INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. XX (20XX) 1–25
DOI: xx.xxxx/itor.xxxxx

A simheuristic-based algorithm for the stochastic long-term maintenance scheduling problem

Diego G. Coelho[a,*], Marcone J. F. Souza[b] and Luciano P. Cota[c]

[a]*Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto, Instituto Tecnológico Vale, Brazil*
[b]*Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, MG, 35400-000, Brazil*
[c]*Instituto Tecnológico Vale, Ouro Preto, MG, 354000-000, Brazil*
E-mail: diego.coelho@aluno.ufop.edu.br [Diego G. Coelho]; marcone@iceb.ufop.br [Marcone J. F. Souza] luciano.p.cota@itv.org [Luciano P. Cota];

Received DD MMMM YYYY; received in revised form DD MMMM YYYY; accepted DD MMMM YYYY

Abstract

This work addresses the problem of assigning preventive maintenance jobs in a 52-week planning horizon. Given a set of machines that need preventive maintenance, a set of maintenance jobs in these machines, a set of work teams, and a planning horizon, the problem consists of assigning each job to a work team in a given instant of the planning horizon, aiming to minimize the cost with work teams and the cost of performing the unscheduled jobs using outsourced teams. We propose an Iterated Local Search (ILS)-based algorithm specialized for this problem. Using real instances, the ILS algorithm achieved the best results in 81% of the instances, outperforming literature algorithms. However, these algorithms only treat the deterministic version of the problem and do not consider the uncertainty in the job duration that may occur in an industry environment. Not considering this aspect can produce an inefficient schedule with many unscheduled jobs. So, this work also proposes a simheuristic-based algorithm (SIM-ILS) capable of capturing this issue. We tested it in three scenarios, which differ in the level of uncertainty regarding the job duration, and compared their results with those provided by the stochastically evaluated ILS solutions. SIM-ILS found the best solution in 61% of the tests. Therefore, the SIM-ILS can be used to support decision-making in different industrial environments, from environments with low variability in job duration to those with high variability.

Keywords: Long-term preventive maintenance scheduling; Iterated Local Search; Variable Neighborhood Descent; Metaheuristics; Dispatching rules; Simheuristic.

[*]Author to whom all correspondence should be addressed (e-mail: marcone@ufop.edu.br).

1. Introduction

Maintenance planning and control are crucial for the industrial sector as they ensure the availability of assets (e.g., machines such as trucks, belt conveyors, and crushers in the mining industry context) at the lowest possible cost. There are two types of maintenance: corrective and preventive. In corrective maintenance, the aim is to correct an asset's failure. On the other hand, preventive maintenance is a scheduled intervention according to the manufacturer's recommendations or the maintenance teams' experience to minimize the probability of assets failing (Viana, 2022).

Industries such as mining, metallurgy, and automotive use a Computerized Maintenance Management System (CMMS) to manage maintenance jobs. This system centralizes information and controls all maintenance operations. The maintenance plans of the assets are registered in it, containing the information related to the preventive maintenance that must be scheduled, its periodicity, duration, and the skill that each team needs to perform the maintenance jobs.

The data extracted from the CMMS enables the creation of a long-term plan known in the industry as a 52-week maintenance plan. This plan is used as a planning tool, allowing the company to anticipate the material and workforce needs required to perform preventive maintenance. If the industry's workforce is insufficient to perform all the maintenance jobs, the industry can plan to hire external teams to carry them out. This annual planning is, therefore, important to ensure that all preventive maintenance jobs are carried out on time to reduce the probability of failure and increase asset life. In addition, it allows maintenance teams to be planned in advance, ensuring that the work teams with the skills required for performing each maintenance job are available when needed.

However, treating the problem of assigning preventive maintenance jobs for a 52-week plan, called the Long-Term Preventive Maintenance Scheduling Problem (LTPMSP), is challenging. This problem, introduced by Aquino et al. (2018a,b) for a mining company, is NP-hard. In the company used as a case study for this problem, the CMMS only lists the maintenance jobs to be performed on the assets for the coming year, typically around 30,000 jobs in 1,000 assets for a single operating unit, with their respective characteristics (such as durations, execution windows, and skills required to perform them). It does not consider the availability and skills of the maintenance teams, nor does it suggest the scheduling of these jobs. When receiving such jobs, the maintenance team manager manually sequences them. As a result of this manual work, the company is only able, with its own staff, to carry out far fewer jobs than expected. Therefore, to maintain production and minimize the occurrence of asset failures, the company hires outsourced teams, which increases operating costs.

On the other hand, uncertainty is present in industrial environments. Not considering it can produce an inefficient schedule with many unscheduled jobs. Despite this, we did not find any study that considered this characteristic in the literature review of the LTPMSP.

One of the most popular methods for dealing with uncertainties in stochastic optimization problems is the simheuristic (Juan et al., 2022). This method combines simulation and metaheuristic procedures and has been used successfully in various stochastic optimization problems (Kizys et al., 2022; Keenan et al., 2021; Santos et al., 2020; Panadero et al., 2020; Rabe et al., 2020; González-Neira et al., 2019; Guimarans et al., 2018; Juan et al., 2014). On the other hand, the Iterated Local Search (ILS) (Lourenço et al., 2019) is one of the metaheuristics with many successful applications in various deterministic or stochastic optimization problems, such as those presented by Ferone et al. (2020), Cunha et al. (2020), Guimarans et al. (2018), and Hatami et al. (2018).

So, to fill this gap in the study of the stochastic version of the LTPMSP, this work proposes a simheuristic-based algorithm to treat it considering the uncertainty in the duration of maintenance jobs. This proposal is justified because humans carry out these jobs, and therefore, there is variability in the time taken to complete these jobs, which depends on the work team and also on the operating environment. The proposed algorithm combines Monte Carlo Simulation (MCS) and an ILS-based algorithm. As shown in the results, it produces high-quality solutions for the LTPMSP with uncertainty in the duration of the maintenance jobs. As it adapts to different industrial scenarios, it can be incorporated into a decision support system to support the development of the 52-week maintenance plan. Another contribution of our work is the introduction of a new algorithm for determining the start times of maintenance jobs. We show that it is more efficient than the one proposed by Aquino et al. (2019). Finally, we also introduce a new constructive algorithm, which is an ensemble of dispatching rules. This algorithm automatically applies several dispatching rules, each prioritizing a specific optimization criterion, and returns the best of the generated schedules.

The remainder of this article is organized as follows. Section 2 presents a literature review of the problem. Section 3 describes the problem in detail. The proposed algorithms are presented in Section 4. Section 5 reports the results of the computational experiments. Finally, Section 6 concludes this study and suggests directions for future work.

2. Literature review

This section reviews studies on job maintenance (Section 2.1) and simulation techniques for dealing with uncertainties in optimization problems (Section 2.2).

2.1. Review on maintenance management

Although there are many studies on maintenance management, preventive maintenance scheduling problems are still little explored in the literature. Among these, some studies focus on solving generic maintenance problems, and others on applications in specific industries. As an example for the first case, Mena et al. (2021) deal with the preventive maintenance planning of a single machine in a medium-term planning horizon. In this problem, there are time window tolerances for advancing or delaying the execution of the maintenance jobs. To solve the problem, the authors introduced a Mixed-Integer Linear Programming (MILP) formulation. This formulation determines the scheduling of the maintenance jobs to minimize the number of stoppages needed to meet a preventive maintenance policy that guarantees adequate equipment availability.

In the energy industry, there are studies such as those by Saraiva et al. (2011), Almakhlafi and Knowles (2015), Gu et al. (2023), and Ruiz-Rodríguez et al. (2024). Saraiva et al. (2011) deal with the generator maintenance scheduling of a system with 29 generator sets. The study considers long-term planning to minimize the generation cost to supply the expected demand over a scheduling period. They developed a MILP formulation to represent the problem. Using a fictitious, high-cost generator, they penalize the energy not supplied in a period by all the other generators. As a method of solving the problem, they applied a Simulated Annealing-based algorithm. In Almakhlafi and Knowles (2015), an ILS-based algo-

rithm was developed to treat a variant of the preventive maintenance scheduling problem, the generator maintenance scheduling problem. Gu et al. (2023) participated in a challenge to solve the maintenance job scheduling problem proposed by Europe's largest electricity transmission operator, *Réseau de Transport d'Électricité*. The authors proposed an ILS-based algorithm with two perturbation levels to treat large-scale instances of the problem. The objective is to minimize the risk of not carrying out maintenance. Ruiz-Rodríguez et al. (2024) study a maintenance scheduling problem that aims to keep machines in operation at all times based on knowledge of their failure distribution and the time work teams consume to repair them. The authors consider uncertainty in the distribution of machine failures and the repair duration. They also consider that each work team can perform maintenance on any machine. The goal is to increase machines' uptime and reduce the average repair duration.

Viveros et al. (2021) and Ertem et al. (2022) present applications to the chemical industry. In Viveros et al. (2021), the authors address the maintenance planning problem in wastewater treatment plants. The goal is to minimize planned interruptions, fixed maintenance costs, and system downtime while improving cost efficiency and quality requirements. They introduced a MILP formulation to solve the problem. Ertem et al. (2022) deal with a workers-constrained shutdown maintenance scheduling with skills' flexibility. The authors proposed two MILP formulations and two constructive heuristic algorithms. A case study of the cement industry was used to validate the methods.

In the mining industry, we highlight the studies by Aquino et al. (2019) and Andrade et al. (2024). Aquino et al. (2019) developed a mathematical model to solve small-scale instances of the LTPMSP and metaheuristic algorithms to treat larger instances. These heuristic algorithms start from a random initial solution and use a job assignment procedure that starts assignments from the beginning of each job's time window. They considered instances with up to 33484 jobs, 1286 machines, and 263 work teams. Later, in Andrade et al. (2024), the authors proposed two new mathematical formulations and a GRASP-based algorithm for the LTPMSP. The first formulation uses time-indexed variables, and the second combines time-indexed variables with precedence variables. With these new formulations, obtaining seven optimal solutions not yet known in medium-scale instances was possible. GRASP outperforms the methods of Aquino et al. (2019) on medium- and large-scale instances.

Although some studies mentioned above deal with maintenance job scheduling problems, only the last two studies consider specific characteristics of LTPMSP simultaneously, such as maintenance job time windows, assignment of work teams to maintenance jobs, the skill required of the work team to execute a maintenance job, the cost for using a work team, the cost for not scheduling a maintenance job, and the maximum instant of availability of the work team. However, these two studies only treat the deterministic version of the problem and do not consider the uncertainty in the job duration that may occur in an industry environment. Not considering this aspect can produce an inefficient schedule with many unscheduled jobs.

Table 1 summarizes the main characteristics of our proposal with those of the previously reviewed studies for maintenance scheduling problems, focusing on the planning horizon, the objective functions, and the solution methods.

Table 1: Summarizing the main characteristics of our proposal in comparison with the reviewed studies.

Works	Planning horizon			Objective functions					Solution methods			
	Short	Medium	Long	[1]	[2]	[3]	[4]	[5]	Exact	Constructive	Metaheuristics	Simheurístics
Saraiva et al. (2011)	-	-	✓	-	✓	✓	-	-	-	-	✓	-
Almakhlafi and Knowles (2015)	-	-	✓	-	✓	-	-	✓	-	-	✓	-
Aquino et al. (2019)	-	-	✓	✓	✓	✓	-	-	✓	-	✓	-
Woller and Kulich (2021)	-	-	✓	✓	-	-	-	✓	-	-	✓	-
Viveros et al. (2021)	✓	-	-	✓	✓	-	✓	-	✓	-	-	-
Mena et al. (2021)	-	✓	-	✓	-	-	✓	-	✓	-	-	-
Ertem et al. (2022)	✓	-	-	-	-	-	✓	-	✓	✓	-	-
Gu et al. (2023)	-	-	✓	-	-	-	-	✓		✓	✓	-
Ruiz-Rodríguez et al. (2024)	✓	-	-	✓	✓	-	✓	✓	-	✓	-	✓
Andrade et al. (2024)	-	-	✓	✓	✓	✓	-	-	✓	✓	✓	-
Our proposal	-	-	✓	✓	✓	✓	-	-	-	✓	✓	✓

Legend:
[1]: More than one work team; [2]: More than one asset; [3]: Cost for unscheduled jobs; [4]: Downtime; [5]: System reliability

Of the studies presented above, only Ruiz-Rodríguez et al. (2024) consider uncertainty in the input data of the maintenance planning problem. However, they address an operational problem that aims to keep machines running at all times. They do not consider the number of work teams to schedule maintenance jobs. Their study also does not consider that the machines may require different team skills to perform the maintenance.

2.2. Review on simulation techniques

As stated previously, simulation techniques have been used successfully to deal with uncertainty in the input data of several optimization problems. Juan et al. (2022) present an introductory tutorial on the concept of simheuristics and show applications in various sectors, such as manufacturing, services, transport, telecommunications, and insurance. Next, we analyze some studies that used this technique in other optimization problems. We aim to verify which metaheuristics are usually employed within a simheuristic framework.

Guimarans et al. (2018) address the two-dimensional vehicle routing problem where customers' demands are composed of sets of non-stackable items and travel times are stochastic. The authors proposed a simheuristic algorithm that combines Monte Carlo simulation, iterated local search, and biased-randomized routing and packing heuristics.

Santos et al. (2020) seek to determine the appropriate number of active equipment in an iron ore crusher circuit to ensure efficiency and production rate. They developed a decision support system that applies an ILS-based simheuristic to achieve the objectives of the problem. This system uses a simulated plant model to evaluate the production rate. They used real production scenarios from a Brazilian mine to perform the computational experiments. The results showed that the simheuristic solutions generated an increase in production rate of up to 9% and a reduction in energy consumption of up to 59%.

Panadero et al. (2020) applied a variable neighborhood search-based simheuristic for the project portfolio selection problem aiming to maximize the expected net present value of the investment. Rabe et al. (2020) developed a genetic algorithm-based simheuristic for a manufacturing system aiming to find a configuration that minimizes the difference between the system flow factor and a target value.

Keenan et al. (2021) deal with the time-capacitated arc routing problem with stochastic demands. The objective is to minimize the expected time needed to serve all customers, considering the variability of demands and service times. They proposed a simheuristic algorithm with an oscillation pattern, which works together with a local search method to create a balance between diversification and intensification strategies.

Kizys et al. (2022) deal with the portfolio optimization problem to minimize risk for an expected return by assigning weights to assets. They proposed a simheuristic algorithm that integrates a variable neighborhood search-based algorithm with Monte Carlo simulation to treat stochastic returns and noisy covariances modeled as random variables.

Rodríguez-Espinosa et al. (2024) applied a simheuristic approach using the NSGA-II with Monte Carlo simulation to deal with a flexible job shop problem. The problem follows the just-in-time philosophy, seeking to minimize two objectives: 1) the weighted sum of the earliness and tardiness and 2) the weighted sum of the earliness and tardiness risk.

Regarding solution methods for the maintenance scheduling problem studied by Ruiz-Rodríguez et al. (2024), they compared genetic algorithm-based simheuristic with reinforcement learning and dispatching rules. In turn, Rabet et al. (2024) proposed a genetic algorithm-based simheuristic for a supply chain scheduling problem.

From the literature review on solution methods using simheuristics, we realized that the most common metaheuristics integrated into simheuristic algorithms are genetic algorithms, variable neighborhood search, and iterated local search. Therefore, our proposal for developing an ILS-based algorithm for the LTPMSP aligns with that of the simheuristic research community. Furthermore, the LTPMSP is a scheduling problem, and this solution method has proven efficient in dealing with both deterministic and stochastic versions of this type of problem.

3. Long-term maintenance scheduling problem

We describe below the sets, input parameters, indexes, decision variables, and the mixed-integer linear programming model proposed by Aquino et al. (2019) to represent the problem under study.

- Sets:
  $\mathcal{E}$**:** Set of industrial machines that will undergo preventive maintenance jobs;
  $\mathcal{T}$**:** Set of preventive maintenance jobs to be performed on the set of machines;
  $\mathcal{W}$**:** Set of maintenance teams available to execute maintenance jobs.
- Indexes:
  i, j**:** Indexes for maintenance jobs;
  k**:** Index for work teams.
- Input parameters:
  Q**:** Number of industrial machines, i.e, $Q = |\mathcal{E}|$;
  N**:** Number of preventive maintenance jobs, i.e., $N = |\mathcal{T}|$;
  M**:** Number of work teams, i.e., $M = |\mathcal{W}|$;
  $\mathcal{S}_i$**:** Skill required to execute maintenance job $i \in \mathcal{T}$;
  E_i**:** Start of the maintenance job execution time window $i \in \mathcal{T}$;
  L_i**:** End of maintenance job execution time window $i \in \mathcal{T}$;

P_i**:** Duration of the maintenance job $i \in \mathcal{T}$;
γ_k**:** Cost for using a work team $k \in \mathcal{W}$;
ω_i**:** Cost for not scheduling maintenance job $i \in \mathcal{T}$;
$\mathcal{E}_i$**:** Industrial machine on which maintenance job i will be executed, $\mathcal{E}_i \subseteq \mathcal{E}$.
$\mathcal{W}_i$**:** Set of maintenance teams available to execute maintenance job i, $\mathcal{W}_i \subseteq \mathcal{W}$;
H_k**:** Maximum instant of availability of the work team $k \in \mathcal{W}$;
$\mathcal{T}_k$**:** Set of maintenance jobs that work team $k \in \mathcal{W}$ can execute, $\mathcal{T}_k \subseteq \mathcal{T}$.

- Decision and auxiliary variables:

x_{ij}^k**:** 1 if maintenance job i is performed immediately before maintenance j by work team k, 0 otherwise;
y_{ik}**:** 1 if maintenance job i is performed by work team k, 0 otherwise;
z_k**:** 1 if work team k is used, 0 otherwise;
c_{ik}**:** instant of completion of maintenance job i when it is performed by work team k;
r_{ij}**:** 1 if maintenance job i is performed immediately before maintenance j, 0 otherwise.

The problem under study can be represented through mathematical expressions (1)-(17):

$$\min \sum_{k \in \mathcal{W}} \gamma_k z_k + \sum_{i \in \mathcal{T}} \omega_i \left(1 - \sum_{k \in \mathcal{W}_i} y_{ik}\right) \tag{1}$$

$$\sum_{k \in \mathcal{W}_i} y_{ik} \leq 1 \qquad i \in \mathcal{T} \tag{2}$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{j\}} x_{ij}^k = y_{jk} \qquad j \in \mathcal{T}, k \in \mathcal{W}_j \tag{3}$$

$$\sum_{j \in \mathcal{T}_k} x_{0j}^k = z_k \qquad k \in \mathcal{W} \tag{4}$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{l\}} x_{il}^k = \sum_{j \in \mathcal{T}_k \cup \{0\} \setminus \{l\}} x_{lj}^k \qquad k \in \mathcal{W}, l \in \mathcal{T}_k \tag{5}$$

$$c_{0k} = 0 \qquad k \in \mathcal{W} \tag{6}$$

$$c_{jk} \geq c_{ik} + P_j - M'_{ij}(1 - x_{ij}^k) \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \cup \{0\}, j \in \mathcal{T}_k \tag{7}$$

$$c_{ik} \geq (E_i + P_i)\, y_{ik} \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \tag{8}$$

$$c_{ik} \leq L_i \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \tag{9}$$

$$c_{jk'} \geq c_{ik} + P_j - M'_{ij}(1 - r_{ij}) \qquad k \in \mathcal{W}, k' \in \mathcal{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_{k'}, \mid k \neq k', i < j, \mathcal{E}_i = \mathcal{E}_j \tag{10}$$

$$c_{jk'} \leq c_{ik} - P_i + M''_{ij} r_{ij} \qquad k \in \mathcal{W}, k' \in \mathcal{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_{k'}, \mid k \neq k', i < j, \mathcal{E}_i = \mathcal{E}_j \tag{11}$$

$$c_{ik} \leq H_k \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \tag{12}$$

$$x_{ij}^{k} \in \{0,1\} \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \cup \{0\}, j \in \mathcal{T}_k \cup \{0\} \tag{13}$$

$$y_{ik} \in \{0,1\} \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \cup \{0\} \tag{14}$$

$$z_k \in \{0,1\} \qquad k \in \mathcal{W} \tag{15}$$

$$c_{ik} \geq 0 \qquad k \in \mathcal{W}, i \in \mathcal{T}_k \tag{16}$$

$$r_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{T}, j \in \mathcal{T}, i < j, \mathcal{E}_i = \mathcal{E}_j \tag{17}$$

In this formulation, a dummy job (0) was defined to precede the first preventive maintenance job immediately and immediately succeed the last job performed by each work team.

The objective function (1) seeks to minimize the cost with the use of the maintenance teams of the company and the cost of performing the unscheduled jobs using outsourced teams.

The set of constraints (2) guarantees that a maximum of one work team will perform each maintenance job. Constraints (3) guarantee that each scheduled maintenance job will have a workforce assigned to it. Constraints (4) guarantee that each work team will only be scheduled if it is used to perform a maintenance job. Constraints (5) guarantee the scheduling of the maintenance jobs performed by each work team. Constraints (6) impose that the completion time of the fictitious maintenance job is zero for all work teams. Constraints (7) control the completion time of maintenance job j and are only activated when it is performed immediately after maintenance job i by team k. Constraints (8) and (9) ensure that each maintenance job is executed in its respective time window. Constraints (10) are activated when maintenance job j immediately precedes maintenance job i in cases where both maintenance jobs concern the same machine ($\mathcal{E}_i = \mathcal{E}_j$) and are performed by different work teams. Constraints (11) are similar to constraints (10) when maintenance job i immediately precedes maintenance job j. To activate or disable these constraints, the constants M'_{ij} and M''_{ij} must take on values greater than or equal to $L_i + P_j$ and $L_j + P_i$, respectively. Constraints (12) ensure that the completion time of job i does not exceed the availability limit time of team k. Finally, constraints (13) to (17) define the domains of the decision and auxiliary variables.

4. Proposed SIM-ILS algorithm

This section details the proposed algorithm and its procedures to provide a solution for the LTPMSP. The main novelties are i) A new algorithm for determining the start times of maintenance jobs called JPA. The development of this algorithm was necessary because the method of Aquino et al. (2019) failed in some instances, not respecting the jobs' time windows. Also, unlike the method of Aquino et al. (2019), which starts the allocation of each job from the beginning of its time window, we show that starting this allocation from the end to the beginning of its time window is more efficient; ii) A new constructive algorithm that automatically applies several dispatching rules, each prioritizing a specific optimization criterion, and returns the best of the generated schedules; iii) The proposed algorithm captures the uncertainties present in the duration of maintenance jobs through a simulation procedure that performs the stochastic evaluation of a solution using the Monte Carlo simulation method; iv) The above procedures are embedded in an ILS-based simheuristic algorithm named SIM-ILS.

This section is organized as follows. Subsection 4.1 presents the operation of the SIM-ILS algorithm proposed to deal with the stochastic version of the problem. The following subsections detail each of

its procedures. Subsections 4.2 and 4.3 show how a solution is represented and evaluated. Subsection 4.4 presents the algorithm responsible for generating an initial solution. Subsections 4.5 and 4.6 present the local search and perturbation methods. Finally, Subsection 4.7 presents the procedure for stochastic evaluation of a solution.

4.1. SIM-ILS

The SIM-ILS algorithm proposed to deal with the stochastic version of the problem under study is based on the classic ILS algorithm and was developed according to the ideas of the algorithms of Guimarans et al. (2018) and Keenan et al. (2021).

The SIM-ILS starts from an initial solution built and refined, considering the deterministic version of the problem. Next, it performs a stochastic evaluation of this solution. In the iterative loop of the SIM-ILS algorithm, a new candidate solution is only accepted if it improves on the current solution in both the deterministic and stochastic evaluations. At the end of this iterative loop, the pool of best solutions found during the search is re-evaluated with a greater number of stochastic simulations. The method returns the solution from the pool with the best stochastic evaluation.

The SIM-ILS pseudo-code is described by Algorithm 1.

Algorithm 1 initializes the time control variables and the solution pool in lines 1 and 2, respectively. In line 3, a solution is built using Algorithm 3. In line 4, the local search *RandomDescent*(.), defined by Algorithm 4, is applied to the constructed solution s. The refined solution is stored in $s^\star$. In line 5, the stochastic cost of the solution $s^\star$ is calculated using the *Simulation*(.) function with n_{Fast} iterations, described by Algorithm 6. It then enters a repeat loop in lines 7-25 until one of the stopping criteria is met. In lines 10-24, an inner repeat loop begins. In line 12, the *Shake*(.) procedure is applied to shake the solution $s^\star$, and in line 13, the perturbed solution s' is refined using the *RandomDescent*(.) local search. The deterministic cost of the refined solution s' is evaluated in line 14 using Algorithm 2. If s' is better than $s^\star$, it is submitted to the *Simulation*(.) procedure on line 15 to find its stochastic cost. If s' has a lower stochastic cost than $s^\star$, the best solution is updated and inserted into a pool of the best solutions. In lines 26-32, each pool solution is evaluated by the *Simulation*(.) procedure, this time with n_{Deep} simulations. As before, each solution is evaluated in terms of its stochastic cost, and if it is less than the stochastic cost of $s^\star$, it is stored as the best solution obtained so far. In the end, the SIM-ILS algorithm returns the best solution for the stochastic version of the problem and its stochastic cost.

The following subsections detail each procedure of the Algorithm 1.

4.2. Representation of a solution

As in Aquino et al. (2019), a solution is represented indirectly by a sequence $s = \langle s_1, s_2, \cdots, s_i, \cdots, s_n \rangle$ of n maintenance jobs to be assigned. In this solution, s_i is the i-th job candidate to be executed. The order in which the maintenance jobs are in the schedule is important, as it defines the assignment priority; the further ahead a given job is in the sequence, the higher its priority.

The indirect representation of the solution simplifies the evaluation of the proposed solution and the use of neighborhood operators to explore the problem's solution space.

Algorithm 1: *SIM-ILS*

```
input  : Number of iterations of the fast simulation (n_Fast), number of iterations of the deep simulation (n_Deep), type of distribution (distrib),
         deviation of the simulation ( desv), number of iterations of the local search (RDMax), number of iterations without improvement (ILSMax),
         k_max, time limit (t_max), and set of criteria (T)
output : s*
1  t ← 0;
2  pool ← ∅;
3  s ← BuildSolution(T);                                                    /* According to Algorithm 3 */
4  s* ← RandomDescent(s, RDMax);                                            /* According to Algorithm 4 */
5  stochCost* ← Simulation(s*, n_Fast, distrib, desv);                      /* According to Algorithm 6 */
6  ILSIter ← 0;
7  while (ILSIter ≤ ILSMax & t ≤ t_max) do
8      ILSIter ← ILSIter + 1;
9      k ← 1;
10     while (k ≤ k_max) do
11         k ← k + 1;
12         s' ← Shake(s*, k);                                               /* According to Algorithm 5 */
13         s' ← RandomDescent(s', RDMax);                                   /* According to Algorithm 4 */
14         if f(s') < f(s*) then
15             stochCost ← Simulation(s', n_Fast, distrib, desv);           /* According to Algorithm 6 */
16             if stochCost < stochCost* then
17                 s* ← s';
18                 stochCost* ← stochCost;
19                 insert(pool, s*);
20                 k ← 1;
21                 ILSIter ← 0;
22             end
23         end
24     end
25 end
26 for (s ∈ pool) do
27     stochCost ← Simulation(s, n_Deep, distrib, desv);                    /* According to Algorithm 6 */
28     if stochCost < stochCost* then
29         s* ← s;
30         stochCost* ← stochCost;
31     end
32 end
33 return s*, stochCost*
```

4.3. Evaluation of a solution

A solution s is evaluated using the Job Positioning Algorithm (JPA), defined by Algorithm 2. Besides calculating the cost of the solution, it also returns the start time of each maintenance job scheduled over the planning horizon. In other words, the JPA defines whether each job will be scheduled, and if it is scheduled, it determines when it will be started and the work team assigned to perform it.

In Algorithm 2, the JPA identifies, for each job s_i, whether there is a time slot available on the machine where it should be executed (line 5). Lines 6-8 identify the candidate teams to execute it, considering their skills and availability. If a team k is skilled and available, the job s_i is assigned to it in line 9. The cost of using this team is updated in line 12 if it has not already been used. For each unscheduled job s_i, a cost ω_{s_i} (line 19) is imposed on the solution. Finally, the algorithm returns the total stochastic cost of the solution s.

To illustrate how the JPA works, consider the sequence $s = \langle 1, 2, 3, 4 \rangle$ with four maintenance jobs (Jobs 1, 2, 3, and 4), one machine (machine A), and two work teams (Teams 1 and 2). Table 2 describes the characteristics of the four jobs of machine A. In turn, Table 3 shows the skills and available time windows of the teams for executing these jobs.

Algorithm 2: JPA - *Job Positioning Algorithm*

input : Solution s
output : Cost of the solution s

Cost $\leftarrow 0$;
$\mathcal{W}' \leftarrow$ Set $\mathcal{W}$ sorted non-increasingly by available time and then by skill;
for *each job* s_i, $i = 1$ *to* N **do**
  $assigned \leftarrow$ **false**;
  if *the machine that will undergo the job* s_i *is available at the time window* $[e_{s_i}, l_{s_i}]$ **then**
    for *each team* $k \in \mathcal{W}'$ **do**
      if *the team* k *has a compatible skill to perform the job* s_i **then**
        if *the team* k *has an available time window to perform the job* s_i **then**
          Assign job s_i to team k;
          $assigned \leftarrow$ **true**;
          if *team* k *is not yet used* **then**
            Cost $\leftarrow$ *Cost* $+ \gamma_k$; /* γ_k is the cost for using the work team k. */
          end
        end
      end
    end
  end
  if $assigned =$ ***false*** **then**
    Cost $\leftarrow$ *Cost* $+ \omega_{s_i}$; /* ω_{s_i} is the cost for not scheduling job s_i. */
  end
end
return *Cost*

Table 2: Maintenance jobs.

Machine A					
#Job	Skill	Time Window Earliest date (E)	Time Window Latest date (L)	Duration (P)	Weight (ω)
1	Mechanic	1	6	2	10
2	Mechanic	3	10	3	20
3	Electric	4	9	3	10
4	Electric	1	4	2	20

Table 3: Maintenance teams.

Team	Skill	Availability window (H)
1	Mechanic	[0, 10]
2	Electric	[0, 10]

Table 4 reports the result of applying the JPA to this example. It shows the representation of the time windows of the machine and the work teams.

Table 4: Job positioning for the sequence $\langle 1, 2, 3, 4 \rangle$.

Planning horizon										
	1	2	3	4	5	6	7	8	9	10
Team 1	1	1	2	2	2					
Team 2						3	3	3		
Machine A	1	1	2	2	2	3	3	3		

In this example, three of the four maintenance jobs were assigned, and the two available teams were used. The value of the objective function is obtained by adding the cost of using these teams (in this case, two) with the cost of not scheduling Job 4, i.e., $Cost = 2 + 20 = 22$.

In JPA, the sequence in which the jobs are submitted has a direct influence on the value of the objective function. In the previous example, the sequence of jobs was the same as that described in Table 2:

$\langle 1, 2, 3, 4\rangle$. To ascertain this influence, if we sort the jobs non-decreasingly according to the end of their L value from Table 2, we have the following sequence: $\langle 4, 1, 3, 2\rangle$.

Table 5 shows the new solution after submitting this sequence of jobs to the JPA. This table shows that Team 1 processes jobs 1 and 2 within the time windows [3, 4] and [8, 10], respectively. On the other hand, Team 2 executes jobs 4 and 2 during the intervals [1, 2] and [5, 7], respectively.

Table 5: Job positioning for the sequence $\langle 4, 1, 3, 2\rangle$.

	Planning horizon									
	1	2	3	4	5	6	7	8	9	10
Team 1			1	1				2	2	2
Team 2	4	4			3	3	3			
Machine A	4	4	1	1	3	3	3	2	2	2

In this new sequence of jobs, it was possible to schedule all of them, generating a better value for the objective function. As only two teams were needed, the cost of this new solution is $Cost = 2$. This result shows that the order in which the jobs are submitted to the JPA algorithm influences the objective function value.

We propose two versions of the JPA: Direct JPA and Inverted JPA. They differ only concerning the assignment of a job within its available interval. In the direct JPA, each job is positioned at the beginning of the interval. In contrast, in the inverted JPA, it is positioned to conclude at the end of the available interval. These versions impact the objective function's value, as will be shown in Section 5.

To illustrate the differences between these JPA versions, we present an example of allocating a job i with the following parameter values: $P_i = 3$, $E_i = 1$, and $L_i = 10$. Figures 1 and 2 illustrate how the JPA versions assign this job.

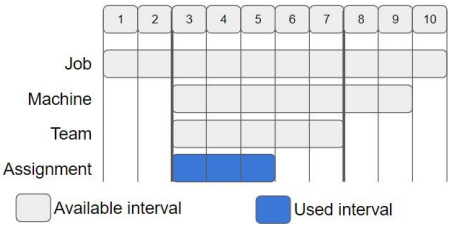

Fig. 1: Job positioning with direct JPA

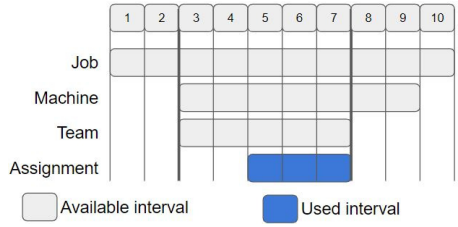

Fig. 2: Job positioning with inverted JPA

As illustrated in Figures 1 and 2, the time window available for processing job i is the interval [3, 7], which is the interval common to the job, machine, and work team windows. In direct JPA, job i starts at instant 3 and ends at instant 5, as illustrated in Figure 1. On the other hand, in inverted JPA, job i starts at instant 5 and ends at the end of the available time window for its processing, which is instant 7, as shown in Figure 2.

4.4. Constructive algorithm

Constructive heuristic algorithms, known as "dispatching rules" in job scheduling problems, are used to generate an initial solution to the problem. They are methods for generating sequences of jobs in which each is subjected to rules that check the benefit of its insertion. Thus, only the job with the most significant benefit is inserted into the sequence at each algorithm step.

Several constructive algorithms can be designed to generate an initial solution, each based on a rule. In order to present these dispatching rules, let T be the set of criteria for ranking jobs, indexed by $t \in T$. Let $a_t = 1$ if criterion $t \in T$ is sorted in descending order and 0 if it is sorted in ascending order.

Let d_l be a dispatching rule, and R be the set of all dispatching rules, i.e., $R = \cup_{l=1}^{|T|!\times 2^{|T|}} d_l$. A dispatching rule $d_l \in R$ consists of choosing a sequence of $|T|$ criteria, that is, $d_l = \langle (t_1, a_{t_1}), (t_2, a_{t_2}), \cdots, (t_{|T|}, a_{t_{|T|}}) \rangle$. Each job is evaluated respecting the order in which the criteria are in the sequence d_l that defines the l-th constructive algorithm. Thus, if there is a tie in a given criterion t_i, the next criterion, t_{i+1}, is used to break the tie. If the tie persists until the last criterion is reached, i.e., $t_{|T|}$, the job chosen to be inserted into the solution is the one with the lowest index.

To explain how this constructive algorithm works, let Table 6 be an example with four criteria for sorting six maintenance jobs.

Table 6: Example of criteria for sorting a set of maintenance jobs.

			Criteria			
#Job	Skill	Machine	E	L	P	ω
1	1	100	0	25	5	10
2	2	100	7	20	7	20
3	1	200	7	25	4	10
4	2	200	10	20	7	20
5	1	100	0	10	3	30
6	2	200	15	30	2	30

In this table, the first three columns identify each job, the skill required, and the machine on which it will be executed. In a given sequence, the last four columns (E, L, P, and ω), which are in columns 4, 5, 6, and 7, are the criteria used for sorting the jobs and so to define the constructive algorithms.

For the above example, there are $4! \times 2^4 = 24 \times 16 = 384$ constructive algorithms since there are four criteria and two ways for sorting each criterion. Let $d_1 = \langle (7, 1), (5, 0), (4, 0), (6, 1) \rangle$ be one of these dispatching rules. This rule means that Column 7 (ω) is used as the first criterion for sorting these jobs. On the other hand, as $a_7 = 1$, the jobs are sorted in descending order according to their weights, i.e., priority is given to the jobs with the highest ω_i values. However, there is a tie when applying this first criterion because $\omega_5 = \omega_6 = 30$ for jobs 5 and 6. So, Column 5 (L) is used as the second criterion to break the tie. In this case, these jobs are sorted in ascending order according to the end of their windows (L_5 and L_6). As $L_5 = 10 < 30 = L_6$, thus Job 5 is the first to be inserted into s, and Job 6 is the second. Among the remaining jobs (1, 2, 3, and 4), jobs 2 and 4 are the next ones to be scheduled because they have the highest value for the first criterion ($\omega_2 = \omega_4 = 20$). As before, there is a tie using the first criterion, so the second criterion must be used. The third criterion must be used since $L_2 = L_4 = 20$. So, Column 4 (E) is applied for sorting jobs 2 and 4 in ascending order. As $E_2 = 7 < E_4 = 10$, Job 2 is inserted into s before Job 4. Now, the remaining jobs, 1 and 3, will be sorted in descending

order concerning the first criterion. The third criterion (E) is used because of ties in the first two criteria. As $E_1 = 0 < 7 = E_3$, Job 1 is inserted into s before Job 3. Therefore, the solution returned by this constructive algorithm is $s = \langle 5, 6, 2, 4, 1, 3 \rangle$.

Algorithm 3 shows the pseudo-code of the proposed ensemble method responsible for evaluating all dispatching rules and returning the best constructed solution.

Algorithm 3: *BuildSolution*

```
   input  : Set of criteria (T)
   output : Solution s
1  s ← ∅;
2  for l ← 1 to |T|! ×2^|T| do
3  |   Let s' be the solution constructed with the rule d_l;
4  |   if f(s') < Cost then
5  |   |   s ← s';
6  |   |   Cost ← f(s');
7  |   end
8  end
9  return s
```

The algorithm receives the set of criteria T as parameter. The main loop occurs between lines 2 and 8. In line 4, a solution s' is constructed using a rule d_l, $l \in T$. Next, in line 5, the solution s' is evaluated. If it has the best cost so far, it is stored. At the end, the algorithm returns the best solution constructed s.

4.5. Random Descent Procedure

As Aquino et al. (2019), we use a neighborhood structure that swaps two jobs in the current solution. With this type of move, it is possible to explore the problem's solution space, starting from any initial solution. To give an example, let $s = \langle 1, 2, 3, 4 \rangle$ be a solution to the problem. One of the neighboring solutions of s is $s' = \langle 1, \mathbf{4}, 3, \mathbf{2} \rangle$, which is obtained by swapping jobs 2 and 4 from s.

We use the Random Descent as the local search method. It consists of randomly selecting a neighbor to be analyzed. If its cost is lower than that of the current solution, it will be accepted; otherwise, another neighbor is randomly selected to be evaluated. This process continues until the determined stopping criterion is reached. Algorithm 4 describes its pseudo-code.

Algorithm 4: *Random Descent*

```
   input  : Solution s, maximum number of iterations without improvement (RDMax)
   output : s
1  Iter ← 0;
2  while (Iter < RDMax) do
3  |   Iter ← Iter + 1;
4  |   Let s' a random neighbor obtained from s;
5  |   if f(s') < f(s) then
6  |   |   s ← s';
7  |   |   Iter ← 0;
8  |   end
9  end
10 return s
```

4.6. Shake Procedure

At the end of the local search process, the solution returned may be a local optimum, i.e., a solution in which all its neighbors have a higher cost, causing the algorithm to get stuck in that region.

The Shake procedure is applied to avoid the algorithm stopping at these optima. It consists of generating k consecutive perturbations in these solutions. In each perturbation, a swap move is applied between two jobs. Algorithm 5 presents the pseudo-code of this procedure.

Algorithm 5: *Shake*

```
   input  : Solution s, level of perturbation (k)
   output : s'
1  s' ← s;
2  i ← 1;
3  while (i ≤ k) do
4  |   s'' ← solution resulting from s' by randomly swapping two jobs;
5  |   s' ← s'';
6  |   i ← i + 1;
7  end
8  return s'
```

4.7. Simulation Procedure

The Simulation procedure performs the stochastic evaluation of a solution using the MCS method. The stochastic evaluation can be fast or deep, depending on the number of simulations. Its pseudo-code is shown in Algorithm 6.

Algorithm 6: *Simulation*

```
   input  : Solution s, number of iterations (n_sim), type of distribution (distrib), percentage deviation (desv)
   output: stochCost
1  stochCost ← 0;                          /* stochCost is the stochastic cost of the solution. */
2  for j ← 1 to n_sim do
3  |   s' ← s;                                                  /* s' is a copy of s. */
4  |   T_stoch ← T;                  /* T_stoch is a copy of the original set of jobs. */
5  |   for each job i ∈ T_stoch do
6  |   |   P'_i ← P_i × Rand(distrib, desv);          /* P'_i is the new job duration in s'. */
7  |   end
8  |   cost ← f(s');                                    /* s' is evaluated by Algorithm 2. */
9  |   stochCost ← stochCost + cost;
10 end
11 stochCost ← stochCost/n_sim;
12 return stochCost
```

In lines 2-10, the Simulation procedure performs n_{sim} evaluations of the objective function resulting from the new job duration. For each job, line 6, its new duration P_i' is calculated from its original duration P_i using the procedure *Rand*(.), described by Algorithm 7. The cost of this stochastic solution is calculated in line 8 through Algorithm 2. Line 11 calculates the average value of the n_{sim} stochastic solutions generated. In the end, the algorithm returns this average value as the stochastic cost of the solution s given as input to the algorithm.

Algorithm 7 describes how the randomness of the *Rand*(.) procedure is generated using the normal distribution. This distribution uses two parameters: μ and σ. The first is the mean of the distribution, i.e., where it is centered, and the second is its standard deviation.

Algorithm 7: *Rand*

input : Type of distribution (*distrib*), percentage deviation (*desv*)
output : r
1 $\mu \leftarrow 0$;
2 $\sigma \leftarrow desv/3$;
3 $r \leftarrow$ *CreateRand*(*distrib*, μ, σ);
4 $r \leftarrow r + 1$;
5 **return** r

Line 1 of Algorithm 7 establishes that the distribution is centered on 0. In line 2, the percentage deviation value *desv* is divided by 3 to ensure that 99.74% of the data generated is within the desired range. In line 3, a random value is generated by the *CreateRand*(.) function. Finally, in line 4, a unit is added to this generated value to create the multiplication factor to be applied to the duration of the jobs.

5. Computational Experiments

The proposed SIM-ILS algorithm for the stochastic LTPMSP was implemented in C++ programming language. We also implemented the ILS algorithm for the deterministic LTPMSP. This implemented ILS is a classic version of the method, which uses the same initial solution, local search, and perturbation procedures from the SIM-ILS algorithm, described in subsections 4.4, 4.5, and 4.6.

All the algorithms developed were run on an Intel Xeon E-2378G @ 2.80 GHz $\times$ 16, 64 GB RAM computer, with a Windows 10 operating system. Although this machine allows multiprocessing, the algorithms developed did not use this feature. The results of the algorithms from the literature, i.e., MSVNS, BRKGA, and BRKMA, are those reported by Aquino et al. (2019). Like them, we use each instance's number of jobs (N) as the maximum runtime for the algorithms in seconds. We have also included the results returned from the GRASP algorithm by Andrade et al. (2024). Both results are available at http://www.decom.ufop.br/prof/marcone/projects/LTPMSP.html.

The rest of this section is organized as follows. Firstly, in Section 5.1, we describe the instances. In Section 5.2, we compare the two versions of the JPA to select the best strategy for positioning the jobs. In Section 5.3, we showed how the parameter values of the proposed ILS were calibrated, reported its results, and compared them with those of four state-of-the-art algorithms of the literature for the deterministic version of the problem. Section 5.4 compares the results of applying the ILS and SIM-ILS algorithms to the stochastic version of the problem.

5.1. Instances

Small-, medium-, and large-scale instances of this problem are available from Aquino and Souza (2016). In these instances, each work team has only a skill and planning horizon of up to 8,670 hours per year.

Initially, we used the set of small-scale instances to validate the convergence of the ILS. The CPLEX solver, version 12.5, with the model (1)-(17), found the optimal solution in 26 of the 100 instances of this set. In contrast, for the others, it does not prove the optimality of the solutions generated in 3600 seconds of processing. On the other hand, ILS found all these optimal solutions and was able to generate solutions with objective function values equal to or lower than those generated by CPLEX.

We tested the proposed algorithms only on medium- and large-scale instances. Table 7 summarizes

their characteristics.

Table 7: Instance characteristics.

Description	Parameter	Values per instance scale		
		Medium	Large	X Large
Number of jobs	N	150 to 600	1200 to 4800	9600 to 33484
Number of work teams	M	57 to 256	41 to 287	132 to 145
Number of machines	Q	91 to 388	252 to 1286	816 to 1032
Start time window (hours)	E_i	0	0	0
End time window (hours)	L_i	3480	4560	10416
Duration (hours)	P_i	0.2 to 108.2	0.1 to 169.8	0.1 to 169.8
Cost for using each work team	γ_k	1	1	1
Cost for an unscheduled job	ω_i	3 to 3246	3 to 5094	4 to 5094
Maximum instant of availability of a work team (hours)	H_k	3480	4560	8760

5.2. Comparison of two job positioning strategies

To decide the best job positioning algorithm (Direct JPA or Inverted JPA), we perform computational experiments using these two strategies in the Constructive Algorithm.

Table 8 reports the results of these strategies in the proposed constructive algorithm. Columns ID, N, M, and Q correspond to the ID, number of jobs, maintenance teams, and machines of each instance. Columns Obj, #N, #M, Time (s), and $d^{\star}$ correspond, respectively, to the value of the objective function generated by the algorithm, the number of scheduled jobs, the number of used teams, the algorithm's execution time, in seconds, and the best dispatching rule used by the constructive algorithm.

Table 8 shows that the best results were achieved by the inverted JPA strategy. Based on these results, it was chosen to be the job positioning algorithm for the proposed constructive algorithm.

5.3. Comparison of the results of the algorithms for the deterministic version of the problem

Given the stochastic nature of metaheuristic algorithms, we first calibrated the parameters of the ILS algorithm. It has three parameters to be calibrated: *ILSMax*, $k_{\max}$, and *RDMax*. To choose the best values for these parameters, the Irace tool (López-Ibáñez et al., 2016) was used. The Irace package receives a set of representative instances of the problem and values for the algorithm's parameters. In the end, it returns the best values found among those provided.

Three representative instances of the data set were selected to carry out the calibration experiments. They have the following characteristics: i) 300, 1200, and 2400 jobs; ii) 112, 122, and 130 teams; and iii) 117, 403, and 603 machines. The values given to Irace and those returned by it are described in Table 9.

Table 8: Results of the constructive algorithm with direct and inverted JPA. Best results are highlighted in bold.

Instances				Constructive Algorithm with Direct JPA					Constructive Algorithm with Inverted JPA				
ID	N	M	Q	Obj	#N	#M	Time (s)	$d^\star$	Obj	#N	#M	Time (s)	$d^\star$
01	150	148	120	1851	143	31	0.8	⟨(4, 0), (5, 1), (6, 0), (3, 0)⟩	**1848**	143	28	0.8	⟨(4, 1), (6, 1), (5, 0), (3, 0)⟩
02	150	75	129	**30**	150	30	0.7	⟨(6, 0), (5, 0), (4, 0), (3, 0)⟩	**30**	150	30	0.7	⟨(6, 0), (5, 0), (4, 0), (3, 0)⟩
03	150	102	91	3281	149	35	0.7	⟨(3, 0), (4, 0), (6, 1), (5, 0)⟩	**3273**	149	27	0.7	⟨(4, 1), (6, 0), (5, 0), (3, 0)⟩
04	150	57	126	25	150	25	0.8	⟨(5, 0), (6, 0), (4, 0), (3, 0)⟩	**24**	150	24	0.8	⟨(6, 1), (5, 0), (4, 0), (3, 0)⟩
05	150	92	93	303	139	52	0.7	⟨(3, 0), (6, 1), (5, 0), (4, 0)⟩	**49**	150	49	0.7	⟨(6, 1), (5, 0), (4, 0), (3, 0)⟩
06	150	71	101	681	146	45	0.7	⟨(5, 1), (4, 0), (6, 0), (3, 0)⟩	**106**	149	34	0.7	⟨(6, 1), (5, 0), (4, 0), (3, 0)⟩
07	300	158	179	5694	275	54	1.8	⟨(6, 1), (5, 0), (4, 1), (3, 0)⟩	**2023**	292	43	1.8	⟨(4, 1), (6, 1), (5, 0), (3, 0)⟩
08	300	221	239	2460	286	56	2.1	⟨(6, 0), (5, 0), (4, 0), (3, 0)⟩	**2470**	285	54	1.9	⟨(4, 1), (6, 0), (5, 0), (3, 0)⟩
09	300	112	177	3720	292	57	1.8	⟨(3, 0), (4, 1), (5, 1), (6, 0)⟩	**3360**	298	42	1.7	⟨(4, 1), (3, 1), (6, 1), (5, 0)⟩
10	300	75	181	182	298	38	1.8	⟨(4, 0), (5, 0), (6, 0), (3, 0)⟩	**35**	300	35	1.8	⟨(4, 1), (6, 1), (5, 0), (3, 0)⟩
11	300	121	162	189	297	79	1.8	⟨(3, 0), (4, 1), (5, 1), (6, 0)⟩	**65**	300	65	1.8	⟨(4, 1), (6, 0), (5, 0), (3, 0)⟩
12	300	119	176	1181	288	59	1.7	⟨(6, 1), (5, 0), (3, 0), (4, 0)⟩	**308**	297	41	1.8	⟨(4, 1), (6, 0), (5, 0), (3, 0)⟩
13	600	165	329	9703	558	73	4.3	⟨(3, 0), (5, 1), (6, 0), (4, 1)⟩	**4879**	580	55	4.3	⟨(4, 1), (5, 1), (6, 0), (3, 0)⟩
14	600	256	388	3556	569	85	4.4	⟨(3, 0), (4, 0), (5, 0), (6, 1)⟩	**3448**	570	77	4.5	⟨(4, 1), (3, 0), (6, 0), (5, 0)⟩
15	600	120	288	10019	571	65	4.0	⟨(5, 1), (6, 1), (3, 1), (4, 0)⟩	**9075**	579	60	4.0	⟨(5, 1), (6, 0), (3, 0), (4, 1)⟩
16	600	77	215	613	594	45	4.0	⟨(6, 0), (5, 0), (3, 1), (4, 0)⟩	**37**	600	37	4.1	⟨(4, 1), (3, 0), (5, 1), (6, 0)⟩
17	600	126	279	732	588	87	4.0	⟨(3, 0), (6, 1), (5, 0), (4, 1)⟩	**410**	598	67	3.9	⟨(4, 1), (6, 1), (5, 0), (3, 0)⟩
18	1200	186	519	18072	1103	88	12.0	⟨(5, 1), (3, 0), (4, 1), (6, 1)⟩	**13067**	1137	71	12.2	⟨(4, 1), (5, 1), (3, 0), (6, 1)⟩
19	1200	263	666	10883	1102	116	11.5	⟨(5, 1), (6, 0), (3, 1), (4, 1)⟩	**9893**	1110	96	11.6	⟨(4, 1), (3, 0), (5, 1), (6, 0)⟩
20	1200	122	470	25543	1143	73	10.6	⟨(5, 1), (3, 1), (4, 0), (6, 0)⟩	**23815**	1162	62	10.7	⟨(6, 1), (5, 0), (3, 1), (4, 0)⟩
21	1200	88	252	1965	1177	51	10.9	⟨(5, 1), (6, 1), (4, 1), (3, 0)⟩	**299**	1199	39	11.3	⟨(4, 1), (5, 0), (6, 1), (3, 0)⟩
22	1200	130	420	2758	1156	92	10.3	⟨(5, 1), (3, 0), (6, 0), (4, 1)⟩	**555**	1195	72	10.3	⟨(4, 1), (6, 1), (5, 0), (3, 1)⟩
23	1200	122	403	13702	1110	83	10.4	⟨(6, 1), (5, 0), (4, 0), (3, 1)⟩	**7402**	1170	69	10.5	⟨(5, 1), (3, 1), (6, 0), (4, 1)⟩
24	2400	188	738	37646	2177	93	34.5	⟨(6, 1), (5, 0), (3, 0), (4, 1)⟩	**31963**	2230	83	35.7	⟨(4, 1), (5, 1), (6, 0), (3, 0)⟩
25	2400	130	603	51367	2231	80	29.3	⟨(5, 1), (6, 0), (4, 0), (3, 0)⟩	**44941**	2303	70	28.5	⟨(5, 1), (4, 1), (6, 0), (3, 1)⟩
26	2400	90	278	6193	2345	67	31.0	⟨(4, 1), (6, 1), (5, 0), (3, 0)⟩	**1510**	2391	54	31.8	⟨(4, 1), (3, 1), (6, 1), (5, 0)⟩
27	2400	132	701	3459	2330	102	27.5	⟨(5, 1), (6, 0), (3, 0), (4, 0)⟩	**1058**	2388	83	27.9	⟨(4, 1), (3, 1), (6, 1), (5, 0)⟩
28	2400	126	561	17595	2268	92	27.3	⟨(6, 1), (5, 0), (3, 0), (4, 1)⟩	**9276**	2346	84	27.8	⟨(5, 1), (3, 0), (6, 0), (4, 1)⟩
29	4800	197	1128	70037	4372	114	107.0	⟨(5, 1), (3, 1), (4, 1), (6, 0)⟩	**62760**	4459	98	110.1	⟨(4, 1), (3, 0), (6, 1), (5, 0)⟩
30	4800	78	1286	55313	4379	177	85.1	⟨(6, 1), (5, 0), (3, 1), (4, 1)⟩	**45143**	4460	144	88.0	⟨(4, 1), (3, 1), (6, 1), (5, 0)⟩
31	4800	130	720	117255	4413	85	82.6	⟨(5, 1), (4, 0), (6, 1), (3, 0)⟩	**102571**	4553	75	84.7	⟨(5, 1), (4, 1), (6, 0), (3, 1)⟩
32	4800	91	294	19221	4646	69	98.6	⟨(4, 1), (6, 1), (5, 0), (3, 0)⟩	**2545**	4781	57	99.6	⟨(4, 1), (3, 1), (6, 1), (5, 0)⟩
33	4800	135	990	16628	4642	110	82.5	⟨(3, 0), (4, 1), (6, 1), (5, 0)⟩	**3194**	4776	90	83.8	⟨(4, 1), (5, 1), (6, 0), (3, 1)⟩
34	4800	129	713	30170	4594	96	81.9	⟨(6, 1), (5, 0), (4, 0), (3, 1)⟩	**16390**	4709	82	83.6	⟨(6, 1), (5, 0), (4, 1), (3, 1)⟩
35	9600	132	816	72860	9184	104	259.7	⟨(5, 1), (6, 1), (3, 0), (4, 1)⟩	**39849**	9407	88	263.4	⟨(5, 1), (6, 0), (4, 1), (3, 0)⟩
36	19200	132	908	156077	18377	105	947.8	⟨(6, 1), (5, 0), (4, 0), (3, 1)⟩	**89924**	18812	94	970.9	⟨(5, 1), (6, 0), (4, 1), (3, 1)⟩
37	33484	145	1032	234613	32231	111	2885.8	⟨(5, 1), (4, 0), (6, 1), (3, 0)⟩	**141902**	32870	92	3006.6	⟨(5, 1), (6, 1), (3, 1), (4, 0)⟩

Table 9: Description of the parameters, its values given to Irace, and the values returned by it.

Parameter	Description	Set of values	Values Returned
ILSMax	Maximum number of iterations without improvement in ILS	{40, 50, 60, 70}	70
$k_{\max}$	Highest level of perturbations	{10, 15, 20, 30}	15
RDMax	Maximum number of iterations without improvement of the local search method	{30, 40, 50, 60}	40

Considering the criteria established in Table 6 for sorting jobs, all 384 constructive algorithms were tested for each instance, each using the two versions of JPA, direct and inverted. At the end of the runs, the best-performing constructive algorithm and its cost are returned. The constructive algorithm with inverted JPA obtained the best performance because it generated the lowest costs in all the instances evaluated. Therefore, only its results were compared with those of the other algorithms. In addition, the solution provided by the best constructive algorithm was used as the initial solution for the ILS algorithm.

Table 10 reports the results of the proposed algorithms (Constructive algorithm with inverted JPA and ILS), those of the metaheuristic algorithms by Aquino et al. (2019): MSVNS, BRKGA, and BRKMA, and those of the GRASP algorithm by Andrade et al. (2024). Columns ID, N, M, and Q represent the instance ID, number of jobs, maintenance teams, and machines in each instance, respectively. Columns Avg, Best, and RPD (%) indicate respectively the average of the results, the best result, and the relative percentage deviation from the best solution found by all algorithms (BKV, Best Known Value) for each

instance, including our ILS results, calculated by Eq. (18):

$$RPD = \frac{f_{Avg} - f_{BKV}}{f_{BKV}} \tag{18}$$

For the constructive algorithm with inverted JPA, the values in the Avg and Best columns are the same because this algorithm is deterministic. For the ILS, the "Avg" column reports the average result of 30 runs for each instance. For MSVNS, BRKGA, and BRKMA, the Avg column reports the values published in Aquino et al. (2019), which ran each instance five times. For GRASP, the Avg column reports the values published in Andrade et al. (2024), which ran each instance five times. The instances marked with an asterisk (*) are those whose results reported by Aquino et al. (2019) are infeasible. The best results for each instance are highlighted in bold.

Table 10: Results of the Constructive Algorithm with Inverted JPA, ILS, MSVNS, BRKGA, BRKMA, and GRASP algorithms.

Instances				Constructive Algorithm				ILS(1)			MSVNS(1)			BRKGA(1)			BRKMA(1)			GRASP(2)		
ID	N	M	Q	BKV	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD
01	150	148	120	1848	**1848**	1848	-	**1848**	1848	-	-	-	-	-	-	-	-	-	-	**1848**	1848	-
02	150	75	129	30	**30**	30	0.00	**30**	30	0.00	**30**	30	0.00	**30**	30	0.00	**30**	30	0.00	**30**	30	0.00
03	150	102	91	3273	**3273**	3273	0.00	**3273**	3273	0.00	**3273**	3273	0.00	**3273**	3273	0.00	**3273**	3273	0.00	**3273**	3273	0.00
04	150	57	126	24	**24**	24	0.00	**24**	24	0.00	**24**	24	0.00	**24**	24	0.00	**24**	24	0.00	**24**	24	0.00
05	150	92	93	49	**49**	49	0.00	**49**	49	0.00	**49**	49	0.00	**49**	49	0.00	**49**	49	0.00	**49**	49	0.00
06	150	71	101	106	**106**	106	0.00	**106**	106	0.00	**106**	106	0.00	**106**	106	0.00	**106**	106	0.00	**106**	106	0.00
07	300*	158*	179*	1931	2023	2023	-	**1931**	1938	-	-	-	-	-	-	-	-	-	-	**1931**	1985.4	-
08	300	221	239	2451	2470	2470	0.78	**2451**	2451	0.01	2452	2452	0.04	2452	2452	0.04	2452	2452	0.04	**2451**	2452.6	0.07
09	300	112	177	3359	3360	3360	0.03	**3359**	3359	0.00	3360	3361	0.06	3361	3362	0.09	3362	3363	0.12	**3359**	3359	0.00
10	300	75	181	35	**35**	35	0.00	**35**	35	0.00	**35**	36	2.86	37	38	8.57	37	38	8.57	**35**	35	0.00
11	300	121	162	65	**65**	65	0.00	**65**	65	0.00	281	281	332.31	282	283	335.38	282	283	335.38	**65**	65	0.00
12	300	119	176	308	**308**	308	0.00	**308**	308	0.00	**308**	308	0.00	**308**	309	0.32	**308**	309	0.32	**308**	308	0.00
13	600*	165*	329*	4494	4879	4879	-	**4494**	4522	-	-	-	-	-	-	-	-	-	-	4566	4624.2	-
14	600	256	388	3380	3448	3448	2.01	**3380**	3392	0.35	3384	3385	0.15	3384	3387	0.21	3388	3398	0.53	3448	3457.6	2.30
15	600	120	288	8576	9075	9075	5.82	8577	8577	0.02	8578	8579	0.03	8580	8581	0.06	8584	8593	0.20	**8576**	8974.4	4.65
16	600	77	215	37	**37**	37	0.00	**37**	37	0.00	42	42	13.51	43	106	186.49	43	84	127.03	38	38	2.70
17	600	126	279	410	**410**	410	0.00	**410**	410	0.00	414	414	0.98	415	416	1.46	416	417	1.71	**410**	410	0.00
18	1200*	186*	519*	10921	13067	13067	-	**10921**	11281	-	-	-	-	-	-	-	-	-	-	10923	10973.2	0.48
19	1200	263	666	9517	9893	9893	3.95	**9517**	9542	0.26	9527	9535	0.19	9587	9634	1.23	9575	9632	1.21	9621	9653.6	1.44
20	1200	122	470	21921	23815	23815	8.64	**21921**	22100	0.82	21930	21932	0.05	22075	22268	1.58	22318	22434	2.34	22673	23068.2	5.23
21	1200	88	252	299	**299**	299	0.00	**299**	299	0.00	309	324	8.36	765	1016	239.8	390	664	122.07	**299**	299	0.00
22	1200	130	420	513	555	555	8.19	**513**	513	0.00	526	528	2.92	532	665	29.63	679	779	51.85	581	597.2	16.41
23	1200	122	403	6830	7402	7402	8.37	6836	6839	0.13	6841	6842	0.18	6841	6852	0.32	6842	6850	0.29	**6830**	6830.2	0.00
24	2400	188	738	26174	31963	31963	22.12	26932	27978	6.89	26777	26991	3.12	26705	27797	6.20	27179	27835	6.35	**26174**	26314.8	0.54
25	2400	130	603	37566	44941	44941	19.63	**37566**	39127	4.15	38094	38293	1.94	39614	41268	9.85	38764	40402	7.55	45537	45846.4	22.04
26	2400	90	278	566	1510	1510	166.78	570	584	3.11	2000	2273	301.59	2026	2333	312.19	1585	1953	245.05	**566**	569.2	0.57
27	2400	132	701	621	1058	1058	70.37	**621**	628	1.11	816	945	52.17	1768	2319	273.43	1608	2244	261.35	760	776.4	25.02
28	2400	126	561	8234	9276	9276	12.65	8242	8244	0.12	8259	8280	0.56	8839	9454	14.82	8775	9363	13.71	**8234**	8234.2	0.00
29	4800	197	1128	56164	62760	62760	11.74	56902	57762	2.85	59580	60525	7.76	62395	62883	11.96	61138	62649	11.55	**56164**	56252.6	0.16
30	4800	78	1286	40953	45143	45143	10.23	**40953**	41219	0.65	41925	42215	3.08	42643	43650	6.59	42781	43981	7.39	41135	41797.2	2.06
31	4800	130	720	86147	102571	102571	19.07	**86147**	88412	2.63	94780	95815	11.22	95629	97950	13.70	92320	95958	11.39	91174	95693	11.08
32	4800	91	294	194868	197978	197978	1.60	194944	195014	0.07	207390	208749	7.12	199117	200655	2.97	198340	199368	2.31	**194868**	194868.4	0.00
33	4800	135	990	1944	3194	3194	64.30	**1944**	1958	0.74	6194	6803	249.95	6008	7279	274.43	6088	7831	302.83	2613	2753.6	41.65
34	4800	129	713	15171	16390	16390	8.04	**15171**	15186	0.10	16745	17181	13.25	19062	19738	30.10	17749	18138	19.56	16212	16737.6	10.33
35	9600	132	816	29943	39849	39849	33.08	**29943**	31075	3.78	50272	51712	72.70	35778	39650	32.42	38004	39445	31.73	33782	35119.2	17.29
36	19200	132	908	69688	89924	89924	29.04	**69688**	73805	5.91	198838	208419	199.07	102773	108320	55.44	103863	108290	55.39	80720	81725.2	17.27
37	33484	145	1032	123538	141902	141902	14.87	**123538**	127514	3.22	616479	621953	403.45	223752	237002	91.85	220048	222586	80.18	141431	149869	21.31

* Infeasible solution returned by the MSVNS. BRKGA. and BRKMA algorithms.

(1) Tested on an Intel Xeon E5-2660 v2 @ 2.20 GHz × 40 computer, with 384 GB of RAM using multiprocessing.

(2) Tested on a computer equipped with an Intel i3-10100T 3.00 GHz processor, 16 GB of RAM, and the Ubuntu 18.04.6 LTS operating system.

According to Table 10, the proposed ILS algorithm achieved the best results in 81% of the instances, followed by GRASP at 57%, the constructive algorithm with inverted JPA at 33%, MSVNS at 19%, BRKGA at 16%, and BRKMA at 16%. Considering the average results, ILS achieved the best average results in 76% of the instances, followed by GRASP in 22%. Analyzing the largest instances (ID 35, 36, and 37), the performance of ILS was much better than the other algorithms.

Figure 3 illustrates the box plot graph of the distribution of the RPDs of the algorithms evaluated.

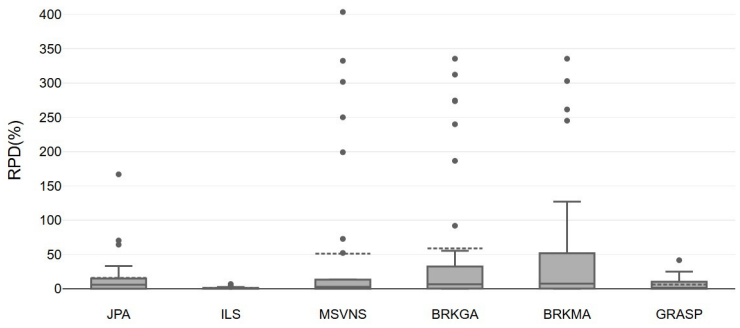

Fig. 3: Box plot of the algorithms' RPD results for the deterministic version of the problem.

According to Figure 3, the dispersion of the RPD values of the ILS was almost zero, much smaller than that of the other algorithms. In addition, the proposed JPA also performed well when compared to the algorithms by Aquino et al. (2019).

We then apply statistical tests to verify if there is a significant difference between the results of the metaheuristic algorithms (ILS, MSVNS, BRKGA, BRKMA, and GRASP). Initially, we checked that the data did not follow a normal distribution using the Shapiro-Wilk test since the p-value was equal to 2.2×10^{-16}. Thus, to verify if the differences between the results presented by the algorithms are statistically significant, we performed the non-parametric Friedman test (Montgomery, 2007) with 95% confidence. The test presented a p-value equal to 1.449×10^{-10}. Therefore, we can conclude that there is a statistical difference in the results. Next, we applied the Paired Wilcoxon signed-rank non-parametric test (Wilcoxon, 1945), with 95% confidence, to identify the pairs of results that present the differences. Table 11 reports the results of this test.

Table 11: p-values of the paired Wilcoxon signed-rank test (α = 0.05).

Group 1	Group 2	p-value
ILS	MSVNS	0.0004729
ILS	BRKGA	1.296×10^{-5}
ILS	BRKMA	8.515×10^{-6}
ILS	GRASP	0.005386
MSVNS	BRKGA	0.0129
MSVNS	BRKMA	0.05023
BRKGA	BRKMA	0.1194
MSVNS	GRASP	0.01205
BRKGA	GRASP	0.0004733
BRKMA	GRASP	0.0005609

Table 11 shows a significant statistical difference between the results of the ILS algorithm and the other algorithms (p-value ≤ 0.05). Thus, these tests confirm that ILS outperforms all other algorithms regarding the RPD results.

5.4. Comparison of the results of the algorithms for the stochastic version of the problem

As the previous subsection showed, the ILS algorithm performed best among the algorithms evaluated for the problem under study. However, all the tests were carried out with deterministic job duration, which is not a reality in the industrial environment. For this purpose, we evaluated in this section the proposed SIM-ILS algorithm (see Algorithm 1) to deal with the uncertainties present in the job duration.

In the computational experiments, we use the normal distribution in the SIM-ILS algorithm to represent the distribution of the job duration. As in Panadero et al. (2020), we set three values for the job duration variability (L, M, and H, representing the levels of uncertainty: Low, Medium, and High). Each value raises a scenario for the application of the SIM-ILS algorithm. SIM-ILS-L considers that there may be a 10% variability in the duration of jobs. SIM-ILS-M and SIM-ILS-H consider the variability of up to 20% and 30% in the duration of jobs, respectively. The parameters n_{Fast} and n_{Deep}, representing the number of objective function evaluations with the simulated job duration values, were set to 100 and 1000. These two values are those used by Guimarans et al. (2018) and Keenan et al. (2021). Finally, as in Aquino et al. (2019), the parameter $t_{\max}$ was set to N, i.e., the number of jobs in each instance.

To compare ILS and SIM-ILS, we stochastically evaluate the best solution generated by ILS for the deterministic problem, considering each of the three levels of uncertainty for the stochastic problem. These ILS evaluations give rise to ILS-L, ILS-M, and ILS-H scenarios. Table 12 reports the results of the stochastic evaluation of the best ILS solution for the deterministic problem and the best SIM-ILS solution for the stochastic problem at each level of uncertainty. In this table, column ILS represents the best solution provided by the ILS for the deterministic problem. Columns ILS-L, ILS-M, and ILS-H show their stochastic evaluations at each level of uncertainty of the stochastic problem. The other columns show the best SIM-ILS results in each uncertainty scenario.

Table 13 reports the values of the stochastic RPDs of the SIM-ILS at each level of uncertainty, given by Equation (19).

$$RPD_I^S = \frac{stochCost_{\textit{SIM-ILS-I}} - stochCost_{\textit{ILS-I}}}{stochCost_{\textit{ILS-I}}} \tag{19}$$

where $stochCost_{\textit{ILS-I}}$ represents the stochastic evaluation of the ILS best solution in Scenario $I \in \{\text{L, M, H}\}$ and $stochCost_{\textit{SIM-ILS-I}}$ represents the stochastic cost of best solutions returned by the algorithm SIM-ILS in the same Scenario I. Negative RPD values indicate that the SIM-ILS solution is better than the stochastically evaluated ILS solution for each level of uncertainty.

The results show that the SIM-ILS solutions are better or equal to the ILS ones in scenarios L, M, and H in 61% of the tests (i.e., in 68 of the 111 tests). The cases in which the RPD was positive indicate that SIM-ILS could not find the best solution generated by ILS. This result occurred because the SIM-ILS and ILS algorithms are limited in execution time, given by the number of jobs in each instance. As a SIM-ILS solution requires more evaluations of the objective function to know its stochastic evaluation, it does not have enough time to better evaluate the solution space compared to ILS. One way of further improving SIM-ILS results is to consider the initial solution provided by ILS as its initial solution. Another way is to increase the execution time of SIM-ILS.

Table 12: Results of stochastic evaluation of the best ILS solution and the best SIM-ILS solution at each level of uncertainty.

Instances				Deterministic evaluation' result	Stochastic evaluation' result at each level of uncertainty					
ID	N	M	Q	ILS	ILS-L	ILS-M	ILS-H	SIM-ILS-L	SIM-ILS-M	SIM-ILS-H
01	150	148	120	1848	1847	1837	1827	1847	1841	1834
02	150	75	129	30	30	30	29	30	30	29
03	150	102	91	3273	3269	3107	2711	3273	3078	2750
04	150	57	126	24	24	24	24	24	24	24
05	150	92	93	49	49	49	49	49	49	49
06	150	71	101	106	707	720	683	696	686	672
07	300	158	179	1931	2117	2118	2124	2022	2058	1972
08	300	221	239	2451	2456	2455	2454	2455	2456	2452
09	300	112	177	3359	3359	3359	3325	3634	3359	3353
10	300	75	181	35	34	34	34	34	34	34
11	300	121	162	65	65	65	65	65	65	65
12	300	119	176	308	913	891	915	903	755	910
13	600	165	329	4494	4545	4584	4607	4744	4796	4775
14	600	256	388	3380	3425	3440	3445	3391	3486	3413
15	600	120	288	8577	8742	8923	8870	8826	8659	7990
16	600	77	215	37	128	179	201	127	180	196
17	600	126	279	410	417	504	542	411	465	556
18	1200	186	519	10921	11864	12066	12271	11807	12020	12288
19	1200	263	666	9517	9656	9653	9667	9641	9610	9874
20	1200	122	470	21921	24455	25327	25274	23594	23021	23769
21	1200	88	252	299	460	534	580	462	531	571
22	1200	130	420	513	518	613	696	517	592	631
23	1200	122	403	6836	7842	8029	8249	8418	7936	9919
24	2400	188	738	26932	30668	31085	31184	28866	28844	30993
25	2400	130	603	37566	40522	41354	41136	41594	40870	42350
26	2400	90	278	570	1575	1694	1709	1479	1693	1746
27	2400	132	701	621	690	828	883	757	718	904
28	2400	126	561	8242	10288	10757	11259	10946	11678	11939
29	4800	197	1128	56902	60238	60835	61356	60352	61553	61768
30	4800	78	1286	40953	43815	44005	44188	43065	43032	44228
31	4800	130	720	86147	93445	94092	93667	93996	96340	92017
32	4800	91	294	194944	198803	199854	199973	198529	199249	200498
33	4800	135	990	1944	2118	2331	2476	2548	2381	2893
34	4800	129	713	15171	17924	18150	18855	19280	19430	19815
35	9600	132	816	29943	36101	36731	37874	37898	40366	40649
36	19200	132	908	69688	80964	82206	84035	78422	80989	82558
37	33484	145	1032	123538	141969	143817	146249	140729	142124	143278

6. Conclusions

This work focused on the problem of allocating preventive maintenance jobs to a 52-week schedule. We develop constructive and job positioning algorithms that work with an ILS-based metaheuristic algorithm to deal with it. The aim is to minimize the cost of using maintenance work teams and the cost of hiring outsourced teams to perform unscheduled jobs.

Initially, we developed a new constructive algorithm capable of automatically testing a set of dispatching rules defined from a set of criteria and returning the best one for each instance. We evaluate it through computational experiments using two versions of this constructive algorithm on medium- and large-scale instances from the literature. We showed that the version of the constructive algorithm that uses the inverted job positioning algorithm (named Inverted JPA) obtained the best performance in all the instances evaluated since it generated the lowest costs. In addition, we observed that this proposed constructive algorithm constructs good-quality solutions in much less computational time than the metaheuristic algorithms of the literature. We also registered that it generated better solutions for some instances than the existing ones, including the largest instance taken from the industry.

We also developed an ILS-based metaheuristic algorithm to improve the solutions generated by the constructive heuristic algorithm. In the computational experiments, the solution provided by the best constructive algorithm, i.e., the constructive algorithm with inverted JPA, was used as the starting point for the ILS algorithm. We showed that the ILS algorithm outperformed other metaheuristic algorithms in the literature, achieving the best results in 81% of the instances.

Table 13: Results of the RPD_I^S for the best SIM-ILS solution at each level of uncertainty.

Instances				RPD_I^S		
ID	N	M	Q	SIM-ILS-L	SIM-ILS-M	SIM-ILS-H
01	150	148	120	0.00	0.22	0.38
02	150	75	129	0.00	0.00	0.00
03	150	102	91	0.12	-0.93	1.44
04	150	57	126	0.00	0.00	0.00
05	150	92	93	0.00	0.00	0.00
06	150	71	101	-1.56	-4.72	-1.61
07	300	158	179	-4.49	-2.83	-7.16
08	300	221	239	-0.04	0.04	-0.08
09	300	112	177	8.19	0.00	0.84
10	300	75	181	0.00	0.00	0.00
11	300	121	162	0.00	0.00	0.00
12	300	119	176	-1.10	-15.26	-0.55
13	600	165	329	4.38	4.62	3.65
14	600	256	388	-0.99	1.34	-0.93
15	600	120	288	0.96	-2.96	-9.92
16	600	77	215	-0.78	0.56	-2.49
17	600	126	279	-1.44	-7.74	2.58
18	1200	186	519	-0.48	-0.38	0.14
19	1200	263	666	-0.16	-0.45	2.14
20	1200	122	470	-3.52	-9.10	-5.95
21	1200	88	252	0.43	-0.56	-1.55
22	1200	130	420	-0.19	-3.43	-9.34
23	1200	122	403	7.35	-1.16	20.24
24	2400	188	738	-5.88	-7.21	-0.61
25	2400	130	603	2.65	-1.17	2.95
26	2400	90	278	-6.10	-0.06	2.17
27	2400	132	701	9.71	-13.29	2.38
28	2400	126	561	6.40	8.56	6.04
29	4800	197	1128	0.19	1.18	0.67
30	4800	78	1286	-1.71	-2.21	0.09
31	4800	130	720	0.59	2.39	-1.76
32	4800	91	294	-0.14	-0.30	0.26
33	4800	135	990	20.30	2.15	16.84
34	4800	129	713	7.57	7.05	5.09
35	9600	132	816	4.98	9.90	7.33
36	19200	132	908	-3.14	-1.48	-1.76
37	33484	145	1032	-0.87	-1.18	-2.03

However, the ILS algorithm developed and others in the literature only deal with the deterministic version of the problem. They do not consider the uncertainties that can occur during the execution of a job. Thus, a high-quality solution for the deterministic problem is not necessarily the best for the stochastic problem.

To deal with this situation, we proposed the SIM-ILS algorithm, a simheuristic algorithm capable of considering these stochastic aspects of a real industry environment. In this algorithm, the solutions are submitted to a stochastic evaluation through Monte Carlo simulations considering the variation in job duration.

We tested the SIM-ILS algorithm in three scenarios, which differ in terms of levels of uncertainty regarding the job duration, and compared their results with those provided by the ILS. We showed that the proposed simheuristic algorithm finds the best solutions in 61% of different industrial environments, from environments with low variability in job duration to those with high variability. For example, for the largest instance (with 33484 jobs), the stochastic evaluation of the best solution produced by ILS costs 1240 units more than the one generated by SIM-ILS, considering the scenario with a level of uncertainty equal to 10%. In other words, the best ILS solution is worse than the SIM-ILS solution, resulting in more unscheduled jobs and/or the need for more work teams.

In future work, we suggest considering the uncertainty in the availability of work teams. In addition, we intend to treat maintenance job windows as flexible. In this case, jobs scheduled outside their windows are penalized in the objective function. This variant of the problem could reduce the need to hire outsourced teams to perform these jobs.

Acknowledgments

The authors are grateful for the support provided by the Universidade Federal de Ouro Preto, Vale S.A, Instituto Tecnológico Vale, and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Finance Code 001), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, grants 303266/2019-8, 311074/2023-5, and 302629/2023-8), and Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG, grant PPM CEX 676/17).

Conflict of interest

The authors declare that they have no conflict of interest.

References

Almakhlafi, A., Knowles, J., 2015. Iterated local search for the generator maintenance scheduling problem. In *7th Multidisciplinary International Conference on Scheduling: Theory and Applications*, Multidisciplinary international scheduling conference: theory & applications, pp. 708–742.

Andrade, J.L.M., Souza, M.J.F., de Sá, E.M., Menezes, G.C., de Souza, S.R., 2024. Formulations and heuristic for the long-term preventive maintenance order scheduling problem. *Computers & Operations Research* 170, 106781. https://doi.org/10.1016/j.cor.2024.106781.

Aquino, R.D., Chagas, J.B.C., Souza, M.J.F., 2018a. A mixed-integer linear programming model and a simulated annealing algorithm for the long-term preventive maintenance scheduling problem. In Abraham, A., Muhuri, P.K., Muda, A.K. and Gandhi, N. (eds), *Intelligent Systems Design and Applications (ISDA 2017)*, *Advances in Intelligent Systems and Computing*. Vol. 736. Springer International Publishing, Cham, pp. 144–153. https://doi.org/10.1007/978-3-319-76348-4_15.

Aquino, R.D., Chagas, J.B.C., Souza, M.J.F., 2018b. A variable neighborhood search algorithm for the long-term preventive maintenance scheduling problem. In *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, INSTICC, SciTePress, Funchal, Madeira, Portugal, pp. 303–310. https://www.scitepress.org/papers/2018/66897/66897.pdf.

Aquino, R.D., Chagas, J.B.C., Souza, M.J.F., 2019. Abordagem exata e heurísticas para o problema de planejamento de ordens de manutenção de longo prazo: Um estudo de caso industrial de larga escala. *Pesquisa Operacional para o Desenvolvimento* 11, 3, 159–182.

Aquino, R.D., Souza, M.J.F., 2016. Instances for the ltpmsp problem. http://www.decom.ufop.br/prof/marcone/projects/LTPMSP/instances2016.zip. Accessed on December 31, 2024.

Cunha, V., Santos, I., Pessoa, L., Hamacher, S., 2020. An ils heuristic for the ship scheduling problem: application in the oil industry. *International Transactions in Operational Research* 27, 1, 197–218.

Ertem, M., As'ad, R., Awad, M., Al-Bar, A., 2022. Workers-constrained shutdown maintenance scheduling with skills flexibility: Models and solution algorithms. *Computers & Industrial Engineering* 172, 108575.

Ferone, D., Hatami, S., González-Neira, E.M., Juan, A.A., Festa, P., 2020. A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem. *International Transactions in Operational Research* 27, 3, 1368–1391.

González-Neira, E.M., Urrego-Torres, A.M., Cruz-Riveros, A.M., Henao-Garcia, C., Montoya-Torres, J.R., Molina-Sánchez, L.P., Jiménez, J.F., 2019. Robust solutions in multi-objective stochastic permutation flow shop problem. *Computers & Industrial Engineering* 137, 106026.

Gu, H., Lam, H.C., Pham, T.T.T., Zinder, Y., 2023. Heuristics and meta-heuristic to solve the roadef/euro challenge 2020 maintenance planning problem. *Journal of heuristics* 29, 1, 139–175.

Guimarans, D., Dominguez, O., Panadero, J., Juan, A.A., 2018. A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times. *Simulation Modelling Practice and Theory* 89, 1–14.

Hatami, S., Calvet, L., Fernández-Viagas, V., Framinán, J.M., Juan, A.A., 2018. A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simulation Modelling Practice and Theory* 85, 55–71.

Juan, A., Li, Y., Ammouriova, M., Panadero, J., Faulin, J., 2022. Simheuristics: An introductory tutorial. In *2022 Winter Simulation Conference (WSC)*, IEEE, Singapore 11-14 Dec., pp. 1325–1339.

Juan, A.A., Barrios, B.B., Vallada, E., Riera, D., Jorba, J., 2014. A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory* 46, 101–117.

Keenan, P., Panadero, J., Juan, A.A., Martí, R., McGarraghy, S., 2021. A strategic oscillation simheuristic for the time capacitated arc routing problem with stochastic demands. *Computers & Operations Research* 133, 105377.

Kizys, R., Doering, J., Juan, A.A., Polat, O., Calvet, L., Panadero, J., 2022. A simheuristic algorithm for the portfolio optimization problem with random returns and noisy covariances. *Computers & Operations Research* 139, 105631.

Lourenço, H.R., Martin, O.C., Stützle, T., 2019. Iterated local search: Framework and applications. In Gendreau, M. and Potvin, J. (eds), *Handbook of metaheuristics*, *International Series in Operations Research & Management Science*. Vol. 272. Springer, Cham, chapter 5, pp. 129–168.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3, 43–58.

Mena, R., Viveros, P., Zio, E., Campos, S., 2021. An optimization framework for opportunistic planning of preventive maintenance activities. *Reliability Engineering & System Safety* 215, 107801.

Montgomery, D., 2007. *Design and Analysis of Experiments* (5th edn.). John Wiley & Sons, New York, NY.

Panadero, J., Doering, J., Kizys, R., Juan, A.A., Fito, A., 2020. A variable neighborhood search simheuristic for project portfolio selection under uncertainty. *Journal of Heuristics* 26, 353–375.

Rabe, M., Deininger, M., Juan, A.A., 2020. Speeding up computational times in simheuristics combining genetic algorithms with discrete-event simulation. *Simulation Modelling Practice and Theory* 103, 102089.

Rabet, R., Ganji, M., Fathi, M., 2024. A simheuristic approach towards supply chain scheduling: Integrating production, maintenance and distribution. *Computers & Operations Research* 153, 111264.

Rodríguez-Espinosa, C.A., González-Neira, E.M., Zambrano-Rey, G.M., 2024. A simheuristic approach using the nsga-ii to solve a bi-objective stochastic flexible job shop problem. *Journal of Simulation* 18, 4, 646–670.

Ruiz-Rodríguez, M.L., Kubler, S., Robert, J., Traon, Y.L., 2024. Dynamic maintenance scheduling approach under uncertainty: Comparison between reinforcement learning, genetic algorithm simheuristic, dispatching rules. *Expert Systems with Applications* 248, 123404.

Santos, M.S., Pinto, T.V., Lopes Júnior, Ê., Cota, L.P., Souza, M.J.F., Euzébio, T.A., 2020. Simheuristic-based decision support system for efficiency improvement of an iron ore crusher circuit. *Engineering Applications of Artificial Intelligence* 94, 103789.

Saraiva, J.T., Pereira, M.L., Mendes, V.T., Sousa, J.C., 2011. A simulated annealing based approach to solve the generator maintenance scheduling problem. *Electric Power Systems Research* 81, 7, 1283–1291.

Viana, H., 2022. *PCM, planejamento e controle da manutenção* (2nd edn.). Qualitymark, Rio de Janeiro, Brazil.

Viveros, P., Miqueles, L., Mena, R., Kristjanpoller, F., 2021. Opportunistic strategy for maintenance interventions planning: A case study in a wastewater treatment plant. *Applied Sciences* 11, 22, 10853.

Wilcoxon, F., 1945. Some uses of statistics in plant pathology. *Biometrics Bulletin* 1, 4, 41–45.

Woller, D., Kulich, M., 2021. The ALNS metaheuristic for the maintenance scheduling problem. In *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO 2021*, SciTePress, pp. 156–164.