JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION

doi:10.3934/jimo.2020177

PLANNING ROLLING STOCK MAINTENANCE: OPTIMIZATION OF TRAIN ARRIVAL DATES AT A MAINTENANCE CENTER

Hanyu Gu, Hue Chi Lam* and Yakov Zinder

School of Mathematical and Physical Sciences
University of Technology Sydney
15 Broadway, Ultimo, NSW 2007, Australia

(Communicated by Bertrand M.T. Lin)

Abstract. A railway network is an indispensable part of the public transportation system in many major cities around the world. In order to provide a safe and reliable service, a fleet of passenger trains must undergo regular maintenance. These maintenance operations are lengthy procedures, which are planned for one year or a longer period. The planning specifies the dates of trains' arrival at the maintenance center and should take into account the uncertain duration of maintenance operations, the periods of validity of the previous maintenance, the desired number of trains in service, and the capacity of the maintenance center. The paper presents a nonlinear programming formulation of the considered problem and several optimization procedures which were compared by computational experiments using real world data. The results of these experiments indicate that the presented approach is capable to be used in real world planning process.

1. **Introduction.** Passenger trains provide one of the major means of public transport in many cities around the world. For example, the suburban passenger rail network in Sydney, Australia connects central Sydney with northern, southern, western, and eastern suburbs with 174 stations. The network spans over 813 kilometers of track and delivered about 359.2 million passenger journeys in 2017/18 [16].

There is a risk of sudden breakdowns or even a derailment if the trains (rolling stock) do not undergo maintenance regularly. There are several mandatory levels of maintenance that differ from each other by their scope and periodicity. Readers are referred to [9] for a detailed description of the various maintenance levels. This paper focuses on the high-level heavy maintenance which is the most involved and time consuming procedure and is performed at a specialized maintenance center. The rolling stock arrives at the maintenance center in groups. Each group is comprised of several cars coupled together and is referred to as a set or a train-set [9]. All cars in a train-set undergo maintenance in the maintenance center simultaneously.

The reliable functioning of a passenger transportation system is not possible without a plan, specifying the availability of the rolling stock. The dates when the train-sets should be withdrawn from service and sent to the maintenance center

2020 *Mathematics Subject Classification.* 90B36, 90C59.

Key words and phrases. Poisson binomial distribution, maintenance planning, mixed integer linear programming, iterated local search, earliness/tardiness.

* Corresponding author.

are crucial for the rolling stock operator as well as for the planning at the maintenance center. Given that the heavy maintenance of a train-set is a long process, both, rolling stock operator and maintenance center, need to specify at least for a year when the heavy maintenance of each train-set should commence [10], [15]. Furthermore, often the transportation of passengers and the rolling stock heavy maintenance are carried out by two separate organizations which operate on the basis of a long term contract that specifies for each train-set the precise date of the commencement of its maintenance.

This paper is concerned with the development of the plan that specifies the dates when the train-sets should arrive at the maintenance center. The presented optimization procedures take into account the specifics of the heavy maintenance procedure, including its uncertain duration and the restriction on the period between consecutive arrivals of the train-sets at the maintenance center, as well as the information provided by the rolling stock operator, including the engineering restrictions on the permissible period between heavy maintenance procedures and the demand for transportation.

The heavy maintenance has a validity period after which another heavy maintenance procedure must take place according to the engineering restrictions. Consequently, the rolling stock operator determines for every train-set a time window within which the maintenance of this train-set should commence (see, for example, [10]). The length of such time window depends on the practice of the rolling stock operator and in some cases can be zero (see, for example, [15]). Although it is equally possible to start maintenance at any point in time within the corresponding time window, for the purpose of the optimization procedures below, the middle of a time window will be considered as a preferred date and the time window will be viewed as a specification of the permissible deviation from this preferred date.

The dates when the train-sets should arrive at the maintenance center, specified by the mentioned above plan, may violate the time windows of the train-sets which are dictated only by the practice of the rolling stock operator and do not take into account the capacity of the maintenance center, the uncertain duration of maintenance, and the demand for transportation. Such violation is highly undesirable which often is modeled by introducing a penalty for the violation of the time windows.

A train-set that arrives at the maintenance center before its time window is referred to as "early", while a train-set that arrives after its time window is referred to as "tardy". If despite the penalties for the violation of the time windows the plan still contains some early and tardy train-sets, this situation is resolved by the consultations between the rolling stock operator and the maintenance center. Similar to the majority of publications on this topic, the consultation phase is beyond the scope of this paper.

Upon arriving at the maintenance center, a train-set is shunted to the first operation line, where thorough inspections and replacement of some components and parts such as air-conditioning units are performed. A train-set can arrive only if the first operation line is not occupied by the previous train-set. Normally, there are several types of trains and each type may require different time at the first operation line.

After the first operation line, the train-set undergoes various maintenance operations such as bogies replacement, system testing, refurbishment, etc. In order to perform these operations, the train-set has to be shunted between several lines. The

duration of each operation depends on the condition of the train-set; the availability and composition of the workforce; the availability of spare parts; and many other factors. Consequently, the dwelling time of a train-set at the maintenance center (also referred to as a cycle time) is uncertain at the time of planning.

On the arrival at the maintenance center, a train-set is completely withdrawn from service and must stay at the maintenance center for at least one month. This long cycle time directly impacts the number of train-sets available in active service. Therefore, as part of the input data for the heavy maintenance planning, a permissible number of train-sets that can be out of service simultaneously are specified for each type of train-sets. Furthermore, the capacity of the maintenance center also imposes the restriction on the number of train-sets which can undergo maintenance simultaneously. Any violation of all these restrictions causes serious problems and therefore must be taken into consideration at the planning stage.

The goal of the planning process is to determine an arrival plan, specifying the arrival dates for all train-sets. The discussion above suggests that it is reasonable to have the objective function as a weighted sum of two components: the total penalty for the deviation (earliness and tardiness) from the arrival time windows, and the expected total penalty for violating the center capacity as well as for violating the permissible number of out-of-service train-sets of all types.

The body of literature on planning the rolling stock maintenance falls into two broad categories: (i) planning the rolling stock low-level maintenance subject to the trains timetable, and (ii) planning the rolling stock high-level maintenance for a long planning horizon. The first category considers only low-level maintenance, such as daily and monthly inspections, which is often combined with the decisions on the rolling stock utilization. Accordingly, the low-level maintenance is often considered as constraints in the planning process of rolling stock utilization (see for example, [9], [7]).

The second category is concerned with the high-level maintenance which has a long cycle time. In authors' opinion, the high-level maintenance planning has not attracted literature which it deserves. To the best of the authors' knowledge, only the publications [15], [5], and [10] are closely related to the planning problem considered in this paper.

The first of these three publications, [15], is concerned with maintenance scheduling at the Hong Kong Mass Transit Railway Corporation. In contrast to our paper, the authors of [15] postulate that the duration of maintenance is given. This simplification allows them to approach the minimization of the earliness and tardiness with respect to the completion of maintenance as a deterministic scheduling problem. The authors of [15] also assume that the given permissible number of trains which can dwell at the maintenance center simultaneously can not be violated, which may not be possible to ensure when the duration of maintenance is uncertain. The resultant deterministic scheduling problem is solved by a genetic algorithm. The authors reported that the proposed approach produced near optimal solutions for randomly generated instances with linear earliness and tardiness objective.

As in [15], [5] assumes that the duration of maintenance is known and that the limit on the number of trains that can dwell at the maintenance center simultaneously cannot be violated. In addition, [5] is concerned with planning under the condition that the maintenance operations must commence prior to their due dates. The planning problem was formulated as a mixed integer linear programming model

and was solved using IBM ILOG CPLEX 11.2. The objective function is a weighted sum of the maintenance cost, the cost of shunting activities, the cost related to the spare parts, and the penalty for early maintenance. The authors reported that the inclusion of the cost related to the spare parts positively affected the quality of the plan.

The publication [10] is concerned with planning the heavy maintenance of the electric multiple units at the Shanghai Railway Bureau. As in [15] and [5], it is assumed that the duration of maintenance is known and that the limit on the number of trains that can dwell at the maintenance center simultaneously cannot be violated. As in our paper, each train has a time window when its maintenance should commence, but in contrast to our paper, [10] postulates that this time window cannot be violated. The problem is formulated as an integer linear program with the objective of minimizing the total penalty for early maintenance. It was reported that small instances can be solved exactly. For large instances, the authors propose a simulated annealing algorithm and report solving instances with up to 124 train-sets and a planning horizon of 607 days.

In summary, the contribution of our paper is that, in contrast to the previous publications, our paper considers the planning problem where

- for each train-set, the duration of maintenance is uncertain;
- the violation of the time windows within which the maintenance should commence is undesirable but possible;
- there are several types of train-sets;
- the violation of the given limit on the number of train-sets that can dwell at the maintenance center simultaneously is undesirable but possible;
- for each type of train-sets, the violation of the given limit on the number of train-sets that can dwell at the maintenance center simultaneously is undesirable but possible.

These differences to the previous publications lead to new optimization procedures designed to be used in a complex planning process involving the negotiations between the rolling stock operator and the maintenance center. More specifically, the paper presents

- a new nonlinear programming formulation of the considered planning problem, which takes into account the uncertain duration of maintenance;
- a method that permits to evaluate the objective function of the introduced nonlinear mathematical program for any feasible solution without resorting to computationally expensive Monte-Carlo simulation;
- a new mixed integer linear programming relaxation that is based on Jensen's Inequality [3] and therefore provides a lower bound to the optimal value of the objective function for the nonlinear mathematical program and hence allows to assess the quality of approximate solutions;
- an Iterated Local Search (ILS) metaheuristic that, in contrast to the previous publications, takes into account the uncertain duration of maintenance;
- a hybrid two-stage optimization procedure that combines the Jensen's Inequality based relaxation (or another mixed integer linear program also introduced in the paper) with either a local search subroutine or the presented ILS subroutine;
- a fast method for evaluation of the neighborhoods for the local search and ILS subroutines;

- results of the comparison, by means of computational experiments on real-world data, of the presented optimization procedures taking into account two main characteristics: the solution quality and the time needed for obtaining the solution.

The remainder of the paper is organized as follows. Section 2 presents a nonlinear mathematical programming formulation of the considered problem, an efficient algorithm for evaluation of the objective function, and a Jensen's Inequality based mixed integer linear programming relaxation. Section 3 presents an alternative mixed integer linear program, local search and iterated local search subroutines, and a fast method for the neighborhood evaluation. Section 4 presents the results of computational experiments that use real-world data provided by a major maintenance center in Australia. The conclusion can be found in Section 5.

2. Mathematical programming formulation.

This section presents a Nonlinear Integer Programming Model (NIPM) for the considered problem and a Mixed Integer Programming Model (MIPM) that provides a lower bound for NIPM. The considered planning problem can be stated as follows. A set $N = \{1, \cdots, n\}$ of train-sets is to undergo maintenance at a maintenance center. The planning period is T days which are numbered from 0 to $T-1$. An arrival plan specifies for each train-set $j \in N$ the day s_j of its arrival at the maintenance center. No two train-sets can arrive on the same day. Furthermore, the train-sets are of m different types and, for each type k, there exists a restriction when the next train-set can arrive after the arrival of a train-set of type k. This restriction is given by the number of days τ_k. If a train-set j of type k arrives on day s_j, then, regardless of the type of the next train-set, it can arrive only on the day $s_j + \tau_k$ or later.

It is convenient to consider the partition F^1, ..., F^m of N, where each F^k is comprised of all train-sets of the same type k. Observe that arrival days that satisfy the restriction on the time between two consecutive arrivals exist if and only if

$$\sum_{1 \le k \le m} |F^k| \tau_k - \max_{1 \le k \le m} \tau_k \le T - 1.$$

In what follows, it is assumed that this inequality holds.

For each $j \in N$, the time that a train-set j spends at the maintenance center (its cycle time) is a discrete random variable D_j which assumes integer values. All these random variables are independently distributed and, for each $1 \le k \le m$, the cycle times of all train-sets in F^k are identically distributed between a_k - the minimal possible duration of a cycle for a train-set of type k, and b_k - the maximal possible duration of a cycle for a train-set of type k. For each $1 \le k \le m$, $\tau_k < a_k$.

Each train-set j has the associated time window $[\theta_j - \Delta, \theta_j + \Delta]$, where θ_j is the preferred day of the commencement of maintenance, i.e. the preferred day of the arrival at the maintenance center, and Δ is the permissible deviation from this preferred day of arrival. The penalty for the violation of time window is defined by the function:

$$g_j(s_j) = \begin{cases} \lambda_1(\theta_j - s_j)^2 & \text{if } s_j < \theta_j - \Delta \\ \lambda_2(\theta_j - s_j)^2 & \text{if } s_j > \theta_j + \Delta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$. For any arrival plan $\sigma = [s_1, \cdots, s_n]$, the total penalty for the violation of time windows will be denoted by $G_1(\sigma) = G_1(s_1, \cdots, s_n)$:

$$G_1(\sigma) = G_1(s_1, \cdots, s_n) = \sum_{j \in N} g_j(s_j). \tag{2}$$

For any arrival plan σ, any $1 \leq k \leq m$, and any day t, denote by $W_t^k(\sigma)$ the number of train-sets in F^k that are at the maintenance center on day t. Since all cycle times are random variables, $W_t^k(\sigma)$ is a random variable. If $W_t^k(\sigma)$ exceeds the given limit C_{kt}, this attracts the penalty $\delta_{kt}(W_t^k(\sigma) - C_{kt})$, where $\delta_{kt} > 0$. The total number of train-sets at the maintenance center on day t is

$$W_t(\sigma) = \sum_{k=1}^{m} W_t^k(\sigma).$$

If this number exceeds the given limit C_t, this attracts the penalty $\delta_t(W_t(\sigma) - C_t)$, where $\delta_t > 0$. Let σ be an arbitrary arrival plan and $G_2(\sigma)$ be the mathematical expectation of the total penalty for the violation of the limits on the number of train-sets that can dwell at the maintenance center simultaneously. Then

$$\begin{aligned} G_2(\sigma) = \sum_{t=0}^{T-1} \Big(& \mathbb{E}\Big[\max\Big\{0, \delta_t\Big(W_t(\sigma) - C_t\Big)\Big\}\Big] \\ & + \sum_{k=1}^{m} \mathbb{E}\Big[\max\Big\{0, \delta_{kt}\Big(W_t^k(\sigma) - C_{kt}\Big)\Big\}\Big]\Big), \end{aligned} \tag{3}$$

where $\mathbb{E}[\cdot]$ is the mathematical expectation. The goal is to minimize

$$G(\sigma) = \alpha G_1(\sigma) + \beta G_2(\sigma) \tag{4}$$

where α and β are two positive weights.

2.1. **Nonlinear integer programming formulation.** For each $t \in \{0, ..., T-1\}$ and each $j \in N$, let

$$x_{jt} = \begin{cases} 1 & \text{if train-set } j \text{ arrives on day } t \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The heavy maintenance planning problem can be formulated as follows.

$$\text{(NIPM)} \quad Z_{NIPM} = \min \alpha G_1(s_1, \cdots, s_n) + \beta G_2(s_1, \cdots, s_n) \tag{6}$$

subject to

$$\sum_{t=0}^{T-1} x_{jt} = 1, \quad \forall j \in N \tag{7}$$

$$\sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=\max(0, t-\tau_k+1)}^{t} x_{js} \leq 1, \quad \forall t \in \{0, ..., T-1\} \tag{8}$$

$$s_j = \sum_{t=0}^{T-1} t x_{jt}, \quad \forall j \in N \tag{9}$$

(1), (2), (3)

$$x_{jt} \in \{0, 1\}, \quad \forall j \in N, \quad \forall t \in \{0, ..., T-1\} \tag{10}$$

In the above formulation, the objective function (6) is the weighted sum of two components: the total penalty for the violation of time windows; and the expected

penalties for the violation of the limits C_t and C_{kt}. Constraint set (7) ensures that each train-set must arrive for maintenance within the planning horizon. Constraint set (8) enforces that at most one train-set can occupy the first operation line on any given day. The arrival day of each train-set can be calculated as (9). Constraint set (10) states that the decision variables are binary.

2.2. **Evaluation of the objective function.** For optimization problems involving random variables, Monte-Carlo simulation is a commonly used approach for approximately evaluating the objective function [2, 8] unless the model can be written in closed form by assuming some special distributions [6, 1]. The objective function (6) can be evaluated exactly using the method below.

Given an arrival plan $\sigma = [s_1, \ldots, s_n]$, let

$$Y_j(s_j, t) = \begin{cases} 1 & \text{if } s_j \le t \text{ and } s_j + D_j \ge t + 1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

which is a Bernoulli random variable that takes value 1 if the train-set j is at the maintenance center on day t and 0 otherwise. Then, the probability of $Y_j(s_j, t) = 1$ can be computed as

$$\begin{cases} \text{Prob}(Y_j(s_j,t)=1) = \text{Prob}(D_j \ge t - s_j + 1) \\ \qquad\qquad\qquad = \displaystyle\sum_{i=t-s_j+1}^{b_k} \text{Prob}(D_j = i), \\ \qquad\qquad\qquad 1 \le k \le m,\ j \in F^k,\ t \in \{s_j, ..., T-1\} \\ \text{Prob}(Y_j(s_j,t)=1) = 0, \quad j \in N,\ t \in \{0, ..., s_j - 1\} \end{cases} \tag{12}$$

For any arrival plan σ, and any day t, $W_t(\sigma) = \sum_{j \in N} Y_j(s_j, t)$ is a sum of Bernoulli random variables with success probabilities according to (12). Therefore, $W_t(\sigma)$ is a random variable that follows Poisson Binomial distribution [4]. For any arrival plan $\sigma = [s_1, \cdots, s_n]$, any $1 \le i \le n$, and any day t, denote $p_i = \text{Prob}(Y_i(s_i, t) = 1)$. Let $\text{Prob}(W_t(\sigma^l) = i)$ be the probability that i train-sets from the partial arrival plan $\sigma^l = [s_1, \ldots, s_l]$ dwell at the maintenance center on day t. Then,

$$\text{Prob}(W_t(\sigma^1) = 1) = p_1 \quad \text{and} \quad \text{Prob}(W_t(\sigma^1) = 0) = 1 - p_1$$

and for all $1 \le l < n$

$$\begin{cases} \text{Prob}(W_t(\sigma^{l+1}) = 0) = (1 - p_{l+1})\text{Prob}(W_t(\sigma^l) = 0) \\ \text{Prob}(W_t(\sigma^{l+1}) = i) = (1 - p_{l+1})\text{Prob}(W_t(\sigma^l) = i) \\ \qquad\qquad + p_{l+1}\text{Prob}(W_t(\sigma^l) = i - 1), \qquad 1 \le i \le l \\ \text{Prob}(W_t(\sigma^{l+1}) = l + 1) = p_{l+1}\text{Prob}(W_t(\sigma^l) = l) \end{cases}. \tag{13}$$

Let PMF be the probability mass function of the Poisson binomial distributed variable. The entire procedure is outlined in Algorithm 1 [4].

Similarly, for any arrival plan σ, any $1 \le k \le m$, and any day t, $W_t^k(\sigma) = \sum_{j \in F^k} Y_j(s_j, t)$ is a sum of Bernoulli random variables with success probabilities according to (12). Then, all probabilities $\text{Prob}(W_t^k(\sigma))$ can be obtained using Algorithm 1.

Algorithm 1 Direct Convolution (DC)

Input: total number of Bernoulli random variables n; success probability p_i of the i-th random variable
Output: PMF
procedure DC$(p_1, \cdots, p_n)$
    $PMF(0) = 1 - p_1$, $PMF(1) = p_1$
    for i from 2 to n **do**
        $j = 1$
        $new\_PMF(0) = (1 - p_i)\ PMF(0)$
        while $j < i$ **do**
            $new\_PMF(j) = p_i\ PMF(j-1) + (1 - p_i)\ PMF(j)$
        end while
        $new\_PMF(i) = p_i\ PMF(i-1)$
        $PMF = new\_PMF$
    end for
    return PMF
end procedure

As a result, the objective function (4) can be computed as

$$\alpha G_1(\sigma) + \beta G_2(\sigma) = \alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \left[\sum_{w=C_t+1}^{|N|} \delta_t(w - C_t) \mathrm{Prob}(W_t(\sigma) = w) + \sum_{k=1}^{m} \sum_{w=C_{kt}+1}^{|F^k|} \delta_{kt}(w - C_{kt}) \mathrm{Prob}(W_t^k(\sigma) = w) \right]. \tag{14}$$

2.3. **Integer linear programming relaxation based on Jensen's Inequality.** As in (3), denoting the mathematical expectation by $\mathbb{E}[\cdot]$, the right-hand side in (14) can be written as

$$\alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \left(\delta_t \mathbb{E}[\max\{0, W_t(\sigma) - C_t\}] + \sum_{k=1}^{m} \delta_{kt} \mathbb{E}[\max\{0, W_t^k(\sigma) - C_{kt}\}] \right)$$

and, taking into account that $\max\{0, \cdot\}$ is convex, by Jensen's Inequality [3],

$$\geq \alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \left(\delta_t \max\{0, \mathbb{E}[W_t(\sigma) - C_t]\} + \sum_{k=1}^{m} \delta_{kt} \max\{0, \mathbb{E}[W_t^k(\sigma) - C_{kt}]\} \right)$$

$$= \alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \left(\delta_t \max\{0, \mathbb{E}[W_t(\sigma)] - C_t\} + \sum_{k=1}^{m} \delta_{kt} \max\{0, \mathbb{E}[W_t^k(\sigma)] - C_{kt}\} \right)$$

Denote

$$G_2'(\sigma) = \sum_{t=0}^{T-1} \left(\delta_t \max\{0, \mathbb{E}[W_t(\sigma)] - C_t\} + \sum_{k=1}^{m} \delta_{kt} \max\{0, \mathbb{E}[W_t^k(\sigma)] - C_{kt}\} \right)$$

Then,

$$\alpha G_1(\sigma) + \beta G_2(\sigma) \geq \alpha G_1(\sigma) + \beta G_2'(\sigma). \tag{15}$$

Using the variables x_{jt}, introduced in (5), for any $t \in \{0, ..., T-1\}$,

$$\mathbb{E}[W_t(\sigma)] = \mathbb{E}\Big[\sum_{k=1}^{m} \sum_{j \in F^k} Y_j(s_j, t)\Big] = \sum_{k=1}^{m} \sum_{j \in F^k} \text{Prob}\Big(Y_j(s_j, t) = 1\Big)$$
$$= \sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js}, \tag{16}$$

and, for any $t \in \{0, ..., T-1\}$ and any $1 \leq k \leq m$,

$$\mathbb{E}[W_t^k(\sigma)] = \mathbb{E}\Big[\sum_{j \in F^k} Y_j(s_j, t)\Big] = \sum_{j \in F^k} \text{Prob}(Y_j(s_j, t) = 1)$$
$$= \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js}. \tag{17}$$

This leads to the following Mixed Integer Programming Model (MIPM):

$$\text{(MIPM)} \quad Z_{MIPM} = \min \alpha \sum_{j \in N} \Big(\lambda_1 \sum_{t=0}^{\theta_j - \Delta - 1} (\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j + \Delta + 1}^{T-1} (t - \theta_j)^2 x_{jt} \Big)$$
$$+ \beta \sum_{t=0}^{T-1} \Big(\delta_t \; w_t + \sum_{k=1}^{m} \delta_{kt} w_t^k \Big) \tag{18}$$

subject to

$$(7), (8), (10)$$

$$\sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js} \leq C_t + w_t,$$
$$\forall t \in \{0, ..., T-1\} \tag{19}$$

$$\sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js} \leq C_{kt} + w_t^k,$$
$$1 \leq k \leq m, \; \forall t \in \{0, ..., T-1\} \tag{20}$$

$$w_t \geq 0, \; w_t^k \geq 0, \quad 1 \leq k \leq m, \; \forall t \in \{0, ..., T-1\}. \tag{21}$$

The nonlinear programming problem NIPM and the mixed integer linear programming problem MIPM use the same variables x_{jt}, introduced in (5), that satisfy the same set of constraints (7), (8), (10). Therefore, for any arrival plan $\sigma = [s_1, ..., s_n]$ that is feasible for NIPM, there exists a feasible solution for MIPM with variables x_{jt}, w_t and w_t^k satisfying, for all $j \in N$, $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$,

$$s_j = \sum_{t=0}^{T-1} t x_{j,t}$$

$$w_t = \max\left\{0,\ \sum_{k=1}^{m}\sum_{j\in F^k}\sum_{s=0}^{t}\sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i)x_{js} - C_t\right\}$$

$$w_t^k = \max\left\{0,\ \sum_{j\in F^k}\sum_{s=0}^{t}\sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i)x_{js} - C_{kt}\right\}.$$

or, by taking into account (16) and (17),

$$w_t = \max\{0,\ \mathbb{E}[W_t(\sigma)] - C_t\}$$
$$w_t^k = \max\{0,\ \mathbb{E}[W_t^k(\sigma)] - C_{kt}\}.$$

In other words, for any arrival plan $\sigma = [s_1, ..., s_n]$ that is feasible for NIPM, there exists a solution for the mixed integer linear programming problem MIPM which variables x_{jt}, w_t and w_t^k satisfy

$$\alpha G_1(\sigma) + \beta G_2'(\sigma) = \alpha \sum_{j\in N}\left(\lambda_1 \sum_{t=0}^{\theta_j-\Delta-1}(\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j+\Delta+1}^{T-1}(t-\theta_j)^2 x_{jt}\right)$$
$$+ \beta \sum_{t=0}^{T-1}\left(\delta_t\ w_t + \sum_{k=1}^{m}\delta_{kt} w_t^k\right).$$

This, by virtue of (15), implies that the optimal value Z_{MIPM} of the objective function for MIPM is a lower bound on the optimal value Z_{NIPM} of the objective function for NIPM.

3. Construction of arrival plans.

As has been discussed in Section 1, the existing literature often ignores the stochastic nature of cycle times and focuses on the optimization procedures that construct arrival plans assuming that the duration of maintenance is known. This also reflects the practice often encountered in industry. Under the assumption of deterministic cycle times, the arrival plan is constructed either by solving a mathematical programming problem [10], [5] or by some metaheuristic [10], [15]. The Mixed Integer Linear Programming Problem (MILP) below also assumes that the cycle times are known constants, but in contrast to the previous publications the resulting arrival plan is evaluated as described in Section 2.2.

As far as metaheuristics are concerned, this paper presents an Iterated Local Search (ILS) algorithm which is embedded into the multi-start framework. In contrast to the previous publications, the presented ILS does not assume that the cycle times are deterministic and evaluates each element in a neighborhood taking into account the stochastic nature of cycle times.

Furthermore, this paper presents a hybrid optimization procedure which generates a starting solution by solving either the mixed integer linear programming problem MILP or the mixed integer linear programming problem MIPM and then enhances the obtained solution using the mentioned above ILS algorithm. As has been discussed above, when the hybrid optimization procedure generates the starting solution using the mixed integer linear programming problem MIPM, the optimal value of the objective function is a lower bound on the optimal value of the original nonlinear programming problem.

Section 4 reports the results of computational experiments which were aimed at the comparison of the quality of the solutions produced by the presented optimization procedures and the times needed to generate these solutions. Since the considered optimization problem is a component of the complex planning process which normally involves a lot of negotiations between the rolling stock operator and the maintenance center, both characteristics, quality and time, are important.

3.1. **Mixed integer linear program MILP.** It is reasonable to model random cycle times using a beta-PERT distribution [12], which is commonly used in project management [13]. In this case, the constant duration is chosen as the most likely value of the cycle time according to the beta-PERT distribution. This approach is adopted in the computational experiments below. Let q_k be the most likely value of the cycle time for the train-sets in F^k. Then, the arrival day for each train-set j can be determined by

$$s_j = \sum_{t=0}^{T-1} t x_{jt}$$

where variables x_{jt} are obtained by solving the following mixed integer linear program

$$\text{(MILP)} \quad Z_{MILP} = \min \ \alpha \sum_{j \in N} \Big(\lambda_1 \sum_{t=0}^{\theta_j - \Delta - 1} (\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j+\Delta+1}^{T-1} (t - \theta_j)^2 x_{jt} \Big)$$

$$\beta \sum_{t=0}^{T-1} \Big(\delta_t \ w_t + \sum_{k=1}^{m} \delta_{kt} w_t^k \Big) \tag{22}$$

subject to

$$(7),\ (8),\ (10)$$

$$\sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=\max(0,t-q_k+1)}^{t} x_{js} \le C_t + w_t, \qquad \forall t \in \{0, ..., T-1\} \tag{23}$$

$$\sum_{j \in F^k} \sum_{s=\max(0,t-q_k+1)}^{t} x_{js} \le C_{kt} + w_t^k, \qquad 1 \le k \le m, \ \forall t \in \{0, ..., T-1\} \tag{24}$$

$$w_t \ge 0, \ w_t^k \ge 0, \quad 1 \le k \le m, \ \forall t \in \{0, ..., T-1\}. \tag{25}$$

3.2. **Local search subroutines.** As has been discussed above, the considered planning problem can be solved either by some local search based metaheuristic or by a hybrid algorithm that produces a starting solution by solving one of the two mixed integer linear programs, MIPM or MILP, and then enhances this solution by some local search based optimization procedure. Four neighborhood operators, $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_1'$, and $\mathcal{N}_2'$, as described below, will be used for this purpose.

3.2.1. *Neighborhood operators.* Consider an arbitrary arrival plan $\sigma = [s_1, ..., s_n]$ and a sequence $t_1, ..., t_n$ of arrival days which are listed in a nondecreasing order and which are obtained by changing a single arrival day in σ, say by changing the arrival day s_g of some train-set g. For each $1 \le j \le n$, let $n(j)$ be the train-set that

arrives at t_j, and let $k(j)$ be the type of $n(j)$. Let t_i be the new arrival day assigned to the train-set g. The replacement of s_g by t_i results in a new feasible arrival plan if

(p1) $i = 1$ and $t_1 + \tau_{k(1)} \leq t_2$;
(p2) $1 < i < n$, $t_{i-1} + \tau_{k(i-1)} \leq t_i$ and $t_i + \tau_{k(i)} \leq t_{i+1}$;
(p3) $i = n$ and $t_{n-1} + \tau_{k(n-1)} \leq t_n$.

The neighborhood explored by the operator $\mathcal{N}_1$ is comprised of all feasible arrival plans (solutions) that can be obtained from the input arrival plan σ by assigning a different arrival day to a single train-set. In other words, the neighborhood explored by the operator $\mathcal{N}_1$ is comprised of all arrival plans that are obtained when the change of a single arrival day in σ results either in (p1), or (p2), or (p3). The benefit of using $\mathcal{N}_1$ is in the fast evaluation of each solution in the neighborhood because each solution is obtained by changing only a single arrival day and this change does not affect any other arrival days. The operator $\mathcal{N}_2$ explores all solutions that can be obtained by changing any two arrival days of the input arrival plan. Similar to $\mathcal{N}_1$, these two changes must not affect any other arrival days in the input arrival plan.

Other two operators $\mathcal{N}_1'$ and $\mathcal{N}_2'$ are similar to $\mathcal{N}_1$ and $\mathcal{N}_2$ but their neighborhoods are constructed without the restriction that the change of an arrival day in σ does not affect any unchanged days in σ. More specifically, the operator $\mathcal{N}_1'$ explores the neighborhood comprised of the solutions that result from a change of a single arrival day in the input arrival plan, but in contrast to $\mathcal{N}_1$, this change is allowed to violate all three feasibility conditions (p1), (p2), and (p3). In such case, the set of arrival days is transformed into a feasible arrival plan by the algorithm TRANSFORMATION and its two subroutines LEFT and RIGHT. Similarly, the operator $\mathcal{N}_2'$ explores all solutions that can be obtained by changing any two arrival days in the input arrival plan, but in contrast to $\mathcal{N}_2$, these two changes may affect some other arrival days in the input solution. In such case, analogously to $\mathcal{N}_1'$, the resultant set of arrival days is transformed into a feasible arrival plan using the same algorithm TRANSFORMATION.

Algorithm 2 TRANSFORMATION

```
if t_i < s_g then
    LEFT(i, t_1, ..., t_n)                    ▷ Algorithms 3
else
    RIGHT(i, t_1, ..., t_n)                   ▷ Algorithm 4
end if
for j = 1; j ≤ n; j++ do
    s_{n(j)} = t_j
end for
return [s_1, ..., s_n]
```

Algorithm 3 LEFT

$t_i^{'} = t_i$
if $i > 1$ **then**
  for $j = i - 1; j > 0; j$-- **do**
    $t_j^{'} = \min\{t_j, t_{j+1}^{'} - \tau_{k(j)}\}$
  end for
  if $t_1^{'} < 0$ **then**
    $t_1 = 0$
    for $j = 2; j \leq i; j$++ **do**
      $t_j = t_{j-1} + \tau_{k(j-1)}$
    end for
  else
    for $j = 1; j \leq i; j$++ **do**
      $t_j = t_j^{'}$
    end for
  end if
end if
if $i < n$ **then**
  for $j = i; j < n; j$++ **do**
    $t_{j+1} = \max\{t_j + \tau_{k(j)}, t_{j+1}\}$
  end for
end if
return $[t_1, ..., t_n]$

Algorithm 4 RIGHT

$t_i^{'} = t_i$
if $i < n$ **then**
  for $j = i; j < n; j$++ **do**
    $t_{j+1}^{'} = \max\{t_{j+1}, t_j^{'} + \tau_{k(j)}\}$
  end for
  if $t_n^{'} > T - 1$ **then**
    $t_n = T - 1$
    for $j = n - 1; j \geq i; j$-- **do**
      $t_j = t_{j+1} - \tau_{k(j)}$
    end for
  else
    for $j = i + 1; j \leq n; j$++ **do**
      $t_j = t_j^{'}$
    end for
  end if
end if
if $i > 1$ **then**
  for $j = i; j > 1; j$-- **do**
    $t_{j-1} = \min\{t_j - \tau_{k(j-1)}, t_{j-1}\}$
  end for
end if
return $[t_1, ..., t_n]$

Let $\mathfrak{N}$ be one of the four operators, $\mathcal{N}_1$, or $\mathcal{N}_1'$, or $\mathcal{N}_2$, or $\mathcal{N}_2'$. For each solution $\tilde{\sigma} = [\tilde{s}_1, ..., \tilde{s}_n]$ in the neighborhood of an input arrival plan σ, the operator $\mathfrak{N}$ computes the value of the objective function

$$\alpha G_1(\tilde{\sigma}) + \beta G_2(\tilde{\sigma}) = \alpha \sum_{1 \leq j \leq n} g_j(\tilde{s}_j) + \beta \sum_{t=0}^{T-1} \Big(\mathbb{E}\left[\max\{0, \delta_t(W_t(\tilde{\sigma}) - C_t)\}\right] + \sum_{k=1}^{m} \mathbb{E}\left[\max\{0, \delta_{kt}(W_t^k(\tilde{\sigma}) - C_{kt})\}\right] \Big). \tag{26}$$

Let $\hat{\sigma}$ be a solution in the neighborhood of σ with the smallest value of the objective function. Denote by $\mathfrak{N}(\sigma)$ the output solution produced by the operator $\mathfrak{N}$. Then

$$\mathfrak{N}(\sigma) = \begin{cases} \sigma & \text{if } \alpha G_1(\sigma) + \beta G_2(\sigma) \leq \alpha G_1(\hat{\sigma}) + \beta G_2(\hat{\sigma}) \\ \hat{\sigma} & \text{otherwise} \end{cases} \tag{27}$$

The Algorithm 5 below outlines the local search procedure for an input arrival plan σ. If $\mathfrak{N}$ is $\mathcal{N}_i$, where $i \in \{1, 2\}$, then this procedure will be referred to as LSi. If $\mathfrak{N}$ is $\mathcal{N}_i'$, where $i \in \{1, 2\}$, this procedure will be referred to as LSi$'$.

Algorithm 5 Local Search

1: **repeat**
2: $\bar{\sigma} = \sigma$
3: $\sigma = \mathfrak{N}(\sigma)$
4: **until** $\alpha G_1(\sigma) + \beta G_2(\sigma) == \alpha G_1(\bar{\sigma}) + \beta G_2(\bar{\sigma})$
5: **return** $\bar{\sigma}$

The four search operators, mentioned above, can be combined in different ways. In this paper, four options are considered: (1) LS1 is a local search that uses only the operator $\mathcal{N}_1$; (2) LS1$'$ is a local search that uses only the operator $\mathcal{N}_1'$; (3) Sequential Local Search (SLS) applies LS1 to the input arrival plan and when LS1 terminates applies LS2 to the output of LS1; and (4) SLS$'$ applies LS1$'$ to the input arrival plan and when LS1$'$ terminates applies LS2$'$ to the output of LS1$'$. Each option has its own merit as demonstrated in the computational study.

3.2.2. *Evaluation of solutions in a neighborhood.* The main computational burden in many local search algorithms is the evaluation of the elements constituting a neighborhood. The local search procedures LS1, LS2, LS1$'$ and LS2$'$ are no exception. Therefore, an algorithm for computing (26) for each element of the neighborhood of an input solution is the key factor that determines the efficiency of these optimization procedures.

As (14) indicates, in order to recompute (26) efficiently, one requires a fast algorithm for recomputing the probability mass functions of the random variables W_t, W_t^k for all $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$. All these random variables have the same nature - they are sums of Bernoulli random variables. Given the structure of neighborhoods explored by the operators $\mathcal{N}_1$ and $\mathcal{N}_2$, each element in the neighborhood of the input arrival plan can be obtained by changing the arrival day of one or two trains-sets in this input solution. For each such train-set, the change in the arrival day results in the change of one of the Bernoulli random variables in the sums defining the random variables W_t, W_t^k for all $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$.

This replacement can be viewed as the elimination of one of the Bernoulli random variables from the sum followed by the addition of another Bernoulli random variable to the result of the elimination. The Algorithm 6 and Algorithm 7 below do this efficiently for any probability mass function of a random variable that is a sum of Bernoulli random variables, and therefore, are used for recomputing probability mass functions of the random variables W_t, W_t^k for all $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$.

Let p be the success probability of the Bernoulli random variable that is to be eliminated, PMF be the original probability mass function, and new_PMF be the probability mass function resulted from the elimination of this Bernoulli random variable. The probability mass function new_PMF is defined for all integers $0 \leq i \leq n-1$, whereas the probability mass function PMF is defined for all integers $0 \leq i \leq n$. For each i from the domain of new_PMF, if $p = 0$, then $new\_PMF(i) = PMF(i)$, whereas if $p = 1$, then $new\_PMF(i) = PMF(i+1)$. If $0 < p < 1$, then

$$PMF(0) = (1-p)\ new\_PMF(0)$$

and, for $1 \leq i \leq n-1$,

$$PMF(i) = p\ new\_PMF(i-1) + (1-p)\ new\_PMF(i)$$

This observations lead to the Algorithm 6 below.

Algorithm 6 Single Train-set Removal (STR)

```
1: if 0 < p < 1 then
2:     new_PMF(0) = PMF(0) / (1 − p)
3:     for i from 1 to n - 1 do
4:         new_PMF(i) = [PMF(i) − new_PMF(i − 1) ∗ p] / (1 − p)
5:     end for
6: else
7:     for i from 0 to n - 1 do
8:         if p = 1 then
9:             new_PMF(i) = PMF(i + 1)
10:        else
11:            new_PMF(i) = PMF(i)
12:        end if
13:    end for
14: end if
15: return new_PMF
```

Let p be the success probability of the Bernoulli random variable that is to be added to a sum of Bernoulli random variables which has the probability mass function PMF. Let new_PMF be the probability mass function of the sum after this addition. The probability mass function new_PMF is defined for all integers $0 \leq i \leq n$, whereas the probability mass function PMF is defined for all integers $0 \leq i \leq n-1$. The reasoning similar to the above lead to the Algorithm 7.

3.3. **Iterated local search.** Iterated local search (ILS) is one of the commonly used metaheuristics which was successful in solving a wide range of optimization problems [11]. The Algorithm 8 below outlines this metaheuristic as it was implemented and used in the computational experiments, the results of which are

Algorithm 7 Single Train-set Addition (STA)

if $0 < p < 1$ **then**
  $new\_PMF(0) = PMF(0) * (1 - p)$
  for i from 1 to $n - 1$ **do**
    $new\_PMF(i) = PMF(i) * (1 - p) + PMF(i - 1) * p$
  end for
  $new\_PMF(n) = PMF(n - 1) * p$
end if
if $p = 0$ **then**
  $new\_PMF(n) = 0$
  for i from 0 to $n - 1$ **do**
    $new\_PMF(i) = PMF(i)$
  end for
end if
if $p = 1$ **then**
  $new\_PMF(0) = 0$
  for i from 1 to n **do**
    $new\_PMF(i) = PMF(i - 1)$
  end for
end if
return new_PMF

reported in Section 4. In this pseudocode, G is the objective function (4) and the parameter U specifies the maximum permissible number of consecutive unsuccessful attempts to improve the current best known arrival plan σ^*.

The Algorithm 8 interchangeably invokes two subroutines, SEARCH and PERTURB. In some computational experiments, reported in Section 4, the subroutine SEARCH is a local search procedure LS1 or LS1$'$ (see Algorithm 5), whereas in the others, the subroutine SEARCH is the sequential local search SLS or SLS$'$. The subroutine PERTURB randomly chooses three train-sets and one by one assigns to them new arrival days without violating the feasibility (that is, without violating the restrictions on the duration of time intervals between any two consecutive arrivals of train-sets). In the process of assigning the arrival days to the three selected train-sets, the subroutine PERTURB does not take into account the new value of the objective function G. In what follows, for any arrival plan $\tilde{\sigma}$, SEARCH($\tilde{\sigma}$) and PERTURB($\tilde{\sigma}$) denote the output of SEARCH and PERTURB, respectively, resulted from their application to $\tilde{\sigma}$. The arrival plan σ in SEARCH(σ) in line 1 is a feasible (in terms of the time between the consecutive arrivals) input solution.

An input arrival plan for the ILS is generated either by solving one of the two mixed integer linear programs, MIPM or MILP, or by the heuristic INITIAL described below. The heuristic INITIAL chooses randomly n different arrival days $t_1 < ... < t_n$ and associates randomly with each t_i one of m types of train-sets in such a manner that each type $1 \leq k \leq m$ receives $|F^k|$ arrival days. Denote by $k(t_i)$ the type associated with t_i. If, for each $1 \leq i < n$,

$$t_i + \tau_{k(t_i)} \leq t_{i+1},$$

then the arrival days are feasible. If they are not feasible, then the heuristic INITIAL transforms the generated arrival days into feasible arrival days, using the

Algorithm 8 Iterated Local Search

$\sigma^* = \text{SEARCH}(\sigma)$
$u = 0$
while $u \leq U$ **do**
  $\sigma = \text{PERTURB}\ (\sigma^*)$
  $\sigma = \text{SEARCH}\ (\sigma)$
  if $G(\sigma^*) > G(\sigma)$ **then**
    $\sigma^* = \sigma$
    $u = 0$
  else
    $u = u + 1$
  end if
end while
return σ^*

Algorithm 9 below. After obtaining feasible arrival days, the heuristic INITIAL generates an arrival plan by assigning the days associated with each type of train-sets to the train-sets of this type in the increasing order of their preferred days θ_j.

Algorithm 9 Arrivals Adjustment

for i from $n-1$ to 1 **do**
  $t_i = \min[t_i, t_{i+1} - \tau_{k(t_i)}]$
end for
if $t_1 < 0$ **then**
  $t_1 = 0$
  for i from 2 to n **do**
    $t_i = \max[t_i, t_{i-1} + \tau_{k(t_{i-1})}]$
  end for
end if
return $t_1, ..., t_n$

4. **Computational results.** The proposed solution approaches are tested and evaluated on data provided by one of the leading maintenance centers in Australia. The planning horizon is one year. There are 35 train-sets of 3 different types. The parameters of train-sets in F^k, $1 \leq k \leq 3$ are given in Table 1. The column '$|F^k|$' reports the total number of train-sets of the same type k; the column 'τ_k' gives the number of days a train-set of the corresponding type spends on the first operation line. Since an increase in transport demand is often expected during public holidays, the number of train-sets of type k which can dwell at the maintenance center simultaneously during these days is normally less than that on any other days of the year. The out-of-service limit 'C_{kt}' for type k on day t is reported in column 'non-PH' if day t is not a public holiday, and in column 'PH' if day t is a public holiday, where PH is an abbreviation of public holiday.

TABLE 1. Parameters for the train types.

Train type k	$\lvert F^k \rvert$	τ_k	C_{kt}	
			non-PH	PH
1	25	4	3	1
2	5	5	2	1
3	5	5	1	1

For any day t, the capacity of the maintenance center imposes a limit $C_t = 5$, and the daily penalty factor δ_t is fixed at 1. For any $1 \leq k \leq 3$, and any day t, the daily penalty factor $\delta_{kt} = 1$ if t is not a public holiday, and $\delta_{kt} = 10$ if t is a public holiday. A larger penalty is applied for public holidays because it is more undesirable to violate the given limit C_{kt} during these periods. With a larger penalty factor δ_{kt}, the violation of the permissible number of train-sets of type k that can dwell at the maintenance center simultaneously on public holidays is still possible in the optimal solution.

The permissible deviation from the preferred day of the commencement of maintenance is 14 days, i.e. $\Delta = 14$. The penalty factors for the violation of time windows are chosen as $\lambda_1 = \lambda_2 = 1$.

As has been discussed in Section 3.1, the random cycle times are modeled using beta-PERT distribution with the minimum, most likely, and maximum values as given in Table 2 for each train type k. Note that the beta-PERT distribution is a continuous probability distribution. So for the computational experiments, beta-PERT distribution is discretized into days.

TABLE 2. Parameters of probability distribution for cycle time by train types.

Train type	Minimum	Most likely	Maximum	Distribution
1	20	25	40	beta-PERT
2	27	30	46	beta-PERT
3	29	30	52	beta-PERT

Extensive experiments were performed on numerous settings for the weights α and β. The choice of values presented in Table 3 corresponds to the appropriate weight coefficients that can generate solutions representing typical scenarios. For case 1, a large relative weight is assigned to the second component of the objective function $G_2(\sigma)$, and the optimization procedures aim at minimizing the expected penalties for the violation of the limits C_t and C_{kt}. The importance of the objective $G_1(\sigma)$ increases when proceeding from case 1 to case 9.

TABLE 3. Assignment of α and β for all the cases.

Case	α	β	Case	α	β
1	1	1000	6	1	100
2	1	300	7	1	50
3	1	200	8	1	10
4	1	180	9	1	1
5	1	150			

All algorithms are implemented in Python 2.7. The mixed integer linear programs are solved with IBM ILOG CPLEX 12.7 via the mathematical programming modeling language PuLP [14]. All tests are run on a computer with Intel i5-6300U 2.4GHz processor and 8GB of RAM.

4.1. **Comparison of the performance of MIPM and MILP.** Table 4 compares the results of the two mixed integer linear programs. As has been discussed in Section 2.3, solving the MIPM produces a lower bound which is reported under the column titled 'LB'. The column 'Time' gives the running time (in seconds) by CPLEX to obtain an optimal solution. For all cases, the objective values of the solutions are computed as described in Section 2.2 and reported under the column titled 'Obj.'. The relative gap is calculated as $\%\text{Rel} = (\text{Obj.} - \text{LB})/\text{LB} \times 100$.

The results in Table 4 shows that the MILP can be solved in short computation time, yet the MIPM has the advantage of providing a lower bound. It is observed that the MIPM gives better solutions in 6 out of 9 cases, and on average 0.54% better than the MILP. For all cases, CPLEX obtains an optimal solution to MILP in less than 11 seconds, whereas more time is needed to solve the MIPM to optimality. By investigating the output of CPLEX in case 1 which takes the longest time, it was found that the optimal solution in fact was obtained in less than 2 minutes. The remaining time was taken by CPLEX for proving optimality, which is a common behavior of this software. As the relative weight of β decreases, the computation time of MIPM reduces substantially. This observation suggests that the second component of the objective function is harder to optimize for the MIPM.

TABLE 4. Comparison of the performance of MIPM and MILP

Case	LB	MIPM			MILP		
		Time	Obj.	%Rel	Time	Obj.	%Rel
1	156,744	2,078	206,060	31.46	7	**201,203**	**28.36**
2	62,612	235	**77,764**	**24.20**	7	80,353	28.33
3	48,629	68	59,178	21.69	8	**59,103**	**21.54**
4	45,768	100	**55,368**	**20.98**	8	56,322	23.06
5	40,190	54	**48,447**	**20.54**	11	49,369	22.84
6	30,789	46	**36,245**	**17.72**	8	36,572	18.78
7	20,940	41	23,624	12.82	8	**23,337**	**11.45**
8	10,594	39	**10,753**	**1.50**	8	10,818	2.11
9	6,706	38	**6,725**	**0.28**	8	6,748	0.63
Average	46,997	300	58,240	16.80	8	58,203	17.46

The improvements in solution quality by the hybrid two-stage optimization procedure that combines the mixed integer linear program with the local search LS1 and the Sequential Local Search (SLS) are reported in Tables 5 and 6, respectively. In both tables, the column 'LS Time' gives the computation time (in seconds) of the local search procedure. Results in Table 5 show that the local search LS1 is effective in improving the solutions of both models for all cases. On average, the relative gap is reduced by 30.56% for MIPM and 29.71% for MILP. The behavior of the local search is consistent for both models: the best performance is achieved in case 5 with a change in the objective value of 11.48% for MIPM, and 11.76% for MILP; whereas the worst performance is observed in case 8 at 0.004% and 0.23%, respectively. The

TABLE 5. Improvements in solution quality of MIPM and MILP by the local search LS1.

Case	MIPM-LS1			MILP-LS1		
	LS Time	Obj.	%Rel	LS Time	Obj.	%Rel
1	46.32	**189,551**	**20.93**	19.59	192,637	22.90
2	44.29	**72,584**	**15.93**	32.72	75,368	20.37
3	36.11	56,483	16.15	46.72	**55,102**	**13.31**
4	28.95	53,125	16.07	60.71	**51,999**	**13.61**
5	67.76	**42,884**	**6.70**	46.84	44,547	10.84
6	22.06	**32,971**	**7.09**	13.01	35,356	14.83
7	7.44	22,948	9.59	26.11	**22,905**	**9.38**
8	10.56	**10,752**	**1.49**	15.11	10,793	1.88
9	6.89	**6,724**	**0.27**	6.85	6,732	0.39
Average	30.04	54,225	10.47	29.74	55,049	11.95

TABLE 6. Improvements in solution quality of MIPM and MILP by the sequential local search SLS.

Case	MIPM-SLS			MILP-SLS		
	LS Time	Obj.	%Rel	LS Time	Obj.	%Rel
1	879	**189,551**	**20.93**	2,366	190,584	21.59
2	871	**72,584**	**15.93**	3,593	72,868	16.38
3	884	56,483	16.15	931	**54,766**	**12.62**
4	802	53,125	16.07	3,245	**51,208**	**11.89**
5	862	**42,884**	**6.70**	3,589	44,173	9.91
6	838	**32,971**	**7.09**	3,592	34,911	13.39
7	3,196	**22,631**	**8.08**	899	22,905	9.38
8	699	**10,752**	**1.49**	687	10,793	1.88
9	708	**6,724**	**0.27**	749	6,732	0.39
Average	1,082	54,189	10.30	2,183	54,327	10.82

MIPM with LS1 performs better than the MILP with LS1. The former produces solutions that are, on average, 1.48% better than the latter, with the same average time of 30 seconds. Moreover, the local search applied to a better initial solution does not necessarily produce a better local optimum, as demonstrated by the result of case 1.

If the SLS is used to enhance the starting solutions of MIPM and MILP, results in Table 6 show that the MIPM with SLS yields better solutions in shorter running times. For MIPM, the search in the neighborhood explored by the operator $\mathcal{N}_2$ finds better solution in only one of the nine cases, i.e. case 7. For MILP, the SLS is seen to be especially useful on cases with large relative weight β. Since the MIPM produces better results over the MILP for most cases, MIPM is used in the remainder of the computational experiments.

4.2. **Comparison of hybrid ILS and multi-start ILS.** In this section, eight algorithms, namely hybrid ILS with LS1, hybrid ILS with LS1′, hybrid ILS with SLS, hybrid ILS with SLS′, multi-start ILS with LS1, multi-start ILS with LS1′,

multi-start ILS with SLS, and multi-start ILS with SLS$'$, are compared by means of computational experiments. For all eight algorithms, a maximum permissible number of iterations without improvement $U = 20$ is used. The computational results are reported in Tables 7 - 10. In the preliminary testing, performing an exhaustive search on the neighborhood by $\mathcal{N}_2$ and $\mathcal{N}_2'$ is too time consuming, which significantly reduces the number of perturbation in the ILS procedure due to the time limit of 1 hour. As a result, the ability of ILS to escape the local optimum is severely impaired. Therefore instead of allowing the arrival days of two train-sets to be changed to any feasible days from $\{0, ..., T-1\}$, the newly assigned arrival day t of each such train-set j is selected from the range $s_j - 36 \leq t \leq s_j + 36$, where s_j is obtained from the input arrival plan σ. Such restriction substantially reduces the size of the search space so that the neighborhood can be explored in a reasonable time. The reduced structure of neighborhood explored by $\mathcal{N}_2$ and $\mathcal{N}_2'$ is employed in the computational experiments of hybrid ILS with SLS, hybrid ILS with SLS$'$, multi-start ILS with SLS, and multi-start ILS with SLS$'$. We will discuss the results of hybrid ILS first.

The hybrid ILS is implemented by solving the MIPM, which provides an input arrival plan to the iterated local search in Algorithm 8. Because of the faster evaluation of solutions in a neighborhood and the smaller neighborhood size, the versions of the hybrid iterated local search with the operators $\mathcal{N}_1$ and $\mathcal{N}_2$ are faster than their counterparts with the operators $\mathcal{N}_1'$ and $\mathcal{N}_2'$ often at a cost of inferior solution quality. The computational experiments took this into account and ran versions with the operators $\mathcal{N}_1$ and $\mathcal{N}_2$ with one hour limit on the permissible computation time, recorded for each such optimization procedure the average of the actual computation times for ten runs, and then set this recorded average time as the limit on the computation time for the version with the corresponding operators $\mathcal{N}_1'$ and/or $\mathcal{N}_2'$.

For each case, i.e. for each choice of the parameters α and β, Tables 7 and 8 present the average computation time in seconds (Time), the average value of the objective function (Obj.), and the average relative gap (%Rel) obtained for ten runs of the hybrid ILS. The column In. Obj. displays the value of the objective function obtained by solving the MIPM. Table 7 indicates that the version with $\mathcal{N}_1'$ obtains better quality solutions in six of the nine cases, with the average relative gap improving from 9.12% to 5.84%. Table 8 also indicates the superior performance of the version with $\mathcal{N}_1'$ and $\mathcal{N}_2'$ in comparison with the version with $\mathcal{N}_1$ and $\mathcal{N}_2$.

The output of the multi-start ILS is the best output obtained by the application of the Algorithm 8 to five different input arrival plans generated by the heuristic INITIAL. These five applications of the Algorithm 8 constitute one run of the multi-start ILS. In the course of the computational experiments, the duration of each run of the multi-start ILS was limited by one hour. For each case, i.e. for each choice of the parameters α and β, Tables 9 and 10 present the average required time (Time), average value of the objective function (Obj.), and average relative gap (%Rel) obtained for ten runs of the multi-start ILS. The column In. Obj. contains the average value of the objective function for the input arrival plans that resulted in the output of the multi-start ILS.

In Table 9, the multi-start ILS with LS1 is superior to the multi-start ILS with LS1$'$ in both time and solution quality. The same observation can be seen in Table 10, in which the multi-start ILS with SLS showing better performance over the version with SLS$'$. It is worth noting that although the multi-start ILS algorithms begin

TABLE 7. Performance of hybrid ILS with LS1 and LS1′.

Case	In. Obj.	Time	Hybrid ILS with LS1		Hybrid ILS with LS1′	
			Obj.	%Rel	Obj.	%Rel
1	206,060	2,685	189,487	20.89	**180,530**	**15.17**
2	77,764	864	71,715	14.54	**67,092**	**7.16**
3	59,178	581	55,624	14.38	**51,344**	**5.58**
4	55,368	932	50,780	10.95	**47,803**	**4.45**
5	48,447	805	42,416	5.54	**41,666**	**3.67**
6	36,245	472	32,828	6.62	**32,371**	**5.14**
7	23,624	817	**22,494**	**7.42**	22,948	9.59
8	10,753	483	10,752	1.49	10,752	1.49
9	6,725	355	6,724	0.27	6,724	0.27
Average	58,240	888	53,647	9.12	51,248	5.84

TABLE 8. Performance of hybrid ILS with SLS and SLS′.

Case	In. Obj.	Time	hybrid ILS with SLS		hybrid ILS with SLS′	
			Obj.	%Rel	Obj.	%Rel
1	206,060	3,600	187,992	19.94	**179,847**	**14.74**
2	77,764	2,482	70,845	13.15	**66,967**	**6.96**
3	59,178	1,640	55,100	13.31	**51,324**	**5.54**
4	55,368	2,281	51,459	12.43	**47,785**	**4.41**
5	48,447	2,496	42,193	4.98	**41,432**	**3.09**
6	36,245	1,634	32,968	7.08	**32,371**	**5.14**
7	23,624	1,413	**22,613**	**7.99**	22,948	9.59
8	10,753	1,191	10,752	1.49	10,752	1.49
9	6,725	1,078	6,724	0.27	6,724	0.27
Average	58,240	1,979	53,405	8.96	51,128	5.69

TABLE 9. Performance of multi-start ILS with LS1 and LS1′.

Case	multi-start ILS with LS1				multi-start ILS with LS1′			
	Time	In. Obj.	Obj.	%Rel	Time	In. Obj.	Obj.	%Rel
1	2,904	731,900	**200,646**	**28.01**	3,600	701,192	221,792	41.50
2	3,206	246,939	**76,713**	**22.52**	3,600	228,459	82,564	31.87
3	3,525	205,225	**56,717**	**16.63**	3,600	161,630	62,662	28.86
4	3,101	157,225	**52,039**	**13.70**	3,600	141,340	57,548	25.74
5	2,946	155,164	**46,623**	**16.01**	3,600	115,812	48,889	21.65
6	2,996	110,872	**35,309**	**14.68**	3,600	97,424	39,585	28.57
7	2,736	80,850	**23,722**	**13.29**	3,600	60,620	29,355	40.19
8	2,548	50,200	**11,109**	**4.86**	3,600	29,562	14,635	38.15
9	2,386	40,485	**6,917**	**3.14**	3,600	30,494	11,311	68.67
Average	2,928	197,651	56,644	14.76	3,600	174,059	63,149	36.13

TABLE 10. Performance of multi-start ILS with SLS and SLS′.

Case	multi-start ILS with SLS				multi-start ILS with SLS′			
	Time	In. Obj.	Obj.	%Rel	Time	In. Obj.	Obj.	%Rel
1	3,600	704,669	**207,738**	**32.53**	3,600	792,732	229,339	46.31
2	3,600	234,564	**76,228**	**21.75**	3,600	242,8870	80,410	28.43
3	3,600	176,729	**55,596**	**14.33**	3,600	149,560	61,766	27.01
4	3,600	165,457	**52,054**	**13.73**	3,600	139,882	56,795	24.09
5	3,600	141,360	**45,025**	**12.03**	3,600	112,386	50,796	26.39
6	3,600	112,397	**35,946**	**16.75**	3,600	109,447	43,065	39.87
7	3,600	78,051	**23,870**	**13.99**	3,600	59,009	29,110	39.02
8	3,600	57,798	**11,192**	**5.64**	3,600	30,803	15,081	42.35
9	3,600	35,437	**6,903**	**2.93**	3,600	28,582	9,512	41.85
Average	3,600	189,607	57,173	14.85	3,600	185,030	63,986	35.04

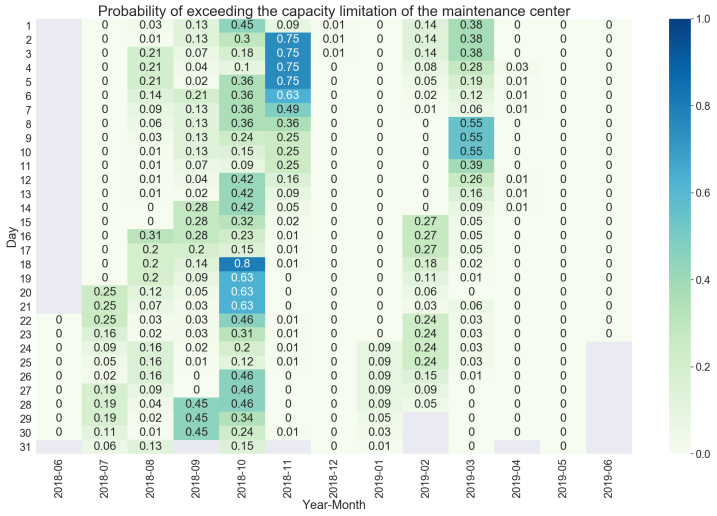
TABLE 11. Summary of the effects of the different neighborhoods.

Tables	$\mathcal{N}$	$\mathcal{N}'$	EQUAL	Winner
7	1	6	2	MIPM + ILS + $\mathcal{N}_1'$
8	1	6	2	MIPM + ILS + $\mathcal{N}_1'$ + $\mathcal{N}_2'$
9	9	0	0	multi-start ILS + $\mathcal{N}_1$
10	9	0	0	multi-start ILS + $\mathcal{N}_1$ + $\mathcal{N}_2$

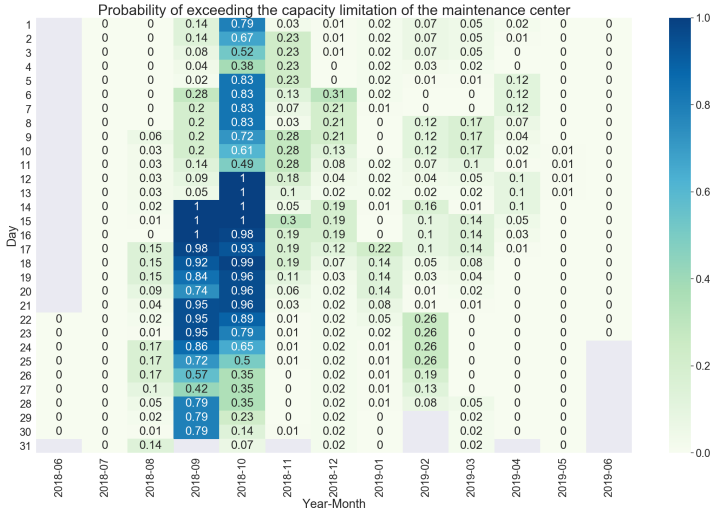
with poor initial solutions, significantly better results are achieved after the local search based enhancement procedures, with the best reported average improvement of 72% observed in the multi-start ILS with LS1.

In summary, the comparison of the eight optimization procedures indicates that the best solution quality is obtained by the combination of the mixed integer linear program MIPM and the iterated local search with the operators $\mathcal{N}_1'$ and $\mathcal{N}_2'$. The comparison of the different neighborhoods is summarized in Table 11. The column Tables contains references to the tables that present the results of computational experiments. The column $\mathcal{N}$ contain the number of cases for which the neighborhood structure that does not require the subroutine TRANSFORMATION yields a better solution quality. The column $\mathcal{N}'$ contain the number of cases for which the neighborhood structure obtained, using the subroutine TRANSFORMATION, yields a better solution quality. The column EQUAL contains the number of cases when both neighborhood types produced the same solution quality.

4.3. **Visualization of quality of arrival plan.** During the negotiations between the rolling stock operator and the maintenance center, it is useful to have information about the risk of violating the center capacity which may occur as a result of the uncertain duration of maintenance. For this reason, a powerful visualization tool, based on the idea of heat map [17], is developed to provide insights into the risk over the planning horizon. Figures 1A and 1B show examples of the risk heat map associated with the arrival plans of cases 1 and 9, respectively. The horizontal axis indicates the month and year (for example, 2018-06 stands for June 2018), while the vertical axis indicates the day of the month. Each cell of the heat map



(A)



(B)

FIGURE 1. Heat maps displaying the probability of having more than 5 train-sets residing in the maintenance center for each day across the planning horizon of one year for cases (A) $\alpha = 1$, $\beta = 1000$; and (B) $\alpha = 1$, $\beta = 1$.

corresponds to a particular day in the planning horizon, and the probability of violating the limit is clearly stated in each cell. The color intensity reflects the level of risk whereby the darker the color, the higher the risk.

The resulting heat map in Figures 1A and 1B show a trade-off example in which the constructed arrival plan must prioritize either the technological restrictions of the maintenance center or the arrival time windows. Figure 1A considers the perspective of the maintenance center who is more concerned about keeping the number of train-sets below the capacity of the maintenance center. As a result, the total penalty for the violation of the capacity limitation is insignificant and it can be seen on Figure 1A that there are few days which have high probability of exceeding the center capacity. However, the total penalty for the violation of time windows, $G_1(\sigma)$, is 23,487. On the other hand, the heat map in Figure 1B is associated with an arrival plan $\sigma^{'}$ that is constructed considering the perspective of the rolling stock operator, the main concern of whom is to satisfy the arrival time windows. In this case, the total penalty $G_1(\sigma^{'})$ is only 6,218 but the maintenance center has a high risk of violating the capacity, i.e. it is harder to have an efficient operational plan.

5. **Conclusion.** The paper contributes to the existing body of literature on train maintenance by introducing a nonlinear programming problem that determines the arrival days of train-sets to a maintenance center, taking into account the uncertain duration of maintenance and the requirements specified by the rolling stock operator as well as the technological restrictions of the maintenance center. A fast method of evaluation of the objective function for any feasible solution of the nonlinear program is presented together with a mixed integer programming relaxation based on Jensen's inequality. This relaxation provides a lower bound on the optimal value of the objective function of the nonlinear program and generates a starting solution for the hybrid optimization procedure which enhances this solution by using iterated local search. This hybrid optimization procedure is compared with iterated local search metaheuristic diversified by the multi-start framework. The results of the computational experiments on real-world data warrant the implementation of the presented approach in the process of maintenance planning. Further research should be focused on the operational level of the maintenance planning for a shorter planning horizon and more detailed information about maintenance procedures.

Acknowledgments. We are grateful to the Editor and three anonymous referees for their constructive comments on the earlier versions of our paper.

REFERENCES

[1] S. Ahmed, Two-stage stochastic integer programming: A brief introduction, in *Wiley Encyclopedia of Operations Research and Management Science* (eds. J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh and J.C. Smith), John Wiley & Sons, (2011).

[2] G. Bayraksan and D. P. Morton, Assessing solution quality in stochastic programs via sampling, in *Informs 2009 Tutorials in Operations Research*, (2009), 102–122.

[3] P. Billingsley, *Probability and Measure*, 3^{rd} edition, John Wiley & Sons, New York, 1995.

[4] W. Biscarri, S. D. Zhao and R. J. Brunner, A simple and fast method for computing the Poisson binomial distribution function, *Computational Statistics & Data Analysis*, **122** (2018), 92–100.

[5] K. Doganay and M. Bohlin, Maintenance plan optimization for a train fleet, in *Computers in Railways XII*, (eds. B. Ning and C.A. Brebbia), WIT Press, (2010), 349–358.

[6] B. Fortz, M. Labbé, F. Louveaux and M. Poss, Stochastic binary problems with simple penalties for capacity constraints violations, *Mathematical Programming*, **138** (2013), 199–221.

[7] G. L. Giacco, D. Carillo, A. D'Ariano, D. Pacciarelli and A. G. Marín, Short-term rail rolling stock rostering and maintenance scheduling, *Transportation Research Procedia*, **3** (2014), 651–659.

[8] T. Homem-de Mello and G. Bayraksan, Monte Carlo sampling-based methods for stochastic optimization, *Surveys in Operations Research and Management Science*, **19** (2014), 56–85.
[9] Y. C. Lai, D. C. Fang and K. L. Huang, Optimizing rolling stock assignment and maintenance plan for passenger railway operations, *Computers & Industrial Engineering*, **85** (2015), 284–295.
[10] B. Lin, J. Wu, R. Lin, J. Wang, H. Wang and X. Zhang, Optimization of high-level preventive maintenance scheduling for high-speed trains, *Reliability Engineering & System Safety*, **183** (2019), 261–275.
[11] H. R. Lourenço, O. C. Martin, and T. Stützle, Iterated local search: framework and applications, in *Handbook of Metaheuristics* (eds. M. Gendreau and J. Potvin), 2^{nd} edition, Springer, Boston, (2010), 363–397.
[12] D. G. Malcolm, J. H. Rooseboom, C. E. Clark and W. Fazar, Application of a technique for research and development program evaluation, *Operations Research*, **7** (1959), 646–669.
[13] J. G. Pérez, M. Martín, C. García and M. Granero, Project management under uncertainty beyond beta: The generalized bicubic distribution, *Operations Research Perspectives*, **3** (2016), 67–76.
[14] S. Mitchell, M. O'Sullivan and I. Dunning, *PuLP: a Linear Programming Toolkit for Python*, 2011. Available from: `http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf`.
[15] C. Sriskandarajah, A. K. S. Jardine and C. K. Chan Maintenance scheduling of rolling stock using a genetic algorithm, *Journal of the Operational Research Society*, **49** (1998), 1130–1145.
[16] Sydney Trains, *Sydney Trains Annual Report 2017-18*, 2018. Available from: `https://www.transport.nsw.gov.au/news-and-events/reports-and-publications/sydney-trains-annual-reports`.
[17] M. Waskom, *Seaborn: v0.8.1*, 2017. Available from: `https://seaborn.pydata.org`.

Received January 2020; revised September 2020.

E-mail address: `hanyu.gu@uts.edu.au`
E-mail address: `hue.lam@student.uts.edu.au`
E-mail address: `yakov.zinder@uts.edu.au`