NURail project ID: NURail2016-UIUC-R17

Optimal Planning of Rail Grinding Activities in Large-scale Networks

By

Chao Lei, Ph.D.
Postdoctoral Research Associate
Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign
E-mail: chaolei@illinois.edu

Yanfeng Ouyang, Ph.D.
Professor
Department of Civil and Environmental Engineering
University of Illinois at Urbana Champaign
Email: yfouyang@illinois.edu

and

Siyang Xie, Ph.D.
Research Assistant
Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign
E-mail: sxie13@illinois.edu

17-04-2018

Grant Number: DTRT13-G-UTC52

DISCLAIMER

Funding for this research was provided by the NURail Center, University of Illinois at Urbana - Champaign under Grant No. DTRT13-G-UTC52 of the U.S. Department of Transportation, Office of the Assistant Secretary for Research & Technology (OST-R), University Transportation Centers Program. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

TECHNICAL SUMMARY

Title

Optimal Planning of Rail Grinding Activities in Large-scale Networks

Introduction

In railroads, safety is one of the most important topics and has attracted tremendous attention recently due to some reported accidents. Among all the causes of train accidents in the United States, track defects are one of the leading reasons. Basically, the natural processes of wear and fatigue of rail steel can proceed at a rapid pace that results in track defects and short service lives. Therefore, maintenance of rails (such as grinding, ballast cleaning, ditching) is very important for regular railroad operations.

One unique feature of these rail maintenance activities is that vehicles may experience variable productivity due to both endogenous and exogenous factors, and thus the actual working duration of a maintenance job is unknown to the operator beforehand and needs to be determined. For example, if a track has not been grinded in a timely fashion, its abrasive condition will further accelerate the wear and tear such that more time has to be spent on that track next time to get the job done. The variability of productivity in this case is caused endogenously by the generated grinding schedule. On the other hand, vehicle productivity may also be affected by exogenous reasons such as traffic interference. Take ballast cleaning as an example, the associated vehicles can work for a longer time during a day if the track to be maintained is under traffic curfew.

Practical instances of the rail maintenance routing and scheduling problem (RMRSP) usually involve hundreds or thousands of jobs and an enormous number of complex constraints. Meanwhile, all the routing decisions must be made in a large-scale railroad network. In current rail industry practice, routing and scheduling decisions are mostly manually determined based on the experiences and knowledge of experts. However, such decision-making process is likely to take a long time while the solution quality may still be poor. In light of these complexities, this research focuses on building a comprehensive mathematical model and developing efficient solution approaches for RMRSP under variable productivities.

Approach and Methodology

In this research, we build a mixed-integer model formulation for RMRSP, which involves many complex side constraints for various business requirements from the industry practice. Adding these side constraints introduces significant difficulties in finding a good feasible solution for large-scale industrial applications.

RMRSP is a VRPTW involving routing and scheduling decisions and many complex side constraints. Therefore, it is extremely difficult, if not impossible, to find an optimal solution for large-scale industrial instances via current exact solvers. To effectively handle the complexity and challenges associated with this type of problems, we propose a customized algorithm framework which includes a rolling horizon approach and a stepwise heuristic procedure with embedded mixed-integer programming (MIP) optimization, insertion heuristic, and local search heuristic.

Findings

We test the performance of the proposed model and solution algorithm for two real-world RMRSP instances in a Class I railroad network in North America (with more than 20,000 miles of track length): one is scheduling rail grinders for grinding rail segments, and the other one is scheduling ballast cleaners for a given set of jobs requests. Both instances are used by the Class I railroad company in its real-world operations. These two empirical case studies show that the schedule produced by the proposed approach leads to a higher utilization of the maintenance vehicles compared with current manual solutions, e.g., the improvement in track miles being maintained is 21% for grinders and 77% for ballast cleaners. This saving is very attractive to the railroad company since most of the maintenance vehicles involve large costs.

Conclusions

This research develops a VRPTW-based MIP model to formulate the core component of RMRSP under variable productivities, with many complex side constraints being used to capture various business requirements. A customized stepwise algorithm procedure involving an MIP optimization model, an insertion heuristic, and a local search module, is designed and embedded in a rolling horizon approach framework to effectively solve the problem. A series of numerical instances show that the proposed approach is able to obtain good solutions effectively and efficiently. Two empirical case studies with rail grinding and ballast cleaning applications are also conducted to demonstrate that the proposed solution algorithm outperforms manual solution approach and provide insights and instructions for industrial applications.

Recommendations

The proposed model and solution method have been adopted by a Class I railroad company to provide insights and instructions for their railroad maintenance scheduling and routing applications. Note that besides rail grinding and ballast cleaning, the proposed model and solution approach can be easily extended to other applications in the railroad maintenance context, such as ditching, tie replacement and tamping.

In the future, it would be interesting to study the problem in a dynamic setting, where the routes and schedules can be modified according to the changing environments. This would become particularly important if real-time data and more accurate predictions can be obtained by applying more advanced sensing technologies to detecting the deterioration status of the tracks or ballasts. In addition, it would be meaningful to investigate how the cyclic scheduling of rail maintenance machines can be achieved, where the segments that need to be routinely maintained can be visited repeatedly based on certain frequencies. Given the frequencies might be significantly different from each other, it is easy to imagine that the problem scale would become much larger and more powerful solution approaches such as metaheuristic algorithms need to be developed.

Publications

Xie, S., Lei, C. and Ouyang, Y. (2018) "A customized hybrid approach to infrastructure maintenance scheduling in railroad networks under variable productivities." *Computer-aided Civil and Infrastructure Engineering*. In press.

Primary Contact

Principal Investigator
Yanfeng Ouyang
Professor
Department of Civil and Environmental Engineering
University of Illinois at Urbana Champaign
205 N Mathews Ave, Urbana, IL, 61801
Email: yfouyang@illinois.edu

Other Faculty and Students Involved

Chao Lei
Postdoctoral Research Associate
Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign
Email: chaolei@illinois.edu

Siyang Xie
Research Assistant
Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign
E-mail: sxie13@illinois.edu

NURail Center
217-244-4444
nurail@illinois.edu
http://www.nurailcenter.org/

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1: INTRODUCTION

In railroads, safety is one of the most important topics and has attracted tremendous attention recently due to some reported accidents (Castillo et al., 2016b,a; Wang et al., 2018). Among all the causes of train accidents in the United States, track defects are one of the leading reasons. According to the Federal Railroad Administration Office of Safety Analysis (2017), 27.77% of train accidents happened from January to December 2017 were caused by track defects. Basically, the natural processes of wear and fatigue of rail steel can proceed at a rapid pace that results in track defects and short service lives. Therefore, maintenance of rails is very important for regular railroad operations. The goal of rail maintenance is to achieve the longest possible rail life without increasing the safety risks and costs associated with unanticipated rail failures. In North America, railroad companies spend millions of dollars to perform periodic maintenance jobs, so as to ensure safety and improve operational efficiency.

As an example, grinding of railroad segments has evolved as an effective maintenance technique to control wear artificially and to maintain wheel/rail contact. Rail grinding is usually performed by rail-bound production grinders, which remove metal from the rail surface using rotating grinding wheels (stones). The volume of removed metal would be related to the number, abrasive condition and arrangement of grinding stones on each rail, the pressure being enforced, the forward speed of the machine, and the hardness of the rail surface being worked on. Proper rail grinding controls the rail (and wheel) surface plastic deformation and the rolling contact fatigue cracks on the surface, and it improves truck steering, as well as the dynamic stability of rolling stock.

Another example of railroad maintenance is ballast cleaning. Track ballast, which is typically made of crushed stones, forms the track-bed upon which railroad ties are laid. The purpose of track ballast is not only to bear the load from trains and railroad ties, but also to facilitate drainage of water and prevent vegetation that might interfere with the track structure. However, the ballast may get worn over time, i.e., losing its angularity and generating fine pieces of material, such as sand. Combined with water in the ballast, the rounded stones and fines may stick together and make the ballast like a lump of concrete. Apparently, this would weaken the effectiveness of track drainage as well as the flexibility of ballast in constraining tracks. Therefore, ballast cleaning is conducted to remove the worn-out ballast, screens it and replaces with the fresh one. The cleaning work is performed by rail-bound machines called ballast cleaners. However, given the huge and largely fixed rental and operating costs for ballast cleaners and their unique requirements (e.g., low working speed, weather dependency, etc.), it is essential to come up with an effective routing and scheduling plan such that the utilization of ballast cleaners can be maximized.

One unique feature of the rail maintenance activities considered in this research is that vehicles may experience variable productivity due to both endogenous and exogenous factors, and thus the actual working duration of a maintenance job is unknown to the operator beforehand and needs to be determined. For example, if a track has not been grinded in a timely fashion, its abrasive condition will further accelerate the wear and tear such that more time has to be spent on that track next time to get the job done. The variability of productivity in this case is caused endogenously by the generated grinding schedule. On the other hand, vehicle productivity may also be affected by exogenous reasons such as traffic interference. Take ballast cleaning as an example, the associated vehicles can work for a longer time during a day if the track to be maintained is under traffic curfew. In light of these complexities, this research focuses on building a comprehensive mathematical model and developing efficient solution approaches for the rail maintenance routing and scheduling problem (RMRSP) under variable productivities.

Rail infrastructure maintenance is so important that it has attracted a lot of attentions from both the rail industry and the academic community. Strategic and tactical maintenance planning of railway infrastructures (Su et al., 2017; Xie et al., 2016), rail inspection scheduling (Lannez et al., 2015; Osman et al., 2017; Peng et al., 2013), as well as the coordination of train scheduling and infrastructure maintenance (D'Ariano et al., 2017; Forsgren et al., 2013; Lidén and Joborn, 2017; Luan et al., 2017; Vansteenwegen et al., 2016), have been widely studied in the literature. See Lidén (2015) for a more detailed review of the rail infrastructure maintenance planning problems. However, the study and analysis of the rail track maintenance scheduling problem under variable productivities are still lacking. Peng et al. (2011) and Peng and Ouyang (2012) studied the similar track maintenance scheduling problems in a railroad network, where side constraints such as time window, mutual exclusion, and precedence constraints were considered. They constructed time-space network based models and developed neighborhood search heuristic algorithms to solve the discrete model. However, since the working durations, as well as travel times, are continuous, the planning horizon of RMRSP cannot be discretized such that it is impossible to formulate RMRSP into a time-space network model. The most related work to our research is conducted by Peng et al. (2013), who studied a rail inspection scheduling problem where each task needs to be routed and scheduled periodically within a continuous planning horizon. However, we notice that the working durations of tasks in RMRSP might be unknown and can be affected by the times that they are scheduled (e.g., a machine can work in a higher efficiency during curfews), rather than being fixed in the rail inspection scheduling problem. Therefore, different model and algorithms need to be specifically developed for RMRSP.

Since one of the primary purposes of RMRSP is to find the optimal route to move maintenance vehicles over the rail network to perform maintenance jobs, it shares great similarities with the well-known vehicle routing problem (VRP) (Toth and Vigo, 2014), where each maintenance job can be considered as a vertex of the network. In addition, since RMRSP involves scheduling decisions in the time dimension and the start time of a job is usually subject to certain time restrictions, it resembles the so-called VRP with time windows (VRPTW). VRPTW is nondeterministic polynomial (NP) hard since it can be easily reduced to a typical VRP. In fact, it has been proved that even finding a feasible solution to VRPTW for a fixed number of vehicles is already an NP-complete problem (Savelsbergh, 1985). Even though many exact solution methods (e.g., branch-and-cut (Kohl et al., 1999; Bard et al., 2002; Kallehauge et al., 2007), branch-and-price (Desrochers et al., 1992; Lysgaard, 2006)) have been proposed for VRPTW, most of the literatures focus on designing heuristics (Bräysy and Gendreau, 2005a) and meta-heuristics (Bräysy and Gendreau, 2005b) to find good feasible solutions quickly. A thorough survey of mathematical models and solution methods to solve VRPTW can be found in Desaulniers et al. (2014). It should be noted that RMRSP is even more complex than typical

VRPTW, since various types of side constraints (e.g., mutual exclusion constraints, various time windows constraints) must be considered in RMRSP in addition to the traditional routing and time window constraints in VRPTW.

One may also notice that RMRSP shares some similarities with the periodic vehicle routing problem (PVRP) (Christofides and Beasley, 1984; Gaudioso and Paletta, 1992; Chao et al., 1995; Cordeau et al., 1997; Francis et al., 2008), where customers are visited periodically over multiple time periods according to a certain set of schedules, and different routes are constructed in different time periods. Francis et al. (2006) later studied PVRP with service choices, which allow customers' visit frequencies to be decision variables. The readers can refer to Campbell and Wilson (2014) for a comprehensive review of PVRP. However, it should be noted that there is a major difference between our RMRSP and PVRP, i.e., the planning horizon for RMRSP is continuous and cannot be discretized. Since the work duration for a maintenance job can be a few minutes or several weeks, it may be unrealistic to define the length for a single period. Moreover, the vehicles in RMRSP follow a continuous trip throughout the planning horizon, which means that a vehicle resumes service right from where it locates by the end of the previous working day, rather than having to start and end at the depot as in PVRP.

Practical instances of RMRSP usually involve hundreds or thousands of jobs and an enormous number of complex constraints. Meanwhile, all the routing decisions must be made in a large-scale railroad network. In current rail industry practice, routing and scheduling decisions are mostly manually determined based on the experiences and knowledge of experts (Peng et al., 2013). However, such decision-making process is likely to take a long time while the solution quality may still be poor. In this research, we build a mixed-integer model formulation for RMRSP, which involves many complex side constraints for various business requirements from the industry practice. Adding these side constraints introduces significant difficulties in finding a good feasible solution for large-scale industrial applications. Hence, we propose a customized algorithm framework which includes a rolling horizon approach and a stepwise heuristic procedure with embedded mixed-integer programming (MIP) optimization, insertion heuristic, and local search heuristic. A series of numerical instances show that the proposed approach is able to obtain good solutions effectively and efficiently. Two empirical case studies with rail grinding and ballast cleaning applications are also conducted to demonstrate that the proposed solution algorithm outperforms manual solution approach and provide insights and instructions for industrial applications. Note that besides rail grinding and ballast cleaning, the proposed model and solution approach can be easily extended to other applications in the railroad maintenance context, such as ditching, tie replacement and tamping.

The remainder of the report is organized as follows. Section 2 presents a VRPTW based core model formulation for RMRSP with multiple sets of side constraints. Section 3 proposes our customized heuristic algorithm. Section 4 demonstrates the proposed model and algorithms with a series of numerical instances and two real-world case studies and draws managerial insights. Section 5 summarizes the report and discusses possible future extensions.

SECTION 2: MODEL FORMULATION

In this section, we present a mixed-integer optimization model for RMRSP. We first formulate the core component of RMRSP as a variant of VRPTW and then introduce various types of side constraints.

2.1. VRPTW-based Core Model

Consider a rail network with a set of segments, denoted by $\mathscr{M}$, where each segment $m \in \mathscr{M}$ is expected to be maintained periodically (with frequency F_m) every Γ_m units of time.[1] Specifically, for each segment $m \in \mathscr{M}$, every time a vehicle finishes performing a maintenance activity on it, another maintenance activity is expected to be scheduled for it after the desired headway Γ_m. As such, each segment is likely to be maintained for $N_m \geq 1$ times within the entire planning horizon, and we denote the set of all expected maintenance activities of segment m as $\mathscr{N}_m = \{1,2,\ldots,N_m\}$. We call each maintenance activity for a segment as a *job*. Let $\mathscr{I}$ denote the set of all maintenance jobs that need to be performed in the planning horizon, a job $i \in \mathscr{I}$ is defined as (m_i, n_i), where $m_i \in \mathscr{M}$ denotes the segment corresponding to job i and $n_i \in \mathscr{N}_{m_i}$ indicates the maintenance activity of segment m_i. We further define $\mathscr{I}_m$ as the set of jobs associated with segment $m \in \mathscr{M}$. Each maintenance job $i \in \mathscr{I}$ is associated with the track length L_i and priority value P_i of its segment m_i. We define decision variable $z_i \in \{0,1\}$ to indicate whether job i is performed or not. Each job i can only be conducted by one vehicle. Denote $\mathscr{K}$ as the set of maintenance vehicles, we further let $z_i^k \in \{0,1\}$ denote whether job i is performed by vehicle $k \in \mathscr{K}$ or not. If job i is not performed by any vehicle, a penalty of $W_{\text{job}} L_i$ is incurred, where W_{job} denote the penalty per mile.

Since a job i of RMRSP typically involves a railroad segment with non-negligible length L_i, the direction of the machine movement actually matters. We assume that the railroad segment m_i corresponding to job i has two ends E_i^1 and E_i^2. Let $\mathscr{D}_m = \{d_1^m, d_2^m, d_3^m, d_4^m\}$ be the set of directions a vehicle could move along between any two consecutive jobs i and j, where $d_1^m = E_i^2 \to E_j^1, d_2^m = E_i^2 \to E_j^2, d_3^m = E_i^1 \to E_j^1, d_4^m = E_i^1 \to E_j^2$. We define decision variable $x_{i,j}^{k,d}$ to denote whether vehicle k moves from job i to j in direction $d \in \mathscr{D}$, and let $T_{i,j}^{k,d}$ denote the corresponding travel time. For simplicity of modeling, we assume a single virtual vehicle depot $\{0\}$ for all vehicles, with zero travel distance to any point in the railroad network. Similarly, we define $x_{0,i}^{k,d_1}$ and $x_{i,0}^{k,d_2}$ to indicate whether vehicle k reaches job i from and leaves job i to the depot in directions d_1 and d_2, respectively. Here the direction set for moving from the depot to the first job is $\mathscr{D}_f = \{d_1^f, d_2^f\}$ with $d_1^f = 0 \to E_i^1, d_2^f = 0 \to E_i^2$, and the direction set for arriving at the depot from the last job is $\mathscr{D}_l = \{d_1^l, d_2^l\}$ with $d_1^l = E_i^2 \to 0, d_2^l = E_i^1 \to 0$. Given that the vehicles require fuel and drivers to function, the travel cost is considered in this scheduling problem as well, which is assumed to be proportional to the travel time, with penalty weight as W_{travel} per time unit.

[1]The railroad company determines the division of its network into segments, and the frequencies at which these segments should be maintained. These decisions are given as inputs to the proposed model.

Moreover, each job i requires S_i^k time for vehicle k to finish, and has a preferred maintaining time depending on the last time the segment was maintained and Γ_m. Either being early or late per time unit from the preferred time incurs a penalty of $W_{\text{time}}P_i$, where W_{time} is the corresponding penalty weight per unit time. For each segment m, its last maintenance activity is at time W_m^0, thus the preferred time for its first expected maintenance activity is $W_m^0 + \Gamma_m$. We denote w_i^k as the start time for vehicle k to perform job i, and u_i and v_i as the number of time units that job i is late and early, respectively. Let $C_{\text{cost item}}$ denote the costs for some other items (those related to side constraints, which will be introduced in next section). The core VRPTW model can be formulated as follows:

(RMRSP)

$$\min \sum_{i \in \mathscr{I}} \Big(W_{\text{job}} \alpha_{\text{job}} L_i (1 - z_i) + W_{\text{time}} \alpha_{\text{time}} P_i (u_i + v_i)$$

$$+ W_{\text{travel}} \alpha_{\text{travel}} \sum_{j \in \mathscr{I}} \sum_{d \in \mathscr{D}_m} \sum_{k \in \mathscr{K}} T_{i,j}^{k,d} x_{i,j}^{k,d} \Big) + \sum_{\text{other cost items}} C_{\text{cost item}} \tag{2.1a}$$

$$\text{s.t.} \sum_{k \in \mathscr{K}} z_i^k = z_i, \ \forall i \in \mathscr{I}, \tag{2.1b}$$

$$\sum_{j \in \mathscr{I}} \sum_{d \in \mathscr{D}_m} x_{i,j}^{k,d} + \sum_{d \in \mathscr{D}_l} x_{i,0}^{k,d} = z_i^k, \ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.1c}$$

$$z_i \le z_j, \ \forall i, j \in \mathscr{I}_m, n_i > n_j, m \in \mathscr{M}, \tag{2.1d}$$

$$\sum_{j \in \mathscr{I}} \sum_{d \in \{d_1^m, d_2^m\}} x_{i,j}^{k,d} + x_{i,0}^{k,d_1^l} = \sum_{j \in \mathscr{I}} \sum_{d \in \{d_1^m, d_3^m\}} x_{j,i}^{k,d} + x_{0,i}^{k,d_1^f}, \ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.1e}$$

$$\sum_{j \in \mathscr{I}} \sum_{d \in \{d_3^m, d_4^m\}} x_{i,j}^{k,d} + x_{i,0}^{k,d_2^l} = \sum_{j \in \mathscr{I}} \sum_{d \in \{d_2^m, d_4^m\}} x_{j,i}^{k,d} + x_{0,i}^{k,d_2^f}, \ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.1f}$$

$$\sum_{j \in \mathscr{I}} \sum_{d \in \mathscr{D}_f} x_{0,j}^{k,d} = \sum_{i \in \mathscr{I}} \sum_{d \in \mathscr{D}_l} x_{i,0}^{k,d} = 1, \ \forall k \in \mathscr{K}, \tag{2.1g}$$

$$\sum_{d \in \mathscr{D}} x_{i,j}^{k,d} \le 1, \ \sum_{d \in \mathscr{D}_f} x_{0,j}^{k,d} \le 1, \ \sum_{d \in \mathscr{D}_l} x_{i,0}^{k,d} \le 1, \ \forall i \in \mathscr{I}, j \in \mathscr{I}, k \in \mathscr{K}, \tag{2.1h}$$

$$w_i^k + s_i^k + T_{i,j}^{k,d} \le w_j^k + A_{i,j}^{k,d} \left(1 - x_{i,j}^{k,d}\right), \ \forall i \in \mathscr{I}, j \in \mathscr{I}, d \in \mathscr{D}_m, k \in \mathscr{K}, \tag{2.1i}$$

$$s_i^k = (1 + \gamma P_i u_i) S_i^k, \ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.1j}$$

$$w_i^k - W_{m_i}^0 - \Gamma_{m_i} \le u_i + B_i^k (1 - z_i^k), \ \forall i \in \mathscr{I}, n_i = 1, k \in \mathscr{K}, \tag{2.1k}$$

$$w_i^k - W_{m_i}^0 - \Gamma_{m_i} \ge -v_i - B_i^k (1 - z_i^k), \ \forall i \in \mathscr{I}, n_i = 1, k \in \mathscr{K}, \tag{2.1l}$$

$$w_i^k - w_{i'}^{k'} - \Gamma_{m_i} \le u_i + B_i^k (1 - z_i^k) + B_{i'}^{k'} (1 - z_{i'}^{k'}),$$
$$\forall k, k' \in \mathscr{K}, i \in \mathscr{I}, n_i > 1, i' = (m_i, n_i - 1), \tag{2.1m}$$

$$w_i^k - w_{i'}^{k'} - \Gamma_{m_i} \ge -v_i - B_i^k (1 - z_i^k) - B_{i'}^{k'} (1 - z_{i'}^{k'}),$$
$$\forall k, k' \in \mathscr{K}, i \in \mathscr{I}, n_i > 1, i' = (m_i, n_i - 1), \tag{2.1n}$$

$$x_{i,j}^{k,d_1}, x_{i,0}^{k,d_2}, x_{0,j}^{k,d_3}, z_i, z_i^k \in \{0, 1\}, w_i^k, u_i, v_i \ge 0,$$
$$\forall i, j \in \mathscr{I}, k \in \mathscr{K}, d_1 \in \mathscr{D}_m, d_2 \in \mathscr{D}_l, d_3 \in \mathscr{D}_f, \tag{2.1o}$$

$$A_{i,j}^{k,d}, B_i^k, \forall i \in \mathscr{I}, j \in \mathscr{I}, d \in \mathscr{D}_m, k \in \mathscr{K}. \tag{2.1p}$$

The objective function (2.1a) minimizes the weighted summation of various costs/penalties, which consist of four components: (i) the penalty for not performing jobs at all; (ii) the cost for traveling; (iii) the penalty for not performing jobs at their preferred times; and (iv) other cost items due to various side constraints. Constraints (2.1b) and (2.1c) ensure that if job i is performed, it should be visited by exactly one vehicle in one direction d. Constraints (2.1d) enforce that later jobs of one segment should occur after corresponding former jobs. Constraints (2.1e) - (2.1g) enforce flow conservation for regular jobs and the virtual depot. Constraints (2.1h) enforce that each job is only possible to be performed in one direction by one vehicle. Constraints (2.1i) establish relationships between the vehicle routes and the job performing times, which are also capable of eliminating subtours possibility. Constraints (2.1j) define the relationship between actual working time and regular working time for maintaining a particular job, due to the variability of productivity. Constraints (2.1k) - (2.1n) compute the number of time units that each job is performed early or late from its preferred time. Constraints (2.1o) define binary and nonnegative variables while constraints (2.1p) indicate those big values used in different constraints.

2.2. Side Constraints

In this section, we will introduce various types of important industrial considerations and business requirements. We design a customized approach to formulate them into several sets of side constraints in the following subsections.

2.2.1. Hard/Soft Time Windows Constraints

As we introduced before, each maintenance job $i \in \mathscr{I}$ has a preferred performing time, and we try not to be late or early to avoid penalty. For example, if a grinding job is performed too late, a railroad segment may face the risk of breaking down and causing severe blockage, while if it is conducted too early, the abrasive condition of the segment is still good and we are wasting resources to grind the segment unnecessarily. Given that different jobs have different priorities, the whole set of jobs can be classified into two sets $\mathscr{I}_{\text{hard}}$ and $\mathscr{I}_{\text{soft}}$. For those maintenance jobs in $\mathscr{I}_{\text{hard}}$ (we call them "hard" jobs), they are assigned high priorities and are forced to be started within certain "hard" time windows. For other maintenance jobs in $\mathscr{I}_{\text{soft}}$ (we call them "soft" jobs), they are assigned with relatively low priorities and are allowed to be started outside the given preferred "soft" time windows at certain penalty cost. The hard and soft time windows constraints can be formulated as follows:

$$C_{\text{HStw}}(\mathbf{u}', \mathbf{v}') = \sum_{i \in \mathscr{I}_{\text{soft}}} W_{\text{time}}^{\text{soft}} P_i (u_i' + v_i') \tag{2.2a}$$

$$u_i \leq U_i, \ \forall i \in \mathscr{I}_{\text{hard}}, \tag{2.2b}$$

$$v_i \leq V_i, \ \forall i \in \mathscr{I}_{\text{hard}}, \tag{2.2c}$$

$$u_i \leq U_i + u_i', \ \forall i \in \mathscr{I}_{\text{soft}}, \tag{2.2d}$$

$$v_i \leq V_i + v_i', \ \forall i \in \mathscr{I}_{\text{soft}}, \tag{2.2e}$$

where C_{HStw} is the total penalty costs incurred by those jobs in $\mathscr{I}_{\text{soft}}$ if they are performed outside their soft time windows. U_i and V_i are the maximum time units that job i could be late and early if it is a "hard" job, and u_i' and v_i' are the time units that "soft" job i is actually late and early beyond its preferred time windows, respectively. Constraints (2.2b)-(2.2c) ensure that the "hard" jobs can only be performed within their hard time windows, while (2.2d)-(2.2e) define how long the "soft" jobs are performed outside their soft time windows.

2.2.2. Preference Time Windows Constraints

Preference time windows constraints would push a job to be performed inside certain preferred time windows. These constraints differ from the hard/soft time windows constraints in that jobs are allowed to be partially performed within the preferred time windows and introduce proportional benefits. Such constraints should be incorporated, for example, if a ballast cleaning job is worked inside its curfew periods when maintenance vehicles can work with higher efficiencies. Define the set of preference time windows as $\mathscr{P}$, and the start and end time of the time window $p \in \mathscr{P}$ are P_p^{start} and P_p^{end}, respectively. We further define the set of jobs that have preference time window $p \in \mathscr{P}$ as $\mathscr{I}_p$. The time preference constraints can be written out as follows:

$$C_{\text{Pref}}(r) = -\sum_{i \in \mathscr{I}_p} \sum_{k \in \mathscr{K}} \sum_{p \in \mathscr{P}} C_{\text{Pref}}^{i,k} t_{ip}^k \tag{2.3a}$$

$$w_i^k + s_i^k - t_{ip}^k \geq P_p^{\text{start}} - A_{ip}^k \left(2 - z_i^k - q_{ip}^k\right), \ \forall i \in \mathscr{I}_p, k \in \mathscr{K}, p \in \mathscr{P}, \tag{2.3b}$$

$$w_i^k + t_{ip}^k \leq P_p^{\text{end}} + A_{ip}^k \left(2 - z_i^k - q_{ip}^k\right), \ \forall i \in \mathscr{I}_p, k \in \mathscr{K}, p \in \mathscr{P}, \tag{2.3c}$$

$$s_i^k \geq S_i^k - (1-\beta) \sum_{p \in \mathscr{P}} t_{ip}^k - A_i^k \left(1 - z_i^k\right), \ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.3d}$$

$$s_i^k \leq S_i^k - (1-\beta) \sum_{p \in \mathscr{P}} t_{ip}^k + A_i^k \left(1 - z_i^k\right), \ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.3e}$$

$$q_{ip}^k \in \{0,1\}, t_{ip}^k \geq 0, \forall i \in \mathscr{I}_p, k \in \mathscr{K}, p \in \mathscr{P}, \tag{2.3f}$$

where β is the percentage of increased efficiency if machines work within preferred time windows (such as curfews), $t_{ip}^k \geq 0$ is the length of time that vehicle $k \in \mathscr{K}$ works on job $i \in \mathscr{I}_p$ in preferred time window $p \in \mathscr{P}$. Furthermore, we define $q_{ip}^k \in \{0,1\}, \forall i \in \mathscr{I}_p, k \in \mathscr{K}, p \in \mathscr{P}$, where q_{ip}^k equals to 1 if vehicle k would work on job i for t_{ip}^k length of time in time window $p \in \mathscr{P}$. $C_{\text{Pref}}(\cdot)$ in (2.3a) is the total time preference constraints penalty cost, where $C_{\text{Pref}}^{i,k}$ is the cost factor associated with job i and vehicle k. Constraints (2.3b) and (2.3c) are the time conservation constraints when vehicle can perform part of a job within its preferred time windows. Constraints (2.3d) and (2.3e) are the formulas for the relationship between s_i^k and t_{ip}^k.

2.2.3. Repulsive Time Windows Constraints

Repulsive time windows constraints are enforced when a maintenance job cannot be worked during certain time windows. Weather condition might be the most common reason for having repulsive time windows constraints in maintenance job scheduling. In addition, many maintenance jobs require that the associated rail tracks must be in dry condition for maintenance. Therefore, it is ideal that the vehicles are not working in an area during its rainy or snowy seasons. For example, ballast cleaning jobs cannot be performed when the air temperature is too low or when the rail tracks are covered by snow. Hence, some ballast cleaning jobs in the northern areas should not be scheduled in winter. Define the set of repulsive time windows as $\mathscr{R}$, the start and end time of the time window $r \in \mathscr{R}$ are R_r^{start} and R_r^{end}, respectively. Given that the set of jobs that should not be performed during $r \in \mathscr{R}$ is denoted as $\mathscr{I}_r$, we can write out the repulsive time windows constraints as follows:

$$w_i^k + S_i^k \le R_r^{\text{start}} + D_{ir}^k \left(1 - z_i^k + p_i^r\right), \ \forall k \in \mathscr{K}, r \in \mathscr{R}, i \in \mathscr{I}_r, \tag{2.4a}$$

$$w_i^k \ge R_r^{\text{end}} - D_{ir}^k \left(2 - z_i^k - p_i^r\right), \ \forall k \in \mathscr{K}, r \in \mathscr{R}, i \in \mathscr{I}_r, \tag{2.4b}$$

$$p_i^r \in \{0,1\}, k \in \mathscr{K}, r \in \mathscr{R}, \forall i \in \mathscr{I}_r, \tag{2.4c}$$

where $p_i^r \in \{0,1\}$ indicates whether job $i \in \mathscr{I}_r$ is performed before or after the repulsive time window r. Constraints (2.4a) ensure that if $p_i^r = 0$ the job $i \in \mathscr{I}_r$ is finished before the repulsive time window, while constraints (2.4b) force job i to be started after the repulsive time window if $p_i^r = 1$.

2.2.4. Mutual Exclusion Constraints

Mutual exclusion constraints require that certain pairs of jobs not to be performed simultaneously due to real-world preferences or restrictions. For example, those grinding jobs that are in the same division of the railroad network or involve the same railroad employees/resources (e.g., roadmasters) cannot be performed at the same time. Basically, all the maintenance jobs can be classified into multiple groups based on some particular criteria, i.e., $\mathscr{I} = \cup_{g \in \mathscr{G}} \mathscr{I}_g$, such that any two jobs in the same group $\mathscr{I}_g$ shall not be performed simultaneously. The mutual exclusion constraints can be formulated as follows:

$$w_{i_1}^{k_1} + S_{i_1}^{k_1} \le w_{i_2}^{k_2} + C_{i_1,i_2}^{k_1,k_2} \left(3 - y_{i_1,i_2} - z_{i_1}^{k_1} - z_{i_2}^{k_2}\right), \ \forall i_1 \ne i_2 \in \mathscr{I}_g, g \in \mathscr{G}, k_1, k_2 \in \mathscr{K}, \tag{2.5a}$$

$$w_{i_2}^{k_2} + S_{i_2}^{k_2} \le w_{i_1}^{k_1} + C_{i_1,i_2}^{k_1,k_2} \left(2 + y_{i_1,i_2} - z_{i_1}^{k_1} - z_{i_2}^{k_2}\right), \ \forall i_1 \ne i_2 \in \mathscr{I}_g, g \in \mathscr{G}, k_1, k_2 \in \mathscr{K}, \tag{2.5b}$$

$$y_{i_1,i_2} \in \{0,1\}, \ \forall i_1, i_2 \in \mathscr{I}, \tag{2.5c}$$

where y_{i_1,i_2} indicates whether job i_1 is performed before job i_2 or not. If $y_{i_1,i_2} = z_{i_1}^{k_1} = z_{i_2}^{k_2} = 1$, i.e., jobs i_1 and i_2 belong to the same group and i_1 is performed by vehicle k_1 before i_2 is performed by vehicle k_2, constraints (2.5a) is effective by forcing i_1 to be finished before i_2 is started. When $y_{i_1,i_2} = 0$ and $z_{i_1}^{k_1} = z_{i_2}^{k_2} = 1$, constraints (2.5b) become effective and require that job i_2 is finished before i_1 is started.

2.2.5. *Precedence Constraints*

The precedence constraints state that certain jobs must be performed in a certain sequence. For example, a rail grinding or ballast cleaning job should always be performed before a corresponding tie job, so as to avoid the possible damage that grinding might bring to the new tie.

Let $\mathscr{G}_{\text{hard}}$ and $\mathscr{G}_{\text{soft}}$ be the sets of all pairs of jobs that should follow the hard and soft precedence constraints, respectively. For any pair $(i_1, i_2) \in \mathscr{G}_{\text{hard}}$, job i_1 is required to be finished at least T_{i_1,i_2} before job i_2 starts, while for any pair $(i_1, i_2) \in \mathscr{G}_{\text{soft}}$, job i_1 is preferred to be finished at least T_{i_1,i_2} before job i_2 starts, and violating this preference incurs penalty $c_{\text{Prec}}^{i_1,i_2}$. The precedence constraints can be formulated as follows:

$$C_{\text{Prec}}(\mathbf{y}) = \sum_{(i_1,i_2)\in\mathscr{G}_{\text{soft}}} c_{\text{Prec}}^{i_1,i_2} y_{i_1,i_2} \tag{2.6a}$$

$$\sum_{k_1\in\mathscr{K}} \left(w_{i_1}^{k_1} + S_{i_1}^{k_1} z_{i_1}^{k_1}\right) + T_{i_1,i_2} \le \sum_{k_2\in\mathscr{K}} w_{i_2}^{k_2} + A_{i_1,i_2}\left(2 - z_{i_1} - z_{i_2}\right), \forall (i_1,i_2) \in \mathscr{G}_{\text{hard}}, \tag{2.6b}$$

$$\begin{aligned}\sum_{k_1\in\mathscr{K}} \left(w_{i_1}^{k_1} + S_{i_1}^{k_1} z_{i_1}^{k_1}\right) + T_{i_1,i_2} \le \sum_{k_2\in\mathscr{K}} w_{i_2}^{k_2}\\ + A_{i_1,i_2}\left(2 + y_{i_1,i_2} - z_{i_1} - z_{i_2}\right),\ \forall (i_1,i_2) \in \mathscr{G}_{\text{soft}},\end{aligned} \tag{2.6c}$$

$$y_{i_1,i_2} \in \{0,1\},\ \forall i_1, i_2 \in \mathscr{I}, \tag{2.6d}$$

where C_{Prec} is the total penalty associated with the precedence constraints, $c_{\text{Prec}}^{i_1,i_2}$ is the penalty for pair (i_1,i_2) violating the soft precedence constraints, and y_{i_1,i_2} indicates whether job i_1 is finished before job i_2 is started or not. (2.6b) and (2.6c) imply the hard and soft precedence constraints, respectively.

2.2.6. *Maintenance Constraints*

Maintenance constraints require that for each vehicle $k \in \mathscr{K}$ we reserve a certain period of downtime for maintenance. During the maintenance period, the vehicles are clearing and must be stored on track away from mainline. Typically, maintenance crews go on duty and take on water, fuel, oil, and materials to maintain the working vehicles.

For simplicity of modeling, we construct a dummy job for each vehicle k to represent the maintenance, with MT_k being the length of the associated maintenance time length. We then formulate the maintenance constraints as follows:

$$w_i^k + \text{MT}_k \cdot m_i^k + S_i^k + T_{i,j}^{k,d} \le w_j^k + A_{i,j}^{k,d}\left(1 - x_{i,j}^{k,d}\right),\ \forall i \in \mathscr{I}, j \in \mathscr{I}, k \in \mathscr{K}, d \in \mathscr{D}, \tag{2.7a}$$

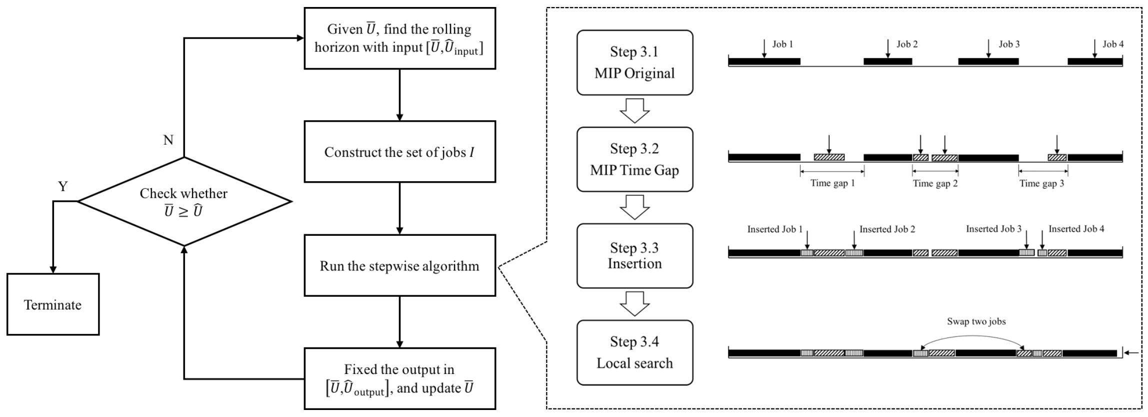
$$m_i^k \le z_i^k,\ \forall i \in \mathscr{I}, k \in \mathscr{K}, \tag{2.7b}$$

$$\sum_{i\in\mathscr{I}} m_i^k = 1,\ \forall k \in \mathscr{K}, \tag{2.7c}$$

where m_i^k indicates whether the maintenance period for vehicle k follows right after finishing job i in its schedule. (2.7a) redefine the relationship between the performing times of the two jobs that are before and after the maintenance period. (2.7b) ensure that the maintenance period for vehicle k comes after a job i only if job i is maintained by vehicle k. (2.7c) require that each vehicle is scheduled for a maintenance exactly once during the scheduling horizon.

SECTION 3: SOLUTION ALGORITHM

RMRSP is a VRPTW involving routing and scheduling decisions and many complex side constraints. Therefore, it is extremely difficult, if not impossible, to find an optimal solution for large-scale industrial instances via current exact solvers. To effectively handle the complexity and challenges associated with this type of problems, we propose a customized algorithm framework based on a rolling horizon approach and a stepwise heuristic procedure, in which the model formulation presented in Section 2 is adjusted and embedded as a module. The overall rolling horizon algorithm framework is shown as Figure 3.1.



Given $\overline{U}$, find the rolling horizon with input $[\overline{U}, \widehat{U}_{\text{input}}]$
Construct the set of jobs I
Run the stepwise algorithm
Fixed the output in $[\overline{U}, \widehat{U}_{\text{output}}]$, and update $\overline{U}$
Check whether $\overline{U} \geq \widehat{U}$
N
Y
Terminate
Step 3.1 MIP Original
Step 3.2 MIP Time Gap
Step 3.3 Insertion
Step 3.4 Local search
Job 1
Job 2
Job 3
Job 4
Time gap 1
Time gap 2
Time gap 3
Inserted Job 1
Inserted Job 2
Inserted Job 3
Inserted Job 4
Swap two jobs

Figure 3.1: Rolling horizon algorithm framework and the embedded stepwise heuristic procedure.

3.1. Rolling Horizon

Suppose that the entire scheduling horizon of RMRSP is $[0, \widehat{U}]$, considering the "periodicity" requirement for maintaining railroad segments, it is very complicated to solve RMRSP for the entire scheduling horizon at once because the preferred maintaining time T_i for many jobs are unknown beforehand. For example, suppose the entire scheduling horizon is 6 months (i.e., $\widehat{U} = 6$ in the unit of month) and a track segment needs maintenance activity every 4 months. If this segment is scheduled at time $t_1 = 1$ for the first maintenance activity, then it is expected to be scheduled for another activity at $t_2 = 5$ (otherwise a penalty will be incurred). If however, the segment is scheduled at time $t_1 = 3$ for the first maintenance activity, then there is no need to schedule any other maintenance activity during the scheduling horizon. Therefore, to address the periodicity requirements, we adopt a rolling horizon approach to decompose the problem in the entire scheduling horizon into multiple smaller versions in shorter horizons. The basic procedure of the approach is shown in Figure 3.1 and can be described as follows:

Step 1: Initially, for each railroad track segment $m \in \mathscr{M}$, we generate a maintenance job i_m with preferred maintaining time $T_{i_m} = W_m^0 + \Gamma_m$. In this way, we construct a set of maintenance jobs as $\mathscr{I}' = \{i_m, \forall m \in \mathscr{M}\}$;

Step 2: Given the start of the current scheduling horizon $\overline{U}$ (initially set as 0 in the first step), we generate the input horizon as $[\overline{U}, \widehat{U}_{\text{input}}]$ with $\widehat{U}_{\text{input}} = (n-1)\Delta U_{\text{output}} + \Delta U_{\text{input}}$ and the output horizon as $[\overline{U}, \widehat{U}_{\text{output}}]$ with $\widehat{U}_{\text{output}} = n\Delta U_{\text{output}}$, where ΔU_{input} is the length of the horizon for considering input segments and ΔU_{output} is the length of the horizon for obtaining scheduling outputs, i.e., we only consider the set of maintenance jobs $\mathscr{I}_{\text{input}}$ with preferred time $T_i \in [\overline{U}, \widehat{U}_{\text{input}}], \forall i \in \mathscr{I}_{\text{input}}$, and we only fix into the solution the set of maintenance jobs $\mathscr{I}_{\text{output}}$ with performing time $w_i^k \in [\overline{U}, \widehat{U}_{\text{output}}], \forall i \in \mathscr{I}_{\text{output}}$. Typically ΔU_{ouput} is smaller than ΔU_{input} so as to allow some flexibility in the scheduling. Figure 3.2 illustrates how the rolling horizon evolves and proceeds over time;

Step 3: We then solve the RMRSP model with respect to $\mathscr{I}_{\text{input}}$. It is worthy noting that Constraints (2.1d), (2.1m)-(2.1n) are no longer required since $|\mathscr{I}_m| = 1, \forall m \in \mathscr{M}$, which significantly reduces the model complexity. From the solution, we take and fix the schedules for those jobs $\mathscr{I}_{\text{output}}$ that are started within $[\overline{U}, \widehat{U}_{\text{output}}]$;

Step 4: We calculate $\overline{U}_k$ as the finish time of the latest job that has already been performed by vehicle $k \in \mathscr{K}$ within $[\overline{U}, \widehat{U}_{\text{output}}]$, and update $\overline{U} = \min_{k \in \mathscr{K}}\{\overline{U}_k\}$. If $\overline{U}$ reaches the end of the scheduling horizon, i.e., $\overline{U} \geq \widehat{U}$, we terminate; otherwise, we go back to Step 1.

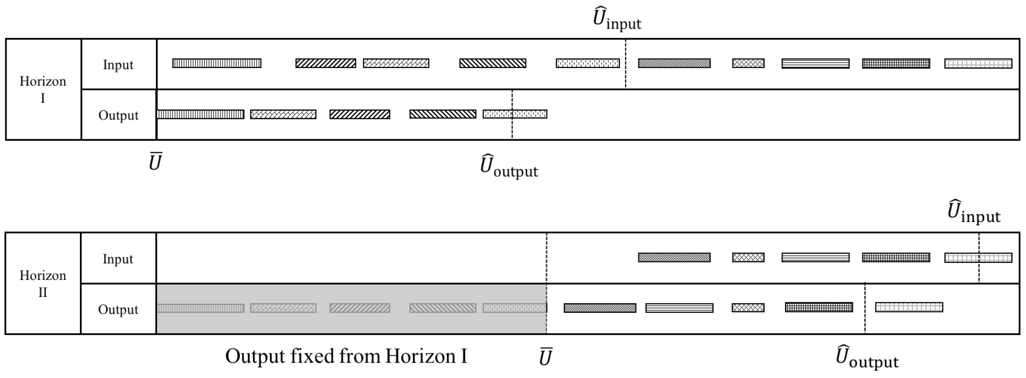

$\widehat{U}_{\text{input}}$
Horizon I
Input
Output
$\overline{U}$
$\widehat{U}_{\text{output}}$
$\widehat{U}_{\text{input}}$
Horizon II
Input
Output
Output fixed from Horizon I
$\overline{U}$
$\widehat{U}_{\text{output}}$

Figure 3.2: Illustration of the rolling horizon algorithm.

3.2. *Stepwise Heuristic Procedure*

In Step 3 of the rolling horizon approach, we solve our VRPTW model with respect to the job set $\mathscr{I}_{\text{input}}$. However, if the number of segments in the railroad network is large, $\mathscr{I}_{\text{input}}$ could still potentially involve many segments, and make the problem size difficult to be simply handled by solvers. Therefore, we further decompose step 3 in the rolling horizon method into the following 4 sub-steps. The proposed stepwise heuristic procedure is also illustrated in Figure 3.1.

Step 3.1: *MIP Original*: we set thresholds $\overline{L}$ and $\overline{P}$ to define the set of "important" jobs as $\mathscr{I}_{\text{important}} = \{i \in \mathscr{I}_{\text{input}} : L_i \geq \overline{L} \text{ or } P_i \geq \overline{P}\}$, and the set of "unimportant" jobs as $\mathscr{I}_{\text{unimportant}} = \mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{important}}$, where $\overline{L}$ and $\overline{P}$ denote the lower thresholds of the track length and the priority value for a segment to be important, respectively. In this way, the total track length of segments that are considered only slightly reduces while the number of segments incorporated can be decreased significantly, leading to a much simpler model.[2] We then solve our MIP formulation with respect to set $\mathscr{I}_{\text{important}}$ to obtain an initial solution, and denote the set of jobs (in the order of performing time) that are scheduled for vehicle $k \in \mathscr{K}$ in the solution as $\mathscr{I}^k_{\text{scheduled}}$, and $\mathscr{I}_{\text{scheduled}} = \cup_{k \in \mathscr{K}} \mathscr{I}^k_{\text{scheduled}}$;

Step 3.2: *MIP TimeGap*: in the initial solution obtained in Step 3.1, if there exist large time gaps (e.g., larger than a certain number of days), say we observe a time gap $[\overline{U}_{\text{gap}}, \widehat{U}_{\text{gap}}]$, we run our MIP model with respect to those jobs in $\mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{scheduled}}$ with preferred performing times falling in $[\overline{U}_{\text{gap}} - \Delta U_{\text{gap}}, \widehat{U}_{\text{gap}} + \Delta U_{\text{gap}}]$ to fill the gap. Here ΔU_{gap} is set to include more jobs with preferred performing times outside but close to the time gap $[\overline{U}_{\text{gap}}, \widehat{U}_{\text{gap}}]$. We finally combine the solutions for all the time gaps with the Step 3.1 initial solution and update $\mathscr{I}^k_{\text{scheduled}}, \forall k \in \mathscr{K}$ and $\mathscr{I}_{\text{scheduled}}$;

Step 3.3: *Insertion*: in the solution obtained in Step 3.2, for any remaining time gap between a pair of consecutive jobs, we try to insert those jobs in $\mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{scheduled}}$ using an insertion heuristic and update $\mathscr{I}_{\text{scheduled}}$. The heuristic will be described in details in Section 3.3;

Step 3.4: *Local search*: we use a local search heuristic algorithm to improve the solution $\mathscr{I}_{\text{scheduled}}$ obtained in Step 3.3. This step will be described in more details in Section 3.4. Finally, we move all scheduled jobs in $\mathscr{I}_{\text{scheduled}}$ forward in terms of performing time to further close any existing time gap. This may leave a small time gap in the end of the scheduling horizon, but can be easily handled in Step 3.4 by carefully updating $\overline{U}$.

3.3. Insertion Heuristic

Step 3.3 in the stepwise heuristic procedure adopts an insertion heuristic to insert remaining jobs one by one into the scheduling solution obtained in step 3.2. Each insertion action includes two decisions: (i) the job to be selected for insertion, and (ii) the position in the solution to insert the selected job. Given the step 3.2 solution $\{\mathscr{I}^k_{\text{scheduled}}\}_{k \in \mathscr{K}}$ and $\mathscr{I}_{\text{scheduled}}$, for each remaining job $i_r \in \mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{scheduled}}$, if we insert it right after $i_s \in \mathscr{I}^{k_{i_s}}_{\text{scheduled}}$ in the schedule of vehicle k_{i_s} with direction d_{i_s,i_r}, by denoting the jobs right before and after $i \in \mathscr{I}_{\text{scheduled}}$ as

[2]How the number of segments can be reduced highly depends on the actual application context and the parameter settings (e.g., values of $\overline{L}$ and $\overline{P}$). For example, in the grinding application in Section 4.3, the total number of segments is 393, while the total number of important segments is only 98 when $\overline{L} = 20$ and $\overline{P} = 25$, respectively.

i.prev and i.next, respectively, we compute the corresponding benefit of this insertion as

$$\begin{aligned} C(i_s,i_r) =& W_{\text{job}} L_{i_r} - W_{\text{travel}} \left(T^{k_{i_s},d_{i_s,i_r}}_{i_s,i_r} + T^{k_{i_s},d_{i_r,i_s.\text{next}}}_{i_r,i_s.\text{next}} - T^{k_{i_s},d_{i_s,i_s.\text{next}}}_{i_s,i_s.\text{next}} \right) - W_{\text{time}} P_{i_r} (u_{i_r} + v_{i_r}) \\ & - \sum_{i \in \mathscr{I}^{k_{i_s}}_{\text{scheduled}}} W_{\text{time}} P_i \left(\Delta u_i + \Delta v_i \right) - \sum_{\text{other cost items}} C_{\text{cost item}} \end{aligned} \tag{3.1}$$

where $C(i_s,i_r)$ is the composition of the benefit of maintaining i_r, the penalty of increased traveling introduced by inserting i_r, the penalty for i_r to miss the preferred maintaining time, the penalty for changing schedules of jobs following i_s in $\mathscr{I}^{k_{i_s}}_{\text{scheduled}}$, and other cost items. Note that to insert i_r, we may need to postpone some subsequent job $i \in \mathscr{I}^{k_{i_s}}_{\text{scheduled}}$, which potentially changes the value of u_i or v_i by Δu_i or Δv_i, respectively, and introduces additional penalty or infeasibility issue. Specifically, we have

$$w^{k_{i_s}}_{i_r} = w^{k_{i_s}}_{i_s} + S^{k_{i_s}}_{i_s} + T^{k_{i_s},d_{i_s,i_r}}_{i_s,i_r}, \tag{3.2a}$$

$$u_{i_r} = \max\{0, w^{k_{i_s}}_{i_r} - T_{i_r}\}, \tag{3.2b}$$

$$v_{i_r} = \max\{0, T_{i_r} - w^{k_{i_s}}_{i_r}\}, \tag{3.2c}$$

$$w^{k_{i_s}}_{i} = \max\{w^{k_{i_s}}_{i}, w^{k_{i_s}}_{i.\text{prev}} + S^{k_{i_s}}_{i.\text{prev}} + T^{k_{i_s},d_{i.\text{prev},i}}_{i.\text{prev},i}\}, \; \forall i \in \mathscr{I}^{k_{i_s}}_{\text{scheduled}}, \tag{3.2d}$$

$$\Delta u_i = \max\{0, \left(w^{k_{i_s}}_{i} - T_i \right) - u_i\}, \; \forall i \in \mathscr{I}^{k_{i_s}}_{\text{scheduled}}, \tag{3.2e}$$

$$\Delta v_i = \max\{0, \left(T_i - w^{k_{i_s}}_{i} \right) - v_i\}, \; \forall i \in \mathscr{I}^{k_{i_s}}_{\text{scheduled}}, \tag{3.2f}$$

where (3.2a)-(3.2c) define the maintaining time of job i_r, the number of time units that i_r is late and early, respectively. (3.2d) update the maintaining time for any job i in the schedule of vehicle k_{i_s} after insertion, and (3.2e) and (3.2f) compute the changes in time that job i is late and early, respectively. We need to verify the feasibility of this insertion by checking whether any related side constraint is violated with these new variable values.

Based on the insertion benefit calculations, the insertion heuristic can be described as follows:

Step 3.3.1. Calculate the insertion benefit $C(i_s,i_r)$ of any pair of jobs $(i_s,i_r), i_s \in \mathscr{I}_{\text{scheduled}}, i_r \in \mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{scheduled}}$;

Step 3.3.2. Find the pair of jobs (i^*_s, i^*_r) that is insertion-feasible and has the maximum value of $C(i_s,i_r)$, i.e.,

$$(i^*_s, i^*_r) = \operatorname{argmax}_{\substack{i_s \in \mathscr{I}_{\text{scheduled}}, \\ i_r \in \mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{scheduled}}}} \{C(i_s,i_r)\}$$

Step 3.3.3. If $C(i^*_s, i^*_r) \leq 0$ or $\mathscr{I}_{\text{scheduled}} = \mathscr{I}_{\text{input}}$, we have inserted all jobs that could benefit us and hence terminate; otherwise we go Step 3.3.4.

Step 3.3.4. If $C(i^*_s, i^*_r)$ is positive, we insert i^*_r right after i^*_s into the schedule of the corresponding vehicle $k_{i^*_s}$, update $\mathscr{I}^{k_{i^*_s}}_{\text{scheduled}}$ and $\mathscr{I}_{\text{scheduled}}$, and update i^*_r.prev $=$

$i_s^*, i_s^*.\text{next.prev} = i_r^*, i_r^*.\text{next} = i_s^*.\text{next}, i_s^*.\text{next} = i_r^*$. Then for any job i after i_r^* in the schedule of $k_{i_s^*}$, we update its maintaining time (after we update those jobs scheduled before it) as

$$w_i^{k_{i_s^*}} = \max\{w_i^{k_{i_s^*}}, w_{i.\text{prev}}^{k_{i_s^*}} + S_{i.\text{prev}}^{k_{i_s^*}} + T_{i.\text{prev},i}^{k_{i_s^*}, d_{i.\text{prev},i}}\}$$

Then we go back to Step 3.3.1 and repeat the process.

3.4. Local Search Heuristic

By implementing the insertion heuristic, we have the scheduling solution $\{\mathscr{I}^k_{\text{scheduled}}\}_{k\in\mathscr{K}}$ occupying almost the entire scheduling horizon for each vehicle. We further use a local search algorithm to improve the solution. Specifically, given a scheduling solution $\{\mathscr{I}^k_{\text{scheduled}}\}_{k\in\mathscr{K}}$, we search and check the solutions in its neighborhood to find any potential for improvement. The neighborhood of a solution is defined by considering the following types of moves:

(1) One vehicle swap: we swap the orders of two jobs i_1, i_2 in the schedule of one vehicle k, as that shown in Step 3.4 in Fig 3.1.

(2) Two vehicles swap: we swap the positions of two jobs i_1 and i_2 that are in the schedules of two different vehicles k_1 and k_2, respectively.

(3) Two vehicles interchange: we remove one job i from the schedule of one vehicle k_1 and insert the job i into the schedule of another vehicle k_2.

In each local search step, we enumerate all the possible feasible neighbors of the current solution, calculate the benefit of the move to each neighbor, and find the one with the best benefit. Note that when constructing neighbor solutions, all the side constraints need to be taken into considerations to check solution feasibility. If the best neighbor solution leads to a lower system cost than the current one, we accept the move and update the current solution with the neighbor solution.

In order to improve the efficiency of the local search procedure, a few more approaches are developed as follows for further enhancement:

(1) We apply hierarchical neighborhood search. Specifically, we first calculate the benefits associated with only the jobs that are swapped or interchanged, and select a small set of moves with the largest benefits. We then add back the benefits associated with those jobs that are indirectly affected by the selected set of moves and find the best one.

(2) We implement "block" swap/interchange, where a block is defined as a sequence of consecutive jobs in the solution. These block operations introduce additional neighborhood search opportunities and could possibly bring improvement when two blocks both correspond to smooth routes and are not performed at their preferred times. Compared with the simple job swap/interchange, the block operations are typically much faster;

(3) We trace the benefits and avoid unnecessary recalculations. For example, if a "one vehicle swap" operation is taken, the schedule of jobs that are performed by other vehicles remain unchanged and their associated benefits that have been calculated before do not need to be revisited in the following step.

SECTION 4: NUMERICAL STUDIES

In this section, we test the performance of the proposed models and algorithms through a series of numerical experiments, as well as two real-world RMRSP instances in one of the largest railroads in North America. All tests are implemented in the Microsoft Visual C# environment and run on a personal computer with 3.4GHz CPU and 8GB RAM.

4.1. Performance Tests

To test the performance of the proposed model and algorithm, we solve a series of numerical instances extracted from the full-scale dataset provided by the company. Both the proposed solution approach and Gurobi solver are used and the results obtained by the two methods are compared.

To make the test cases being solvable by using Gurobi, we simplify the problem by only considering the core model Eqs. (2.1a)-(2.1p), and selecting up to 120 segments and 4 vehicles from the database. Specifically, we test four sets of cases: (i) 20 segments with frequency 3 times/year, 2-4 vehicles; (ii) 40 segments with frequency 3 times/year, 2-4 vehicles; (iii) 40 segments with frequency 3 times/year and 40 segments with frequency 2 times/year, 2-4 vehicles; and (iv) 40 segments with frequency 3 times/year, 40 segments with frequency 2 times/year, and 40 segments with frequency 1 time/year, 2-4 vehicles. Since the maximum frequency of maintaining a segment in any case is 3, we solve three *MIP Original* with the rolling horizon approach. The basic parameters for the rolling horizon are set as $\widehat{U} = 12$, $\Delta U_{\text{input}} = 6$ and $\Delta U_{\text{output}} = 4$. The time limit of using Gurobi to solve each *MIP Original* and *MIP TimeGap* is set accordingly to ensure that the total solution time is no larger than 30min, while the time of using Gurobi to solve the original formulation is set to 2h. For the *Insertion* step, the thresholds are set as $\overline{L} = 20$ and $\overline{P} = 2$.

The computational results are presented in Table 4.1. It can be observed that our proposed algorithm always outperforms Gurobi in terms of both solution quality and computation time. When the instance size is relatively large, i.e., the number of segments is larger than 40 or the number of vehicles is larger than 2, it becomes significantly difficult for Gurobi to solve the original model. For example, an instance with 3 vehicles and 80 segments has more than 220000 rows, 175000 columns and 173000 binary variables, and the difference between Gurobi solution and our heuristic solution is almost 200%. Moreover, the quality of the Gurobi solution looks quite random and fails to provide any insight.

Taking a closer look at the solution, we can observe that when the number of segments is small such as 20, two vehicles are enough to finish almost all the jobs (the cost for missing jobs

# of segments	# of jobs	# of vehicles	Heuristic approach (0.5h)					Gurobi (2h)	Diff (%)
			# of finished jobs	Cost components $					
				Missing jobs	Travel	Earliness&Lateness	Total		
20	60	2	58	1923	6670	1558	10151	11630	14.6
		3	60	0	3296	1864	5160	8183	58.6
		4	60	0	2355	1717	4072	14494	255.9
40	120	2	92	19575	3332	5250	28156	29267	3.9
		3	115	1066	4874	3830	9769	64199	557.2
		4	115	867	2391	3934	7191	34265	476.5
80	200	2	132	44017	5733	5602	55351	85242	54.0
		3	167	10825	6935	7470	25229	73523	191.4
		4	177	3338	5704	5960	15002	107865	619.0
120	240	2	114	72911	4551	4851	82313	124369	51.1
		3	178	31512	7114	5666	44293	155896	252.0
		4	207	7809	9899	7476	25184	136386	541.6

Table 4.1: Algorithm performance for the hypothetical numerical instances.

is very small). When the number of segments increases, more vehicles are required to avoid the high penalty of not being able to finish some jobs. For example, when 40 segments are considered, 2 vehicles are only able to finish 92 jobs, while 3 or 4 vehicles can finish up to 115 out of 120 (5 jobs are missing because of the high penalty for other components if they are scheduled). Furthermore, when the number of segments increases to 120, even 4 vehicles are not enough, and the cost for missing jobs (also the total cost) increases significantly as the number of vehicles reduces from 4 to 2.

We further test the effect of the insertion and local search heuristic by investigating the objective value improvements obtained by running these two procedures. The detailed costs with and without running the insertion and local search heuristic for the hypothetical cases (with 80 and 120 segments) are summarized in Table 4.2 below. We do not show the cases with 20 and 40 segments because these cases involve mostly important segments and there is very little room for improvement. From the comparison, we see that the insertion heuristic brings a large benefit to the solution by adding more jobs (although less important ones) into the schedule, and as a result, shortening the time gaps between adjacent scheduled jobs and reduces the total system costs. In addition, as compared with the heavy computational burden needed for solving the MIP models in Steps 3.1 and 3.2 (i.e., ≈ 2 h), the execution times of the insertion and local search procedures (i.e., < 1 min) can actually be considered negligible.

4.2. *Sensitivity Analysis*

To show the applicability of our formulation and algorithms, and to draw managerial insights, we further conduct sensitivity analysis on some system parameters with the case involving 80 segments, 200 jobs, and 3 vehicles. Different weights of the various cost components in the objective function, and different values of $\Delta\widehat{U}_{\text{input}}$ and $\Delta\widehat{U}_{\text{output}}$. The results are summarized in the following Table 4.3 and Table 4.4, respectively.

From Table 4.3, we can observe that as we change the relative weights of different cost com-

# of segments	# of jobs	# of vehicles	With or without insertion & local search	# of finished jobs	Cost components $ Missing jobs	Travel	Earliness&Lateness	Total
80	120	2	without	94	48978	5800	4185	58962
			with	132	44017	5733	5602	55351
		3	without	133	15275	6999	6246	28520
			with	167	10825	6935	7470	25229
		4	without	143	7779	5790	4825	18394
			with	177	3338	5704	5960	15002
120	200	2	without	85	77018	4438	3532	84988
			with	114	72991	4551	4851	82313
		3	without	131	36259	7170	4445	47894
			with	178	31512	7114	5666	44293
		4	without	156	14073	10207	5737	30018
			with	207	7809	9899	7476	25814

Table 4.2: Comparison of results with and without the insertion and local search heuristic.

ponents, the values of these cost components in the solution change as expected. Specifically, when we increase the weight of one component, the cost of that component will decrease to avoid the associated high penalty. We want to emphasize here that the total costs under different ratios of weights are not supposed to be compared with each other as it highly depends on how different cost components can be transformed into monetary values following certain rules, which is not the focus of our research.

$W_{\text{job}} : W_{\text{time}} : W_{\text{travel}}$	Cost components $ Missing jobs	Travel	Earliness&Lateness	Total
1:1:1	8255	8005	6234	22494
4:1:1	4949	13307	6864	25120
1:4:1	23254	3448	6738	33440
1:1:4	18952	8152	4309	31414

Table 4.3: Results for different weights of cost components in the objectives.

Table 4.4 presents the results under different values of $\Delta\widehat{U}_{\text{input}}$ and $\Delta\widehat{U}_{\text{output}}$. When we increase the value of $\Delta\widehat{U}_{\text{input}}$ (from 6 to 9), the total cost reduces because we consider more job options and enlarge the feasible solution space. When we reduce the value of $\widehat{U}_{\text{output}}$ (from 4 to 3), the total system cost increases. This is because with smaller value of $\widehat{U}_{\text{output}}$, more model runs are needed, and thus it is more likely that some of the model runs have insufficient jobs to schedule and leave time gaps. This contributes to a smaller job completion ratio and a worse occupancy of the whole planning horizon.

4.3. Real-world Applications

We test the performance of the proposed model and solution algorithm for two real-world RMRSP instances in a Class I railroad network in North America (with more than 20,000 miles of track length): one is scheduling rail grinders for grinding rail segments, and the other one is

$\Delta\widehat{U}_{\text{input}}$	$\Delta\widehat{U}_{\text{output}}$	Cost components $			
(month)	(month)	Missing jobs	Travel	Earliness&Lateness	Total
6	4	8255	8005	6234	22494
6	3	13476	8022	7056	28554
9	4	8143	7831	5297	21271
9	3	10938	5572	6512	23023

Table 4.4: Results for different choices of input and output horizon length.

scheduling ballast cleaners for a given set of jobs requests. Both instances are used by the Class I railroad company in its real-world operations.

In the grinder scheduling instance, we plan the yearly grinding schedule for 393 rail track segments with two grinders. Note that each segment may need multiple jobs to be performed during the one year period; see Figure 4.1 for the frequency distribution of segments. The constraints considered for the grinder scheduling include (i) hard/soft time windows constraints, (ii) mutual exclusion constraints, (iii) precedence constraints; and (iv) maintenance constraints. Considering the large size of the network and the complicated nature of routing problems, we partition the original problem into multiple smaller pieces, as we discussed in Section 3.1. Specifically, we set ΔU_{input} to be 6 months and ΔU_{output} to be 4 months. This means that the solution algorithm will run Step 1 to 4 for three times to plan the schedules for the whole year. In addition, the time limit and acceptable gap for each *MIP Original* step are set as 1h and 5%, respectively. While for each *MIP TimeGap* step, they are set as 10min and 1%, respectively. Given the problem settings, the total solution time of the proposed algorithm solving one real-world instance is always less than 4h, which is acceptable in practice since the schedule is updated every one month. The solution obtained by applying the algorithm is compared with the manual solution provided by experts from the company. To protect data confidentiality, we only report some summaries of the solutions but not the exact numbers or statistics.

There are several key performance metrics to evaluate the grinder scheduling solutions: (i) the total track miles that are finished within the planning horizon (Figure 4.2a); (ii) the total travel time that the grinders spend on moving from segment to another (Figure 4.2b); (iii) the job completion ratio and the number of missed grinding jobs (Figure 4.2c); and (iv) the number of jobs that are grinded too early or too late (beyond the soft time window), (Figure 4.2d). Compareing our model solution to the manual solution, we have the following observations:

(1) Our approach achieves a 21% improvement in total track miles being maintained.

(2) Our approach yields a significant 41% reduction in total travel time.

(3) Only 74 out of total 785 grinding jobs are missed in our model solution.

(4) Only 9 out of 711 grinding jobs that have been performed are grinded too early.

(5) Only 4 out of 711 grinding jobs that have been performed are grinded too late.

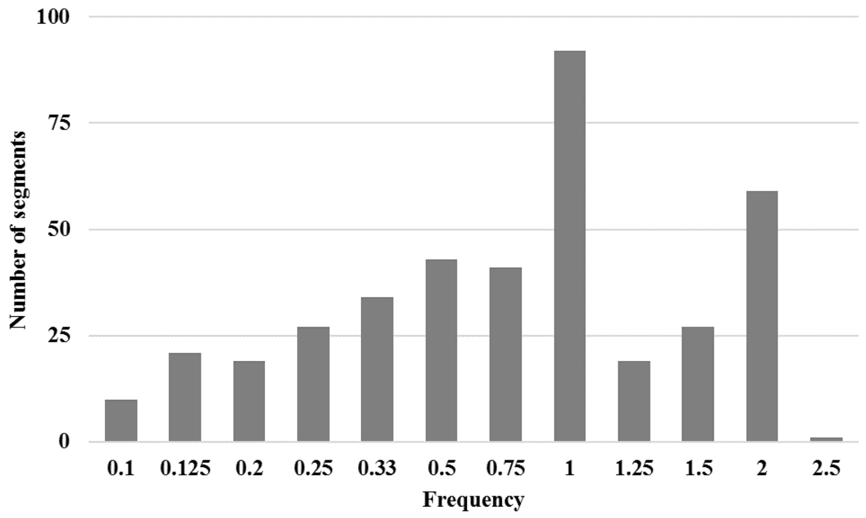
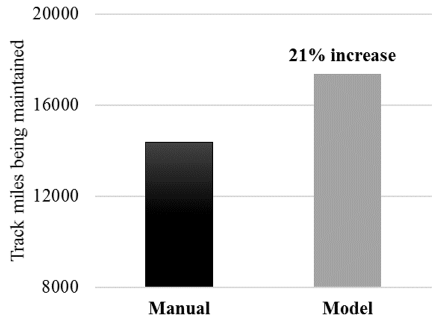


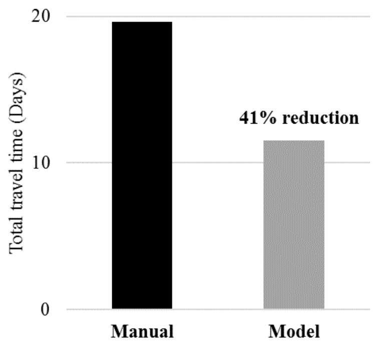
Figure 4.1: Frequency distribution.

The second application context of our model is yearly scheduling of the ballast cleaners over the large-scale railroad network. There are 449 job requests being submitted, with the total track length of nearly 5500 miles.[3] The constraints are similar to those of the grinding scheduling, except that (i) the preference time windows constraints are added considering the curfew periods, where the percentage of increased efficiency β equals to 60%; (ii) the repulsive time windows are enforced due to weather requirements; and (iii) some other factors, such as whether a segment is constructed as concrete ties, or whether it can be worked inside a curfew, would determine the importance of segments, and in turn, influence the overall scheduling plan. Since there are usually a large number of job requests being submitted, some segments cannot be all scheduled within one year planning horizon even if the ballast cleaners work without stopping. Thus, it is critical to identify the most important segments (e.g., the ones with concrete ties or needs tie replacement, etc.) and come up with a schedule with reasonably short travel distances. The solution generated from the proposed model is able to cover nearly 50.6% of total track miles of the job requests, which is 77% longer than the manual solution (as shown in Figure 4.3a). Almost all segments with concrete ties (i.e., 99.9%) and nearly 45.6% of tracks that are going to have tie replacement are scheduled. Meanwhile, as shown in Figure 4.3b, the total traveling times are less than 8% of total working days. Considering the fact that the job requests are not necessarily adjacent to each other over the network, such productivity is very satisfactory in real practice. Moreover, by incorporating the weather conditions into the model, the newly generated schedule would encourage the ballast cleaners to go toward south during winter and work at dry areas in the summer. In this way, the layoff of cleaners due to bad weather conditions can be significantly reduced by following the schedule, such that the utilization of the ballast cleaners can be improved.

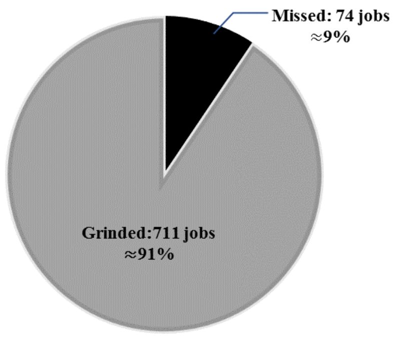
[3] To protect data confidentiality, this mileage number is calculated by multiplying the real data with a constant value.



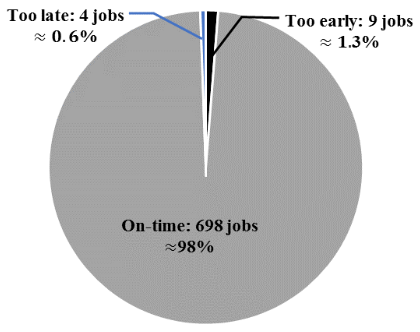
(a) Total track miles maintained



(b) Total travel time

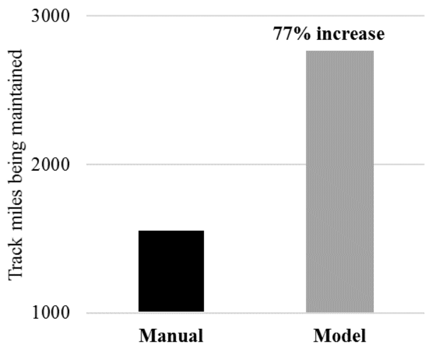


(c) Job completion ratio

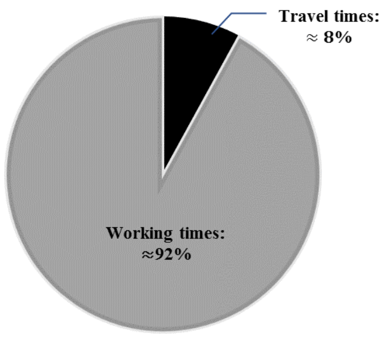


(d) Timeliness of schedule

Figure 4.2: Results of real-world grinder scheduling.



(a) Total track miles maintained



(b) Productivity of vehicles

Figure 4.3: Results of real-world ballast cleaner scheduling.

SECTION 5: CONCLUSION

This research develops a VRPTW-based MIP model to formulate the core component of RMRSP under variable productivities, with many complex side constraints being used to capture various business requirements. A customized stepwise algorithm procedure involving an MIP optimization model, an insertion heuristic, and a local search module, is designed and embedded in a rolling horizon approach framework to effectively solve the problem. The results of a series of numerical instances show that the proposed approach is able to yield much better solutions as compared to the existing commercial solver. For example, for the test instance with 200 jobs and 4 vehicles in Table 1, the cost provided by directly solving the original model using Gurobi MIP solver is more than 6 times larger than the one provided by our algorithm. Moreover, two empirical case studies show that the schedule produced by the proposed approach leads to a higher utilization of the maintenance vehicles compared with current manual solutions, e.g., the improvement in track miles being maintained is 21% for grinders and 77% for ballast cleaners. This saving is very attractive to the railroad company since most of the maintenance vehicles involve large costs. The proposed model and solution method have been adopted by a Class I railroad company to provide insights and instructions for their railroad maintenance scheduling and routing applications.

This work can be potentially extended in several directions. We are interested in studying the problem in a dynamic setting, where the routes and schedules can be modified according to the changing environments. This would become particularly important if real-time data and more accurate predictions can be obtained by applying more advanced sensing technologies to detecting the deterioration status of the tracks or ballasts. In addition, it would be interesting to investigate how the cyclic scheduling of rail maintenance machines can be achieved, where the segments that need to be routinely maintained can be visited repeatedly based on certain frequencies. Given the frequencies might be significantly different from each other, it is easy to imagine that the problem scale would become much larger and more powerful solution approaches such as metaheuristic algorithms need to be developed.

APPENDIX: NOTATION LIST

$\mathscr{M}$	Set of segments;
$\mathscr{N}_m$	Set of all expected maintenance activities of segment $m \in \mathscr{M}$;
$\mathscr{I}$	Set of all maintenance jobs that need to be performed in the planning horizon;
$\mathscr{I}_m$	Set of jobs associated with segment $m \in \mathscr{M}$;
$\{0\}$	Virtual depot;
$\mathscr{K}$	Set of maintenance vehicles;
$\mathscr{D}_m$	Set of directions a vehicle could move along between any two consecutive jobs i and j, $\mathscr{D}_m = \{d_1^m, d_2^m, d_3^m, d_4^m\}$, where $d_1^m = E_i^2 \to E_j^1, d_2^m = E_i^2 \to E_j^2, d_3^m = E_i^1 \to E_j^1, d_4^m = E_i^1 \to E_j^2$;

$\mathscr{D}_f$	Set of directions for moving from the depot to the first job, with $\mathscr{D}_f = \{d_1^f, d_2^f\}$, where $d_1^f = 0 \rightarrow E_i^1, d_2^f = 0 \rightarrow E_i^2$;
$\mathscr{D}_l$	Set of directions for arriving at the depot from the last job, with $\mathscr{D}_l = \{d_1^l, d_2^l\}$, where $d_1^l = E_i^2 \rightarrow 0, d_2^l = E_i^1 \rightarrow 0$;
$\mathscr{I}_{\text{hard}}$	Set of "hard" jobs that are assigned high priorities and are forced to be started within certain "hard" time windows;
$\mathscr{I}_{\text{soft}}$	Set of"soft" jobs that are assigned with relatively low priorities and are allowed to be started outside the given preferred "soft" time windows at certain penalty cost;
$\mathscr{P}$	Set of preference time windows;
$\mathscr{I}_p$	Set of jobs that have preference time window $p \in \mathscr{P}$;
$\mathscr{R}$	Set of repulsive time windows;
$\mathscr{I}_r$	Set of jobs that should not be performed during repulsive time window $r \in \mathscr{R}$;
$\mathscr{G}_{\text{hard}}, \mathscr{G}_{\text{soft}}$	Sets of all pairs of jobs that should follow the hard and soft precedence constraints, respectively.
$\mathscr{I}_{\text{input}}$	Set of maintenance jobs with preferred time $T_i \in [\overline{U}, \widehat{U}_{\text{input}}]$;
$\mathscr{I}_{\text{output}}$	Set of jobs that are started within $[\overline{U}, \widehat{U}_{\text{output}}]$;
$\mathscr{I}_{\text{important}}$	Set of "important" jobs, where $\mathscr{I}_{\text{important}} = \{i \in \mathscr{I}_{\text{input}} : L_i \geq \overline{L} \text{ or } P_i \geq \overline{P}\}$;
$\mathscr{I}_{\text{unimportant}}$	Set of "unimportant" jobs as $\mathscr{I}_{\text{unimportant}} = \mathscr{I}_{\text{input}} \backslash \mathscr{I}_{\text{important}}$;
$\mathscr{I}^k_{\text{scheduled}}$	Set of jobs (in the order of performing time) that are scheduled for vehicle $k \in \mathscr{K}$ in the solution;
F_m	Desired maintenance frequency of segment $m \in \mathscr{M}$;
Γ_m	Desired maintenance headway of segment $m \in \mathscr{M}$;
N_m	Number of times that segment $m \in \mathscr{M}$ is likely to be maintained within the entire planning horizon;
L_i	Track length associated with maintenance job $i \in \mathscr{I}$;
P_i	Priority value associated with maintenance job $i \in \mathscr{I}$;
W_{job}	Penalty per mile for not performing a job;
W_{travel}	Penalty per time unit for traveling;
W_{time}	Penalty per unit time for being late or early from the preferred time;
α_{job}	Cost factor of the penalty for not performing a job;
α_{travel}	Cost factor of the penalty for traveling;
α_{time}	Cost factor of the penalty for being late or early;
E_i^1, E_i^2	The two ends of segment m_i corresponding to job $i \in \mathscr{I}$;
$T_{i,j}^{k,d}$	Travel time for vehicle $k \in \mathscr{K}$ moving from job i to j in direction $d \in \mathscr{D}$;
S_i^k	Regular working time needed to finish job $i \in \mathscr{I}$ by vehicle $k \in \mathscr{K}$;
γ	Deterioration rate of tracks;
W_m^0	Maintenance time of last maintenance activity for segment $m \in \mathscr{K}$;
$A_{i,j}^{k,d}, B_i^k$	Big values associated with different constraints;
U_i, V_i	Maximum time units that job $i \in \mathscr{I}_{\text{hard}}$ could be late and early if it is a "hard" job, respectively;
$P_p^{\text{start}}, P_p^{\text{end}}$	Start and end time of the preference time window $p \in \mathscr{P}$, respectively;
$R_r^{\text{start}}, R_r^{\text{end}}$	Start and end time of repulsive time window $r \in \mathscr{R}$, respectively;
β	Percentage of increased efficiency if vehicles work within preferred time windows;

t_{ip}^{k}	Length of time that vehicle $k \in \mathscr{K}$ works on job $i \in \mathscr{I}_p$ in preferred time window $p \in \mathscr{P}$;
$C_{\text{cost item}}$	Costs related to side constraints;
C_{HStw}	Total penalty costs incurred by jobs in $\mathscr{I}_{\text{soft}}$ if they are performed outside their soft time windows;
$C_{\text{Pref}}^{i,k}$	Total time preference constraints penalty cost factor associated with job $i \in \mathscr{I}_p$ and vehicle $k \in \mathscr{K}$;
C_{Prec}	Total penalty associated with the precedence constraints;
$c_{\text{Prec}}^{i_1,i_2}$	Penalty for job pair (i_1, i_2) violating the soft precedence constraints;
MT_k	Length of the maintenance time length for vehicle $k \in \mathscr{K}$;
T_i	Preferred maintenance time for job $i \in \mathscr{I}$;
$\widehat{U}$	Length of the entire scheduling horizon;
$\overline{U}$	Start time of the current scheduling horizon, initially set as 0;
$[\overline{U}, \widehat{U}_{\text{input}}]$	Input horizon in the solution algorithm;
$[\overline{U}, \widehat{U}_{\text{output}}]$	Output horizon in the solution algorithm;
ΔU_{input}	Length of the horizon for considering input segments;
ΔU_{output}	Length of the horizon for obtaining scheduling outputs;
$\overline{L}$,$\overline{P}$	Lower thresholds of the track length and the priority value for a segment to be important, respectively;
$\overline{U}_{\text{gap}}$	Start time of a time gap in the schedule;
$[\overline{U}_{\text{gap}}, \widehat{U}_{\text{gap}}]$	Time gap in the schedule;
$C(i_s, i_r)$	Composition of the benefit of maintaining i_r, the penalty of increased traveling introduced by inserting i_r, the penalty for i_r to miss the preferred maintaining time, the penalty for changing schedules of jobs following i_s in $\mathscr{I}_{\text{scheduled}}^{k_{i_s}}$, and other cost items;
z_i	Binary decision variable indicating whether job $i \in \mathscr{I}$ is performed or not;
z_i^k	Binary decision variable indicating whether job $i \in \mathscr{I}$ is performed by vehicle $k \in \mathscr{K}$ or not;
$x_{i,j}^{k,d}$	Routing decision variable, indicating whether vehicle $k \in \mathscr{K}$ move from job i to j in direction $d \in \mathscr{D}$;
$x_{0,i}^{k,d_1}, x_{i,0}^{k,d_2}$	Binary decision variables indicating whether vehicle k reaches job i from the depot in direction d_1 and leaves job i to the depot in direction d_2, respectively;
s_i^k	Actual working duration of job $i \in \mathscr{I}$ by vehicle $k \in \mathscr{K}$;
w_i^k	Start time for vehicle k to perform job $i \in \mathscr{I}$;
u_i, v_i	Number of time units that job $i \in \mathscr{I}$ is late and early, respectively;
u_i', v_i'	Time units that "soft" job $i \in \mathscr{I}_{\text{soft}}$ is actually late and early beyond its preferred time windows, respectively;
q_{ip}^k	Binary decision variable indicating whether vehicle $k \in \mathscr{K}$ would work on job $i \in \mathscr{I}_p$ for t_{ip}^k length of time in time window $p \in \mathscr{P}$;
p_i^r	Binary decision variable indicating whether job $i \in \mathscr{I}_r$ is performed before or after the repulsive time window $r \in \mathscr{R}$;
y_{i_1,i_2}	Binary decision variable indicating whether job $i_1 \in \mathscr{I}$ is performed before job $i_2 \in \mathscr{I}$ or not;

m_i^k	Binary decision variable indicating whether the maintenance period for vehicle $k \in \mathscr{K}$ follows right after finishing job $i \in \mathscr{I}$ in its schedule.

References

Bard, J. F., Kontoravdis, G., Yu, G., 2002. A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows. Transportation Science 36 (2), 250–269.

Bräysy, O., Gendreau, M., 2005a. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. Transportation Science 39 (1), 104–118.

Bräysy, O., Gendreau, M., 2005b. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. Transportation Science 39 (1), 119–139.

Campbell, A. M., Wilson, J. H., 2014. Forty years of periodic vehicle routing. Networks 63 (1), 2–15.

Castillo, E., Calviño, A., Grande, Z., Sánchez-Cambronero, S., Gallego, I., Rivas, A., Menéndez, J. M., 2016a. A Markovian-Bayesian network for risk analysis of high speed and conventional railway lines integrating human errors. Computer-Aided Civil and Infrastructure Engineering 31 (3), 193–218.

Castillo, E., Grande, Z., Calviño, A., 2016b. Bayesian networks-based probabilistic safety analysis for railway lines. Computer-Aided Civil and Infrastructure Engineering 31 (9), 681–700.

Chao, I.-M., Golden, B. L., Wasil, E., 1995. An improved heuristic for the period vehicle routing problem. Networks 26 (1), 25–44.

Christofides, N., Beasley, J. E., 1984. The period routing problem. Networks 14 (2), 237–256.

Cordeau, J.-F., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30 (2), 105–119.

D'Ariano, A., Meng, L., Centulio, G., Corman, F., 2017. Integrated stochastic optimization approaches for tactical scheduling of trains and railway infrastructure maintenance. Computers & Industrial Engineering.

Desaulniers, G., Madsen, O., Ropke, S., 2014. Chapter 5: The Vehicle Routing Problem with Time Windows. In: Vehicle Routing. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, pp. 119–159.

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40 (2), 342–354.

Federal Railroad Administration Office of Safety Analysis, 2017. One year accident/incident overview by region/state/county. `http://safetydata.fra.dot.gov/officeofsafety`, Accessed: 12/11/2017.

Forsgren, M., Aronsson, M., Gestrelius, S., 2013. Maintaining tracks and traffic flow at the same time. Journal of Rail Transport Planning & Management 3 (3), 111–123.

Francis, P., Smilowitz, K., Tzur, M., 2006. The period vehicle routing problem with service choice. Transportation Science 40 (4), 439–454.

Francis, P. M., Smilowitz, K. R., Tzur, M., 2008. The period vehicle routing problem and its extensions. In: The vehicle routing problem: latest advances and new challenges. Springer, pp. 73–102.

Gaudioso, M., Paletta, G., 1992. A heuristic for the periodic vehicle routing problem. Transportation Science 26 (2), 86–92.

Kallehauge, B., Boland, N., Madsen, O. B., 2007. Path inequalities for the vehicle routing problem with time windows. Networks 49 (4), 273–293.

Kohl, N., Desrosiers, J., Madsen, O. B. G., Solomon, M. M., Soumis, F., 1999. 2-path cuts for the vehicle routing problem with time windows. Transportation Science 33 (1), 101–116.

Lannez, S., Artigues, C., Damay, J., Gendreau, M., Aug. 2015. A railroad maintenance problem solved with a cut and column generation matheuristic. Networks 66 (1), 40–56.

Lidén, T., 2015. Railway infrastructure maintenance - A survey of planning problems and conducted research. Transportation Research Procedia 10, 574–583.

Lidén, T., Joborn, M., 2017. An optimization model for integrated planning of railway traffic and network maintenance. Transportation Research Part C: Emerging Technologies 74, 327–347.

Luan, X., Miao, J., Meng, L., Corman, F., Lodewijks, G., 2017. Integrated optimization on train scheduling and preventive maintenance time slots planning. Transportation Research Part C: Emerging Technologies 80, 329–359.

Lysgaard, J., 2006. Reachability cuts for the vehicle routing problem with time windows. European Journal of Operational Research 175 (1), 210–223.

Osman, M. H. B., Kaewunruen, S., Jack, A., Jul. 2017. Optimisation of schedules for the inspection of railway tracks. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 1–11.

Peng, F., Kang, S., Li, X., Ouyang, Y., Somani, K., Acharya, D., 2011. A heuristic approach to the railroad track maintenance scheduling problem. Computer-Aided Civil and Infrastructure Engineering 26 (2), 129–145.

Peng, F., Ouyang, Y., 2012. Track maintenance production team scheduling in railroad networks. Transportation Research Part B: Methodological 46 (10), 1474–1488.

Peng, F., Ouyang, Y., Somani, K., 2013. Optimal routing and scheduling of periodic inspections in large-scale railroad networks. Journal of Rail Transport Planning & Management 3 (4), 163–171.

Savelsbergh, M. W. P., 1985. Local search in routing problems with time windows. Annals of Operations Research 4 (1), 285–305.

Su, Z., Jamshidi, A., NÃžÃśez, A., Baldi, S., De Schutter, B., 2017. Multi-level condition-based maintenance planning for railway infrastructures – A scenario-based chance-constrained approach. Transportation Research Part C: Emerging Technologies 84, 92–123.

Toth, P., Vigo, D. (Eds.), 2014. Vehicle Routing. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics.

Vansteenwegen, P., Dewilde, T., Burggraeve, S., Cattrysse, D., 2016. An iterative approach for reducing the impact of infrastructure maintenance on the performance of railway systems. European Journal of Operational Research 252 (1), 39–53.

Wang, J., Liu, X.-Z., Ni, Y.-Q., 2018. A Bayesian probabilistic approach for acoustic emission-based rail condition assessment. Computer-Aided Civil and Infrastructure Engineering 33 (1), 21–34.

Xie, W., Ouyang, Y., Somani, K., 2016. Optimizing location and capacity for multiple types of locomotive maintenance shops. Computer-Aided Civil and Infrastructure Engineering 31 (3), 163–175.