|}\}\) be the set of shunting tracks where for each \(e = (i, j) \in E\) and \(i, j \in N\). The tracks mentioned above, including operation and shunting tracks, are all two-way tracks on which trains can move in both \textit{in} and \textit{out} directions.

2.1.2 Train data

Let \(K_0\) and \(K_1\) be the sets of short trains and long trains, respectively. Let \(K\) be the set of all of the considered trains; i.e., \(K = K_0 \cup K_1\). For each train \(k \in K\), the input data include (i) the planned arrival time \(t_{ak}\) for train \(k\) to the depot; (ii) the planned departure time \(t_{dk}\) for train \(k\) from the depot; (iii) the required cleaning time \(\alpha_k\);
Table 1: Summary of input data.

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depot data</td>
<td>$N$</td>
<td>set of nodes, $N = {n_1, n_2, \ldots, n_{</td>
</tr>
<tr>
<td></td>
<td>$N_0$</td>
<td>set of joint nodes, $N_0 \subseteq N$</td>
</tr>
<tr>
<td></td>
<td>$N_1$</td>
<td>set of joint nodes corresponding to the physical joint node between two crossover tracks, $N_1 \subseteq N_0$</td>
</tr>
<tr>
<td></td>
<td>$S$</td>
<td>set of storage tracks, $S = {s_1, s_2, \ldots, s_{</td>
</tr>
<tr>
<td></td>
<td>$R$</td>
<td>set of repair tracks, $R = {r_1, r_2, \ldots, r_{</td>
</tr>
<tr>
<td></td>
<td>$W$</td>
<td>set of cleaning tracks, $W = {w_1, w_2, \ldots, w_{</td>
</tr>
<tr>
<td></td>
<td>$E$</td>
<td>set of shunting tracks, $E = {e_1, e_2, \ldots, e_{</td>
</tr>
<tr>
<td></td>
<td>$h_i$</td>
<td>minimum headway between two trains traversing through the same node $i \in N$</td>
</tr>
<tr>
<td>Train data</td>
<td>$K_0$</td>
<td>set of short trains, e.g., 8-car trains</td>
</tr>
<tr>
<td></td>
<td>$K_1$</td>
<td>set of long trains, e.g., 16-car trains</td>
</tr>
<tr>
<td></td>
<td>$K$</td>
<td>set of all trains, $K = {k_1, k_2, \ldots, k_{</td>
</tr>
<tr>
<td></td>
<td>$t_a^k$</td>
<td>planned arrival time for train $k$ to the depot, $k \in K$</td>
</tr>
<tr>
<td></td>
<td>$t_d^k$</td>
<td>planned departure time for train $k$ from the depot</td>
</tr>
<tr>
<td></td>
<td>$\alpha_k$</td>
<td>required cleaning time for train $k$</td>
</tr>
<tr>
<td></td>
<td>$\beta_k$</td>
<td>required repair time for train $k$</td>
</tr>
<tr>
<td></td>
<td>$\tau_p^k$</td>
<td>minimum required time to draw train $k$ to traverse operation track $p \in S \cup R \cup W \cup E$</td>
</tr>
<tr>
<td></td>
<td>$c_k$</td>
<td>operating cost per unit time for train $k$ when it is running on a track or undergoing operations on repair or cleaning tracks</td>
</tr>
<tr>
<td></td>
<td>$c_k'$</td>
<td>operating cost per unit time for train $k$ when it is waiting on a repair or cleaning track</td>
</tr>
<tr>
<td></td>
<td>$c_k''$</td>
<td>unit penalty of train $k$’s shift</td>
</tr>
<tr>
<td></td>
<td>$\pi_k$</td>
<td>penalty of rejecting train $k$’s maintenance requirement</td>
</tr>
</tbody>
</table>

(iv) the required repair time $\beta_k$; (v) the minimum required time $\tau_p^k$ to draw train $k$ to traverse track segment $p$, where $p \in S \cup R \cup W \cup E$; (vi) the operating cost $c_k$ incurred per time unit when the train is running on a track segment or undergoing operations on repair or cleaning tracks; and (vii) the operating cost $c_k'$ incurred per time unit when the train is waiting on a repair track or cleaning track. Note that because operation tracks are very short, the time for drawing a train on an operation track $p \in S \cup R$ is mainly consumed in the ancillary work rather than in traversing the track. Hence, the time needed to draw a train between any two points on the same operation track $p$ is set to be the same $\tau_p^k$. Using the storage track $s_0$ in Figure 1 as an example, the time needed to draw a short train to traverse track $s_0$’s position I and the time needed to draw a short train to traverse the entire track $s_0$ (including both positions I and II) are the same $\tau_{s_0}^k$.

Due to the limited maintenance capacity, a train’s maintenance work may not be finished before its planned departure time. With this consideration, we impose a penalty measurement on the maintenance schedule if a train’s actual departure time $\hat{t}_d^k$ from the depot is later than the planned $t_d^k$. For each train $k \in K$, a penalty
c_k^d \zeta_k is imposed on train k’s shift \zeta_k, which is defined as \max\{0, \hat{t}_d^k - t_d^k\}. In the worst-case scenario, train k’s maintenance requirement may be rejected by the depot; that is, train k is unscheduled, where a penalty of \pi_k is incurred.

2.2 Objective and constraints

The objective of the considered problem is to determine the shunting schedule and the routing solution for each train, such that the total cost is minimized. The route for train k begins and ends at a storage track; these two storage tracks may be different. A feasible solution must satisfy the following constraints.

- Daily maintenance requirements: Each train must be cleaned and repaired at the minimum required operation time during the planning horizon if this train is not canceled.
- Headway constraints at nodes: For each node i \in N, the arrivals of two trains at this node must be at least \( h_i \) time units apart, the departures of two trains from this node must be at least \( h_i \) time units apart, and one train’s arrival and another train’s departure from this node must be at least \( h_i \) time units apart.
- Track capacity constraints: Two or more trains may not occupy the same point on the track at the same time. For example, a train is not allowed to overtake another train on the track segment.

3 Time-Space Network Formulation

In this section, we formulate the considered problem as a minimum-cost multi-commodity network flow problem with several restrictions, where each commodity represents a train. The underlying network is an acyclic directed two-layer time-space network \( G = (V, A) \). In what follows, we first introduce the construction of our two-layer time-space network in Section 3.1. We then present the flow restrictions in Section 3.2. Finally, we propose the network flow formulation for the considered problem in Section 3.3.

3.1 Time-space network

![Figure 2: Two-layer time-space network structure.](image)
In this section, we present the construction of our two-layer time-space network, including an inward layer and an outward layer (see Figure 2). A train traversing the arcs in the inward layer represents the situation in which this train is heading toward or entering the running shed, and a train traversing the arcs in the outward layer represents the situation in which this train is heading toward the station (see, for example, Figure 1). The set of all possible time instants in the planning horizon is \(\{0, 1, \ldots, T\}\), which forms the “time” dimension of \(G\). The time dimension is the same in both the inward layer and the outward layer, whereas the “space” dimensions in the inward and outward layers are different. Details of the vertices and arcs in network \(G\) are introduced in what follows.

### 3.1.1 Vertices in inward layer network

For each storage track \(s \in S\), there are five components in the “space” dimension in the inward layer network, as shown in Figure 3. These components include: (i) \(\tilde{\rho}_S(s)\), which represents a train’s arrival at storage track \(s\); (ii) \(\bar{\rho}_S(s)\), which represents a train’s occupation of position I of storage track \(s\); (iii) \(\tilde{\rho}'_S(s)\), which represents a train’s occupation of position II of storage track \(s\); (iv) \(\bar{\rho}''_S(s)\), which represents a train’s occupation of positions I and II of storage track \(s\); and (v) \(\hat{\rho}_S(s)\), which represents a train’s departure from storage track \(s\).

For each cleaning track \(w \in W\), there are three components in the “space” dimension in the inward layer network. These components are as follows: (i) \(\tilde{\rho}_W(w)\), which represents a train’s arrival at cleaning track \(w\); (ii) \(\rho_W(w)\), which represents a train’s movement on cleaning track \(w\); and (iii) \(\tilde{\rho}_W(w)\) which represents a train’s occupation of cleaning track \(w\).

For each repair track \(r \in R\), there are seven components in the “space” dimension in the inward layer network, as shown in Figure 4. These components are as follows: (i) \(\tilde{\rho}_R(r)\), which represents a train’s arrival at repair track \(r\); (ii) \(\rho_R(r)\), which represents a train’s arrival at position I of repair track \(r\); (iii) \(\hat{\rho}_R(r)\), which represents a train’s occupation of position I of repair track \(r\); (iv) \(\tilde{\rho}'_R(r)\), which represents a train’s arrival at position II of repair track \(r\); (v) \(\tilde{\rho}''_R(r)\), which represents a train’s occupation of position II of repair track \(r\); (vi) \(\rho''_R(r)\), which represents a train’s arrival at positions I and II of repair track \(r\); and (vii) \(\rho'''_R(r)\), which represents a train’s occupation of positions I and II of repair track \(r\).

For each node \(i \in N\), there is one component \(\rho(i)\) (see Figure 3) in the “space” dimension in the inward layer network. Mathematically, we let

\[
\Omega_{in} = \{\tilde{\rho}_S(s), \bar{\rho}_S(s), \tilde{\rho}'_S(s), \tilde{\rho}''_S(s), \tilde{\rho}_S(s) \mid s \in S\} \cup \{\tilde{\rho}_W(w), \rho_W(w), \tilde{\rho}_W(w) \mid w \in W\} \\
\cup \{\tilde{\rho}_R(r), \rho_R(r), \bar{\rho}_R(r), \tilde{\rho}'_R(r), \tilde{\rho}''_R(r), \tilde{\rho}'_R(r), \rho'''_R(r) \mid r \in R\} \cup \{\rho(i) \mid i \in N\}
\]

denote the “space” dimension in the inward layer time-space network, and

\[
V_{in} = \{(\omega, t) \mid \omega \in \Omega_{in}; t = 0, 1, \ldots, T\}
\]

denote the set of vertices in the inward layer time-space network.
3.1.2 Vertices in outward layer network

For each repair track \( r \in R \), there are four components in the “space” dimension in the outward layer network, as shown in Figure 4. These components are as follows: (i) \( \bar{\varrho}_R(r) \), which represents a train’s occupation of position I of repair track \( r \); (ii) \( \bar{\varrho}'_R(r) \), which represents a train’s occupation of position II of repair track \( r \); (iii) \( \bar{\varrho}''_R(r) \), which represents a train’s occupation of positions I and II of repair track \( r \); and (iv) \( \hat{\varrho}_R(r) \), which represents a train’s departure from repair track \( r \).

For each cleaning track \( w \in W \), there are also three components in the “space” dimension in the outward layer network, as shown in Figure 5. These components are as follows: (i) \( \ddot{\varrho}_W(w) \), which represents a train’s arrival at cleaning track \( w \); (ii) \( \varrho_W(w) \), which represents a train’s movement on cleaning track \( w \); and (iii) \( \bar{\varrho}_W(w) \), which represents a train’s occupation of cleaning track \( w \).

For each storage track \( s \in S \), there are also five components in the “space” dimension in the outward layer network, as shown in Figure 5. These components are as follows: (i) \( \ddot{\varrho}_S(s) \), which represents a train’s arrival at storage track \( s \); (ii) \( \bar{\varrho}_S(s) \), which represents a train’s occupation of position I of storage track \( s \); (iii) \( \bar{\varrho}'_S(s) \), which represents a train’s occupation of position II of storage track \( s \); (iv) \( \bar{\varrho}''_S(s) \), which represents a train’s occupation of positions I and II of storage track \( s \); and (v) \( \hat{\varrho}_S(s) \), which represents a train’s departure from storage track \( s \).

For each node \( i \in N \), there is also one component \( \varrho(i) \) (see Figure 5) in the “space” dimension in the outward layer network. Mathematically, we let

\[
\Omega_{out} = \{ \ddot{\varrho}_S(s), \bar{\varrho}_S(s), \bar{\varrho}'_S(s), \bar{\varrho}''_S(s), \hat{\varrho}_S(s) \mid s \in S \}
\]

\[
\cup \{ \ddot{\varrho}_W(w), \varrho_W(w), \bar{\varrho}_W(w) \mid w \in W \}
\]

\[
\cup \{ \bar{\varrho}_R(r), \bar{\varrho}'_R(r), \bar{\varrho}''_R(r), \hat{\varrho}_R(r) \mid r \in R \}
\]

\[
\cup \{ \varrho(i) \mid i \in N \}
\]

denote the “space” dimension in the outward layer time-space network, and let

\[
V_{out} = \{ (\omega, t) \mid \omega \in \Omega_{out}; t = 0, 1, \ldots, T \}
\]

denote the set of vertices in the outward layer time-space network.

Finally, the vertex set of the time-space network \( G \) is

\[
V = \{ o, d \} \cup V_{in} \cup V_{out},
\]

where vertex \( o \) and vertex \( d \) are the artificial source and the artificial sink, respectively, for the multi-commodity flow.

3.1.3 Arcs

The arc set \( A \) of the time-space network \( G \) contains the following several types of arcs:

- Starting arcs: For the intermediate nodes between the station and depot, there exist some starting arcs which allow a train \( k \) to start its operation in the depot at or after its planned arrival time \( t_{k}^a \).
• Ending arcs: For the intermediate nodes between the station and depot, there exist some ending arcs which allow a train $k$ to complete its operation in the depot at or after its planned departure time $t^d_k$.

• Drawing arcs: For tracks in $S \cup R \cup W \cup E$, there exist some drawing arcs. A train traversing a drawing arc implies that this train is traversing the track that corresponds to the drawing arc.

• Cleaning arcs: For cleaning tracks, there exist some cleaning arcs, which allow a train to be cleaned with a minimum required cleaning time $\alpha_k$ when it dwells at a cleaning track and travels in the outward direction.

• Repairing arcs: For repair tracks, there exist some repairing arcs, which allow a compatible train to be repaired with the minimum required time $\beta_k$ when it is dwelling on the repair track $r$.

• Waiting arcs: For tracks in $S \cup R \cup W$, there exist some waiting arcs, which allow a compatible train to park on the considered track.

• Departure arcs: For tracks in $S \cup R$, there exist some departure arcs, which represent the situation in which a train is about to leave the considered track.

• Transfer arcs: For tracks in $S \cup W$, there exist some transfer arcs, which allow trains to traverse the nodes between operation tracks and shunting tracks.

• Switch arcs: For repair tracks, there exist some switch arcs, which represent the situation in which a train has been repaired on the considered track and switches its travel from the inward direction to the outward direction. Note that these arcs change “layer” components from the inward layer to the outward layer.

• Dummy arc: There is a dummy arc $\bar{o} \rightarrow \bar{d}$ in network $G$. One can say that a train traversing this arc represents the situation in which this train’s daily maintenance requirement is rejected by the depot.

Figures 3, 4, and 5 illustrate different vertices and arcs described above. Note that these arcs have a common characteristic that each arc’s ending time instant is not earlier than its starting time instant. Hence, network $G$ is acyclic when all time-related parameters are positive. Moreover, there is a vector of cost coefficients $(\xi^k_{uv}, \ldots, \xi^{|K|}_{uv})$ associated with each arc $u \rightarrow v \in A$, where cost coefficient $\xi^k_{uv}$ represents the cost for train $k$ to traverse arc $u \rightarrow v$. Each arc $u \rightarrow v \in A$ has a unit capacity per train. More details of the arcs described above.
3.2 Constraints

A path from vertex $o$ to vertex $d$ in our two-layer time-space network corresponds to a schedule of a train that determines the train’s sequence of operations and route (passing through track segments). This schedule is feasible for train $k$ if and only if all cost coefficients of train $k$ along this path are finite. Given the constructed two-layer time-space network, our purpose is to determine feasible paths for all trains such that the total cost is minimized. However, these paths may not satisfy the constraints discussed in Section 2.2. Hence, in addition to the standard network flow constraints such as flow balance constraints and supply/demand constraints, our multi-commodity flow model also includes the following requirements and constraints:

- Cleaning operation requirements: For each $k \in K$, the train must be cleaned once during daily maintenance if it is not cancelled; we then impose the constraint that train $k$’s total flow along the cleaning arcs in the
Repair operation requirements: For each \( k \in K \), the train must be repaired once during daily maintenance if it is not cancelled. Therefore, for each \( k \in K_0 \), we impose the constraint that train \( k \)'s total flow along the repairing arcs in the arc subset

\[
B^k_W = A \cap \left\{ \{o \rightarrow d\} \cup \{(\rho_W(w), t) \rightarrow (\rho_W(w), t'), (\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t') \mid w \in W; \ t, t' = 0, 1, \ldots, T\} \right\}
\]

must be one. For each \( k \in K_1 \), we impose the constraint that train \( k \)'s total flow along the repairing arcs in the arc subset

\[
B^k_R = A \cap \left\{ \{o \rightarrow d\} \cup \{(\rho_R(r), t) \rightarrow (\rho_R(r), t'), (\hat{\rho}_R(r), t) \rightarrow (\hat{\rho}_R(r), t') \mid r \in R; \ t, t' = 0, 1, \ldots, T\} \right\}
\]

must be one.

- Headway constraints: For each joint node \( i \in N_0 \), the time difference between two trains that traverse node \( i \) must be at least \( h_i \) units apart (see Section 2.2). For each \( i \in N_0 \) and each \( t_1 = 0, 1, \ldots, T - h_i + 1 \) we allow no more than one train to arrive at or depart from node \( i \) during \([t_1, t_1 + h_i - 1]\). Denote

\[
\tilde{C}^1_{it_1} = A \cap \left\{ \{(\rho(i), t) \rightarrow (\rho_S(s), t), (\rho(i), t) \rightarrow (\tilde{\rho}_S(s), t) \mid s \in S; \ t = 0, 1, \ldots, T; \ t_1 \leq t \leq t_1 + h_i - 1\} \right\}
\]

\[
\cup \{(\rho(i), t) \rightarrow (\tilde{\rho}_W(w), t), (\rho(i), t) \rightarrow (\tilde{\rho}_W(w), t) \mid w \in W; \ t = 0, 1, \ldots, T; \ t_1 \leq t \leq t_1 + h_i - 1\}
\]

\[
\cup \{(\rho(i), t) \rightarrow (\rho_R(r), t), (\rho(i), t) \rightarrow (\tilde{\rho}_R(r), t) \rightarrow r \in R; \ t = 0, 1, \ldots, T; \ t_1 \leq t \leq t_1 + h_i - 1\}
\]

\[
\cup \{(\rho(i), t) \rightarrow (\rho(j), t'), (\rho(i), t) \rightarrow (\tilde{\rho}(j), t') \mid j \in N \setminus \{i\}; \ t, t' = 0, 1, \ldots, T; \ t_1 \leq t \leq t_1 + h_i - 1\}
\]

Hence, for each node \( i \in N_0 \setminus N_1 \) and time instant \( t_1 = 0, 1, \ldots, T - h_i + 1 \), we impose the constraint that the total train flow along the arcs in the arc subset \( \tilde{C}^1_{it_1} \) is at most one. For each node \( i \in N_1 \) and time instant \( t_1 = 0, 1, \ldots, T - h_i + 1 \), we impose the constraint that the total train flow along the arcs in the arc subset \( \tilde{C}^1_{it_1} \cup \tilde{C}^1_{jt_1} \) is at most one, where \( j = \phi(i) \); that is, nodes \( i \) and \( j \) correspond to the same physical joint node between two crossover tracks in the depot (see, for example, nodes 29 and 32 in Figure 1). For convenience, denote

\[
C^1_{it_1} = \begin{cases} 
\tilde{C}^1_{it_1}, & \text{if } i \in N_0 \setminus N_1, t_1 = 0, 1, \ldots, T - h_i + 1; \\
\tilde{C}^1_{it_1} \cup \tilde{C}^1_{jt_1}, & \text{if } i, j \in N_1 \text{ and } j = \phi(i), t_1 = 0, 1, \ldots, T - h_i + 1.
\end{cases}
\]

- Track capacity constraints: For each track segment in the depot, two or more trains cannot park in the same position of a track segment at any time. Hence, for each storage track \( s \in S \) and time instant \( t_1 = 0, 1, \ldots, T \), considering position 1 of storage track \( s \), we impose the constraint that the total train flow along the arcs in
the arc subset

\[
C_{st1}^2 = A \cap \left\{ (\bar{\rho}_S(s), t_1 - 1) \to (\bar{\rho}_S(s), t_1), (\bar{\rho}''_S(s), t_1 - 1) \to (\bar{\rho}''_S(s), t_1), \\
(\bar{\rho}_S(s), t_1 - 1) \to (\bar{\rho}_S(s), t_1), (\bar{\rho}''_S(s), t_1 - 1) \to (\bar{\rho}''_S(s), t_1) \right\}
\]

\[
\cup \left\{ (\bar{\rho}S(s), t) \to (\bar{\rho}S(s), t'), (\bar{\rho}S(s), t') \to (\bar{\rho}S(s), t'), \\
(\bar{\rho}S(s), t) \to (\bar{\rho}S(s), t'), (\bar{\rho}S(s), t) \to (\bar{\rho}S(s), t') \right\} \text{ if } t, t' = 0, 1, \ldots, T, t \leq t_1 < t'
\]

is at most one. Considering position II of storage track \( s \), we impose the constraint that the total train flow along the arcs in the arc subset

\[
C_{st1}^3 = A \cap \left\{ (\bar{\rho}S(s), t_1 - 1) \to (\bar{\rho}S(s), t_1), (\bar{\rho}''_S(s), t_1 - 1) \to (\bar{\rho}''_S(s), t_1), \\
(\bar{\rho}S(s), t_1 - 1) \to (\bar{\rho}S(s), t_1), (\bar{\rho}''_S(s), t_1 - 1) \to (\bar{\rho}''_S(s), t_1) \right\}
\]

\[
\cup \left\{ (\bar{\rho}S(s), t) \to (\bar{\rho}S(s), t'), (\bar{\rho}S(s), t') \to (\bar{\rho}S(s), t'), \\
(\bar{\rho}S(s), t) \to (\bar{\rho}S(s), t'), (\bar{\rho}S(s), t) \to (\bar{\rho}S(s), t') \right\} \text{ if } t, t' = 0, 1, \ldots, T, t \leq t_1 < t'
\]

is at most one.

For each cleaning track \( w \in W \) and time instant \( t_1 = 0, 1, \ldots, T \), we impose the constraint that the total train flow along the arcs in the arc subset

\[
C_{wt1}^4 = A \cap \left\{ (\bar{\rho}W(w), t_1 - 1) \to (\bar{\rho}W(w), t_1), (\bar{\rho}W(w), t_1 - 1) \to (\bar{\rho}W(w), t_1) \right\}
\]

\[
\cup \left\{ (\bar{\rho}W(w), t) \to (\bar{\rho}W(w), t'), (\bar{\rho}W(w), t) \to (\bar{\rho}W(w), t') \right\} \text{ if } t, t' = 0, 1, \ldots, T, t \leq t_1 < t'
\]

is at most one.

For each repair track \( r \in R \) and time instant \( t_1 = 0, 1, \ldots, T \), considering \( r \)'s position I, we impose the constraint that the total train flow along the arcs in the arc subset

\[
C_{rt1}^5 = A \cap \left\{ (\bar{\rho}R(r), t_1 - 1) \to (\bar{\rho}R(r), t_1), (\bar{\rho}''_R(r), t_1 - 1) \to (\bar{\rho}''_R(r), t_1) \right\}
\]

\[
\cup \left\{ (\bar{\rho}R(r), t) \to (\bar{\rho}R(r), t'), (\bar{\rho}R(r), t) \to (\bar{\rho}R(r), t''), \\
(\rho R(r), t) \to (\bar{\rho}R(r), t'), (\rho R(r), t) \to (\bar{\rho}R(r), t''), \\
(\rho'' R(r), t) \to (\bar{\rho}R(r), t'), (\rho'' R(r), t) \to (\bar{\rho}R(r), t'') \right\} \text{ if } t, t' = 0, 1, \ldots, T, t \leq t_1 < t'
\]

is at most one. Considering \( r \)'s position II, we impose the constraint that the total train flow along the arcs in the arc subset

\[
C_{rt1}^6 = A \cap \left\{ (\bar{\rho}R(r), t_1 - 1) \to (\bar{\rho}R(r), t_1), (\bar{\rho}'' R(r), t_1 - 1) \to (\bar{\rho}'' R(r), t_1) \right\}
\]

\[
\cup \left\{ (\bar{\rho}R(r), t) \to (\bar{\rho}R(r), t'), (\bar{\rho}R(r), t) \to (\bar{\rho}R(r), t''), \\
(\rho R(r), t) \to (\bar{\rho}R(r), t'), (\rho R(r), t) \to (\bar{\rho}R(r), t''), \\
(\rho'' R(r), t) \to (\bar{\rho}R(r), t'), (\rho'' R(r), t) \to (\bar{\rho}R(r), t'') \right\} \text{ if } t, t' = 0, 1, \ldots, T, t \leq t_1 < t'
\]
is at most one.

For each shunting track \((i,j) \in E\) and time instant \(t_1 = 0,1,\ldots,T\), we impose the constraint that the total train flow along the arcs in the arc subset

\[
C_{ijt_1}^7 = A \cap \{(\rho(i),t) \rightarrow (\rho(j),t'), (\rho(j),t) \rightarrow (\rho(i),t'') | t, t', t'' = 0,1,\ldots,T; t \leq t_1 < t'; t \leq t_1 < t''\}
\]
is at most one.

Denote

\[
C = \{C_{it}^1 | i \in \mathbb{N}_0; t = 0,1,\ldots,T - h_i + 1\} \cup \{C_{st}^2, C_{st}^3 | s \in S; t = 0,1,\ldots,T\}
\]
\[
\cup \{C_{we}^4 | w \in \mathbb{W}; t = 0,1,\ldots,T\} \cup \{C_{rt}^5, C_{rt}^6 | r \in \mathbb{R}; t = 0,1,\ldots,T\} \cup \{C_{ijt}^7 | (i,j) \in E; t = 0,1,\ldots,T\}.
\]

Then, for any arc set \(C \in \mathcal{C}\), the total flow along the arcs in \(C\) cannot exceed one.

### 3.3 Integer programming formulation

For each train \(k \in \mathbb{K}\) and \(u \rightarrow v \in \mathbb{A}\), let \(x_{uv}^k = 1\) if arc \(u \rightarrow v\) is traversed by train \(k\), and \(x_{uv}^k = 0\) otherwise.

If each train is taken as a commodity, then the considered problem can be formulated as a minimum-cost multi-commodity network flow formulation represented by the following integer program.

\[
P: \quad \text{Minimize} \quad \sum_{k \in \mathbb{K}} \sum_{u \rightarrow v \in \mathbb{A}} g_{uv} x_{uv}^k \tag{1}
\]

subject to

\[
\sum_{\{v:o \rightarrow v \in \mathbb{A}\}} x_{ov}^k = 1, \quad \text{for all } k \in \mathbb{K} \tag{2}
\]

\[
\sum_{\{u:d \rightarrow u \in \mathbb{A}\}} x_{ud}^k = 1, \quad \text{for all } k \in \mathbb{K} \tag{3}
\]

\[
\sum_{\{u:v \rightarrow u \in \mathbb{A}\}} x_{uv}^k = \sum_{\{v:u \rightarrow v \in \mathbb{A}\}} x_{vu}^k \quad \text{for all } k \in \mathbb{K}, v \in \mathbb{V} \setminus \{o,d\} \tag{4}
\]

\[
\sum_{u \rightarrow v \in \mathbb{B}_w} x_{uv}^k = 1, \quad \text{for all } k \in \mathbb{K} \tag{5}
\]

\[
\sum_{k \in \mathbb{K}} \sum_{u \rightarrow v \in \mathbb{C}} x_{uv}^k \leq 1, \quad \text{for all } C \subset \mathcal{C} \tag{6}
\]

\[
x_{uv}^k \in \{0,1\}, \quad \text{for all } k \in \mathbb{K}, u \rightarrow v \in \mathbb{A} \tag{7}
\]

The objective function (1) minimizes the total cost of the system for the schedule duration; that is, the total cost of operating services at the cleaning tracks and repair tracks, waiting for operations, and penalization on the shift of trains. Constraints (2) are the supply constraints that require the outflow of each train at vertex \(o\) to be 1. Constraints (3) are the demand constraints that require the inflow of each train at vertex \(d\) to be 1. Constraints (4) are the flow balance constraints for trains. Constraints (5) cover the cleaning operation requirement constraints introduced in Section 3.2. Constraints (6) cover the departure headway constraints, arrival constraints, and track capacity constraints introduced in Section 3.2. Constraints (7) are the binary constraints of the decision variables.

The following Proposition 1 implies that the repair operation requirements presented in Section 3.2 are implicitly satisfied in our network flow model, which benefits from the construction of the two-layer time-space network. Thus, there is no need to construct constraints related to repair operation requirements in model \(P\).
Proposition 1 Repair operation requirements are always satisfied if problem $P$ has a solution.

Proof. See Appendix B.

4 Lagrangian Relaxation Heuristic

In this section, we present the Lagrangian relaxation heuristic to solve the proposed model $P$. Lagrangian relaxation has been widely and successfully used to solve transportation problems, see, e.g., Caprara et al. (2002), Fisher (2004), Mahmoudi and Zhou (2016), and Cacchiani et al. (2012).

4.1 Lagrangian relaxation

We use the Lagrangian relaxation technique to relax constraints (5) and (6) of problem $P$ and bring them into the objective function with associated Lagrangian multipliers $\lambda_k \geq 0 \ (k \in K)$ and $\mu_C \geq 0 \ (C \in C)$. Let $\lambda$ and $\mu$ denote the vector of $\lambda_k$ values and the vector of $\mu_C$ values, respectively. The Lagrangian relaxed problem associated with the original optimization problem $P$ can then be formulated as

$$\tilde{P}(\lambda, \mu) : \text{Minimize } \sum_{k \in K} \sum_{u \rightarrow v \in A} \xi_{uv}^k x_{uv}^k + \sum_{k \in K} \lambda_k \left( \sum_{u \rightarrow v \in B \cup \lambda, \mu} x_{uv}^k - 1 \right) + \sum_{C \in C} \mu_C \left( \sum_{k \in K} \sum_{u \rightarrow v \in C} x_{uv}^k - 1 \right)$$

subject to

$$\sum_{(v : o \rightarrow v \in A)} x_{uv}^k = 1, \quad \text{for all } k \in K$$

$$\sum_{(w : u \rightarrow v \in A)} x_{uv}^k = 1, \quad \text{for all } k \in K$$

$$\sum_{(u : v \rightarrow v \in A)} x_{uv}^k = \sum_{(u : v \rightarrow v \in A)} x_{vu}^k \quad \text{for all } k \in K, v \in V \setminus \{o, d\}$$

$$x_{uv}^k \in \{0, 1\}, \quad \text{for all } k \in K, u \rightarrow v \in A$$

After removing the constant $\sum_{k \in K} \lambda_k - \sum_{C \in C} \mu_C$ from the objective of problem $\tilde{P}(\lambda, \mu)$, the reduced problem can be decomposed into $|K|$ independent subproblems. For each $k \in K$, the subproblem is

$$\tilde{P}_k(\lambda, \mu) : \text{Minimize } \sum_{u \rightarrow v \in A} \xi_{uv}^k x_{uv}^k + \sum_{(u \rightarrow v \in B \cup \lambda, \mu)} \lambda_k x_{uv}^k + \sum_{C \in C} \mu_C \sum_{u \rightarrow v \in C} x_{uv}^k$$

subject to

$$\sum_{(v : o \rightarrow v \in A)} x_{uv}^k = 1,$$

$$\sum_{(w : u \rightarrow v \in A)} x_{uv}^k = 1,$$

$$\sum_{(u : v \rightarrow v \in A)} x_{uv}^k = \sum_{(u : v \rightarrow v \in A)} x_{vu}^k \quad \text{for all } v \in V \setminus \{o, d\}$$

$$x_{uv}^k \in \{0, 1\}, \quad \text{for all } u \rightarrow v \in A$$

Each subproblem $\tilde{P}_k(\lambda, \mu)$ is a standard shortest path problem with arc length $\delta_{uv}^k = \xi_{uv}^k + \sum_{(u \rightarrow v : u \rightarrow v \in B \cup \lambda, \mu)} \lambda_k + \sum_{(C \in C : u \rightarrow v \in C \mu_C) \mu_C}$. This shortest path problem in the acyclic network $G$ can be solved efficiently with a standard dynamic programming algorithm. Furthermore, these independent subproblems $\tilde{P}_k(\lambda, \mu)$ can be solved in parallel when solved with a multicore computer processor. Given any nonnegative vectors $\lambda$ and $\mu$, we can obtain a lower bound on the optimal objective value of problem $P$ by solving problem $\tilde{P}(\lambda, \mu)$. Therefore, we
can find a tight lower bound by solving the following problem
\[
\max_{\lambda,\mu} \tilde{P}(\lambda, \mu),
\]
which is referred to as the Lagrangian multiplier problem associated with the original optimization problem \( P \), see, e.g., Ahuja et al. (1993). In this work, we use a modified subgradient optimization technique to search for the near-optimal vector \( \lambda \) and vector \( \mu \); see Section 4.3 for more details.

### 4.2 Upper bound heuristic

In this section, we present a constructive heuristic for obtaining a feasible solution of the considered problem. In what follows, we first present the framework of our upper bound heuristic in Section 4.2.1 and then present three train order ranking heuristics in Section 4.2.2.

#### 4.2.1 Framework of the upper bound heuristic

Given a ranked train set \( \tilde{K} = (\tilde{k}_1, \tilde{k}_2, \ldots, \tilde{k}_{|K|}) \), our upper bound heuristic derives a feasible solution of problem \( P \) as follows. We schedule these ranked trains \( k := \tilde{k}_1, \tilde{k}_2, \ldots, \tilde{k}_{|K|} \) one by one according to their ranks via the following six steps:

(i) set the temporary cost \( \tilde{\xi}_{uvw}^k \) of train \( k \) traversing arc \( u \to v \in A \) according to the lower bound solution and the schedules of trains with higher ranks; readers are referred to Caprara et al. (2002) and Xu et al. (2018) for similar treatments;

(ii) fix the maintenance process as “first cleaning then repair” by setting \( \tilde{\xi}_k^{(\hat{\rho}_W(w),t), (\hat{\rho}_W(w),t')} = +\infty \) and \( \tilde{\xi}_k^{(\hat{\rho}_W(w),t), (\hat{\rho}_W(w),t')} = +\infty \) for each cleaning arc \( (\hat{\rho}_W(w),t) \to (\hat{\rho}_W(w),t') \) and each drawing arc \( (\hat{\rho}_W(w),t) \to (\hat{\rho}_W(w),t') \) in the inward layer network;

(iii) use a standard dynamic programming algorithm to find a shortest path for train \( k \), denoted by \( P_k^{(1)} \), from vertex \( o \) to vertex \( d \) in network \( G \), where the schedules of the trains with higher ranks than \( k \) are kept unchanged;

(iv) reset \( \tilde{\xi}_{uvw}^k \) and \( \tilde{\xi}_{uvw}^k \) to their original values presented in Section 3.1.3, and set \( \tilde{\xi}_k^{(\hat{\rho}_W(w),t), (\hat{\rho}_W(w),t')} = +\infty \) and \( \tilde{\xi}_k^{(\hat{\rho}_W(w),t), (\hat{\rho}_W(w),t')} = +\infty \) for each cleaning arc \( (\hat{\rho}_W(w),t) \to (\hat{\rho}_W(w),t') \) and each drawing arc \( (\hat{\rho}_W(w),t) \to (\hat{\rho}_W(w),t') \) in the outward layer network to fix the maintenance process as “first repair then cleaning;”

(v) use a standard dynamic programming algorithm to find a shortest path for train \( k \), denoted by \( P_k^{(2)} \), from vertex \( o \) to vertex \( d \) in network \( G \), where the schedules of the trains with higher ranks than \( k \) are kept unchanged;

(vi) assign the shorter path in \( \{P_k^{(1)}, P_k^{(2)}\} \) to train \( k \).

Note that in steps (iii) and (v), we need to ensure that the obtained path does not violate any incompatibility constraint of problem \( P \). Because we use dynamic programming twice for each train \( k \), we then call the schedule
generating method *two-phase dynamic programming*. See Algorithm 1 for a summary of this basic upper bound heuristic.

**Algorithm 1** Upper Bound Heuristic

1: **Input:** a ranked train set with the order of \((\tilde{k}_1, \tilde{k}_2, \ldots, \tilde{k}_{|K|})\)
2: for \(k := \tilde{k}_1, \tilde{k}_2, \ldots, \tilde{k}_{|K|}\) do
3: calculate the *two-phase dynamic programming* to construct a schedule for \(k\), where the schedules of the trains with higher ranks than \(k\) are kept unchanged
4: end for
5: **Output:** a feasible train schedule

### 4.2.2 Train order ranking methods

Because different priority sequences of the trains may result in train schedules with different qualities obtained by the above upper bound heuristic, we now present the following three methods to rank the priorities of trains.

**Greedy heuristic:** The greedy heuristic, GH for short, determines the priority sequence of trains following the first come first serve rule; that is, a train with an earlier arrival time at the depot is given higher priority in scheduling.

**Relaxed solution-based heuristic:** After obtaining the lower bound solution, the relaxed solution-based heuristic, RSH for short, ranks trains \(k \in K\) in nondecreasing order of the optimal objective values of \(\bar{P}_k(\lambda, \mu)\).

**Relatedness-based ranking heuristic:** We now introduce a two-phase relatedness-based ranking heuristic, RRH for short, to rank the priority sequence of trains. Specifically, mimicking the way of Ropke and Pisinger (2006) and Shaw (1997), we define the relatedness measure \(\Lambda(k, k')\) for each pair of trains \(k\) and \(k'\). The lower \(\Lambda(k, k')\) is, the more related are these two trains, which means that there is greater interaction between these two trains. The relatedness measure defined in this paper consists of three terms: a path-length term, a time term and a type term. Given term weights \(\alpha\), \(\beta\) and \(\gamma\), the relatedness for trains \(i\) and \(j\) is defined as

\[
\Lambda(k, k') = \alpha(|l_k - l_{k'}|) + \beta(|t^d_k - t^d_{k'}| + |t^d_k - t^d_{k'}|) + \gamma(1 - \omega_{kk'})
\]

where \(l_k\) denotes the length of train \(k\)'s path found in the lower bound solution. \(\omega_{kk'} = 1\) if the trains \(k\) and \(k'\) are the same type; otherwise, \(\omega_{kk'} = 0\). The proposed RRH randomly selects \(\lfloor \varphi \cdot |K|/2 \rfloor\) pairs of related trains, collocated in set \(K\), with given parameters \(\varphi \in (0, 1)\) and \(\sigma \geq 1\) in the first phase. We select a pair of related trains via the following three steps:

(i) randomly select a train \(k\) from train set \(K'\) \((K'\ equals K\ at\ the\ beginning)\) and delete this train from train set \(K'\);

(ii) rank trains \(k' \in K'\) in nondecreasing order of the values of relatedness \(\Lambda(k, k')\);

(iii) randomly generate a number \(y \in [0, 1]\), we select train pair of \(k\) and the \([y^\sigma|K'|]-th\ train in the ranked train set \(K'\) and insert this train pair, denoted by \((k, K'[\lfloor y^\sigma|K'| \rfloor])\), into train pair set \(\mathcal{K}\).

In the second phase of the proposed RRH, we construct a new train order by swapping the positions of two related trains in a selected train pair in \(\mathcal{K}\). Algorithm 2 summarises this heuristic.
Algorithm 2 Relatedness-based Ranking Heuristic (RRH)

1. **Input:** a number \( \varphi \in (0, 1) \), a determinism parameter \( \sigma \geq 1 \) and train order \( \hat{K} = (\hat{k}_1, \hat{k}_2, \ldots, \hat{k}_{|\hat{K}|}) \), an initialized train pair set \( \mathcal{K} = \emptyset \), and a train set \( K' = K \)
2. **Phase 1:** Construct train pair set
3. **while** \(|\mathcal{K}| \leq \lfloor \varphi \cdot |K|/2 \rfloor\) **do**
   4. **step 1:** randomly select a train \( k \) from \( K' \), set \( K' \leftarrow K' \setminus \{k\} \)
   5. **step 2:** rank \( K' \) such that \( k' < k'' \Rightarrow \Lambda(k, K'[k']) < \Lambda(k, K'[k'']) \)
   6. **step 3:** choose a random number \( y \) from \([0, 1]\),
      update \( \mathcal{K} \leftarrow \mathcal{K} \cup \{(k, K'[[y^m][-\ell]])\}, K' \leftarrow K' \setminus \{K'[[y^m][-\ell]]\} \)
7. **end while
8. **Phase 2:** Construct new train order
9. **while** \(|\mathcal{K}| \neq \emptyset\) **do**
   10. **step 1:** select one train pair \((k, k')\) from \( \mathcal{K} \)
   11. **step 2:** update \( \hat{K} \) by swapping positions of trains \( k \) and \( k' \) in \( \hat{K} \)
   12. **step 3:** update \( \mathcal{K} \leftarrow \mathcal{K} \setminus \{(k, k')\} \)
13. **end while
14. **Output:** re-ranked priority sequence of trains

### 4.3 Subgradient optimization procedure

In this section, we present a subgradient optimization procedure to search for near-optimal \( \lambda_k \) and \( \mu_C \) values.

Note that the solution of problem \( \tilde{P}(\lambda, \mu) \) is a relaxed solution of problem \( P \) that may violate constraints (5) and (6) of problem \( P \). In each iteration of the subgradient optimization procedure, the value “\( \sum_{u \rightarrow v \in B_w} x_{uv}^{k} - 1 \)” for each \( k \in K \) and the value “\( \sum_{k \in K} \sum_{u \rightarrow v \in C} x_{uv}^{k} - 1 \)” for each \( C \in C \) form a subgradient vector \( \eta = \{\eta_1, \ldots, \eta_{|K|}, \eta_{|K|+1}, \ldots, \eta_{|K|+|C|}\} \) of the relaxed solution. Let \( \eta^\ell (\lambda^\ell \text{ and } \mu^\ell) \) denote the \( \eta \) (\( \lambda \) and \( \mu \)) vector in the \( \ell \)-th iteration of the procedure, let \( \eta_m^\ell \) be the \( m \)-th component of \( \eta^\ell \) for \( m = 1, 2, \ldots, |K| + |C| \), let \( \lambda_m^\ell \) be the \( m \)-th component of Lagrangian multiplier vector \( \lambda^\ell \) for each \( m = 1, 2, \ldots, |K| \), and let \( \mu_m^\ell \) be the \( m \)-th component of Lagrangian multiplier vector \( \mu^\ell \) for \( m = 1, 2, \ldots, |C| \). In the initial iteration (i.e., \( \ell = 0 \)), the components in \( \eta \) are all initialized as 0, and the Lagrangian multipliers are all set to 0. In the other iterations (i.e., \( \ell > 0 \)), we update each multiplier according to the following formulas (see, e.g., Held and Karp 1971)

\[
\lambda_m^\ell \leftarrow \lambda_{m}^{\ell-1} + \theta \cdot \frac{UB - LB(\lambda, \mu)}{\| \eta^\ell \|^2} \cdot \eta_{m}^{\ell-1} \quad (m = 1, 2, \ldots, |K|)
\]

and

\[
\mu_m^\ell \leftarrow \max \left\{ \mu_{m}^{\ell-1} + \theta \cdot \frac{UB - LB(\lambda, \mu)}{\| \eta^\ell \|^2} \cdot \eta_{m+|K|}^{\ell-1}, 0 \right\} \quad (m = 1, 2, \ldots, |C|),
\]

where \( \theta > 0 \) is a prespecified step size parameter, \( UB \) is the best feasible solution of problem \( P \) found thus far, and \( LB(\lambda, \mu) \) is the optimal objective value of \( \tilde{P}(\lambda, \mu) \) corresponding to the current multipliers \( \lambda \) and \( \mu \).

To improve the convergence of the procedure and avoid the “zig-zag” behavior of the Lagrangian multipliers’ values, we further apply the modified subgradient technique proposed by Camerini et al. (1975). We use a modified subgradient vector \( \tilde{\eta} \) instead of \( \eta \) to update the Lagrangian multipliers \( \lambda_m \) and \( \mu_m \). In the \( \ell \)-th (\( \ell > 0 \)) iteration, the modified subgradient vector \( \tilde{\eta}^\ell \) is updated by

\[
\tilde{\eta}^\ell \leftarrow \eta^\ell + b\tilde{\eta}^{\ell-1},
\]

17
where \( b \) is a scalar defined as

\[
b = \begin{cases} 
-a \cdot \frac{\tilde{\eta}^{\ell-1} \cdot \eta^\ell}{\| \tilde{\eta}^{\ell-1} \|_2}, & \text{if } \tilde{\eta}^{\ell-1} \cdot \eta^\ell < 0; \\
0, & \text{otherwise};
\end{cases}
\]

and \( a \) is a prespecified value such that \( 0 \leq a \leq 2 \). The components in \( \tilde{\eta}^0 \) are all set to 0 in the initial iteration (i.e., \( \ell = 0 \)). Moreover, because the number of constraints (6) may be large in practice, we use a dynamic constraint-generation scheme to handle the relaxed constraints and to determine the corresponding multipliers. To implement this scheme, we initialize an empty constraint pool at first. In each iteration, the constraints violated by the relaxed solution are put into the constraint pool. The multipliers corresponding to these constraints in the pool are then updated with the method described above. After determining the multipliers, the constraints whose corresponding multipliers are updated to 0 are removed from the constraint pool.

In each iteration of the subgradient optimization procedure, we may implement the following five parts: (i) obtain a relaxed solution of problem \( P \) by solving \( |K| \) shortest path problems \( \tilde{P}_k(\lambda, \mu) \); (ii) obtain a feasible solution of problem \( P \) using the upper bound heuristic presented in Section 4.2; (iii) identify constraints that the current relaxed solution has violated; (iv) update the subgradient vector and modified subgradient vector; and (v) update the Lagrangian multipliers. Because part (ii) is time-consuming, we may skip this part in some iterations to save computation time. This subgradient optimization procedure is terminated when one of the following two situations occurs: (i) the number of iterations reaches a prespecified limit; and (ii) the computational time reaches a prespecified limit. See Section 5 for more details regarding parameter settings.

5 Computational Study

In this section, we conduct a computational study to illustrate the effectiveness of the network flow model and the performance of our Lagrangian heuristic. Our method was implemented in C# using a personal computer with a 3.70 GHz 10-core processor (Intel Core i9-10900 Processor) and 32 GB RAM. In what follows, we first present the generation of the test instances in Section 5.1. We then examine our solution methods in some instances at different scales and conduct a study to analyze the performance of the relatedness-based ranking heuristic with different parameter settings in Section 5.2.

5.1 Generation of test instances

In this section, we adopt a depot network at different scales. The first depot network, Network 1, is generated by capturing the structure characteristics of Hefei-Nan depot’s yard 2 in China’s high-speed railway, which has 60 nodes, 6 repair tracks, 1 cleaning track, 9 storage tracks, and 54 shunting tracks. The second depot network, Network 2, is a larger extension of Network 1 that has 163 nodes, 18 repair tracks, 3 cleaning tracks, 30 storage tracks, and 129 shunting tracks. Figures A1 and A2 in Appendix C, respectively, display Networks 1 and 2 in detail.
For Network 1, we set the number of trains $|K| = 4, 8, 12, 16, 20$. Because Network 2 has a larger maintenance capacity, we set $|K| = 8, 16, 24, 32, 40$. We then have 10 combinations of depot network and number of trains. We generate one instance for each combination. For test instances underlying both Network 1 and Network 2, we set the length of the planning horizon to $T = 720$ minutes (e.g., from 18:00 one day to 06:00 the next), and the minimum headway between two trains that traverse the same node $i$ is set to 2 minutes, i.e., $h_i = 2$. In each instance, we generate the train data as follows. For each $k \in K$, we randomly select each train’s type from $\{0, 1\}$, where type 0 (represents a short train) is selected with a probability of $3/5$, and type 1 (represents a long train) is selected with a probability of $2/5$. For each short train $k \in K_0$ (respectively long train $k \in K_1$), the required cleaning time $\alpha_k$ is set to 15 minutes (respectively 20 minutes), the required repair time $\beta_k$ is set to 120 minutes (respectively 150 minutes). To capture the characteristic that trains undergo maintenance each night, we generate the trains’ arrival and departure times at the depot as follows. For each train $k$, an arrival time $t_a^k$ is randomly generated from a discrete uniform distribution between 0 and 360 (e.g., from 18:00 to 24:00) and its departure time $t_d^k$ is randomly generated from a discrete uniform distribution between 600 and 720 (e.g., from 4:00 to 6:00). In addition, for each $k \in K$, we set $\tau_p^k = 2$ minutes if $p \in S \cup R \cup W$, and $\tau_p^k = 1$ minute if $p \in E$. Let $c = 1.0$ denote the unit operating cost for a short train $k \in K_0$ when it is waiting on a track. We then set $c_k = (2.0)c$, $c_k' = (1.0)c$, $c_k'' = (2.4)c$, and $\pi_k = (400)c$ for each train $k \in K_0$, and we set $c_k = (3.0)c$, $c_k' = (1.5)c$, $c_k'' = (3.6)c$, and $\pi_k = (750)c$ for each train $k \in K_1$.

### 5.2 Computational results

In this section, we conduct two sets of experiments to test the effectiveness of our solution methods. We first generate one test instance for each combination of depot network and problem size to test the effectiveness of our solution method. Thus, there are a total 10 instances. We use Lagr. RSH (Lagr. RRH) to denote the Lagrangian heuristic where RSH (RRH) is applied to solve the feasible solution of the problem $P$. In our implementation of the Lagrangian heuristics, parameter $a$ is set to 0.5 and parameter $\theta$ is set to 2.0, and it will be reduced by 20% if the best lower bound identified shows no improvement for 20 consecutive iterations. For the solution of each test instance obtained by these methods, we determine their optimality gaps defined as

$$\text{Gap} = \frac{UB - LB}{LB} \times 100\%,$$

where $UB$ is the objective value of the corresponding solution method, and $LB$ is the best lower bound value between the values obtained by Lagr. RSH and Lagr. RRH. The prespecified limit of computational time is set to 3 hours, and the prespecified limit of iterations is set to 2000 for each instance. For the first 300 iterations, we execute the upper bound heuristic and update the upper bound at each iteration. To save computation time, after 300 iterations, we execute the upper bound heuristic and update the upper bound with a probability of 20% at each iteration. In addition, in Lagr. RRH, we set $\alpha = 0.2$, $\beta = 0.3$, $\gamma = 0.6$, $\varphi = 0.8$, and $\sigma = 2.0$.

Table 2 summarizes the computational results of these test instances, with each row representing the result of the test instance. The “$|K_0|$” column (respectively “$|K_1|$”) reports the number of canceled short trains.
(respectively long trains), the “$\bar{\zeta}$” column reports the value of the average shift of the not cancelled trains in the test instance, the “Gap” column reports the value of the optimality gap, the “T(s)” column lists the value of time (in CPU second) to solve a test instance in each problem set. Table 2 shows that the optimality gaps obtained by all three heuristics tend to increase as $|K|$ increases and as the problem scale increases. Specifically, for these instances with less trains, the Lagrangian relaxation heuristic can output solutions with high quality, as there exist less incompatible constraints, see, e.g., instances S1, S2, S3, S4, L1, L2, and L3. We also observe that limited by the depot capacity, a large number of trains leads to an increase in canceled trains, which may in turn result in a larger optimality gap. This indicates either that the large-scale problem is not easy to solve or that the lower bound obtained by the Lagrangian relaxation heuristic is less tight for problems at a larger scale. Another natural result is that with an increase in problem scale, the required computational time for the Lagrangian heuristics increases significantly.

| inst | $|K_0|$ | $|K_1|$ | LB | $UB$ | $|K_0'|$ | $|K_1'|$ | $\bar{\zeta}$ | Gap | T(s) | $UB$ | $|K_0'|$ | $|K_1'|$ | $\bar{\zeta}$ | Gap | T(s) | $UB$ | $|K_0'|$ | $|K_1'|$ | $\bar{\zeta}$ | Gap | T(s) |
|------|--------|--------|-----|------|--------|--------|-----------|------|------|------|--------|--------|-----------|------|------|--------|--------|-----------|------|------|--------|--------|-----------|------|------|--------|--------|-----------|
| Network 1 |
| S1 | 2 | 2 | 1764.0 | 1787.0 | 0 | 0 | 0.0 | 1.3% | 0 | 1772.0 | 0 | 0 | 0.0 | 0.5% | 320 | 1772.0 | 0 | 0 | 0.0 | 0.5% | 320 |
| S2 | 4 | 4 | 3551.2 | 3625.0 | 0 | 0 | 0.0 | 2.1% | 0 | 3600.0 | 0 | 0 | 0.0 | 1.4% | 510 | 3593.0 | 0 | 0 | 0.0 | 1.2% | 520 |
| S3 | 10 | 2 | 4263.4 | 4440.2 | 0 | 0 | 0.0 | 4.2% | 1 | 4405.5 | 0 | 0 | 0.0 | 3.3% | 881 | 4393.0 | 0 | 0 | 0.0 | 3.0% | 885 |
| S4 | 8 | 8 | 7205.1 | 7883.0 | 2 | 2 | 0.0 | 9.4% | 1 | 7687.8 | 0 | 2 | 0.0 | 6.7% | 930 | 7688.0 | 2 | 2 | 0.0 | 6.7% | 947 |
| S5 | 15 | 5 | 7597.9 | 8552.0 | 2 | 3 | 0.0 | 12.6% | 1 | 8542.0 | 1 | 4 | 0.0 | 12.4% | 1561 | 8506.8 | 3 | 1 | 0.0 | 12.0% | 1486 |
| Avg. | 8 | 4 | 4876.3 | 5257.4 | 1 | 1 | 0.0 | 5.9% | 1 | 5201.5 | 0 | 1 | 0.0 | 4.9% | 840 | 5190.6 | 1 | 1 | 0.0 | 4.7% | 834 |
| Network 2 |
| L1 | 5 | 3 | 3237.3 | 3359.0 | 0 | 0 | 0.0 | 3.8% | 2 | 3339.0 | 0 | 0 | 0.0 | 3.1% | 1911 | 3333.0 | 0 | 0 | 0.0 | 3.0% | 1920 |
| L2 | 6 | 10 | 7454.5 | 8034.3 | 0 | 0 | 0.0 | 7.9% | 4 | 7937.6 | 0 | 0 | 0.0 | 6.6% | 3229 | 7928.3 | 0 | 0 | 0.0 | 6.5% | 3288 |
| L3 | 16 | 8 | 9604.0 | 10403.5 | 0 | 0 | 0.0 | 8.3% | 6 | 10263.5 | 0 | 0 | 0.0 | 6.9% | 4633 | 10230.5 | 0 | 0 | 0.0 | 6.5% | 4813 |
| L4 | 21 | 11 | 12909.4 | 14280.8 | 0 | 0 | 0.0 | 10.6% | 9 | 14134.5 | 0 | 0 | 0.5 | 9.5% | 6763 | 14121.0 | 0 | 0 | 0.4 | 9.4% | 6862 |
| L5 | 22 | 18 | 17330.9 | 19539.1 | 5 | 0 | 0.0 | 12.7% | 9 | 19290.0 | 2 | 1 | 0.0 | 11.3% | 7519 | 19255.3 | 3 | 1 | 0.1 | 11.1% | 7575 |
| Avg. | 14 | 10 | 10105.4 | 11123.3 | 1 | 0 | 0.0 | 8.7% | 6 | 10992.9 | 0 | 0 | 0.1 | 7.5% | 4815 | 10973.6 | 1 | 0 | 0.1 | 7.3% | 4892 |

The average optimal gaps obtained by Lagr. RSH and Lagr. RRH are less than 5.0% (respectively 8.0%) for instances on Network 1 (respectively Network 2), while the average optimal gaps obtained by GH is 5.9% (respectively 8.7%) for instances on Network 1 (respectively Network 2). This computational result indicates that the solutions obtained by both Lagr. RSH and Lagr. RRH are better than that obtained by GH. This result illustrates the effectiveness of the proposed network flow model and the developed Lagrangian relaxation heuristic. We also observe that the solutions obtained by Lagr. RRH are slightly better than those obtained by Lagr. RSH in all test instances, implying that the RRH may find a better train schedule order than RSH. We can also see that GH can provide high-quality solutions for problems at a small scale, see, e.g., the first two instance groups in Networks 1 and 2. This may benefit from the tailored two-layer time-space network for the considered problem because it may make the problem easier to solve. In addition, the Lagrangian relaxation heuristic requires a much longer computation time than GH, especially for the large-scale problems, which
shows that GH is highly efficient and indicates its utility in real-time applications. In addition, we see that although the value of the train shift increases as the number of trains increases, the value of the train shift is small. Typically, a train shift may be more acceptable than a train cancelation. In practice, we may sacrifice the punctuality of the train schedule to reduce the number of canceled trains.

Next, we conduct a computational study of sensitivity analysis to show how the computational results obtained by the relatedness-based ranking heuristic are affected by the values of parameter $\varphi$ and parameter $\sigma$. In this computational study, we choose two instances S5 and L5 from the 10 instances in the first experimental set. That is, the first instance with underlying Network 1 considers 15 short trains and 5 long trains. The second instance with underlying Network 2 considers 22 short trains and 8 long trains. For each instance, we set $\varphi = 0.2, 0.4, 0.6, 0.8$ and $\sigma = 1, 2, 3, 4$. There are thus 16 combinations of $\varphi$ and $\sigma$, and 32 computational instances. The other parameter settings are the same as in the first experimental set.

Table 3: Computational results with different $\varphi$ and $\sigma$ (using instance S5 data).

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\sigma$ = 1</th>
<th>$\sigma$ = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LB1$</td>
<td>$UB$</td>
<td>$</td>
</tr>
<tr>
<td>0.2</td>
<td>7583.9</td>
<td>8553.8</td>
</tr>
<tr>
<td>0.4</td>
<td>7625.2</td>
<td>8570.4</td>
</tr>
<tr>
<td>0.6</td>
<td>7591.0</td>
<td>8494.8</td>
</tr>
<tr>
<td>0.8</td>
<td>7598.3</td>
<td>8436.6</td>
</tr>
</tbody>
</table>

Table 4: Computational results with different $\varphi$ and $\sigma$ (using instance L5 data).

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\sigma$ = 1</th>
<th>$\sigma$ = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LB1$</td>
<td>$UB$</td>
<td>$</td>
</tr>
<tr>
<td>0.2</td>
<td>7599.1</td>
<td>8507.0</td>
</tr>
<tr>
<td>0.4</td>
<td>7604.7</td>
<td>8570.2</td>
</tr>
<tr>
<td>0.6</td>
<td>7705.3</td>
<td>8437.0</td>
</tr>
<tr>
<td>0.8</td>
<td>7614.6</td>
<td>8507.8</td>
</tr>
</tbody>
</table>

Tables 3 and 4 summarize the computational results of the sensitivity analysis study, where optimality gaps
Gap1 and Gap2 are, respectively, defined as

\[
\text{Gap1} = \frac{UB - LB_1}{LB_1} \times 100\%, \quad \text{Gap2} = \frac{UB - LB_2}{LB_2} \times 100%,
\]

where \(UB\) and \(LB_1\) are, respectively, the objective value of corresponding solution and the lower bound value in the instance with the same setting of \(\varphi\) and \(\sigma\), while \(LB_2\) (see the value with over line in both Table 3 and Table 4) is the best lower bound value in these instances with different settings of \(\varphi\) and \(\sigma\). We can see that in the results with underlying Network 1, the maximum optimality gap is 12.9%, while the minimum optimality gap is 9.5%. This indicates that the values of \(\varphi\) and \(\sigma\) can affect the performance of Lagr. RRH. We can also see that in the results with underlying Network 2, the maximum optimality gap is 12.0%, while the minimum optimality gap is 10.8%. That is, there exists a less difference among the optimality gaps in the results with underlying Network 2. Moreover, we can also see that the difference between Gap1 and Gap2 for these instances on Network 1 is larger than that on Network 2. A comparison of the results with the underlying Network 1 and the results with the underlying Network 2 indicates that the number of canceled trains in the former is larger than that in the latter, even though the former has less trains than the latter. This means the situation with 20 trains on Network 1 is denser than that with 40 trains on network 2. Combining the above observations, we can infer that Lagr. RRH’s performance may be highly influenced by the values of \(\varphi\) and \(\sigma\) in a dense situation. In addition, the results in Table 3 allows us to roughly conclude that the optimality gaps obtained by Lagr. RRH tend to decrease as the value of \(\varphi\) increases.

6 Conclusions

This study examined the model and solution method for the passenger train shunting and routing problem in a depot. We presented a minimum-cost multi-commodity network flow model for the considered problem on the basis of a tailored two-layer time-space network. We then developed Lagrangian relaxation heuristics and a benchmark solution method to solve the problem. The computational results demonstrate the effectiveness of the proposed model and the solution methods.

The construction of a two-layer time-space network was an important part of this study. For example, we carefully designed several vertices for each repair track and each storage track to deal with a practical operation requirement that allows only one long train or two short trains to park on a repair or storage track at each time instant. Our network is flexible and can be easily modified to accommodate other practical requirements. For example, if wheel repair is required, we can design vertices and arcs corresponding to a wheel repair track following the method to design vertices and arcs corresponding to cleaning tracks, because wheel repair equipment is also installed along tracks rather than in a running shed. In addition, in this study, we considered only one kind of depot layout in China’s high-speed railways. An interesting extension of our work would be to apply our solution method to other depots with different layouts by modifying our time-space network according to the considered layout.
It is worth mentioning that some uncertainty factors exist; for example, the train’s arrival time at the depot may be delayed by malfunctioning infrastructure or rolling stock, and these disturbances may make the schedule less efficient or even infeasible. It would thus be worthwhile to study the optimization model and solution method for scheduling trains in the depot to allow an optimal response to such disturbances.

**Acknowledgement**

This study is supported by the National Natural Science Foundation of China (No. 72071059), China Postdoctoral Science Foundation (No. 2019M662144). We thank Mr. Wei Zhang from Hefei Railway Station for his help on data collection and problem definition.

**References**


Appendix A. Description of Arcs

- Starting arcs: For each node $i \in N$ and time instant $t = 0, 1, \ldots, T$, there is a starting arc $o \rightarrow (\rho(i), t)$ if this node is a joint node between the station and depot (i.e., nodes 2 and 3 in Figure 1). For each train $k \in K$, if $t \geq t_k^a$, then $\xi^k_{((i), t)} = 0$; otherwise $\xi^k_{((i), t)} = +\infty$.

- Ending arcs: For each node $i \in N$ and time instant $t = 0, 1, \ldots, T$, there is an ending arc $(\rho(i), t) \rightarrow d$ if this node is a joint node between the station and depot. As mentioned in Section 2.1.2, a shift cost is imposed if a train’s actual departure time is later than planned. Considering this, we impose the shift cost $\Gamma_k(t) = \max \left\{ 0, c_k^\prime(t - t_k^a) \right\}$ on an ending arc. For each train $k \in K$, $\xi^{k}_{((i), t), d} = \Gamma_k(t)$ if $t_k^a \leq t \leq T$; otherwise, $\xi^{k}_{((i), t), d} = +\infty$.

- Drawing arcs: For each storage track $s \in S$ and time instants $t, t' = 0, 1, \ldots, T$, there are three drawing arcs, including $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$, $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$, and $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$, in the inward layer network and three drawing arcs, including $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$, $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$, and $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$, in the outward layer network if there exists $k \in K_0$ such that $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$. For each train $k \in K$ and each drawing arc $a \rightarrow v$ mentioned above, if $k \in K_0$, $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$, then $\xi^k_{(h, a), d} = c_k(t' - t)$ where $a$ and $t'$ are the time instants corresponding to vertices $u$ and $v$. There is one drawing arc $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$ in the inward layer network and one drawing arc $(\tilde{\rho}_S(s), t) \rightarrow (\tilde{\rho}_S(s), t')$ in the outward layer network if there exists $k \in K_1$ such that $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$. For each train $k \in K$, if $k \in K_1$, $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$, then $\xi^k_{(h, a), d} = c_k(t' - t)$; otherwise, $\xi^k_{(h, a), d} = +\infty$. For each repair track $r \in R$ and time instants $t, t' = 0, 1, \ldots, T$, there are three drawing arcs $(\tilde{\rho}_R(r), t) \rightarrow (\tilde{\rho}_R(r), t')$, $(\tilde{\rho}_R(r), t) \rightarrow (\tilde{\rho}_R(r), t')$, and $(\tilde{\rho}_R(r), t) \rightarrow (\tilde{\rho}_R(r), t')$ if there exists $k \in K_0$ such that $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$. For each train $k \in K$, if $k \in K_0$, $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$, then $\xi^k_{(h, a), d} = c_k(t' - t)$; otherwise, $\xi^k_{(h, a), d} = +\infty$. There is one drawing arc $(\tilde{\rho}_R(r), t) \rightarrow (\tilde{\rho}_R(r), t')$ if there exists $k \in K_1$ such that $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$. For each train $k \in K$, if $k \in K_1$, $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$, then $\xi^k_{(h, a), d} = c_k(t' - t)$; otherwise, $\xi^k_{(h, a), d} = +\infty$. For each cleaning track $w \in W$ and time instants $t, t' = 0, 1, \ldots, T$, there are two drawing arcs $(\tilde{\rho}_W(w), t) \rightarrow (\tilde{\rho}_W(w), t')$ and $(\tilde{\rho}_W(w), t) \rightarrow (\tilde{\rho}_W(w), t')$ if there exists $k \in K$ such that $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$. For each $k \in K$, if $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$, then $\xi^k_{(h, a), d} = c_k(t' - t)$; otherwise, $\xi^k_{(h, a), d} = +\infty$. For each shunting track $e = (i, j) \in E$ and time instants $t, t' = 0, 1, \ldots, T$, there are two drawing arcs $(\rho(i), t) \rightarrow (\rho(j), t')$ and $(\rho(j), t) \rightarrow (\rho(i), t')$ if there exists $k \in K$ such that $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$. For each $k \in K$, if $t \geq t_k^a$ and $t' = t + \tau_k^a \leq T$, then $\xi^k_{(h, a), d} = c_k(t' - t)$; otherwise, $\xi^k_{(h, a), d} = +\infty$. A train traversing a drawing arc implies that this train is traversing the track that corresponds to the drawing.
Cleaning arcs: For each $k \in K$, $w \in W$, and $t, t' = 0, 1, \ldots, T$, there are two cleaning arcs $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t')$ and $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t')$ if $t \geq t_k^a$ and $t' = t + \alpha_k \leq T$. For each train $k \in K$, if $t \geq t_k^a$ and $t' = t + \alpha_k \leq T$, then $\xi^k_{(\hat{\rho}_W(w), t), (\hat{\rho}_W(w), t')} = \xi^k_{(\hat{\rho}_W(w), t), (\hat{\rho}_W(w), t')} = c_k(t' - t)$; otherwise, $\xi^k_{(\hat{\rho}_W(w), t), (\hat{\rho}_W(w), t')} = \xi^k_{(\hat{\rho}_W(w), t), (\hat{\rho}_W(w), t')} = +\infty$. The arc $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t')$ allows train $k$ to be cleaned with a minimum required cleaning time $\alpha_k$ when it dwells on cleaning track $w$ and travels in the inward direction. The clean arc $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t')$ allows train $k$ to be cleaned with a minimum required cleaning time $\alpha_k$ when it dwells on cleaning track $w$ and travels in the outward direction.

Repairing arcs: For each $k \in K_0$, $r \in R$ and $t, t' = 0, 1, \ldots, T$, there are two repairing arcs $(\rho_R(r), t) \rightarrow (\rho_R(r), t')$ and $(\rho'_R(r), t) \rightarrow (\rho'_R(r), t')$ if $t \geq t_k^a$ and $t' = t + \beta_k \leq T$. For each train $k \in K$, if $k \in K_0$, $t \geq t_k^a$ and $t' = t + \beta_k \leq T$, then $\xi^k_{(\rho_R(r), t), (\rho'_R(r), t')} = \xi^k_{(\rho'_R(r), t), (\rho_R(r), t')} = c_k(t' - t)$; otherwise, $\xi^k_{(\rho'_R(r), t), (\rho_R(r), t')} = \xi^k_{(\rho_R(r), t), (\rho'_R(r), t')} = +\infty$. The arc $(\rho_R(r), t) \rightarrow (\rho'_R(r), t')$ (arc $(\rho'_R(r), t) \rightarrow (\rho_R(r), t')$) allows short train $k$ to be repaired with the minimum required repairing time $\beta_k$ when it is dwelling at repair track $r$’s position I (position II). For each $k \in K_1$, $r \in R$ and $t, t' = 0, 1, \ldots, T$, there is also a repairing arc $(\rho''_R(r), t) \rightarrow (\rho''_R(r), t')$ if $t \geq t_k^a$ and $t' = t + \beta_k \leq T$. For each train $k \in K$, if $k \in K_1$, $t \geq t_k^a$ and $t' = t + \beta_k \leq T$, then $\xi^k_{(\rho''_R(r), t), (\rho'_R(r), t')} = c_k(t' - t)$; otherwise, $\xi^k_{(\rho''_R(r), t), (\rho'_R(r), t')} = +\infty$. This arc allows long train $k$ to be repaired with the minimum required time $\beta_k$ when it travels along the entire repair track $r$.

Waiting arcs: For each storage track $s \in S$ and time instant $t = 0, 1, \ldots, T - 1$, there are four waiting arcs, including $(\hat{\rho}_S(s), t) \rightarrow (\hat{\rho}_S(s), t + 1)$, $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$, $(\hat{\rho}_S(s), t) \rightarrow (\hat{\rho}_S(s), t + 1)$, and $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$ if there exists $k \in K_0$ such that $t \geq t_k^a$, and there are two waiting arcs $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$ and $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$ if there exists $k \in K_1$ such that $t \geq t_k^a$. For each $k \in K$, if $k \in K_0$, $t \geq t_k^a$, $\xi^k_{(\rho''_S(s), t), (\rho''_S(s), t + 1)} = \xi^k_{(\hat{\rho}_S(s), t), (\hat{\rho}_S(s), t + 1)} = \xi^k_{(\rho''_S(s), t), (\hat{\rho}_S(s), t + 1)} = 0$; otherwise, $\xi^k_{(\rho''_S(s), t), (\rho''_S(s), t + 1)} = \xi^k_{(\hat{\rho}_S(s), t), (\rho''_S(s), t + 1)} = \xi^k_{(\rho''_S(s), t), (\hat{\rho}_S(s), t + 1)} = +\infty$. The arc $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$ (respectively arc $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$) allows a short train to wait at storage track $s$’s position I (position II) when traveling in the inward (outward) direction. The arc $(\hat{\rho}_S(s), t) \rightarrow (\hat{\rho}_S(s), t + 1)$ (arc $(\rho''_S(s), t) \rightarrow (\rho''_S(s), t + 1)$) allows a short train to wait at storage track $s$’s position I (position II) when traveling in the outward (inward) direction.

For each cleaning track $w \in W$ and time instant $t = 0, 1, \ldots, T - 1$, there are two waiting arcs $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t + 1)$ and $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t + 1)$ if there exists train $k \in K$ such that $t \geq t_k^a$. For each $k \in K$, if $t \geq t_k^a$, then $\xi^k_{(\hat{\rho}_W(w), t), (\hat{\rho}_W(w), t + 1)} = \xi^k_{(\rho_w(w), t), (\rho_w(w), t + 1)} = \beta_k$; otherwise, $\xi^k_{(\hat{\rho}_W(w), t), (\hat{\rho}_W(w), t + 1)} = \xi^k_{(\rho_w(w), t), (\hat{\rho}_W(w), t + 1)} = +\infty$. The arc $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t + 1)$ and arc $(\hat{\rho}_W(w), t) \rightarrow (\hat{\rho}_W(w), t + 1)$ allow a train to wait at the cleaning track $w$ when traveling in the inward and outward directions, respectively.
For each repair track \( r \in R \) and time instant \( t = 0, 1, \ldots, T - 1 \), there are two waiting arcs \((\bar{s}_R(r), t) \rightarrow (\bar{s}_R(r), t+1)\) and \((\bar{s}'_R(r), t) \rightarrow (\bar{s}'_R(r), t+1)\) if there exists \( k \in K_0 \) such that \( t = t^k_0 \), and there is one waiting arc \((\bar{s}''_R(r), t) \rightarrow (\bar{s}''_R(r), t+1)\) if there exists \( k \in K_1 \) such that \( t = t^k_0 \). For each \( k \in K \), if \( k \in K_0 \) and \( t \geq t^k_0 \), then \( \xi^k_{(\bar{s}_R(r), t), (\bar{s}_R(r), t+1)} = \xi^k_{(\bar{s}'_R(r), t), (\bar{s}'_R(r), t+1)} = \xi^k_{(\bar{s}''_R(r), t), (\bar{s}''_R(r), t+1)} = \xi^k_{(\bar{s}_R(r), t), (\bar{s}_R(r), t+1)} = +\infty \). The arc \((\bar{s}_R(r), t) \rightarrow (\bar{s}_R(r), t+1)\) and arc \((\bar{s}'_R(r), t) \rightarrow (\bar{s}'_R(r), t+1)\) allow a short train to park at repair track \( r \)'s position I and II, respectively, when traveling in the outward direction. For each \( k \in K \), if \( k \in K_1 \), \( t \geq t^k_0 \), then \( \xi^k_{(\bar{s}_R(r), t), (\bar{s}''_R(r), t+1)} = \xi^k_{(\bar{s}'_R(r), t), (\bar{s}''_R(r), t+1)} = +\infty \). The arc \((\bar{s}''_R(r), t) \rightarrow (\bar{s}''_R(r), t+1)\) allows a long train to park on the entire repair track \( r \) when traveling in the outward direction.

• Departure arcs: For each storage track \( s \in S \) and time instant \( t = 0, 1, \ldots, T \), there are two departure arcs \((\bar{s}'_S(s), t) \rightarrow (\bar{s}'_S(s), t)\) and \((\bar{s}''_S(s), t) \rightarrow (\bar{s}''_S(s), t)\) if there exists \( k \in K_0 \) such that \( t = t^k_0 \), and there are two departure arcs \((\bar{s}'_S(s), t) \rightarrow (\bar{s}'_S(s), t)\) and \((\bar{s}''_S(s), t) \rightarrow (\bar{s}''_S(s), t)\) if there exists \( k \in K_1 \) such that \( t = t^k_0 \). For each \( k \in K_0 \), if \( t \geq t^k_0 \), then \( \xi^k_{(\bar{s}'_S(s), t), (\bar{s}'_S(s), t)} = \xi^k_{(\bar{s}''_S(s), t), (\bar{s}''_S(s), t)} = 0 \); otherwise, \( \xi^k_{(\bar{s}'_S(s), t), (\bar{s}'_S(s), t)} = \xi^k_{(\bar{s}''_S(s), t), (\bar{s}''_S(s), t)} = +\infty \). Arcs \((\bar{s}'_S(s), t) \rightarrow (\bar{s}'_S(s), t)\) and \((\bar{s}''_S(s), t) \rightarrow (\bar{s}''_S(s), t)\) represent the situation in which a short train is about to leave the storage track \( s \) when traveling in the inward and outward directions, respectively. For each \( k \in K_1 \), if \( t \geq t^k_0 \), then \( \xi^k_{(\bar{s}'_S(s), t), (\bar{s}'_S(s), t)} = \xi^k_{(\bar{s}''_S(s), t), (\bar{s}''_S(s), t)} = 0 \); otherwise, \( \xi^k_{(\bar{s}'_S(s), t), (\bar{s}'_S(s), t)} = \xi^k_{(\bar{s}''_S(s), t), (\bar{s}''_S(s), t)} = +\infty \). Arcs \((\bar{s}'_S(s), t) \rightarrow (\bar{s}'_S(s), t)\) and \((\bar{s}''_S(s), t) \rightarrow (\bar{s}''_S(s), t)\) represent the situation in which a long train is about to leave the storage track \( s \) when traveling in the inward and outward directions, respectively.

For each repair track \( r \in R \) and time instant \( t = 0, 1, \ldots, T \), there is a departure arc \((\bar{s}_R(r), t) \rightarrow (\bar{s}_R(r), t)\) if there exists \( k \in K_0 \) such that \( t = t^k_0 \), and there is a departure arc \((\bar{s}'_R(r), t) \rightarrow (\bar{s}'_R(r), t)\) if there exists \( k \in K_1 \) such that \( t = t^k_0 \). For each \( k \in K_0 \), if \( t \geq t^k_0 \), then \( \xi^k_{(\bar{s}_R(r), t), (\bar{s}_R(r), t)} = 0 \); otherwise, \( \xi^k_{(\bar{s}_R(r), t), (\bar{s}_R(r), t)} = +\infty \). The arc \((\bar{s}_R(r), t) \rightarrow (\bar{s}_R(r), t)\) represents the situation in which a short train has been repaired on repair track \( r \) and is about to leave this repair track. The arc \((\bar{s}'_R(r), t) \rightarrow (\bar{s}'_R(r), t)\) represents the situation in which a long track has been repaired at repair track \( r \) and is about to leave this repair track.

• Transfer arcs: For each storage track \( s \in S \) and time instant \( t = 0, 1, \ldots, T \), there are four transfer arcs \((\rho(i), t) \rightarrow (\rho_1(s), t), (\rho_2(s), t) \rightarrow (\rho_2(s), t), (\rho(i'), t) \rightarrow (\rho_1(s), t), (\rho(i'), t) \rightarrow (\rho_2(s), t), (\rho_3(s), t) \rightarrow (\rho(i), t)\) if there exists \( k \in K \) such that \( t \geq t^k_0 \) and nodes \( i \) and \( i' \) are the endpoints of storage track \( s \).

For each repair track \( r \in R \) and time instant \( t = 0, 1, \ldots, T \), there are also two transfer arcs \((\rho(i), t) \rightarrow (\rho_1(r), t)\) and \((\rho(i), t) \rightarrow (\rho_2(r), t)\) if there exists \( k \in K \) such that \( t \geq t^k_0 \) and nodes \( i \) is the endpoint of repair track \( r \).

For each cleaning track \( w \in W \) and time instant \( t = 0, 1, \ldots, T \), there are six transfer arcs \((\rho(i), t) \rightarrow (\rho_1(w), t), (\rho_1(w), t) \rightarrow (\rho_2(w), t), (\rho_2(w), t) \rightarrow (\rho(i'), t), (\rho(i), t) \rightarrow (\rho_1(w), t), (\rho_2(w), t) \rightarrow (\rho_1(w), t), (\rho(i'), t) \rightarrow (\rho(i), t)\) if there exists \( k \in K \) such that \( t \geq t^k_0 \) and nodes \( i \) and \( i' \) are the endpoints of cleaning track \( w \). These transfer arcs allow a train to traverse the nodes between operation tracks and
slunting tracks. For each train $k \in K$, $\xi_{uv}^k = 0$ if arc $u \rightarrow v \in A$ is a transfer arc.

- **Switch arcs:** For each repair track $r \in R$, there are two switch arcs $(\bar{\rho}_R(r), t) \rightarrow (\bar{\varrho}_R(r), t)$ and $(\bar{\rho}_R'(r), t) \rightarrow (\bar{\varrho}_R'(r), t)$ if there exists $k \in K_0$ such that $t \geq t_a^k$, and there is one switch arc $(\bar{\rho}_R''(r), t) \rightarrow (\bar{\varrho}_R''(r), t)$ if there exists $k \in K_1$ such that $t \geq t_a^k$. These arcs represent the situation in which a train has been repaired on track $r$ at time $t$ and switches its travel from the inward direction to the outward direction. Note that these arcs change “layer” components from the inward layer to the outward layer. For each $k \in K_0$, if $t \geq t_a^k$, then $\xi_{(\bar{\rho}_R(r), t), (\bar{\varrho}_R(r), t)}^k = \xi_{(\bar{\rho}_R'(r), t), (\bar{\varrho}_R'(r), t)}^k = 0$; otherwise, $\xi_{(\bar{\rho}_R(r), t), (\bar{\varrho}_R'(r), t)}^k = \xi_{(\bar{\rho}_R'(r), t), (\bar{\varrho}_R'(r), t)}^k = +\infty$. For each $k \in K_1$, if $t \geq t_a^k$, then $\xi_{(\bar{\rho}_R''(r), t), (\bar{\varrho}_R''(r), t)}^k = 0$; otherwise, $\xi_{(\bar{\rho}_R''(r), t), (\bar{\varrho}_R''(r), t)}^k = +\infty$.

- **Dummy satisfied-demand arc:** There is a dummy satisfied-demand arc $o \rightarrow d$. One can also say that a train traversing this arc represents the situation in which this train’s daily maintenance requirement is rejected by the depot. A high penalty $\pi_k$ is incurred if train $k$ traverses arc $o \rightarrow d$; then we have $\xi_{od}^k = \pi_k$. 


Appendix B. Proof of Proposition 1

As mentioned in Section 3.2, a path from vertex $o$ to vertex $d$ in our two-layer time-space network corresponds to a schedule of a train in the feasible solution. Clearly, a train path in the two-layer time-space network takes either the form $o \rightarrow d$ or the form $o \rightarrow (\rho(i), t) \rightarrow \cdots \rightarrow (\rho(j), t') \rightarrow d$. For each train $k \in K$, let $\tilde{A}_k$ be the set of the arcs along train $k$’s path in a feasible solution of problem $P$. If train $k$’s path takes the form $o \rightarrow d$ (i.e., $\tilde{A}_k = \{o \rightarrow d\}$), we can claim that train $k$ is rejected by the depot (see Section 3.1.3). That is, train $k$ is canceled, which indicates that the repair operation requirement for train $k$ can be omitted. However, because only switch arcs can connect both the in-layer time-space network and the out-layer time-space network, if train $k$’s path takes the form $o \rightarrow (\rho(i), t) \rightarrow \cdots \rightarrow (\rho(j), t') \rightarrow d$ rather than the form $o \rightarrow d$, there must be a switch arc in set $\tilde{A}_k$; otherwise, train $k$’s path must be infeasible. Note that a switch arc’s preceding arcs include only repairing arcs. To consider the flow balance constraints, before train $k$ traverses a switch arc, it must traverse a repairing arc along its feasible path, which indicates that train $k$’s repair operation requirement is satisfied.

This completes the proof of the proposition.
Appendix C. Depot Networks

Figure A1: Network 1.

Figure A2: Network 2.