sustainability

MDPI

Article

A Time-Space Network-Based Optimization Method for Scheduling Depot Drivers

Fei Peng [1], Xian Fan [2], Puxin Wang [1] and Mingan Sheng [2,*]

[1] School of Automotive Engineering, Beijing Polytechnic, Beijing 100176, China
[2] School of Automotive and Transportation Engineering, Hefei University of Technology, Hefei 230009, China
* Correspondence: 2021171026@mail.hfut.edu.cn

Abstract: The driver scheduling problem at Chinese electric multiple-unit train depots becomes more and more difficult in practice and is studied in very little research. This paper focuses on defining, modeling, and solving the depot driver scheduling problem which can determine driver size and driver schedule simultaneously. To solve this problem, we first construct a time-space network based on which we formulate the problem as a minimum-cost multi-commodity network flow problem. We then develop a Lagrangian relaxation heuristic to solve this network flow problem, where the upper bound heuristic is a two-phase method consisting of a greedy heuristic and a local search method. We conduct a computational study to test the effectiveness of our Lagrangian relaxation heuristic. The computational results also report the significance of the ratio of driver size to task size in the depot.

Keywords: driver scheduling; electric multiple-unit (EMU) train depot; optimization; time-space network; Lagrangian relaxation; local search

Citation: Peng, F.; Fan, X.; Wang, P.; Sheng, M. A Time-Space Network-Based Optimization Method for Scheduling Depot Drivers. *Sustainability* **2022**, *14*, 14431. https://doi.org/10.3390/su142114431

Academic Editors: Xin Yang, Songpo Yang and Xiaoming Xu

Received: 1 August 2022
Accepted: 27 October 2022
Published: 3 November 2022

1. Introduction

In the Chinese high-speed railway system, electric multiple-unit (EMU) trains need the maintenance to be executed at a specific level that is determined according to its accumulated running time or mileage reaches which are at a predefined threshold. Typically, there are five maintenance levels, in which the first-level maintenance (called operational maintenance) for an EMU train takes place when this EMU's accumulated running time reaches 48 h or its accumulated running mileage reaches 4000 km [1]. Since operational maintenance has the highest execution frequency, it becomes the main work for EMU train depots. During the first-level maintenance in a depot, an EMU train may be first cleaned at a cleaning track, then shunted to a repairing track for inspection tasks to be performed, and lastly returned to a storing track to wait for departure. Practically, because of the limited capacity of operation tracks, the task order mentioned above may be changed, and an EMU train may be stored at a storing track between different tasks.

Given a determined train shunting plan of the first-level maintenance, depot drivers are required to drive EMU trains from a storing track to a cleaning track, from a cleaning track to a repair track and, so on, according to a depot driver schedule. The depot driver scheduling problem aims to build a set of duties to cover driving tasks (including shunting and cleaning tasks), where many practical requirements should be satisfied, e.g., each duty can be picked up at most once by at most one driver, and the time between two adjacent tasks in duty must be longer than the walk time for a driver. A well-planned depot driver schedule is desired for the railway operator because it can reduce labor costs and improve the maintenance capacity of a depot. Moreover, with the increased use of EMU trains and the expansion of EMU depots, the scale of EMU maintenance work is expanding quickly, and it is difficult to obtain optimal or even feasible depot driver schedules manually, especially during important holidays, such as the Chinese Spring Festival.

In this paper, we design a time-space network for the considered depot driver scheduling problem with first-level maintenance. We then present a network flow optimization

model that determines the depot driver schedule, the assignment of schedule to depot drivers, and the routes for the depot drivers. We develop a Lagrangian relaxation heuristic to solve the formulated model and conduct a series of experiments to test the efficiency and effectiveness of the proposed solution method.

2. Literature Review

In the literature, Heil et al. [2] present a comprehensive review of studies of railway crew scheduling problems in the past twenty years, they mention that few papers study the maintenance crews and depot shunting drivers scheduling. To the best of our knowledge, two works related to depot crew scheduling in the literature can be found. Wang et al. [3] study the depot shunting driver assignment problem with workload balance considerations, and Pour et al. [4] study the signaling maintenance crew scheduling problem in the Danish railway system.

The depot driver scheduling problem is one kind of crew scheduling problem (CSP) since depot shunting drivers can be regarded as crew. The CSP can be described as follows: "Given a set of tasks (trips, flights, etc.) with fixed starting and ending times and locations, and given a set of rules and criteria, find the minimum cost set of duties such that each task is included in a duty and all rules are satisfied". A lot of studies have been conducted on CSP in the past few years. Many researchers formulate the problem as a set partitioning problem or a set covering problem that can be solved by exact algorithms, see, e.g., Souza et al. and Gharaei et al. [5–9]. Some exact algorithms are usually developed based on a column generation technique, see, e.g., Jütte et al., Nishi et al., and Liu et al. [10–12]. As the problem's scale is usually too large to be solved using the exact algorithm, researchers turn to developing a meta-heuristic to handle larger-scale crew scheduling problems. For example, Guillermo and Josś [13] develop a hybrid approach that combines tabu search and integer programming to solve a crew scheduling problem in railways. Elizondo et al. [14] put forward an evolutionary approach to deal with a CSP in underground passenger transport. Khmeleva et al. [15] presented a fuzzy-logic controlled genetic algorithm designed for the solution of the crew-scheduling problem in the rail-freight industry.

The crew planning problem is usually decomposed into two subproblems, including the crew scheduling problem and the crew rostering problem. In this respect, the crew scheduling problem forms anonymous duties covering all the trips for a defined period of time, and the crew rostering problem combines the duties into sequences which are subsequently assigned to individual crew members, see, e.g., Caprara et al. [16]. Many exact algorithms and meta-heuristics are developed to solve scheduling problems, see, e.g., Amjadian and Gharaei, Askari et al., and Taleizadeh et al. [17–19]. For example, Chew et al. [20] designed an approach based on tabu search to solve the driver scheduling problem in Singapore rapid transit. Practically, since the crew rostering problem is closely related to the crew scheduling problem, in some papers, both the crew scheduling problem and the crew rostering problem are taken into consideration, such as Amaya and Uribe, Zhou et al., and Borndörfer et al. [21–23], who apply Lagrangian relaxation and Benders decomposition to solve the integrated problem of duty scheduling and rostering.

The characteristic of the depot driver scheduling problem is that the time interval between two consecutive tasks assigned to the same driver must be taken into consideration, while this time interval should be determined according to the practical working places rather than the types of these two consecutive tasks. A simple way to solve this problem is to give a sufficient long walking time between different tasks. However, this method may cause a lot of unnecessary waiting and walk for depot drivers which results in a waste of human resources. In the worst case, the depot driver schedule would be infeasible if a depot manager ignores the characteristics mentioned above. In this paper, the walking time between different tasks is particularly taken into consideration, and the drivers are allowed to return to the depot lounge several times if needed; thus, the decisions of waiting time and waiting places for each crew need to be determined. As we consider a long practical

planning horizon, we also consider the determination of crew rostering and the maximum working time requirements for depot drivers. Our problem bears some similarities with the problem of Wang et al. [3] as both of these two problems focus on assigning driving tasks to depot drivers with given train shunting schedules. In Wang et al. [3], the objective is to minimize the walking distance and the fairness of the depot driver schedule, while the different required working time for different tasks and the waste of labor resources caused by waiting are not taken into consideration, and they do not allow the depot drivers to return to driver lounge more than once.

The contributions of this paper are as follows. First, we define an important practical depot driver scheduling problem for Chinese high-speed railways, which is studied in very little research. Second, we present an integrated optimization model for the considered problem which can determine driver size and driver schedule simultaneously and develop a network flow model-based Lagrangian relaxation heuristic. Our method is flexible in that it can be easy to modify arc costs or types to model and solve the problem that contains more practical constraints. Third, our computational results report the significance of the ratio of driver size to task size in the depot, which can help railway planners to estimate the driver size that is needed for these depots with different layouts and scales.

The rest of this paper is organized as follows. Section 3 gives a detailed description of the considered depot driver scheduling problem. Section 4 presents a time-space network and formulates a min-cost multi-commodity network flow model for the considered problem. Section 5 describes the development of a Lagrangian relaxation-based solution method. Computational experiments are conducted to test the solution method and to assess the benefits obtained by our solution approach, and the results are reported in Section 6. Lastly, some conclusions are made in Section 7.

3. Problem Description

The studied problem, hereinafter referred to as the depot driver scheduling problem, aims to simultaneously assign driving tasks to drivers and determine the drivers' detailed work schedules and walking routes while minimizing the operational cost. Some important characteristics of our problem are as follows: (i) our problem considers both the labor legislations (e.g., maximum working time requirement) and the depot regulations (e.g., the specified time window for depot drivers to sign in); (ii) a maintenance service may be canceled due to the limited driver scale, and a penalty is incurred when there is a cancellation; (iii) the maintenance services are predetermined and cannot be changed during planning horizon; and (iv) the planning horizon of each working day is discretized, and the time units are integers for which each time unit is 1 min. To be more precise, we offer the following definitions.

- Task: A task represents an operation a driver may execute, such as signing in, signing out, and driving trains. Tasks can be divided into three categories: (i) driving tasks, which are indicated in the EMU train shunting schedule, such as, shunting an EMU from one track to another, and driving trains at cleaning tracks; (ii) shift tasks which include sign-in and sign-out operations; and (iii) walking and waiting tasks. Practically, the required time for driving tasks is fixed, while the timings for shift tasks is uncertain. Moreover, the numbers and the required time of walking and waiting tasks need to be determined when making solutions.
- Shunting schedule: A shunting schedule determines the arrival and departure time of each train at each repairing track (optional), cleaning track, and shunting track it may traverse in the depot, such that certain operational requirements, such as track capacity, minimum headway constraints, and required maintenance procedures, etc., are taken into consideration. The shunting schedule is determined in the shunting schedule planning phase, which is used as input data for the problem studied in this paper.
- Duty: A duty is a chain of tasks that can be assigned to at most one depot driver. A feasible duty must satisfy various labor laws and legislation.

In this paper, we study the depot driver scheduling problem of EMU trains' first-level maintenance in the depot, which aims at assigning a set of predetermined driving tasks to drivers such that certain labor laws and legislation (e.g., maximum working time constraint) and the depot regulations (e.g., the specific time window for depot drivers to sign in), should be satisfied. During the first-level maintenance, two maintenance processes, including the inspection process and the cleaning process must be performed. Other operations, such as shunting trains are auxiliary in the process. The order of these two maintenance processes is unfixed which should be determined in when making a train shunting schedule. Moreover, different processes should be executed on the corresponding tracks with specific equipment, that is, the inspection process should be executed on the inspection tracks that are equipped with fault inspection and correction equipment, while the cleaning process should be executed on the cleaning tracks that are equipped with train cleaning equipment (e.g., automatic train cleaner). It is obvious that without scheduling these driving tasks, our problem can be reduced to a set covering problem, which has been proven NP-hard.

Figure 1 shows a typical layout of EMU train depot in China, which contains a driver lounge, a stabling yard, a cleaning area, a repairing shed, and several shunting tracks connecting different operation tracks. The stabling yard can be used to store trains and serve as the buffer between different maintenance operations or a temporary stop before the train departs from the depot. For an EMU train, there are driver cabs at both ends of this train, and it can move by itself in two directions without locomotives. A driver must be in the cab corresponding to the running direction to drive the train. When a train stays at an operating track, there are two positions of the track corresponding to the positions of the train's cabs for drivers to get on/off the train to start or end a driving task (which can be seen in Figure 1). For the sake of description convenience, we call the position of a track close to the entrance of the depot near working point (see, e.g., points numbered with odd numbers) and the other position which is far from the entrance of the depot far working point (see, e.g., points numbered with even numbers).

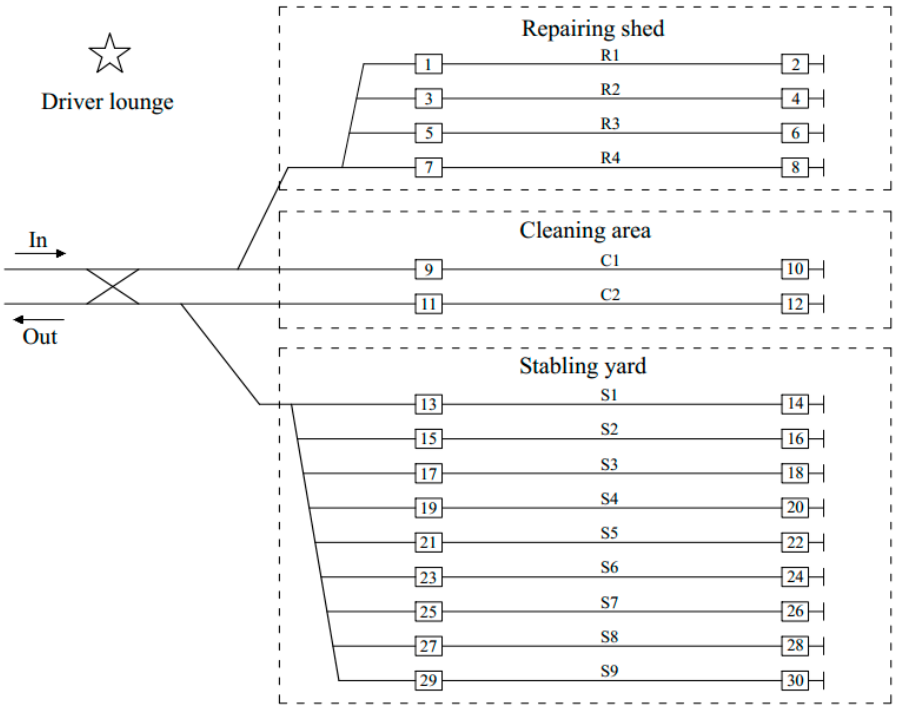

Figure 1. A typical layout of an EMU train depot.

Given a chart of the depot shunting schedule, we can derive all the driving tasks with detailed information, such as the start and end time and the start and end working point. Take the EMU-1 in Figure 2 as an example, the train arrives at the depot and stops before the cleaning track C1 at time 30. Then, a driver in the depot takes over the train at once and starts to perform the cleaning task of the train at the near working point of track C1, i.e., working point 9. After finishing the cleaning task at time 65, the driver will get off the train at the far working point of track C1, i.e., working point 10. Another driver will get on the train at working point 9 at time 65 to shunt the train to track R4 to perform the repairing task. During the period of the shunting task, the driver has to switch the cab once, the time of switching cab has been included in the shunting task. After finishing the shunting task, the driver will get off the train at the far working point of track R4, i.e., working point 8. Finally, when the main driving tasks of EMU-1 have been finished, a driver will drive the train from the operating track R4 to operating track S1, to be specific, shunting the train from working point 7 at time 136 to working point 14 at time 148, then all the driving tasks for EMU-1 have been finished and the train will wait on the track S1 for departure. In a depot, there are usually several fixed hours for drivers to sign in at the driver lounge. After signing in, drivers will receive the working schedule, then in order to perform tasks in their duties, they have to walk to the working point corresponding to the tasks. If the driver arrives at the working point before the task begins, he will wait until the task begins. When the driver finishes the duty assigned to him, he will walk to the drivers' lounge and sign out.

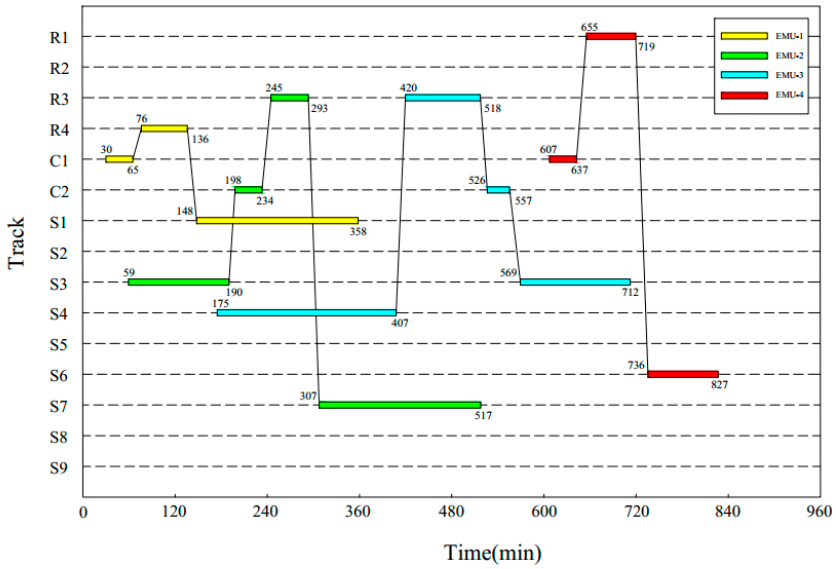

Figure 2. Train shunting schedule.

Before introducing the input data, we first state the assumptions of our study.

- All trains considered in the paper are homogenous and can be moved bidirectionally on tracks.
- All drivers considered in the paper are homogenous and able to drive all trains.
- All repairing, cleaning, and stabling tracks are first-in-last-out tracks.
- Drivers are smart, they would find the shortest path to reposition themselves between two points.

3.1. Input Data

Table 1 summarizes the input parameters of our problem, in which all time-related parameters are integer-valued. These parameters are described in detail as follows. The planning horizon $[0,T]$ is discretized and the time units are expressed as integers (e.g., $T = 960$, if the planning horizon is 16 h and each time unit is 1 min).

Table 1. Summary of input parameters.

Type of Data	Notation	Description
Depot data	N	set of working points in the depot, including the driver lounge n_0, $N = \left\{n_0, n_1, \cdots, n_{\lvert N\rvert}\right\}$
	N_o	set of working points that could be used as origin points of driving tasks, $N_o \subset N$
	E	set of track segments $i \to j$ in the depot, where $i, j \in N$
	d_{ij}	walking distance between working point n_i and n_j
Task data	R	set of driving tasks indicated in the shunting schedule
	o_r	origin working point of driving task r
	d_r	destination working point of driving task r
	p_r	start time of driving task r
	q_r	completion time of driving task r
	π_r	penalty for canceling driving task r
Driver data	K	set of depot drivers, $K = \left\{k_1, k_2, \cdots, k_{\lvert K\rvert}\right\}$
	T_{sign}	set of time points for drivers to sign in, $T_{sign} = \left\{t_1, t_2, \cdots, t_{\lvert T_{sign}\rvert}\right\}$
	$\overline{\tau}$	maximum allowed working time for drivers during the planning horizon
	v_k	average walking speed of driver k
	f_k	fixed cost for driver k if he/she is on-duty during the planning horizon
	α_k	unit cost for driver k when he/she is walking
	β_k	unit cost for driver k when he/she is waiting
	γ_k	unit cost for driver k to shunt trains in the depot
	δ_k	unit cost for driver k to drive trains at cleaning tracks

3.1.1. Depot Data

Let N be the set of nodes and E be the set of track segments in the depot, among which n_0 denoted the driver lounge and the other nodes $n_0, n_1, \cdots n_{|N|}$ represent working points along tracks. Each track segment $i \to j \in E$ represents the segment from working point i to working point j, with no intermediate working point in between. In nodes set N some points may be the origin points of driving tasks, which are collected in node set N_0. Note that a working point may be both an origin working point of a driving task and a destination working point of another driving task, e.g., in Figure 1, point 9 may be an origin working point of the driving task of driving a train at cleaning track C1, and point 9 may also be a destination working point of shunting task. Obviously, driver lounge n_0 is not in set N_o. The distance between working point n_i and n_j is denoted by d_{ij}.

3.1.2. Task Data

Let R be the set of considered driving tasks. For each driving task $r \in R$, the input data of each task r include: (i) the origin working point $o_r \in N \backslash \{n_0\}$; (ii) the destination working point $d_r \in N \backslash \{n_0\}$; (iii) the start time p_r at its start working point o_r; (iv) the completion time q_r at its destination working point d_r; and (v) a penalty π_r that is incurred if driving task r is canceled.

3.1.3. Driver Data

Let K be the set of considered drivers. For each driver $k \in K$, the input data include: (i) the average walking speed v_k; (ii) the operating cost α_k per unit time for driver k when he/she is walking; (iii) the operating cost β_k per unit time for driver k when he/she is waiting; (iv) the operating cost γ_k per unit time for driver k when he/she is executing shunting tasks; (v) the operating cost δ_k per unit time for driver k when he/she is executing cleaning tasks, i.e., driving a train at a cleaning track; and (vi) the fixed cost f_k for driver k if he/she is on duty, i.e., this driver works in the planning horizon, during the planning horizon. Considering the practical concern that the cleaning task is more difficult and fatiguing than the shunting task as the driver should control train speeds carefully, we require that $\gamma_k < \delta_k < \beta_k < \alpha_k$. Moreover, according to the depot regulation and labor

legislations, drivers are required to sign in at the given time points in set T_{sign}, and each driver can work at most $\overline{\tau}$ time during the planning horizon.

3.2. Objective and Constraints

The considered problem is an integrated problem which simultaneously generates duties to cover driving tasks and assign these obtained duties to depot drivers, such that the total cost is minimized. The obtained solution for each on-duty driver indicates this driver's sign-in time, sign-out time, driving tasks, timings for having rests, and walking route, etc., in the depot. A feasible depot driver schedule must satisfy the following constraints:

- Driving task assignment constraints: Each driving task $r \in R$ can be executed by a driver at most once.
- Sign-in constraints: If a driver $k \in K$ is on-duty, he/she must sign in at the specific time points in T_{sign} in the driver lounge and finally signs out in the driver lounge.
- Walking time constraints: The time interval between executing two consecutive driving tasks for the same driver should be no less than the walking time for the driver to walk from the proceeding task's destination working point to the succeeding task's origin working point.
- Maximum working time constraints: For each driver $k \in K$, the maximum duration of duty cannot exceed a fixed value $\overline{\tau}$.

The total cost of a solution is the sum of (i) total operating cost of executing (uncanceled) driving tasks, (ii) total cancellation cost of canceled driving tasks, (iii) total auxiliary cost for on-duty drivers to sign in, sign out, wait somewhere, and walk in the depot, and (iv) total fixed cost for on-duty drivers.

4. Time-Space Network Formulation

In this section we formulate the considered problem as a minimum-cost multi-commodity network flow problem with several restrictions, in which each commodity represents a depot driver, and the underlying network is an acyclic directed time-space network $G = (V, A)$. In what follows, we first introduce the construction of our time-space network in Section 4.1. We then discuss the flow restrictions in Section 4.2. Lastly, we present the minimum-cost multi-commodity network flow model in Section 4.3.

4.1. Time-Space Network Construction

The set of all possible time instants in the planning horizon $[0, T]$ is $\{0, 1, \cdots, T\}$, which forms the "time" dimension of our network G. The "space" dimension of G covers all working points. Corresponding to each working point $i \in N_o$ and each time instant t are three vertices, including ρ_i^t, $\overline{\rho}_i^t$ and $\hat{\rho}_i^t$, while corresponding to each working point $i \in N \backslash N_o$ and each time instant t are two vertices ρ_i^t and $\overline{\rho}_i^t$. In these vertices, ρ_i^t represents that a crew is staying at working point i and time t, $\overline{\rho}_i^t$ represents a crew's departure for walking, and $\hat{\rho}_i^t$ represents a crew's departure for executing driving tasks. Mathematically, the considered possible states that covered by "space" dimension is given by

$$V = \{o, d\} \cup \left\{\rho_i^t, \overline{\rho}_i^t, \hat{\rho}_i^t \middle| i \in N_o, t = 0, 1, \cdots T\right\} \cup \{\rho_i^t, \overline{\rho}_i^t | i \in N \backslash N_o, t = 0, 1, \cdots T\} \quad (1)$$

where vertices o and d are, respectively, the dummy source and sink for the muti-commodity flow.

The arc set A of G contains several types of arcs. The cost for driver $k \in K$ to traverse arc $u \to v$ is denoted by $\xi^k(u, v)$. In what follows, we present a detailed description of different types of arcs considered in this paper.

- Starting arcs: For working point $i = n_0$, i.e., the driver lounge, and each time instant $t \in T_{sign}$, there exists a starting arc $o \to \rho_i^t \in A_{sign}$. A driver traverses this arc represents that this driver signs in for the duty during the planning horizon, where a fixed cost of $\xi^k(o, \rho_i^t) = f_k$ is incurred when driver k is on-duty.

- Ending arcs: For working point $i = n_0$, i.e., the driver lounge, and each time instant $t = 0,\ 1,\ 2 \cdots T$, there exists an ending arc $\rho_i^t \to d$. For each driver $k \in K, \xi^k\left(\rho_i^t, d\right)$. This arc allows drivers to complete its operation at time t.
- Transfer arcs: There are transfer arcs in network G which represents the situation where a driver has finished dwelling at a working point and is about to leave current working point. There are two types of transfer arcs depending on what the next operation the driver will do. The first type of transfer arc is for the situation where the driver will walk to leave current point. For each working point $i \in N$ and time instant $t = 0,\ 1,\ 2 \cdots T$, there is a transfer arc $\rho_i^t \to \overline{\rho}_i^t$ of this type. For each $k \in K, \xi^k\left(\rho_i^t, \overline{\rho}_i^t\right) = 0$. The second type of transfer arc is for the situation where the driver will execute a driving task. For each working point $i \in N_o$ and time instant $t = 0, 1, \cdots T$, there is a transfer arc $\rho_i^t \to \hat{\rho}_i^t$ of this type if there exists one driving task r such that $i = o_r$ and $t = p_r$. For each $k \in K, \xi^k\left(\rho_i^t, \hat{\rho}_i^t\right) = 0$.
- Walking arcs: For each $i \to j \in E$ and time instants $t, t' = 0,\ 1,\ 2 \cdots T$, there is a walking arc $\overline{\rho}_i^t \to \rho_j^{t'}$ if there exists at least one driver $k \in K$ such that $t' - t$ equals to the round walking time of d_{ij}/v_k. For each $k \in K, \xi^k\left(\overline{\rho}_i^t, \rho_j^{t'}\right) = \alpha_k(t - t')$. This arc allows driver k to walk from working point i to working point j.
- Waiting arcs: For each working point $i \in N$ and time instant $t = 0,\ 1,\ 2 \cdots T - 1$, there exists a waiting arc $\rho_i^t \to \rho_i^{t+1}$. For each $k \in K, \xi^k\left(\rho_i^t, \rho_i^{t+1}\right) = \beta_k$. This arc allows a driver to wait at working point i when he/she is not working.
- Driving arcs: For each pair of working points $i \in N_o$ and $j \in N_d$, and time instants $t, t' = 0,\ 1,\ 2 \cdots T$, there is a driving arc $\rho_i^t \to \rho_i^{t+1}$ if there exists at least one driving task r such that $i = o_r,\ \ t = p_r,\ \ j = d_r$ and $t' = p_r$. For each $k \in K, \xi^k\left(\hat{\rho}_i^t, \rho_i^{t'}\right) = \gamma_k(t' - t) - \pi_r$ if task r is a shunting task, and $\xi^k\left(\hat{\rho}_i^t, \rho_j^{t'}\right) = \delta_k(t' - t) - \pi_r$ if task r is a driving task at a cleaning track. This arc allows driver k to shunt a train from one track to another or drive a train at a cleaning track. The cost coefficient $\xi^k\left(\hat{\rho}_i^t, \rho_j^{t'}\right)$ includes not only the operating cost $\gamma_k(t' - t)$ but also a cost reduction of π_r since task r would not be canceled if task r is executed by driver k.
- Dummy arcs: For the working point that related to the driver lounge, there is a dummy arc $o \to d$. A driver k traverses the dummy arc means the driver has no work to do during the planning horizon, we have $\xi^k(o, d) = 0$.

4.2. Constraints

A path from vertex o to vertex d in the time-space network G corresponds to a duty for depot drivers. A duty is feasible for depot driver $k \in K$ if all cost coefficients of driver k along this path are finite. The objective of this problem is to find feasible paths for all depot drivers such that the total cost is minimized. The path should satisfy the constraints in Section 3.2. So, besides the standard network flow constraints such as flow conservation constraints and supply/demand constraints, our multi-commodity flow model also has the following constraints:

- Driving task operation requirements: For each $r \in R$, the driving task r is executed at most once if it is uncanceled. Therefore, for each $r \in R$, we impose the constraint that the depot driver k's flow in the arc subset is at most one.

$$C_r^1 = A \cap \left\{ \hat{\rho}_{o_r}^{q_r} \to \rho_{d_r}^{q_r} \middle| r \in R \right\}, \tag{2}$$

- Maximum duty time requirements: For each $k \in K$, the duty time cannot exceed the maximum allowed working time for drivers during the planning horizon if the driver is on duty. For each arc $w \to d \in A$, we denote an arc subset as follows:

$$C_o^2 = \{o \to d\}, \tag{3}$$

in case of $w = o$, and

$$C_w^2 = A \cap \{o \to \rho_{n_0}^t | t = 0,\ 1, \cdots T\ s.t. \tau(w) - t \leq \overline{\tau}\}, \tag{4}$$

in case of $w \neq o$, where $\tau(w)$ is the time component of vertex w. For convenience, denote $V_d = \{o\} \cup \{\rho_{n0}^t | t = 0,\ 1, \cdots T\}$ and $C^2 = \{C_w^2 | w \in V_d\}$. For each $k \in K$, if driver k traverses arc $w \to d \in A$, then this driver must traverse an arc in arc in C_w^2 exactly once.

4.3. Time-Space Network Formulation

With the developed time-space network, we formulate the studied problem using a network flow model, where for each driver $k \in K$ and $u \to v \in A$, we use decision x_{uv}^k to indicate whether driver k traverses ($x_{uv}^k = 1$) arc $u \to v$ or not ($x_{uv}^k = 0$). The minimum cost multi-commodity network flow model for the considered problem is formulated as below:

$$\mathrm{P}: \ \min \sum_{r \in R} \pi_r + \sum_{k \in K} \sum_{u \to v \in A} \xi^k(u, v) x_{uv}^k \tag{5}$$

$$\text{s.t.} \sum_{\{v: o \to v \in A\}} x_{ov}^k = 1, \text{for all } k \in K \tag{6}$$

$$\sum_{\{v: v \to d \in A\}} x_{vd}^k = 1, \text{for all } k \in K \tag{7}$$

$$\sum_{\{u: u \to v \in A\}} x_{uv}^k = \sum_{\{u: v \to u \in A\}} x_{vu}^k, \text{for all } k \in K, v \in V \backslash \{o, d\} \tag{8}$$

$$\sum_{k \in K} \sum_{u \to v \in C_r^1} x_{uv}^k \leq 1, \text{for all } r \in R \tag{9}$$

$$x_{wd}^k \leq \sum_{u \to v \in C_w^2} x_{uv}^k, \text{for all } k \in K, w \in V_d \tag{10}$$

$$x_{uv}^k \in \{0,\ 1\}, \text{ for all } k \in K, u \to v \in A \tag{11}$$

In the objective function (5), the constant term "$\sum_{r \in R} \pi_r$" is the total cancellation penalty of all driving tasks, and the term "$\sum_{k \in K} \sum_{u \to v \in A} \xi^k(u, v) x_{uv}^k$" is the total cost of the solution less the cancellation penalties saved by successfully completing driving tasks. Then, objective function (5) represents the total cost of the solution. Constraints (6) are the supply constraints which require the outflow of each driver at vertex o to be 1. Constraints (7) are the demand constraints which require the inflow of each driver at vertex d to be 1. Constraints (8) are the flow conservation constraints for drivers at each vertex. Constraints (9) guarantee that each task is performed at most once. Constrains (10) required that the drivers must work no more than the specific time if he/she is on-duty. Constraints (11) are the binary constraints of the decision variables.

5. Lagrangian Relaxation Heuristic

Lagrangian relaxation provides an efficient mean to find optimality bounds for large-scale integer linear programs that permits us to decompose problems to exploit their special structure. Lagrangian relaxation has been widely used to solve railway planning problems; for example, see Dauzère-Pérès et al. and Xu et al. [24,25].

5.1. Lagrangian Relaxation

We relax constraints (9) and constraints (10) of problem P by Lagrangian way, and bring them into the objective function with Lagrangian multipliers $\lambda_r \geq 0$ for each $r \in R$ and μ_{wd}^k for each $k \in K$ and $w \in V_d$. We obtain the following relaxed problem, where λ is the vector of the λ_r values, and μ is the vector of the μ_{wd}^k values.

$$\mathrm{P}(\lambda,\mu):\ \min \sum_{r\in R}\pi_r+\sum_{k\in K}\sum_{u\to v\in A}\xi_{uv}^k x_{uv}^k+\sum_{r\in R}\lambda_r\left(\sum_{k\in K}\sum_{u\to v\in C_r^1}x_{uv}^k-1\right)+\sum_{k\in K}\sum_{w\in V_d}\mu_{wd}^k\left(x_{wd}^k-\sum_{u\to v\in C_w^2}x_{uv}^k\right) \tag{12}$$

$$\text{s.t.}\sum_{\{v:o\to v\in A\}}x_{ov}^k=1,\ \text{for all } k\in K \tag{13}$$

$$\sum_{\{v:v\to d\in A\}}x_{vd}^k=1,\ \text{for all } k\in K \tag{14}$$

$$\sum_{\{u:u\to v\in A\}}x_{uv}^k=\sum_{\{u:v\to u\in A\}}x_{vu}^k,\ \text{for all } k\in K, v\in V\backslash\{o,d\} \tag{15}$$

$$x_{uv}^k\in\{0,1\},\ \text{for all } k\in K, u\to v\in A \tag{16}$$

Furthermore, after removing the constant item " $\sum_{r\in R}\pi_r-\sum_{r\in R}\lambda_r$", the Lagrangian relaxation problem can be decomposed into $|K|$ independent problems. The subproblem corresponding to each $k\in K$ is given as follows

$$\mathrm{P}_k(\lambda,\mu):\ \min \sum_{u\to v\in A}\xi_{uv}^k x_{uv}^k+\sum_{r\in R}\sum_{u\to v\in C_r^1}\lambda_r x_{uv}^k+\sum_{w\in V_d}\mu_{wd}^k\left(x_{wd}^k-\sum_{u\to v\in C_w^2}x_{uv}^k\right) \tag{17}$$

$$\text{s.t.}\sum_{\{v:o\to v\in A\}}x_{ov}^k=1 \tag{18}$$

$$\sum_{\{v:v\to d\in A\}}x_{vd}^k=1 \tag{19}$$

$$\sum_{\{u:u\to v\in A\}}x_{uv}^k=\sum_{\{u:v\to u\in A\}}x_{vu}^k,\ \text{for all } v\in V\backslash\{o,d\} \tag{20}$$

$$x_{uv}^k\in\{0,1\},\ \text{for all } u\to v\in A \tag{21}$$

Obviously, each subproblem $\mathrm{P}_k(\lambda,\mu)$ is a standard shortest path problem with arc length $\delta_{uv}^k=\xi^k(u,v)+\sum_{\{u\to v\in C_r^1:r\in R\}}\lambda_r+\sum_{\{w\in V_d:u=w,v=d\}}\mu_{wd}^k-\sum_{\{w\in V_d:u\to v\in C_w^2\}}\mu_{wd}^k$.This shortest path problem can be solved easily with a standard dynamic programming algorithm. Given any vector λ of nonnegative λ_r values and vector μ of nonnegative μ_{wd}^k values, a lower bound on the optimal objective value of problem P can be obtained by solving the relaxed problem $\mathrm{P}(\lambda,\mu)$. To provide a tight lower bound, we would need to solve the following optimization problem $\max_{\lambda,\mu}\mathrm{P}(\lambda,\mu)$ which is referred to as the Lagrangian multiplier problem associated with original optimization P. If drivers are heterogenous, subproblems $\mathrm{P}_k(\lambda,\mu)$ can be solved in parallel for different $k\in K$ when they are solved by a multi-core computer processor.

5.2. Upper Bound Heuristic

In this section we present an upper bound heuristic algorithm, which contains two phases. In the first phase, we use a greedy algorithm to solve the problem P which can be considered as a shortest path problem of resource constraints. We solve the shortest path problem considering the maximum schedule duration time for each $k\in K$ with revised arc length δ_{uv}^k following the method used in Xu et al. [25]. For each $k\in K$, we first solve the shortest path problem without considering maximum schedule duration constraints using a dynamic programming whose computational complexity is $O(n)$, where n is the number of arcs in the time-space network. If the obtained assignment scheme for driver k satisfies the maximum schedule duration constrain, we will set the cost of the tasks assigned to the driver to infinite for avoiding this task will be covered by another driver. In contrast, if the obtained assignment scheme for driver k violates the maximum schedule

duration constrain, we temporarily block the tasks whose start time is less than the start time of the first task or whose start time is no less than the start time of the last task in the obtained assignment scheme. Then, we continue to solve the shortest path problem to get an assignment scheme for driver k. If the re-obtained assignment scheme satisfies the maximum duration constraint, we will release the tasks that are temporarily blocked and then block the tasks which are assigned to the driver permanently. Otherwise, we repeat the process above till find a feasible solution.

In the second phase, we try to use local search to improve the solution obtained in the first phase. The computational complexity of local search is obvious $O(n \log n)$, as there are exchange, swap, and insert operators, where n is the number of driver tasks. We then added a sentence. The operation that local changing a solution s to a neighbor s' in neighborhood $N(s)$ of solution s is called *move operator*. In this paper, we defined the following local search moves which are enlightened by Ma et al. [26]. It is worth noting that the specific move operator can be conducted only when the time permits.

- Exchange operator: Given two drivers' duties, choose one task in each duty of the drivers, and exchange the task with the subsequent tasks of the two duties. The procedure of this move is illustrated in Figure 3. The original *duty* 1 contains task *A*, *B*, and *C* sequentially, and the original *duty* 2 contains task *D*, *E*, and *F* sequentially, see Figure 3a. When conducting the "exchange" move, we exchange the tasks after task *A* in *duty* 1 and the tasks after task *D* in *duty* 2. After the "exchange" operator, *duty* 1 will contain task *A*, *E*, and *F* sequentially; *duty* 2 will contain task *D*, *B*, and *C* sequentially, see Figure 3b.
- Insert operator: Given two drivers' duties, choose one task and delete it in one of the duties; insert the task into an appropriate position in the other duty. The procedure of this move is shown in Figure 4. The original *duty* 1 contains task *A*, *B*, and *C* sequentially; the original *duty* 2 contains task *D* and *E* sequentially. When conducting the "insert" move, we delete task *B* in *duty* 1 and insert it into an appropriate position in *duty* 2 (such as between task *D* and task *E* in *duty* 2). After the "insert" operator, *duty* 1 will contain task *A* and *C* sequentially; *duty* 2 will contain task *D*, *B*, and *E* sequentially.
- Swap operator: Given two drivers' duties, choose one task and delete it in one of the duties; insert the task into an appropriate position in the other duty. The procedure of this move is illustrated in Figure 5. The original *duty* 1 contains task *A*, *B* and *C* sequentially; the original *duty* 2 contains task *D* and *E* sequentially. When conduct the "swap" move, we delete task *B* in *duty* 1 and insert it into an appropriate position in *duty* 2 (such as between task *D* and task *E* in *duty* 2), delete task *E* in *duty* 2 and insert it into an appropriate position in *duty* 1 (such as between task *A* and task *C* in *duty* 1). After the "swap" operator, *duty* 1 will contain task *A*, *E* and *C* sequentially; *duty* 2 will contain task *D*, *B* and *F* sequentially.

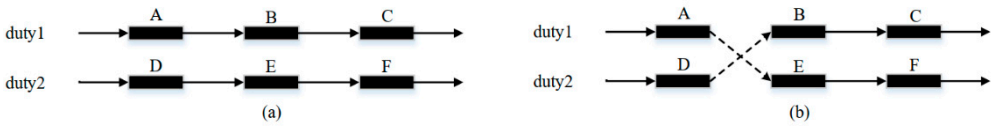

Figure 3. Exchange operator: (**a**) before exchange operation; (**b**) after exchange operation.

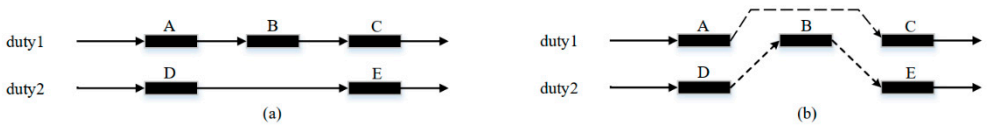

Figure 4. Insert operator: (**a**) before insert operation; (**b**) after insert operation.

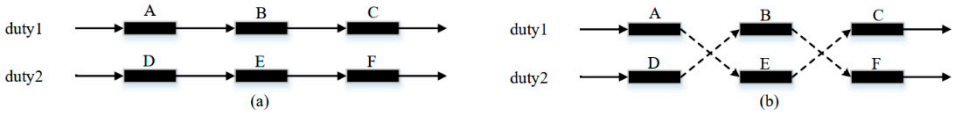

Figure 5. Swap operator: (**a**) before swapping; (**b**) after swapping.

We adopt some ideas of the paper of Hoogeboom et al. [27] when we use above moves to generate new duties, it probably generates infeasible solutions. It is worth pointing out that Hoogeboom allows the violation of constraints to exist by adding a penalty to the infeasible solution to reach different areas of the solution space. In this paper, we do not allow infeasible solutions to exist. Thus, before performing a move operation, we will check the feasibility of the move. In the local search, we explore the three different neighborhoods which are in increasing order of complexity. If an improved solution is identified, the search will restart at the first neighbor of the new solution found. The local search will finish if all neighbors are explored. The pseudocode for this local search is given in Algorithm 1.

Algorithm 1 Pseudocode for the local search: local_search(s)

Input: An initial solution s, move operators set $M = \{m_1, m_2, m_3\}$, let $N_{m(s)}$ denote the neighborhood of solution s with move operator m
1: $m \leftarrow 1$
2:**while** $m \leq |M|$ **do**
3: **for all** $N_{m(s)}$ **do**
4: $s' \leftarrow N_{m(s)}$
5: **if** s' is better than s **then**
6: $s = s'$,
7: $m = 0$,
8: **break**
9: $m \leftarrow m + 1$
Output: s

Because the local search method is time-consuming, we execute the second phase with a certain probability after obtaining solutions in the first phase. The overall framework of the upper bound heuristic algorithm is shown in the Algorithm 2.

Algorithm 2 Framework of the upper bound heuristic

Input: Driver set K, tasks set R, the probability of local search p_L
1:use the greedy heuristic *ub_greedy* (K, R) to obtain solution s
2:randomly generate p from (0,1)
3:if $p \leq p_L$ then
4:$s \leftarrow local\_search(s)$
Output: s

5.3. Subgradient Optimization Procedure

In this section, we present a subgradient optimization procedure which is used to search the near-optimal value of λ_r and μ_{wd}^k. In the iteration of the subgradient procedure, the value "$\sum_{k\in K}\sum_{u\to v\in C_r^1} x_{uv}^k - 1$" for each $r \in R$ and the value "$x_{wd}^k - \sum_{u\to v\in C_w^2} x_{uv}^k$" for each $k \in K$ and $w \in V_d$ form a subgradient vector $\eta = \left\{\eta_1, \cdots, \eta_{|R|}, \eta_{|R|+1}, \cdots, \eta_{|R|+|V_d|}\right\}$ of the relaxed solution.Let η^l (respectively λ^l and μ^l) denote the η (respectively λ and μ) vector in the lth iteration of the procedure let η_m^l be the mth component of η^l for $m = 1, 2, \cdots, |R| + |V_d|$, let λ_m^l be the mth component of Lagrangian multiplier vector μ^l for $m = 1, 2, \cdots, |V_d|$. In the initial iteration, the components in η are all initialized as

0, and the Lagrangian multipliers are all set to 0. In the following iterations ($l > 1$), we update each multiplier according to the following formulas, e.g., see Xu et al. [25].

$$\lambda_m^l \leftarrow \max\left\{\lambda_m^{l-1} + \theta \cdot \frac{UB - LB(\lambda,\mu)}{\|\eta^l\|^2} \cdot \eta_m^l, 0\right\} \ (m = 1,\ 2, \cdots, |R|), \tag{22}$$

and

$$\mu_m^l \leftarrow \max\left\{\mu_m^{l-1} + \theta \cdot \frac{UB - LB(\lambda,\mu)}{\|\eta^l\|^2} \cdot \eta_{m+|R|}^l, 0\right\} \ (m = 1,\ 2, \cdots, |V_d|) \tag{23}$$

where $\theta > 0$ is a prespecified step size parameter, UB is the best feasible solution of problem P been found and $LB(\lambda, \mu)$ is the optimal objective value of $P(\lambda, \mu)$ corresponding to the current multipliers λ and μ.

The procedure of each iteration of the subgradient optimization is described as follows: (i) obtain a relaxed solution of problem P by solving $|K|$ shortest path problems $P_k(\lambda, \mu)$; (ii) obtain a feasible solution of problem P using the upper bound heuristic presented in Section 4.2; (iii) identify constraints that violated by current relaxed solution; (iv) update the 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 situations occurs: (i) the gap between the upper bound and lower bound is below a prespecified threshold; (ii) the computational time reaches a prespecified limit; or (iii) the number of iterations reaches a prespecified limit. See Section 5 for more details regarding parameter settings.

6. Computational Study

In this section, we conduct computational experiments to test the method we presented in this paper. In Section 5.1, we introduce the generation method of test instances and the parameter setting. In Section 5.2, we conduct a computational study to evaluate our model and method. All the test instances are compiled with C# Language using Visual Studio 2019 and run on a personal computer with a 3.70 GHz Intel Core i9-10900k processor and 32 GB of internal memory.

6.1. Generation of Test Instances

We adopt two different sizes of depot networks to design instances to test the effectiveness of the proposed method in this paper. Specifically, the first depot network, namely Network 1, is originated from the network of Shanghai South Depot in China, see Figure 1. Network 1's track layout is of parallel-arrangement type. It has 4 repairing tracks, 2 cleaning tracks, and 9 storage tracks with 31 nodes (including a node that represents the driver lounge). Considering the development of high-speed railway and the expansion of the depots, we extend the current network of Shanghai South Depot. The extended depot network, called Network 2, has 8 repairing tracks, 4 cleaning tracks, and 18 storage tracks as well as 61 nodes (including a node that represents the driver lounge); for more details, see Figure A1.

In our computational study, the input parameters are estimated based on the characteristics of Shanghai South Depot in China. We first test a set of instances with the same task size while different driver sizes in Network 2 to determine a proper ratio of task size to driver size. We here consider one task size of $|R| = 160$ and five driver sizes of $|K| = 8$, 10, 12, 14, 16. For each combination of task size and driver size, we randomly generate 5 instances, which gives 25 test instances. After determining such ratio, we then consider three task sizes with $|R| = 40$, 80, 120 for Network 1, and three task sizes with $|R| = 160$, 200, 240 for Network 2. Since there are four driving tasks, including three shunting tasks and one cleaning task for each EMU train that is performed a level-one maintenance. We then need to consider EMU train shunting schedules with three train sizes of 10, 20 and 30 for Network 1, and three train sizes of 40, 50 and 60 for Network 2. For each task size, we generate 5 instances by randomly generating 5 EMU train shunting schedules,

which gives another 30 instances. For each test instance with these two networks, we set the length of the planning horizon to T = 960 min, e.g., from 16:00 of one day to 08:00 of the next day, which contains two crew shifts. We consider two sign-in time windows for drivers, including [0, 60] and [480, 540], i.e., $T_{sign} = \{0, 1, \cdots, 60, 480, 481, \cdots, 540\}$. The maximum allowed working time $\overline{\tau}$ for each driver is set to 480 min.

For each test instance, we generate train shunting and cleaning tasks as follows. Firstly, for each considered EMU train ι the arrival time a_ι at the depot is randomly generated from a discrete uniform distribution between 20 and $T - 180$, the departure time from the depot is randomly generated from a discrete uniform distribution between $a_\iota + 180$ and $T - 20$. Here, the time of 20 min is the minimum required walking time for drivers and the time of 180 min is the minimum required in-depot time for EMU trains. Secondly, the time for shunting EMU train ι in the depot is randomly selected from the set {4,5,6}, where the probability of each value being selected is $1/3$. The time forcleaning EMU train ι in the depot is randomly selected from the set {20,21,$\cdots$,30}, where the probability of each value being selected is $1/11$. The time for repairing EMU train ι is randomly generated from a discrete uniform distribution between 80 and 100. Thirdly, with these parameters mentioned above, we generate a train shunting schedule by using a greedy first-in-first-served method. Finally, we can obtain task data on the basis of the generated train shunting schedule, see Section 2.

For each driver $k \in K$, let c denote the unit cost for driver k to shunt trains in the depot, i.e., $\gamma_k = c$, we set unit cleaning cost $\delta_k = (1.1)c$, unit walking cost $\alpha_k = (1.3)c$, unit waiting cost $\beta_k = (1.2)c$ and the fixed attendance cost $\alpha_k = (10.0)c$. Specifically, we set the unit waiting cost in driver lounge as 0.3 times the unit waiting cost outside the driver lounge because drivers can get a better rest in the lounge, and we encourage drivers to go back to the lounge and wait. Moreover, the penalty of canceling driving task r is set equal to $(1.2)\cdot[60\cdot\max(\alpha, \beta) + c_r \cdot t_r]$, where $\alpha = \max\left\{\alpha_1, \cdots, \alpha_{|K|}\right\}$, $\beta = \max\left\{\beta_1, \cdots, \beta_{|K|}\right\}$, $c_r = \max\left\{\gamma_1, \cdots, \gamma_{|K|}\right\}$, if task r is a shunting task and $c_r = \max\left\{\beta_1, \cdots, \beta_{|K|}\right\}$ if task r is a cleaning task, as well as t_r is the duration of task r. For simplicity, the monetary unit is scaled by setting c = 1.

In our implementation of the Lagrangian relaxation heuristic, we set the parameter θ to 2.0 which will be reduced by 5% if the best lower bound identified shows no improvement for 15 consecutive iterations. If the optimality gap is less than 1.00%, the subgradient optimization procedure is terminated. The prespecified CPU time is set to two hours, and the prespecified maximum iterations is set to 1000. In addition, to save CPU time, for the first 200 iterations of the subgradient optimization procedure, we execute the upper bound heuristic and update UB at each iteration. After 200 iterations, we execute the upper bound heuristic with probability 0.1 at each iteration. Moreover, since local search method is also time-consuming, we execute local search method with probability 0.1 when the upper bound heuristic is executed.

6.2. Computational Results

Table 2 summarizes the results of the first part of the computational study. The "UD" column reports the number of drivers that do not work during the planning horizon. The column "UT" reports the number of tasks that are not covered by drivers. The column "Gap" reports the optimality gap between the lower boundand the upper bound, which is defined as:

$$Gap = \frac{UB - LB}{LB} \times 100\%, \tag{24}$$

where the UB is the objective value of the corresponding solution method, and LB is obtained by the Lagrangian relaxation heuristic. The "Time (s)" column reports the CPU time (in seconds) used in each test instance.

Table 2. Computational results (Part 1).

Instance	$\|K\|$	$\|R\|$	*LB*	*UB*	*UD*	*UT*	*Gap*	Time (s)
1			4609.16	6168.60	0.00	21.00	33.83%	493.00
2			4811.90	7061.22	0.00	29.00	46.74%	504.00
3	8	160	4644.21	6304.68	0.00	21.00	35.75%	520.00
4			4778.32	5957.60	0.00	19.00	24.68%	517.00
5			4791.81	6291.20	0.00	20.00	31.29%	523.00
Average:			4727.08	6356.66	0.00	22.00	34.46%	511.40
6			4640.82	4932.12	0.00	4.00	6.28%	642.00
7			4856.78	5363.40	0.00	7.00	10.43%	577.00
8	10	160	4682.38	5220.34	0.00	7.00	11.49%	674.00
9			4787.06	5008.66	0.00	4.00	4.63%	657.00
10			4824.05	5444.64	0.00	7.00	12.86%	655.00
Average:			4758.22	5193.83	0.00	5.80	9.14%	641.00
11			4620.82	4795.22	1.00	1.00	3.78%	791.00
12			4850.18	5070.78	0.00	1.00	4.55%	713.00
13	12	160	4687.47	4954.82	0.00	1.00	5.70%	1019.00
14			4763.78	4908.48	1.00	3.00	3.04%	806.00
15			4821.67	5088.62	0.00	0.00	5.54%	850.00
Average:			4748.74	4963.58	0.40	1.20	4.52%	835.80
16			4623.38	4874.24	2.00	0.00	5.43%	857.00
17			4849.61	5034.34	2.00	1.00	3.81%	796.00
18	14	160	4685.95	5000.32	2.00	2.00	6.71%	1086.00
19			4752.28	4954.78	2.00	2.00	4.26%	836.00
20			4812.11	5081.06	2.00	0.00	5.59%	990.00
Average:			4744.67	4988.95	2.00	0.80	5.16%	913.00
21			4617.25	4894.12	4.00	0.00	6.00%	960.00
22			4845.50	5054.16	4.00	1.00	4.31%	910.00
23	16	160	4687.87	4950.14	5.00	2.00	5.59%	1242.00
24			4733.67	4935.52	4.00	2.00	4.26%	989.00
25			4809.03	5090.32	4.00	0.00	5.85%	1097.00
Average:			4738.66	4984.85	4.20	1.00	5.20%	1039.60

From Table 2, we can see that the number of drivers has an important impact on the optimality gap. From the results of instances with 8 drivers we can see that the optimality gap is approximately proportional to the number of tasks not performed, i.e., the more tasks not performed, the higher the optimality gap. From the computational results of instances with 8, 10, and 12 drivers, we can see that as the number of drivers increases, the optimality gap fails rapidly. From the computational results of instances with 12, 14, and 16 drivers, we can obtain a managerial insight that driver size of 12 seems large enough for 160 tasks, since we cannot reduce the optimality gap by using more drivers. Therefore, we set the ratio of the number of drivers to the number of driving tasks as 3:40 for the instances tested in the second part of our computational study. From Table 2, we can also see that the CPU time for each test instance tend to be larger as the driver size increases.

Table 3 summarizes the results of the second part of the computational study. In order to evaluate the improvement of the local search to solutions, we conduct two kinds of upper bound heuristics. One only uses the greedy algorithm to obtain feasible solutions, while the other one further uses the local search method to improve the solutions obtained by the greedy algorithm in the first phase. In Table 3, "NL" denotes the local search is not used in the upper bound heuristic and "LS" denotes the upper bound heuristic used local search. Table 3 shows that our approach is efficient tools that can help railway operators to obtain high-quality schedules for depot drivers. That is, the proposed approaches can not only relieve managers' pressure of scheduling drivers, but also reduce the driver-related cost as much as possible.

Table 3. Computational results (Part 2).

			NL						LS						
Instance	$\|K\|$	$\|R\|$	***LB***	***UB***	***UD***	***UT***	***Gap***	**Time (s)**	***LB***	***UB***	***UD***	***UT***	***Gap***	**Time (s)**	**Impv.**
Network 1															
1			1210.43	1319.28	0.00	3.00	8.99%	89.00	1210.43	1319.28	0.00	3.00	8.99%	81.00	0.00%
2			1547.27	1976.70	0.00	9.00	27.75%	93.00	1547.27	1976.70	0.00	9.00	27.75%	115.00	0.00%
3	3	40	1315.73	1619.36	0.00	6.00	23.08%	99.00	1315.73	1619.36	0.00	6.00	23.08%	143.00	0.00%
4			1240.58	1647.62	0.00	6.00	32.81%	84.00	1240.58	1647.62	0.00	6.00	32.81%	109.00	0.00%
5			1229.81	1374.58	0.00	3.00	11.77%	104.00	1229.81	1374.58	0.00	3.00	11.77%	149.00	0.00%
Average:			1308.76	1587.51	0.00	5.40	20.88%	93.80	1308.76	1587.51	0.00	5.40	20.88%	119.40	0.00%
6			1985.43	2039.56	0.00	0.00	2.73%	173.00	1984.89	2034.94	0.00	0.00	2.52%	257.00	0.23%
7			2122.64	2262.82	0.00	1.00	6.60%	176.00	2123.04	2259.22	0.00	1.00	6.41%	270.00	0.16%
8	6	80	2043.96	2186.74	0.00	0.00	6.99%	170.00	2044.46	2183.86	0.00	0.00	6.82%	250.00	0.13%
9			2066.76	2160.18	0.00	1.00	4.52%	163.00	2066.73	2158.78	0.00	1.00	4.45%	214.00	0.06%
10			2165.23	2230.88	0.00	1.00	3.03%	196.00	2164.10	2228.94	0.00	1.00	3.00%	211.00	0.09%
Average:			2076.80	2176.04	0.00	0.60	4.77%	175.60	2076.64	2173.15	0.00	0.60	4.64%	240.40	0.13%
11			2856.43	3029.94	1.00	0.00	6.07%	274.00	2860.75	3025.84	1.00	0.00	5.77%	377.00	0.14%
12			2834.61	3010.22	0.00	0.00	6.20%	236.00	2834.89	2991.98	0.00	0.00	5.54%	360.00	0.61%
13	9	120	2815.78	2989.74	1.00	0.00	6.18%	225.00	2815.61	2979.06	1.00	0.00	5.81%	337.00	0.36%
14			2754.48	2902.80	1.00	0.00	5.38%	223.00	2756.59	2897.64	1.00	0.00	5.12%	344.00	0.18%
15			2788.21	2938.80	1.00	0.00	5.40%	218.00	2788.21	2938.80	1.00	0.00	5.40%	347.00	0.00%
Average:			2809.90	2974.30	0.80	0.00	5.85%	235.20	2811.21	2966.66	0.80	0.00	5.53%	353.00	0.26%
Network 2															
16			4625.77	4915.36	1.00	1.00	6.26%	747.00	4620.59	4795.22	1.00	1.00	3.78%	819.00	2.51%
17			4850.56	5132.46	0.00	1.00	5.81%	700.00	4850.18	5070.78	0.00	1.00	4.55%	781.00	1.22%
18	12	160	4686.25	5114.44	0.00	1.00	9.14%	716.00	4687.47	4954.82	0.00	1.00	5.70%	1018.00	3.22%
19			4765.90	5005.18	1.00	3.00	5.02%	682.00	4763.78	4908.48	1.00	3.00	3.04%	834.00	1.97%
20			4826.73	5199.06	0.00	0.00	7.71%	793.00	4822.72	5089.96	0.00	0.00	5.54%	896.00	2.14%
Average:			4751.04	5073.30	0.40	1.20	6.79%	727.60	4748.95	4963.85	0.40	1.20	4.52%	869.60	2.21%
21			5987.43	6390.68	1.00	1.00	6.74%	944.00	5981.75	6222.48	1.00	1.00	4.02%	1255.00	2.70%
22			5644.36	6178.52	0.00	1.00	9.46%	992.00	5641.55	5988.44	0.00	1.00	6.15%	1676.00	3.17%
23	15	200	5641.91	6059.82	1.00	0.00	7.41%	895.00	5641.61	5867.12	1.00	0.00	4.00%	1332.00	3.28%
24			5854.48	6404.70	0.00	1.00	9.40%	999.00	5854.78	6183.52	0.00	1.00	5.61%	1453.00	3.58%
25			5997.43	6507.26	0.00	0.00	8.50%	969.00	5996.43	6348.74	0.00	0.00	5.88%	1212.00	2.50%
Average:			5825.12	6308.20	0.40	0.60	8.30%	959.80	5823.22	6122.06	0.40	0.60	5.13%	1385.60	3.05%
26			7158.63	7701.54	2.00	3.00	7.58%	1088.00	7139.62	7401.90	2.00	3.00	3.67%	2461.00	4.05%
27			6944.08	7626.12	1.00	2.00	9.82%	1175.00	6938.38	7316.78	1.00	2.00	5.45%	2053.00	4.23%
28	18	240	6632.26	7229.12	2.00	0.00	9.00%	1119.00	6622.07	7042.46	2.00	0.00	6.35%	1970.00	2.65%
29			6716.87	7169.36	2.00	0.00	6.74%	1100.00	6715.53	7008.32	2.00	0.00	4.36%	2535.00	2.30%
30			6545.21	7026.84	2.00	0.00	7.36%	1102.00	6532.70	6759.46	2.00	0.00	3.47%	1605.00	3.96%
Average:			6799.41	7350.60	1.80	1.00	8.10%	1116.80	6789.66	7105.78	1.80	1.00	4.66%	2124.80	3.44%

From Table 3, it can also be observed that local search method can improve the initial solution obtained by greedy heuristic algorithm, especially for those instances with large depot network scale and intensive driving tasks. It shows that the optimality gap improvements obtained by the local search method tend to be larger as the task size and driver size increase with the underlying Network 2. In contrast, for those instances with small depot network and less driving task, the optimality gap improvements are very little, in which most gap improvements are less than 0.5% for the instances with the underlying Network 1. It is probably because the solution space is small for the small-scale instances and the greedy heuristic can obtain a good enough or even the best solution.

From *UD* and *UT* columns of Table 3, we can observe that the ratio of task size to driver size is not fixed. Particularly, the ratio of task size to driver size for the small-scale instances is bigger than that for large-scale instances. This coincides with our practice experience. That is, for the small-scale instances where the driving tasks are less, the

time duration between two tasks is long, which results in more driver idle time. For the large-scale instances where the driving tasks are intensive, drivers can perform more tasks on average. This result can help railways operators to determine the number of drivers when build or extend EMU train depots. Specifically, from the managerial perspective, drivers could be paid more in a busy depot where there are more driving tasks.

Moreover, we should mention that although the local search can improve the initial solution, it takes a lot of time. Even though we run local search with a probability of 0.1, the upper bound heuristic is still time consuming. Especially, the bigger the instance, the more time local search takes. For the instances with the largest scale, e.g., instances 26–30, the computation time of the algorithm with local search is twice that of the algorithm without local search.

7. Conclusions

In this paper, we studied a practical depot driver scheduling problem in the railway system that has been noticed by a few researchers. We formulate the problem as a minimum-cost multi-commodity network flow model with integer flow restrictions in a time-space network. We developed a Lagrangian relaxation heuristic for the proposed network flow model and designed an upper bound heuristic with a local search method. We conducted a computational study with a practical underlying railway depot in China. The computational results demonstrate the effectiveness of our solution method. The obtained results also show that our approach can help railway operators to find the preferred ratio of the number of drivers to the number of driving tasks, and drivers could be paid more in a busy depot.

It is worth mentioning that our research has several limitations. For example, though the local search method can improve the initial solutions obtained by the greedy heuristic, the CPU time it requires is long. Hence, one interesting future research topic would be to design a fast and effective heuristic algorithm to solve the problem. Similarly, if the task size is huge, the scale of the constructed time-space would be larger and the feasible path for a driver would be long, which needs more CPU memory, and the computational burden becomes heavy. Therefore, developing mathematical techniques to reduce the network size is also an interesting future research topic. Moreover, in this paper, we do not consider the workload balance of drivers and take the shunting schedule as known input data, so the problem of driver scheduling with the equity consideration and the integrated optimization to depot driver scheduling would also be one of the future research interests. In addition, railway depot operations are vulnerable to unexpected disruptions. Once a disruption occurs, we need to use efficient methods to reschedule drivers in the depot. So, another interesting future research direction is to develop efficient driver rescheduling methods for railway operators.

Author Contributions: Conceptualization, F.P., X.F., P.W. and M.S.; methodology, F.P., X.F., P.W. and M.S.; software, F.P., P.W.; validation, F.P., X.F., P.W. and M.S.; formal analysis, F.P., X.F., P.W. and M.S.; investigation, F.P.; resources, X.F.; data curation, F.P., P.W.; writing—original draft preparation, F.P., X.F.; writing—review and editing, F.P., X.F., and M.S.; visualization, F.P.; supervision, F.P. and X.F.; project administration, P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This study is supported by the R&D Program of Beijing Municipal Education Commission (KM202210858004) and the Key Program of Beijing Polytechnic (2021Z018-KXZ).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The depot networks adopted in Section 6 are presented in Figure A1.

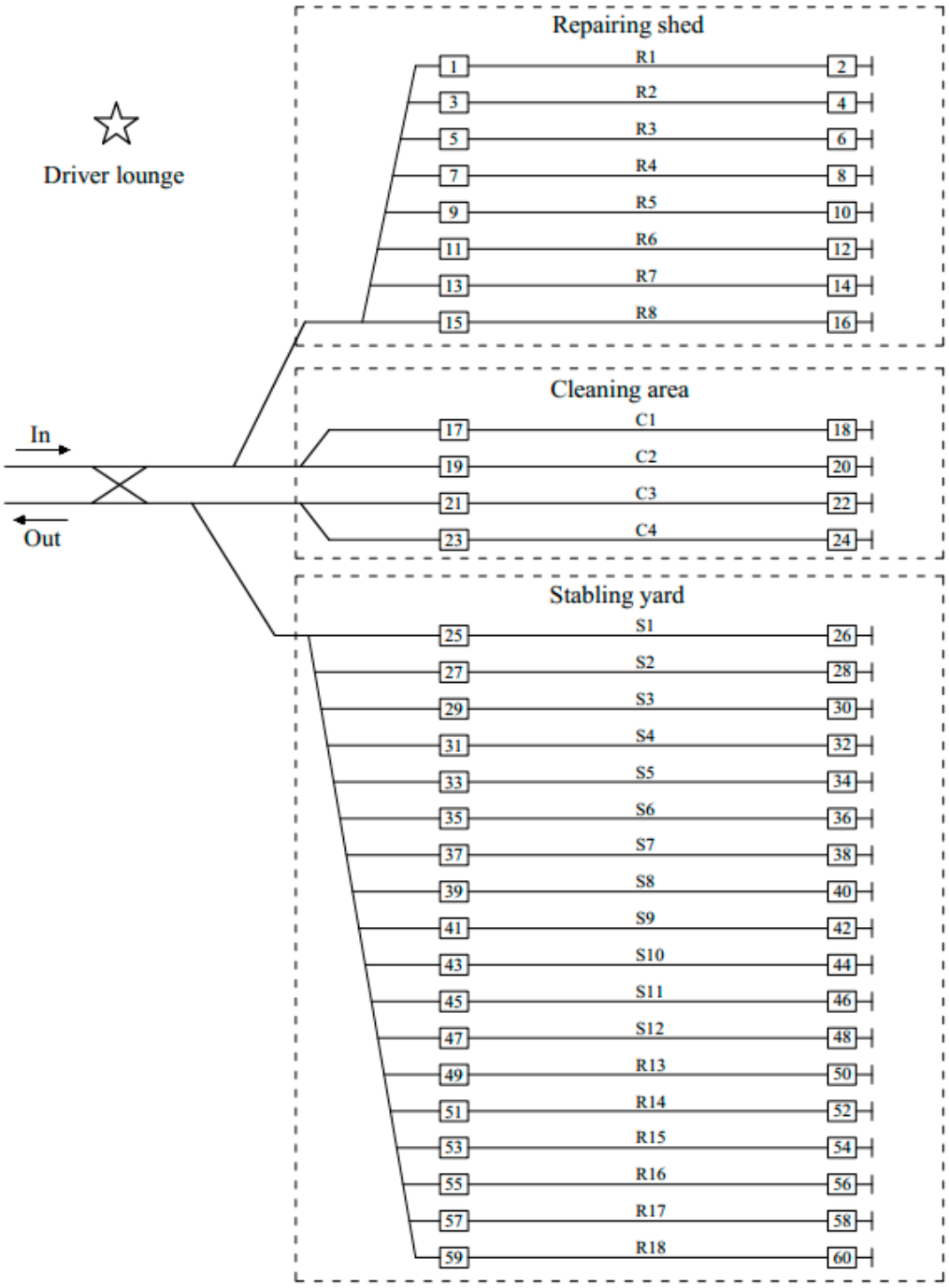

Figure A1. Network 2.

References

1. Lin, B.; Wu, J.; Lin, R.; Wang, J.; Wang, H.; Zhang, X. Optimization of high-level preventive maintenance scheduling for high-speed trains. *Reliab. Eng. Syst. Saf.* **2019**, *183*, 261–275. [CrossRef]
2. Heil, J.; Hoffmann, K.; Buscher, U. Railway crew scheduling: Models, methods and applications. *Eur. J. Oper. Res.* **2020**, *283*, 405–425. [CrossRef]
3. Wang, J.; Gronalt, M.; Sun, Y. A two-stage approach to the depot shunting driver assignment problem with workload balance considerations. *PLoS ONE* **2017**, *12*, e0181165. [CrossRef] [PubMed]

4. Pour, S.M.; Drake, J.H.; Ejlertsen, L.S.; Rasmussen, K.M.; Burke, E.K. A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem. *Eur. J. Oper. Res.* **2018**, *269*, 341–352. [CrossRef]
5. Souza, R.L.C.; Ghasemi, A.; Saif, A.; Gharaei, A. Robust job-shop scheduling under deterministic and stochastic unavailability constraints due to preventive and corrective maintenance. *Comput. Ind. Eng.* **2022**, *168*, 108130. [CrossRef]
6. Gharaei, A.; Amjadian, A.; Amjadian, A.; Shavandi, A.; Hashemi, A.; Taher, M.; Mohamadi, N. An integrated lot-sizing policy for the inventory management of constrained multi-level supply chains: Null-space method. *Int. J. Syst. Sci. Oper. Logist.* **2022**, 1–14. [CrossRef]
7. Gharaei, A.; Hoseini Shekarabi, S.A.; Karimi, M. Optimal lot-sizing of an integrated EPQ model with partial backorders and re-workable products: An outer approximation. *Int. J. Syst. Sci. Oper. Logist.* **2021**, 7–11. [CrossRef]
8. Gharaei, A.; Amjadian, A.; Shavandi, A. An integrated reliable four-level supply chain with multi-stage products under shortage and stochastic constraints. *Int. J. Syst. Sci. Oper. Logist.* **2021**, 1–22. [CrossRef]
9. Gharaei, A.; Almehdawe, E. Optimal sustainable order quantities for growing items. *J. Clean. Prod.* **2021**, *307*, 127216. [CrossRef]
10. Jütte, S.; Albers, M.; Thonemann, U.W.; Haase, K. Optimizing railway crew scheduling at DB Schenker. *Interfaces* **2011**, *41*, 109–122. [CrossRef]
11. Nishi, T.; Muroi, Y.; Inuiguchi, M. Column generation with dual inequalities for railway crew scheduling problems. *Public Transp.* **2011**, *3*, 25–42. [CrossRef]
12. Liu, M.; Haghani, A.; Toobaie, S. Genetic algorithm-based column generation approach to passenger rail crew scheduling. *Transp. Res. Rec.* **2010**, *2159*, 36–43. [CrossRef]
13. Guillermo, C.G.; José, M.R.L. Hybrid algorithm of tabu search and integer programming for the railway crew scheduling problem. In Proceedings of the 2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA), Wuhan, China, 28–29 November 2009; Volume 2, pp. 413–416. [CrossRef]
14. Elizondo, R.; Parada, V.; Pradenas, L.; Artigues, C. An evolutionary and constructive approach to a crew scheduling problem in underground passenger transport. *J. Heuretics* **2010**, *16*, 575–591. [CrossRef]
15. Khmeleva, E.; Hopgood, A.A.; Tipi, L.; Shahidan, M. Fuzzy-logic controlled genetic algorithm for the rail-freight crew-scheduling problem. *KI—Künstliche Intelligenz* **2018**, *32*, 61–75. [CrossRef]
16. Caprara, A.; Kroon, L.; Monaci, M.; Peeters, M.; Toth, P. Passenger railway optimization. In *Handbooks in Operations Research and Management Science*; Barnhart, C., Laporte, G., Eds.; NorthHolland: Amsterdam, The Netherlands, 2007; Volume 14, pp. 129–187.
17. Amjadian, A.; Gharaei, A. An integrated reliable five-level closed-loop supply chain with multi-stage products under quality control and green policies: Generalised outer approximation with exact penalty. *Int. J. Syst. Sci. Oper. Logist.* **2021**, *9*, 1–21. [CrossRef]
18. Askari, R.; Sebt, M.V.; Amjadian, A. A Multi-product EPQ Model for Defective Production and Inspection with Single Machine, and Operational Constraints: Stochastic Programming Approach. In *Logistics and Supply Chain Management*; LSCM 2020. Communications in Computer and Information Science; Molamohamadi, Z., Babaee Tirkolaee, E., Mirzazadeh, A., Weber, G.W., Eds.; Springer: Cham, Switzerland, 2021; pp. 161–193. [CrossRef]
19. Taleizadeh, A.A.; Safaei, A.Z.; Bhattacharya, A.; Amjadian, A. Online peer-to-peer lending platform and supply chain finance decisions and strategies. *Ann. Oper. Res.* **2022**, *315*, 397–427. [CrossRef]
20. Chew, K.-L.; Pang, J.; Liu, Q.; Ou, J.; Teo, C.-P. An optimization based approach to the train operator scheduling problem at Singapore MRT. *Ann. Oper. Res.* **2001**, *108*, 111–122. [CrossRef]
21. Amaya, J.; Uribe, P. A model and computational tool for crew scheduling in train transportation of mine materials by using a local search strategy. *TOP* **2018**, *26*, 383–402. [CrossRef]
22. Zhou, J.; Xu, X.; Long, J.; Ding, J. Integrated optimization approach to metro crew scheduling and rostering. *Transp. Res. Part C Emerg. Technol.* **2021**, *123*, 102975. [CrossRef]
23. Borndörfer, R.; Schulz, C.; Seidl, S.; Weider, S. Integration of duty scheduling and rostering to increase driver satisfaction. *Public Transp.* **2017**, *9*, 177–191. [CrossRef]
24. Dauzère-Pérès, S.; De Almeida, D.; Guyon, O.; Benhizia, F. A Lagrangian heuristic framework for a real-life integrated planning problem of railway transportation resources. *Transp. Res. Part B Methodol.* **2015**, *74*, 138–150. [CrossRef]
25. Xu, X.; Li, C.-L.; Xu, Z. Integrated train timetabling and locomotive assignment. *Transp. Res. Part B Methodol.* **2018**, *117*, 573–593. [CrossRef]
26. Ma, J.; Ceder, A.; Yang, Y.; Liu, T.; Guan, W. A case study of Beijing bus crew scheduling: A variable neighborhood-based approach. *J. Adv. Transp.* **2016**, *50*, 434–445. [CrossRef]
27. Hoogeboom, M.; Dullaert, W.; Lai, D.; Vigo, D. Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. *Transp. Sci.* **2020**, *54*, 400–416. [CrossRef]