New heuristics for the stochastic tactical railway maintenance problem

New Heuristics for the Stochastic Tactical Railway Maintenance Problem

Mauro M. Baldi[a], Roberto Tadei[a,*], Franziska Heinicke[b], Axel Simroth[b]

[a] *Politecnico di Torino, Turin, Italy*
[b] *Fraunhofer IVI, Dresden, Germany*

Abstract

Efficient methods have been proposed in the literature for the management of a set of railway maintenance operations. However, these methods consider maintenance operations as deterministic and known *a priori*. In the stochastic tactical railway maintenance problem (STRMP), maintenance operations are not known in advance. In fact, since future track conditions can only be predicted, maintenance operations become stochastic. STRMP is based on a rolling horizon. For each month of the rolling horizon, an adaptive plan must be addressed. Each adaptive plan becomes deterministic, since it consists of a particular subproblem of the whole STRMP. Nevertheless, an exact resolution of each plan along the rolling horizon would be too time-consuming. Therefore, a heuristic approach that can provide efficient solutions within a reasonable computational time is required. Although STRMP has already been introduced in the literature, little work has been done in terms of solution methods and computational results. The main contributions of this paper include new methodology developments, a linear model for the deterministic subproblem, three efficient heuristics for the fast and effective resolution of each deterministic subproblem, and extensive computational results.

Keywords: railway maintenance, heuristics, greedy randomized adaptive search procedure, genetic algorithm.

1. Introduction

Development of an efficient plan for preventive maintenance is crucial in many fields [28, 30, 19]. The plethora of activities for which preventive maintenance must be scheduled includes railway transportation. To ensure a good daily service in terms of punctuality and safety, railway infrastructure managers have to plan and perform maintenance operations for whole railway networks. Most of these operations involve preventive maintenance at a tactical level within a medium time horizon, usually of 1 year. The aim of preventive maintenance activities is to control track failure probability to guarantee a stable and safe service and minimize maintenance costs. ACEM-Rail [26] is a European project to improve the optimization and automation of railway infrastructure maintenance. One of the objectives of the ACEM-Rail project is to develop new algorithms for efficient planning of railway infrastructure maintenance tasks based on stochastic data drawn from predictions. Several studies have addressed railway maintenance planning. Cheung et al. [9] studied the railway track possession assignment problem (RTPAP) using constraint satisfaction. The problem involves assigning railway tracks to scheduled maintenance tasks according to the satisfaction of a set of constraints. The RTPAP aim is to produce a plan that maximizes the assignment of jobs with the highest priority. To effectively achieve this goal, an engineering work track possession assignment system based on the CHIP constraint programming language [27] substituted manpower used to find a manual solution for the RTPAP. The performance of the system was ten times more efficient than the manual method, and its solutions are free of human errors. Budai et al. [6, 7] proposed a preventive maintenance scheduling problem in which routine activities and projects were scheduled within a given time horizon by minimizing possession costs. The authors provided an integer programming model for the problem and proved that it is $\mathcal{NP}$-hard. Moreover, they provided three fast but simple heuristics. To reduce the gap yielded by these heuristics, Budai et al. [8] tackled the same problem using genetic and memetic algorithms. van Zante-de Fokkert et al. [29] studied a problem in

*Corresponding author:
Department of Control and Computer Engineering, Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
Tel.: +39 0110907032, fax: +39 0110907099 E-mail: roberto.tadei@polito.it

which the whole network was divided into basic working zones named single-track grids. The authors proposed a method yielding a maintenance schedule in two phases. Moreover, they provided a mixed integer programming model with a lexicographic objective function that first minimizes the number of nights (i.e. the length of the plan) and then the sum of the maximum workloads scheduled.

Although these methods efficiently address railway maintenance planning at a tactical level, they deal with a known set of maintenance tasks to be scheduled within a given time horizon. However, within the predictive setting of the ACEM-Rail project, future track conditions are not known in advance and the development of the deterioration process can be only predicted. Therefore, maintenance tasks are affected by uncertainty resulting from unknown deterioration processes. The resulting problem involving creation and solution of an efficient adaptive maintenance plan to minimize overall costs when maintenance tasks are not known *a priori* but subject to uncertainty is the stochastic tactical railway maintenance problem (STRMP). We call the deterministic STRMP subproblem that the infrastructure manager has to solve in each month of the rolling horizon deterministic tactical railway maintenance problem (DTRMP). Very little has been proposed so far in the literature to address STRMP. Heinicke et al. [16] outlined a preliminary methodology in which capacity and risk constraints are treated as soft constraints. Nevertheless, their results were limited to three instances. The aim of this study was to complete the work started by Heinicke et al. [16] and provide a formal definition of the problem, improved heuristics to address the problem, and extensive computational results. We extend the preliminary work of Heinicke et al. [16] with a number of contributions. We provide a formal definition of the problem setting for both STRMP and DTRMP. Moreover we formulate an integer linear programming model for DTRMP and treat capacity and risk constraints as hard constraints. STRMP and DTRMP show a number of analogies with the bin packing problem (BPP) [18], in particular with one of the latest variants, namely the stochastic generalized BPP (SGBPP) [22], and its deterministic variant, the generalized BPP (GBPP) [2]. Using a linear model and exploiting the analogies with these BPPs, we present three efficient heuristics that can address DTRMP. The first heuristic, named ADAPTED FIRST FIT DECREASING (AFFD), is an adaptation of the FIRST FIT DECREASING (FFD) heuristic [18], which is widely used for BPPs. The second heuristic is a GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE (GRASP), which uses AFFD in each iteration of the algorithm. The last heuristic is a GENETIC ALGORITHM (GA) that uses GRASP in the initialization phase and AFFD in each iteration of the algorithm. We also present extensive computational results for 40 instances, each consisting of 36 plans, for a 3-year overall rolling horizon.

The remainder of the paper is organized as follows. In Section 2 we define STRMP. Section 3 describes DTRMP and the three heuristics developed to address it. In Section 4 we present computational results. Section 5 concludes.

2. The stochastic tactical railway maintenance problem

STRMP involves scheduling of predictive and preventive maintenance activities within a given time horizon (*planning horizon*) over a long-term *rolling horizon*. The goal is to bring the degradation process under control by performing appropriate maintenance activities at minimum cost.

A set of maintenance activities, called *warnings*, is assigned to resolve a set of glitches on the track. Each warning can be assigned to a temporal portion of the planning horizon, called a *time slot*. In general, one glitch can be resolved by more than one warning. The final assignment of warnings to time slots is called a *plan*.

Each warning is characterized by a cost, an amount of resources for resolving the warning, and a risk. Depending on the maintenance carried out, warnings can be at different degradation levels. The degradation process for warnings is modeled using a set of degradation levels and a Markov chain that defines the transition between the degradation levels. Therefore, the cost, the resource requirement, and the warning risk depend on the degradation level for a given warning. Because of the different probabilities for degradation levels in the time slots, the above values also depend on the time slot in which a warning is allocated. Each time slot is characterized by the resource amount available to resolve assigned warnings.

Maintenance planning also involves mandatory maintenance periods that require track possession called *fixed warnings*. Track possession clearly alters the quality of the service offered by railway operators. When a warning has a long working duration (longer than one night), the work has to be continued the next day and passenger traffic is disturbed. To have enough time to reroute passenger traffic, the maintenance manager has to book the track 6 months in advance. Therefore, warnings with a long working duration have to be allocated from at least time slot 7 (i.e., 6 months).

Each month new data are obtained from the maintenance management system and the plan is adapted, with reallocation of updated warnings and allocation of new warnings. At the beginning of any month within the whole rolling horizon, three events occur:

1. Warnings allocated in the current month are resolved and deleted from the planning process.
2. The track is measured and the degradation level for warnings in the current month becomes known. On the basis of these measurements, the transition probabilities for the degradation level of these warnings can be updated.
3. Track measurements can reveal possible failures. This implies the introduction of further warnings.

Therefore, a plan is created at the beginning of each month with predicted warnings, while resolved warnings are removed and no longer appear in future plans.

In general, given a Δ-month plan Π_i starting at month i and ending at month $i + \Delta - 1$, plan Π_{i+1} will start at month $i + 1$ and end at month $i + \Delta$ and will contain the warnings allocated to month $i + 1$ up to $i + \Delta - 1$ with updated probabilities and possible new warnings arising from track measurements. Thus, each month we produce a new plan based on the previous plan and on predicted future track conditions. This involves addressing a new static problem each month (DTRMP presented in Section 3.1) but with different data.

Because of uncertainty, it is impossible to know the future degradation level of warnings (i.e., the exact working effort in terms of resources and costs) when track possession is booked. Since booking of track possession implies future fixation of warnings to a time slot, this decision must be robust against uncertainty. In fact, if future track conditions were known, then warning information would be known and this would be enough to solve a deterministic problem (presented in Section 3.1) to find the best maintenance plan. However, as future track conditions are not known, warning information can only be predicted. The result is an adaptive stochastic problem. This is the innovative contribution of STRMP.

In the ACEM-Rail project, warnings are characterized using data provided by the Ferrovie del Gargano [13] operator on the San Severo–Peschici railroad in Italy (see Section 4.1 for detailed information on these data). The rolling horizon is 3 years (36 months) and the planning horizon is 1 year. Each planning horizon is made up of 12 time slots, whereby one time slot corresponds to one month. Therefore, without loss of generality, we use the words *month* and *time slot* interchangeably.

3. The deterministic tactical railway maintenance problem

In this section we describe DTRMP, which provides a new maintenance plan based on the previous plan and on predicted future track conditions.

3.1. The model

We define the following input data:

- T: set of time slots.
- m: total number of time slots.
- $d := |T| + 1$: dummy time slot hosting deferred warnings that cannot be allocated to any time slot $t \in T$.
- P: set of glitches on the railway track.
- W: set of warnings.
- $W_p \subseteq W$: set of warnings able to resolve a glitch $p \in P$. It is clear that $\bigcup_{p \in P} W_p = W$. Note that one warning might be able to resolve a number of glitches at the same time. Such warnings are called combined warnings.
- $W_f \subseteq W$: set of fixed warnings. These warnings require track possession and have to be booked 6 months ahead. We assume that $|T| > 6$.
- $C(w, t)$: expected cost for allocating warning $w \in W$ to $t \in T$.

- $C(w,d) := C(w,|T|+1)$: penalty cost for allocating $w \in W$ to the dummy time slot d, with $C(w,d) > C(w,t), \forall t \in T$.
- $R(w,t)$: expected resources requirement vector for warning $w \in W$ when allocated to time slot $t \in T$.
- $c(t)$: capacity of time slot $t \in T$.
- $S(w,t)$: risk associated with warning $w \in W$ and time slot $t \in T \cup \{d\}$.
- $S_{\max}$: maximum risk allowed.

Moreover, we define the decision binary variable $x(w,t)$ as equal to 1 if warning $w \in W$ is allocated to time slot $t \in T \cup \{d\}$, and 0 otherwise.

An integer linear model for DTRMP can then be formulated as follows:

$$\min \quad \sum_{w \in W} \sum_{t=1}^{|T|+1} C(w,\, t) \cdot x(w,\, t) \tag{1}$$

$$\text{s.t.} \quad S(w,t) \cdot x(w,t) \leq S_{\max} \qquad \forall w \in W,\ \forall t \in T \cup \{d\} \tag{2}$$

$$\sum_{t=1}^{6} x(w,\, t) = 0 \qquad \forall w \in W_f \tag{3}$$

$$\sum_{w \in W_p} \sum_{t=1}^{|T|+1} x(w,\, t) = 1 \qquad \forall p \in P \tag{4}$$

$$\sum_{t=1}^{|T|+1} x(w,\, t) \leq 1 \qquad \forall w \in W \tag{5}$$

$$\sum_{w \in W} R(w,t) \cdot x(w,t) \leq c(t) \qquad \forall t \in T \tag{6}$$

$$x(w,t) \in \{0,1\} \qquad \forall w \in W,\ \forall t \in T \cup \{d\}. \tag{7}$$

In this model, (1) minimizes the sum of the expected costs of the allocated warnings and the penalty costs of the deferred warnings. Constraint (2) is the risk constraints. Constraint (3) ensures that all fixed warning are allocated starting from time slot 7, as they require track possession, which must be booked 6 months ahead. Constraint (4) states that exactly one warning able to resolve a glitch must be allocated. Constraint (5) prevents allocation of a warning to more than one time slot. Constraint (6) is the capacity constraint and (7) is the integrality constraint.

As stated in the Introduction, DTRMP shows a number of affinities with BPPs. All BPPs consist of a set of items characterized by volume to be loaded into a set of bins. Bins are characterized by a capacity, as in the original BPP [18], and by a cost, as in the variable size BPP [14] and variable cost and size BPP [10]. Warnings in DTRMP correspond to BPP items. Similarly, time slots in DTRMP correspond to BPP bins. Moreover, all BPPs have capacity constraints like (2) [20] and require items to be loaded in a similar way to (4). Finally, in GBPP [2, 4, 3], items are also characterized by profits that depend on the bin into which items are loaded [23]. Similar behavior can be observed in DTRMP, whereby warning costs depend on the time slot to which they are allocated. These strong analogies among BPPs and DTRMP motivated us to exploit model (1)–(7) in concert with a widely used BPP heuristic, namely FFD. Therefore, model (1)–(7) and the FFD heuristic are the starting point for the development of our heuristics, which are described in the next section.

3.2. The heuristics

In this section we present three efficient heuristics for addressing STRMP that consistently solve DTRMP.

The first heuristic, AFFD, is a variant of the heuristic used by Garey et al. [15] for the original BPP. The second is a GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE (GRASP), which exploits the AFFD heuristic as an internal procedure. Finally, we present a GA algorithm that uses GRASP in the initialization procedure and AFFD in each subsequent iteration.

3.2.1. The AFFD *heuristic*

This heuristic is a generalization of the original FFD heuristic introduced by Garey et al. [15] to address BPP. In the original FFD heuristic, items are sorted by decreasing volume and each sorted item is accommodated in the first bin able to contain it. An item can be accommodated in a bin if the bin has enough residual space, that is, when the sum of volumes of items already accommodated in the bin plus the volume of the candidate item is less than or equal to the capacity of the bin. In BPP problems, the sum of volumes of items accommodated in a bin is called the *level* of the bin, usually denoted by β.

AFFD extends FFD to DTRMP. It is applied to a list UW of unallocated warnings, which, as stated in Section 3.1, play the role of items in the FFD heuristic. These warnings are unallocated warnings from the previous maintenance plan and new warnings predicted at the beginning of the current plan. We compute the priority s_w for each warning $w \in W$. Then the warnings are sorted by decreasing priority and each sorted warning is accommodated in the first time slot able to contain it. In line with BPP studies, we use $\beta(t)$ to identify the level of time slot t, that is, the sum of the resources for warnings accommodated in time slot t. Algorithm 1 reports the pseudo-code for our AFFD heuristic.

Algorithm 1 The AFFD heuristic

```
1: UW: set of selected and unallocated warnings
2: sort UW by decreasing priority s_w
3: for all t ∈ T ∪ {d} do
4:    for all w ∈ UW do
5:       if R(w, t) + β(t) ≤ c(t) then
6:          # load warning w into time slot t
7:          β(t) := β(t) + R(w, t)
8:          UW := UW \ {w}
9:       end if
10:   end for
11: end for
```

When the AFFD heuristic is used as the base heuristic in the Monte Carlo rollout method, we assign score s_w to warning $w \in W$ according to its priority, which is a measure of the urgency of the warning. A warning is more urgent if the expected cost increases more intensively over time or if the warning is at a higher degradation level or is risky. Therefore, sorting items by decreasing priority corresponds to managing and placing the most urgent warnings first.

The AFFD heuristic is also exploited as a subheuristic of our GA. In this case, priorities are replaced by scores that take another meaning, as illustrated in Section 3.2.3.

3.2.2. The greedy randomized adaptive search procedure

In the AFFD heuristic described in Section 3.2.1 warnings are sorted by decreasing priority. However, there are many ways to sort warnings, each leading — in principle — to a different solution. Our greedy randomized adaptive search procedure heuristic (GRASP) uses the following principle: AFFD is consistently applied, but with a different ordering of the warnings each time. At the end of the overall procedure, the best solution is retained. This idea comes from the GASP heuristic [21, 11], designed to address different multidimensional packing problems. Algorithm 2 reports the pseudo-code for our GRASP heuristic.

Initialization. Given the set UW of warnings to be allocated, we compute two different initial sortings (step 1 of Algorithm 3), which are a trade-off between costs (included in the priorities) and resource requirements.

1. sort warnings by decreasing priority and then by decreasing resources
2. sort warnings by decreasing resources and then by decreasing priorities.

Positions of the warnings in the two sortings are stored respectively in vectors s_1 and s_2. These vectors will be used to generate different sortings in the main loop of the GRASP (steps 6–8). The initial solution $initSol$ is the AFFD heuristic (Section 3.2.1) with the warnings sorted by decreasing priority (step 2). The best solution is initially set to the initial solution (step 3).

Algorithm 2 The GRASP heuristic

1: sorting initialization: produce sortings s_1 and s_2
2: compute an initial solution *initSol* with the AFFD heuristic
3: $bestSol := initSol$
4: **for** $i := 1$ **to** $I_{\max}$ **do**
5: $\alpha := \mathcal{U}[0, 1]$
6: **for all** $w \in UW$ **do**
7: $s_w(\alpha) := -\alpha \cdot s_1[w] - (1 - \alpha) \cdot s_2[w]$
8: **end for**
9: sort warnings by increasing scores
10: compute the current solution *currSol* with the AFFD heuristic and with the sorted warnings
11: **if** $currSol < bestSol$ **then**
12: $bestSol := currSol$
13: **end if**
14: **end for**

Main loop. The main loop consists of $I_{\max}$ iterations (step 4). For each iteration a new solution is computed by generating a different sorting of the warnings (steps 5–9) and then applying AFFD to the resulting list UW of sorted warnings (step 10). This is accomplished by assigning a score to each warning. For warning $w \in W$, its score s_w is computed as

$$s_w(\alpha) = -(\alpha \cdot s_1[w] + (1 - \alpha) \cdot s_2[w]), \tag{8}$$

where $s_1[w]$ and $s_2[w]$ are the positions of warning $w \in W$ in sortings s_1 and s_2, respectively, computed in the initialization phase, and $\alpha \in [0, 1]$ is a coefficient randomly extracted from a uniform distribution in each iteration (step 5). Warnings are sorted by increasing score (step 9).

One important issue in GRASP design is the maximum number of iterations $I_{\max}$. It is clear that the higher the number of iterations, the better is the final solution because more solutions are produced. However, generation of a higher number of solutions increases the computational time. Thus, a compromise between solution quality and computation effort is needed. We generated 100 test instances consisting of 36 plans with approximately 500 warnings per plan. For each plan, we executed GRASP for a maximum number of iterations ranging from 1 up to 150. Table 1 lists the percentage gap for GRASP improvement over AFFD. This gap is computed as

$$100 \times \frac{\text{AFFD} - \text{GRASP}}{\text{AFFD}}.$$

It is evident from Table 1 that the gap becomes constant after 40 iterations. Therefore ,we set the maximum number of iterations $I_{\max}$ to 40 when GRASP is used as a stand-alone heuristic. We increase $I_{\max}$ when GRASP is exploited in the GA initialization procedure because we want an initial population of 100 chromosomes (see Section 3.2.3 for further details).

3.2.3. The genetic algorithm

A GA works on a *population* $\mathcal{S}$ of *chromosomes*. Each chromosome consists of a string of *genes* that represent the codification of a solution. Therefore, the set of chromosomes is also the set of current solutions. In Section 3.2.2 we presented the GRASP heuristic and showed how different sortings lead to different solutions. Therefore, sortings can be used as a codification of solutions. Moreover, if each warning is labeled with a unique number, the genes become the labels for the sorted warnings, and the corresponding chromosome consists of the sequence of labels. The quality of a chromosome is indicated by its *fitness*, which is related to the objective function for the corresponding solution. We computed the fitness of each chromosome relative to the entire population. Given chromosome $i \in \mathcal{S}$ with objective function OF_i, its fitness is computed as $f_i = \frac{OF_i}{\sum_{k \in \mathcal{S}} OF_k}$. Another feature of a chromosome is its *age*, which corresponds to the number of iterations for which the chromosome has been in the population set.

GAs are also characterized by genetic operators, which work on one or two chromosomes (the *parents*) and produce new chromosomes (the *children* or *offspring*) and provide new and possibly improved solutions. The genetic operators we use in our GA are order crossover, mutation, inversion, and translation.

- *Order crossover.* In order crossover, two parent chromosomes generate two child chromosomes. The strings of the genes of the two parents are aligned and two different positions in the strings are randomly selected. These positions delimit the so-called crossing (or matching) section. Each child has the same genes in the crossing section as the corresponding parent. The remaining positions in the string of genes are filled according to the genes of the other parent, starting from the right of the crossing section and avoiding duplicates. This operator is mainly used during the intensification phase.

- *Mutation.* This operator mutates one chromosome by swapping the genes in two randomly selected positions. Mutation is mainly used during the diversification phase.

- *Inversion* This operator mutates one chromosome by inverting the genes within a randomly selected crossing section. Inversion is mainly used during the diversification phase.

- *Translation* This operator mutates one chromosome by randomly selecting one position p and cyclically translated all genes to the right by length p. The reason for this choice comes from packing theory. In packing solutions, residual spaces in the most profitable bins are often filled with less profitable items. Applying this principle to DTRMP, if we are given a chromosome with urgent warnings to the left requiring more resources and less urgent warnings to the right requiring less resources, translation by p can be beneficial in finding better solutions. This operator is mainly used during the diversification phase.

In our GA the solution associated with a new chromosome is evaluated using the AFFD heuristic, with warnings sorted in the order of the genes of the chromosome. The main steps of our GA are reported in Algorithm 3.

The initial population is created using the GRASP heuristic presented in Section 3.2.2 (step 1). In each iteration of GRASP, a chromosome is created, evaluated with AFFD, and added to the population set $\mathcal{S}$.

In each iteration, the age of the chromosomes in the population set is increased (step 4). Moreover, a subset $sub\mathcal{S}$ of chromosomes is randomly drawn from the population set (step 5).

A series of genetic operators is randomly applied to the chromosomes in the extracted subset $sub\mathcal{S}$ (step 6) using roulette-wheel extraction [1]. This is the so-called *reproduction* phase of the genetic algorithm. In this phase the new chromosomes are evaluated through the AFFD procedure and added to the population set $\mathcal{S}$. If an improving solution is found, then the best solution is updated. The size of the population set must be constant. Thus, some of the oldest chromosomes must be killed to accommodate the new chromosomes generated in the reproduction phase (step 7). According to the most recent techniques [24, 25, 5], we adopt an *elitist* approach in which the best chromosomes are preserved by extinction. This means in our case that the 30 chromosomes with the smallest fitness values are replaced by the best chromosomes generated in the reproduction phase.

Diversification (steps 8–10) is an operation that occurs after $MAXNONIMPROVING$ consecutive non-improving evaluations. This procedure avoids situations in which the GA gets stuck in a local minimum. Diversification consists of increasing the probability of mutation and translation genetic operators, which diversify the genes of future chromosomes. The algorithm ends when the maximum number of evaluations $MAXGENERATIONS$ is reached (steps 11–13).

Algorithm 3 The GA heuristic

```
1:  compute the initial population S through the GRASP algorithm
2:  STOP := false
3:  while STOP = false do
4:     increase the age of chromosomes in the population
5:     draw a subset subS of chromosomes from the population S
6:     perform the reproduction procedure on subset subS
7:     kill a subset of chromosomes
8:     if the number of non-improving solutions ≥ MAXNONIMPROVING then
9:        perform the diversification procedure
10:    end if
11:    if the number of generations ≥ MAXGENERATIONS then
12:       STOP := true
13:    end if
14: end while
```

We used an adaptive approach to find appropriate values for the aforementioned parameters. We started with a high number of *MAXGENERATIONS* and *MAXNONIMPROVING* and varied the probability of each genetic operator in the roulette-wheel extraction and the number of chromosomes drawn for reproduction. According to practical experience, we set the population size to 100 [25]. The best value for the number of chromosomes drawn was 30. Each of the 30 chromosomes drawn undergoes order crossover with the remaining 29 chromosomes. Moreover, each drawn chromosome undergoes one of the following genetic operators, with probability 0.5 for inversion, 0.25 for mutation, and 0.25 for translation. The probability is low for mutation and translation because these operators tend to diversify the population. For this reason, in the diversification phase (which occurs after *MAXNONIMPROVING* non-improving generations) the probability is 0.2 for inversion, 0.4 for mutation, and 0.4 for translation. The diversification procedure continues until a new improving solution is found, when the probability is again set to 0.5 for inversion, 0.25 for mutation, and 0.25 for translation.

Once we fixed the population size and the number of chromosomes drawn, we varied *MAXGENERATIONS* in the range {1000, 2500, 5000, 10000} and *MAXNONIMPROVING* in the range {100, 500, 800, 1000, 1500}. For each combination of these two parameters we executed the GA algorithm over ten instances with 36 plans and approximately 500 warnings per plan. We computed the percentage gap with respect to CPLEX as

$$100 \times \frac{\text{GA} - CPLEX}{CPLEX}.$$

Table 2 reports these gaps and Table 3 lists the corresponding computational times. From Table 2 it is evident that the best choice is 10,000 for *MAXGENERATIONS* and 800 for *MAXNONIMPROVING*. Table 3 shows that this accuracy involves the highest computational time of approximately 40 s, which is nevertheless acceptable and still low.

4. Computational results

This section describes the computational tests carried out. We generated 40 instances consisting of 36 monthly plans based on data provided by Ferrovie del Gargano [13]. Section 4.1 presents detailed information on the generation process, while Section 4.2 compares our heuristics with the optima computed by the CPLEX 12.5 optimizer [17].

4.1. Instance generation

Instances were generated based on data provided by Ferrovie del Gargano [13]. There are 22 types of warning, each consisting of a set of tasks, as reported in Table 4. For each warning and task, the columns in Table 4 denote (1) the acronym, (2) the nature of the operation, (3) the name, (4) the amount, and (5) the unit cost at the initial (lowest) degradation level.

Table 5 shows the occurrence probabilities (column 2) and degradation levels (column 3) for all warning types (column 1). An increase in degradation level implies more resources to resolve warnings (column 4), possible resolution of extra warnings (column 5), and an increase in risk (column 6). Column 7 reports how the degradation level, resource requirements, and risks change with time.

Since we did not obtain real data from monthly track measurements, warnings were randomly generated for each plan for each instance considering the occurrence probabilities in Table 5. We varied the number of warnings generated from 500 to 5000.

4.2. Results

Computational tests were performed in Java 8 and executed on a workstation with 4 GB of RAM and a 3.40-GHz processor. We considered instances with a different number of warnings n ranging from 500 to 5000. In particular, we created ten instances for each value of $n \in \{500, 1000, 2500, 5000\}$, for a total of 40 instances. These values reflect the number of warnings for single, regional, and national tracks [12]. Each instance consists of 36 plans. Therefore, each heuristic runs over $40 \times 36 = 1440$ plans. Our heuristics were validated by comparing their performance with that of the CPLEX 12.5 solver [17]. We set a time limit of 5 h for CPLEX.

Table 6 shows the average percentage gap for the three heuristics with respect to the CPLEX solver for all values of n. Data are reported as the mean gap in overall cost for ten instances over 36 plans. It is evident that all the proposed heuristics provide good-quality results, with an overall mean gap of <1%. The most effective heuristic is the GA algorithm, with an overall gap of 0.36%.

Table 7 lists the number of optima found by each heuristic. Again, the most effective heuristic is the GA algorithm, which finds 1142 optima over 1440 plans.

From Tables 6 and 7 we can conclude that GA is the most effective heuristic, while AFFD is the least effective, with an overall gap of 0.90% and 289 optima over 1440 plans.

Tables 8 and 9 list the computational times for all the proposed methods. In particular, Table 8 shows the average computational time for each value of n, while Table 9 reports the maximum computational time.

It is evident that CPLEX can be used when the number of warnings is 500 because the mean time is competitive compared to that of AFFD and GRASP. However, as the number of warnings n increases, CPLEX use becomes prohibitive because the computational time limit of 5 h is reached, while the computational time for GA is <2 min. The CPLEX computational time would clearly increase if the time limit were removed.

Tables 8 and 9 show that GA is the most effective heuristic, but is also the slowest, with a mean and maximum computational times of approximately 44 s and 88 s, respectively. AFFD is the fastest heuristic, but the overall gap for the solutions provided is 0.9%.

From Tables 6–9 we can conclude that our heuristics offer a great degree of flexibility according to the goal of decision-makers. For immediate solutions, AFFD should be implemented, but this choice leads to lower solution quality. For better solutions, the GA algorithm is recommended, but the average computational time increases to 44 s. From Tables 6, 7, and 8 it is evident that GRASP is a compromise between AFFD and GA. It yields an intermediate gap of 0.74% and 331 optima and still offers fast computation.

5. Conclusions

We described extensive work on STRMP, a novel problem for maintenance planning at a tactical level. The main innovations of STRMP with respect to previous problems in the literature are the introduction of uncertainty for future track conditions and the possibility of creating and managing an adaptive maintenance plan rather than a fixed one. Exploiting analogies between STRMP and a number of BPPs, we proposed a model for the deterministic subproblem and three efficient heuristics that effectively address STRMP. The AFFD, GRASP, and GA heuristics offer decision-makers a high degree of flexibility according to the computational time available and the solution quality required. Extensive computational results demonstrate the efficiency and effectiveness of these heuristics for STRMP.

Acknowledgments

This research was developed under the European Research Project ACEM-Rail, Automated and Cost-effective Railway Infrastructure Maintenance, funded by DG Research (Call FP7-SST-2010-RTD-1).

References

[1] M.S. Arumugam, M.V.C. Rao, and Ramaswamy Palaniappan. New hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems. *Applied Soft Computing*, 6(1):38–52, 2005.

[2] M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei. The generalized bin packing problem. *Transportation Research Part E*, 48(6): 1205–1220, 2012. doi: 10.1016/j.tre.2012.06.005.

[3] M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei. Asymptotic results for the generalized bin packing problem. *Procedia - Social and Behavioral Sciences*, 111:663–671, 2013. doi: 10.1016/j.sbspro.2014.01.100. DOI 10.1016/j.sbspro.2014.01.100.

[4] M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei. Branch-and-price and beam search algorithms for the variable cost and size bin packing problem with optional items. *Annals of Operations Research*, 222(1):125–141, 2014. doi: 10.1007/s10479-012-1283-2. DOI 10.1007/s10479-012-1283-2.

[5] R. Bolaños, M. Echeverry, and J. Escobar. A multiobjective non-dominated sorting genetic algorithm (nsga-ii) for the multiple traveling salesman problem. *Decision Science Letters*, 4(4):559–568, 2015.

[6] G. Budai, D. Huisman, and R. Dekker. Scheduling preventive railway maintenance activities. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 4171–4176, 2004.

[7] G. Budai, D. Huisman, and R. Dekker. Scheduling preventive railway maintenance activities. *Journal of the Operational Research Society*, 57:1035–1044, 2006.

[8] G. Budai, R. Dekker, and U. Kaymak. Genetic and memetic algorithms for scheduling railway maintenance activities. Technical Report Econometric Institute Report EI 2009-30, Erasmus University Rotterdam, 2009.

[9] B. S.N. Cheung, K.P. Chow, L. C.K. Hui, and A. M.K. Yong. Railway track possession assignment using constraint satisfaction. *Engineering Applications of Artificial Intelligence*, 12(5):599–611, 1999.

[10] T. G. Crainic, G. Perboli, W. Rei, and R. Tadei. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research*, 38:1474–1482, 2011.

[11] T. G. Crainic, G. Perboli, and R. Tadei. Recent advances in multi-dimensional packing problems. In C. Volosencu, editor, *New Technologies - Trends, Innovations and Research. ISBN: 978-953-51-0480-3*, pages 91–110. InTech, 2012. doi: 10.5772/33302.

[12] DB NETZE. Die bauschwerpunkte der db netz ag. die hier bereitgestellte karte informiert in einer 12-wochen-vorschau ber die bauschwerpunkte auf dem schienennetz der db netz ag. `http://fahrweg.dbnetze.com/fahrweg-de/produkte/trassen/baustelleninformation/bauschwerpunkte.html`, last access August 22, 2015.

[13] Ferrovie del Gargano. URL `http://www.ferroviedelgargano.com/, last access August 22, 2015`.

[14] D. K. Friesen and M. A. Langston. Variable sized bin packing. *SIAM Journal on Computing*, 15:222–230, 1986.

[15] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, STOC '72, pages 143–150, New York, NY, USA, 1972.

[16] F. Heinicke, A. Simroth, and R. Tadei. On a novel optimisation model and solution method for tactical railway maintenance planning. In *Proceedings of the 2nd International Conference on Road and Rail Infrastructure*, pages 421–427. Department of Transportation, Faculty of Civil Engineering, University of Zagreb, 2012.

[17] IBM ILOG Optimization Studio v. 12.5.1. URL `http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.5.1/maps/ic-homepage.html last access August 22, 2015`.

[18] D. S. Johnson, A. Demeters, J. D. Hullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:299–325, 1974.

[19] A. C. Marquez and J.N.D. Gupta. Contemporary maintenance management: process, framework and supporting pillars. *Omega*, 34(3): 313–326, 2006.

[20] S. Martello and P. Toth. *Knapsack Problems - Algorithms and computer implementations*. John Wiley & Sons, Chichester, UK, 1990.

[21] G. Perboli, T. G. Crainic, and R. Tadei. An efficient metaheuristic for multi-dimensional multi-container packing. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on Automation Science and Engineering*, pages 563 –568, 2011. doi: 10.1109/CASE.2011.6042476.

[22] G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45:364–380, 2011.

[23] G. Perboli, R. Tadei, and M. M. Baldi. The stochastic generalized bin packing problem. *Discrete Applied Mathematics*, 160:1291–1297, 2012.

[24] J. Roupec. Advanced genetic algorithms for engineering design problems. *Engineering Mechanics*, 17:407–417, 2010.

[25] J. Roupec, P. Popela, D. Hrabec, J. Novotny, A. Olstad, and K. Haugen. Hybrid algorithm for network design problem with uncertain demands. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, WCECS 2013, October 23–25, 2013, San Francisco, USA 2013.

[26] The ACEM-Rail project. URL `http://www.acem-rail.eu/summary.html, last access August 22, 2015`.

[27] P. van Hentenryck. *Constraint satisfaction in logic programming*, volume 5. MIT press, Cambridge, MA, 1989.

[28] A. van Vlietan Horenbeek and L. Pintelon. Development of a maintenance performance measurement frameworkusing the analytic network process (anp) for maintenance performance indicator selection. *Omega*, 42(1):33–46, 2014.

[29] J. I. van Zante-de Fokkert, D. den Hertog, F. J. van den Berg, and J. H. M. Verhoeven. The netherlands schedules track maintenance to improve track workers' safety. *Interfaces*, 37:133–142, 2007.

[30] H. M. Wee and G. A. Widyadana. A production model for deteriorating items with stochastic preventive maintenance time and rework process with {FIFO} rule. *Omega*, 41(6):941–954, 2013.

Appendix

Table 1: GRASP calibration

Max # of iterations	0	1	10	20	30	40	50	60	70
Gap	0.00	0.09	0.13	0.14	0.14	0.15	0.15	0.15	0.15
Max # of iterations	80	90	100	110	120	130	140	150	
Gap	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	

Table 2: GA calibration: percentage gap

MAXNONIMPROVING / *MAXGENERATIONS*	**100**	**500**	**800**	**1000**	**1500**
1000	0.2628	0.252	0.2511	0.2426	0.2426
2500	0.188	0.1815	0.1774	0.1833	0.1677
5000	0.1448	0.1402	0.1249	0.1563	0.1514
10000	0.1275	0.1085	0.1069	0.1227	0.1246

Table 3: GA calibration: computation time (in seconds)

MAXNONIMPROVING / *MAXGENERATIONS*	**100**	**500**	**800**	**1000**	**1500**
1000	4.6	4.5308	4.542	4.6687	4.5748
2500	10.3449	10.3403	10.3355	10.3398	10.3371
5000	20.1509	20.145	20.2122	20.1692	20.1839
10000	40.3635	40.3777	40.3258	40.3454	40.3749

Table 4: Warning features

CODE	CHARACTER	NAME	AMOUNT	COST
ACEM_RC	Pack	Rail resurfacing	1	1.63
ACEM_GRA	Pack	Mechanic rail grinding	1	0.64
ACEM_GR	Pack	Manual rail grinding	1	0.38
ACEM_BT	Pack	Tamper by tamping machine	1	1.41
ACEM_BL	Pack	Tamper by manual methods	1	108.54
ACEM_BC	Pack	Ballast cleaning	1	10.52
ACEM_SBR	Pack	Subballast replacing	1	34.89
ACEM_RP_RP_J	Pack	Rail replacement in rail with joint (without acstrhom)	1	1721.93
ACSTRHOM	Pack	Stress homogenization per rail	1	2.81
ACEM_BR	Pack	Ballast replacing	1	43.36
ACEM_BP	Pack	Ballast restoration curb profile	1	20.95
ACEM_GD	Pack	Gauge deviation manual field work	1	7.33
ACEM_LD	Pack	Longitudinal defects manual field work	1	49.22
ACEM_HD	Pack	Horizontal defects manual field work	1	47.79
ACEM_FP	Pack	Rail fishplating	1	324.07
ACEM_SF	Pack	Sleepers flanged holes sanitizing	1	11.42
ACEM_SRC	Pack	Concrete sleepers replacement	1	191.93
ACEM_FRT	Pack	Tight fastening	1	8.42
ACEM_FRL	Pack	Large fastening replacement	1	18.92
ACEM_SBL	Pack	Subballast local replacing	1	2806.60
ACEM_DDC	Pack	Dreinage ditches cleaning	1	0.12
ACEM_SGV	Pack	Chemical spray to avoid vegetation	1	0.16

Table 5: Development of the maintenance warnings

CODE	PROBABILITY	DEGRADATION LEVEL	UNITS (U)	EXTRA TASKS	RISK	ENTRY MONTH
ACEM_RC	0.043	1	300 - 400		0	0 - 6
		2	375 - 500		1	3 - 12
		3	750 - 1000		2	6 - 24
		4	750 - 1000		3	18 - 100
ACEM_GRA	0.016	1	50 - 100		0	0 - 6
		2	63 - 125		1	3 - 12
		3	125 - 250		2	6 - 24
		4	0 - 0	U/9 ACEM_RP_RP_J	3	18 - 100
ACEM_GR	0.016	1	50 - 100		0	0 - 6
		2	63 - 125		1	3 - 12
		3	125 - 250		2	6 - 24
		4	0 - 0	U/9 ACEM_RP_RP_J	3	18 - 100
ACEM_BT	0.064	1	1800 - 2000		0	0 - 12
		2	3600 - 4000		1	6 - 18
		3	5400 - 6000		2	10 - 36
		4	5400 - 6000		3	24 - 100
ACEM_BL	0.064	1	5 - 5		0	0 - 12
		2	10 - 10		1	6 - 18
		3	15 - 15		2	10 - 36
		4	15 - 15		3	24 - 100
ACEM_BC	0.02	1	1450 - 1616		0	0 - 6
		2	1450 - 1616	U ACEM_BT	1	4 - 24
		3	1450 - 1616	U ACEM_BT	2	20 - 40
		4	1450 - 1616	U ACEM_BT	3	36 - 100
ACEM_SBR	0.008	1	200 - 300		0	0 - 6
		2	200 - 300	U ACEM_BT	1	4 - 24
		3	200 - 300	U ACEM_BT	2	20 - 40
		4	200 - 300	U ACEM_BT	3	36 - 100
ACEM_RP_RP_J	0.003	1	1 - 1		0	0 - 1
		2	1 - 1		1	0 - 2
		3	1 - 1		2	1 - 3
		4	1 - 1		3	3 - 100
ACSTRHOM	0.16	1	100 - 900		0	0 - 6
		2	100 - 900		1	6 - 12
		3	100 - 900	U ACEM_HD	2	12 - 24
		4	100 - 900	U ACEM_HD	3	18 - 100
ACEM_BR	0.016	1	60 - 100		0	0 - 12
		2	60 - 100	U ACEM_BT	1	8 - 18
		3	60 - 100	U ACEM_SBR	2	16 - 30
		4	60 - 100	U ACEM_SBR	3	24 - 100
ACEM_BP	0.032	1	60 - 100		0	0 - 6
		2	60 - 100	U ACEM_HD	1	6 - 15
		3	60 - 100	U ACEM_BR	2	15 - 30
		4	60 - 100	U ACEM_BR	3	24 - 100
ACEM_GD	0.008	1	10 - 20		0	0 - 2
		2	10 - 20	U ACEM_HD	1	2 - 4
		3	10 - 20	U ACEM_HD	2	4 - 6
		4	10 - 20	U ACEM_HD	3	6 - 12
ACEM_LD	0.016	1	10 - 50		0	0 - 2
		2	10 - 50		1	2 - 4
		3	10 - 50		2	4 - 6
		4	10 - 50	U/9 ACEM_RP_RP_J	3	6 - 12
ACEM_HD	0.016	1	10 - 50		0	0 - 2
		2	10 - 50		1	2 - 4
		3	10 - 50		2	4 - 6
		4	10 - 50	U/9 ACEM_RP_RP_J	3	6 - 12
ACEM_FP	0.008	1	1 - 4		0	0 - 3
		2	1 - 4		3	3 - 100
ACEM_SF	0.008	1	15 - 50		0	0 - 5
		2	15 - 50		2	4 - 7
		3	15 - 50		3	6 - 100
ACEM_SRC	0.006	1	1 - 5		0	0 - 3
		2	1 - 5	1 ACEM_RP_RP_J	3	3 - 100
ACEM_FRT	0.16	1	20 - 100		0	0 - 5
		2	20 - 100		2	4 - 7
		3	20 - 100		3	6 - 100

Table 6: Percentage gap of the proposed heuristics

Method \ n	**500**	**1000**	**2500**	**5000**	**OVERALL GAP**
AFFD	0.85	1.06	0.79	0.90	**0.90**
GRASP	0.70	0.89	0.65	0.73	**0.74**
GA	0.13	0.46	0.35	0.48	**0.36**

Table 7: Number of optima of the proposed heuristics

Method \ n	**500**	**1000**	**2500**	**5000**	**# OPTIMA over 1440**
AFFD	166	69	42	21	**298**
GRASP	171	75	54	31	**331**
GA	301	280	275	286	**1142**

Table 8: Average computation time of the proposed heuristics (in seconds)

Method \ n	**500**	**1000**	**2500**	**5000**	**OVERALL AVERAGE TIME**
CPLEX	0.23	56.03	103.66	444.69	**151.15**
AFFD	0.00	0.00	0.00	0.00	**0.00**
GRASP	0.19	0.20	0.21	0.22	**0.21**
GA	40.66	44.24	46.12	46.37	**44.35**

Table 9: Maximum computation time of the proposed heuristics (in seconds)

Method \ n	**500**	**1000**	**2500**	**5000**
CPLEX	10	18032	18252	18478
AFFD	0.02	0.02	0.02	0.02
GRASP	0.27	0.34	0.41	0.64
GA	47.81	61.59	70.98	87.72