The joint maintenance operation selection and technician routing problem

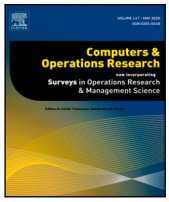
Florian Delavernhe, Bruno BC Castanier, Christelle Guéret, Jorge Mendoza

Computers & Operations Research 167 (2024) 106667

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



The joint maintenance operation selection and technician routing problem

Florian Delavernhe [a,*], Bruno Castanier [b], Christelle Guéret [b], Jorge E. Mendoza [c]

[a] Laboratoire DRIVE, Université de Bourgogne, Nevers, France
[b] Univ Angers, LARIS, SFR MATHSTIC, F-49000 Angers, France
[c] HEC Montréal, CIRRELT, Canada

ARTICLE INFO

Keywords:
Maintenance
Combinatorial optimization
Technician routing
Heuristic
Linear program

ABSTRACT

Numerous studies have explored the challenge of optimizing maintenance grouping, aiming to efficiently manage maintenance resources and reduce costs associated with various maintenance opportunities. Maintenance grouping is even more important when the systems to maintain are geographically distributed because it significantly reduces travel costs. In this research, we tackle a novel problem that integrates condition-based maintenance with the selection of maintenance operations and technician routing. The problem involves the selection of machines requiring maintenance for each time period, determining the nature of required operations (based on the uncertain degradation state of the machines), selecting suitable technicians based on their skills and availability, and planning their routes. We formulate this problem as a mixed-integer program and propose a heuristic approach for its solution. Our method constructs a solution by iteratively adding maintenance operations to technician routes. To assess the method's performance, we conduct experiments that evaluate both running times and solution quality.

1. Introduction

The cost of maintaining an industrial system is a non-negligible part of its total life-cycle costs. For example, in the offshore wind power industry, maintenance can represent up to 25% of the life-cycle costs of a turbine (Snyder and Kaiser, 2009), and poorly managed maintenance operations (MOs) can drastically increase this figure. For instance, Aoudia et al. (2008) studied the main industrial plants of a major oil and gas group and concluded that the impact of ineffective maintenance management costs the company around 566 million USD. In the opposite vein, Al-Najjar (2007) used data from Swedish paper mills to show how investing in the generation of smart maintenance plans could lead to significant savings (up to ten times the investment).

However, optimizing maintenance planning is a daunting task. The first difficulty is to make the MOs decision (which MOs and when to perform them) given the system performance and an associated failure/degradation forecast. A second challenge is to integrate into these decisions the logistic support part (which operations to group on the same day, which technician to send to perform these operations) to minimize the costs. For example, if a production line needs to be stopped for a full day whenever one of its machines is maintained, it is clear that grouping on the same day the MOs that need to be done on the machines of this production line avoids paying multiple site-downtime costs. Recently, the scientific literature has started to study the problem of making MO decisions driven by both logistic consideration and degradation forecast.

Failure/degradation-driven maintenance decision is the process of selecting the MOs to be performed by forecasting the degradation state of the machine (typically with a probability law) and forecasting the impact of the MOs on the degradation state. An example is the use of the rejuvenation parameter, which measures the efficiency of maintenance activity on the machine state (Nguyen and Chou, 2019). This parameter represents the percent reduction of the degradation after an MO with the degradation of a machine modeled with, for example, a Weibull distribution. Nguyen and Chou (2019) use a rejuvenation parameter equal to one to indicate a replacement (as good as new), zero to indicate no maintenance done (as bad as old), and values in the interval $(0, 1)$ to indicate an imperfect repair. The authors seek a method to determine a maintenance schedule for offshore wind turbines that minimizes the maintenance costs while guaranteeing system reliability (i.e., a failure rate that does not exceed a given threshold), but they ignore the logistic support which is particularly costly in this context.

In the literature on logistics problems related to MOs, decision-making is typically influenced by resource management (e.g., technicians, spare parts) and various logistic costs (such as inventory and traveling expenses). A significant logistical concern frequently discussed in the literature pertains to the traveling costs associated with

* Corresponding author.
E-mail addresses: florian.delavernhe@u-bourgogne.fr (F. Delavernhe), bruno.castanier@univ-angers.fr (B. Castanier), christelle.gueret@univ-angers.fr (C. Guéret), jorge.mendoza@hec.ca (J.E. Mendoza).

https://doi.org/10.1016/j.cor.2024.106667
Received 12 April 2023; Received in revised form 9 April 2024; Accepted 13 April 2024
Available online 25 April 2024

geographically dispersed systems. Due to market and client proximity requirements, production systems often tend to be more decentralized (Srai et al., 2016; Matt et al., 2015). However, in such decentralized systems, maintenance management is often centralized, with spare parts and technicians centralized at a single depot (Blakeley et al., 2003).

In this context, one approach to cost reduction is the consolidation of maintenance operations at a single site and optimizing the technicians' routes between different production sites. Notably, the offshore wind power industry has been a particular focus of these efforts (Dai et al., 2015; Irawan et al., 2017; Schrotenboer et al., 2018). The primary objective in these cases is to optimize the routing of maintenance for wind turbines, considering time windows defined by weather conditions. However, the maintenance component in these approaches is typically limited to classical periodic preventive maintenance. In other words, the maintenance models are only exploited to establish the due dates for each maintenance operation (Schrotenboer et al., 2018; Stålhane et al., 2016). More complex maintenance decision processes exist in the offshore wind power industry, with preventive and corrective MOs such as in Gonzalo et al. (2022), Irawan et al. (2023). For instance, in Irawan et al. (2023), the authors consider the computation of an MO plan over a 30 day time period (tactical decisions). They build routes for preventive MOs with the possibility to modify the maintenance plan at the beginning of each day (operational decisions) to adapt to stochastic events and perform corrective MOs.

Another critical logistical aspect, addressed in numerous studies, involves the skill levels of the technicians. For example, in the work presented by Zhao et al. (2022a), skill levels are represented by various domains of expertise, each associated with a proficiency level for each technician. A technician's skill level dictates the range of machines on which that technician can perform maintenance operations. Building upon this skill modeling, studies often focus on forming technician teams dispatched collectively to production sites to maintain multiple components, to minimize routing expenses. It is worth noting that incorporating varying skill levels into the technician modeling may also be useful to model additional factors, such as the levels of tools and equipment utilized by maintenance operators during interventions. It is important to emphasize that these studies on routing primarily address logistical decisions and do not encompass the complexity of making these decisions while considering degradation or failure forecasts, as they typically assume that maintenance operations and their due dates are given.

In recent literature, there has been a surge of papers addressing the problem of geographically dispersed systems, where decisions are driven by both degradation states and logistic costs, such as travel. It is worth noting that most of these studies characterize the degradation state using a virtual age, which, in turn, influences the failure probability. In these works, MOs are not given as instance parameters but may be determined to optimize the overall system's reliability. In many cases, MO types are simplified, often reduced to either preventive or corrective maintenance, as seen in López-Santana et al. (2016), Nguyen et al. (2019), Rashidnejad et al. (2018), Si et al. (2021), Urbani et al. (2023). The actual maintenance performed depends on the degradation state when the technician arrives at the machine: a failure induces corrective maintenance, while preventive maintenance is performed in the absence of a failure.

These studies derive their MO decisions from the calculation of an optimal maintenance cycle, representing the ideal interval between maintenance activities. Shorter cycles lead to more frequent MOs, with a higher likelihood of preventive maintenance, which is typically more cost-effective than corrective maintenance. A mathematical formula, linked to machines' failure probabilities over time, is used to determine the optimal cycle and subsequently, the optimal dates for MOs. Thereafter, the decision problem is often simplified by transforming it into a routing problem with time windows centered around the optimal dates, as seen in López-Santana et al. (2016, 2023) and Rashidnejad et al. (2018), or into a routing problem with penalty costs, accounting for the disparity between the actual MO dates and the optimal dates, as demonstrated in Nguyen et al. (2019), Si et al. (2021), Urbani et al. (2023).

However, it is important to note that these works primarily focus on full replacements and do not consider imperfect maintenance, meaning that a machine is consistently restored to a 'good-as-new' state after maintenance. Si et al. (2022) consider two types of maintenance: preventive and replacement. In this work, the authors account for the influence of the actual age of the machine on the outcomes of preventive maintenance. Notably, the effectiveness of successive preventive maintenance diminishes.

There are relatively few studies that explore models involving more than two types of MOs. An example of this can be found in Jia and Zhang (2020), where the model encompasses four distinct maintenance types: perfect, high-level imperfect, low-level imperfect, and minimal repair. Consequently, the authors tackle a complex problem, requiring not only addressing routing decisions and MO planning but also determining the specific type of MO to be executed by the technician. Failures are managed through minimal repairs, where local technicians restore machines without altering their ages, resulting in a fixed penalty cost for failures.

Similarly, O'Neil et al. (2023) consider multiple imperfect maintenance levels as part of the decision process, with each level leading to different virtual age reductions. However, it is important to mention that in this work, minimal repair is not considered; instead, downtime costs are taken into account.

On a different note, Jafar-Zanjani et al. (2022) addresses a related concern by examining the challenge of selecting maintenance centers and backup part suppliers within a geographically dispersed system. While the logistical decisions in this context may vary (no routing), the MO decisions remain a central focus.

Unlike our problem, in these previous studies, there is no consideration for condition-based maintenance. Indeed, in these studies, the degradation state of a machine is planned over time. Consequently, the costs, duration, and impacts of the chosen MOs are precisely known, as the virtual age before and after the MO is predetermined. With a condition-based policy, the problem becomes more intricate, as these elements are uncertain, and the MO performed by a technician on a machine actually depends on the degradation state of the machine when the technician arrives. In other words, the degradation state (or virtual age) remains stochastic after the technician's visit. Studies involving only preventive and corrective maintenance consider condition-based maintenance, but they have an oversimplified view of MOs and reliability, as they only account for 'good-as-new' MOs with two possible conditions: failure or no failure.

In conclusion, the maintenance literature has historically addressed separately the MOs decisions driven by a degradation model and MOs decisions driven by logistic considerations. This distinction was particularly evident in cases involving geographically dispersed systems. Nonetheless, recent literature is shifting its attention towards rectifying this as there is a surge of work considering the joint optimization of MOs and routing decisions. However, in most of these recent studies, MOs are categorized as either preventive or corrective, with always a resulting good-as-new state for the machine maintained. There is limited literature that addresses the concurrent optimization of technician routing and the selection of more than two different MO types, with imperfect maintenance each linked to varying levels of efficiency. However, these works do not yet fully capture the intricacies of systems where planned MOs depend on the uncertain condition of the machines. This simplified perspective on maintenance issues falls short of incorporating present-day maintenance policies such as condition-based and predictive maintenance which are more dynamic and efficient from an economic and availability point of view.

Table 1 provides a summary of the key aspects covered in the optimization papers referenced in this work. The column labeled *Logistic decisions?* refers to decision problems considering logistic aspects,

Table 1
Summary of the key aspects covered in the literature.

Reference	Logistic decisions?	Maintenance selection?	Reliability modelization?	Imperfect maintenances?	Condition based?
Dai et al. (2015)	X				
Flood (1956)	X				
Gonzalo et al. (2022)	X	X			
Irawan et al. (2023)	X	X			
Irawan et al. (2017)	X	X			
Jafar-Zanjani et al. (2022)	X	X	X		
Jia and Zhang (2020)	X	X	X	X	
López-Santana et al. (2016)	X	X	X		
López-Santana et al. (2023)	X	X	X		
Nguyen et al. (2019)	X	X			
Nguyen and Chou (2019)	X	X			
O'Neil et al. (2023)	X	X	X	X	
Rashidnejad et al. (2018)	X	X			
Schrotenboer et al. (2018)	X	X			
Si et al. (2021)	X	X	X	X	
Snyder and Kaiser (2009)	X	X			
Stålhane et al. (2016)	X				
Urbani et al. (2023)	X	X			
Zhang et al. (2022)	X	X		X	
Present work	X	X	X	X	X

such as travel costs between machines and technician allocation. It is important to note that our review focused solely on papers involving these logistic decisions, as it aligns with the problem addressed in our work. However, it is worth mentioning that most recent optimization works on maintenance selection now incorporate logistic considerations. The *Maintenance selection?* column denotes papers where the selection of MOs is part of the decision process, and not predetermined, e.g., the problem of adjusting the exact date of the MO. The *Reliability modelization?* column refers to papers where machine degradation is explicitly modeled, and decisions made in the optimization process have a direct impact on this degradation state. For instance, it includes the use of a virtual age, that is impacted by each MO performed. This aspect is opposed to problems where reliability is represented solely by a penalty cost based on the deviation from a given due date for the MO. The *Imperfect maintenances?* column highlights papers that explore more than two types of MOs, going beyond the conventional preventive and corrective categories that are often simplified into the computation of optimal cycles. It is noteworthy that there are no existing works that simultaneously consider maintenance selection, routing decisions, and condition-based maintenance. Condition-based maintenance introduces an additional element of stochasticity, as the maintenance performed depends on the machine's state when the technician arrives on-site, leading to a stochastic degradation state even after the visit of the technician. Nevertheless, even though they do not encompass routing decisions and maintenance selection, it is worth noting that Zhang et al. (2022) explores condition-based maintenance policies alongside spare parts inventory management, and Zhao et al. (2022b) investigates the optimization of condition-based performance control and maintenance policies. Both of these works exhibit similarities to the current study in their approach to modeling degradation processes.

In this paper, we address a realistic joint maintenance and routing problem, taking into consideration many different aspects studied in the literature but independently. First of all, this problem fits into a realistic framework seen in many previous publications (see, for example, (Rashidnejad et al., 2018; Schrotenboer et al., 2018; Stålhane et al., 2016)), with a set of geographically dispersed machines, a time horizon divided into periods (days or weeks) and a probabilistic model of the degradation state of the machines. Secondly, as a major novelty and contrary to the maintenance-routing literature, we do condition-based MO decisions, taking into account the stochastic degradation state of a machine upon the technician's arrival. Moreover, to further enhance the realism of our model, we also consider that each technician has a known level of expertise that restricts his/her interventions to machines with a level of degradation below a certain threshold. Therefore, since the level of degradation of a machine is probabilistic when a technician arrives at a site, he or she may not be able to perform a planned MO on a machine if the degradation is not within his or her level of expertise. Thus, the degradation state of a machine is not known exactly after a technician's visit. To resume, the optimization problem considered here aims to plan: (i) the machines maintained during each period, (ii) the MOs performed on the machines maintained, (iii) the technicians performing the MOs, and (iv) the technicians' routes. In the present work, we propose a tailored heuristic method. Results are satisfactory both in terms of computation time and solution quality, compared to the best solutions produced by CPLEX.

This paper is organized as follows: Section 2 presents the problem, the notations, an illustrative example and the mathematical formulation. Section 3 describes the algorithm and all its components, Section 4 gives the results of the experiments, and Section 6 concludes this work.

2. Problem definition

We consider that a machine has a virtual age affecting its performance level. The performance level reflects how efficiently the machine operates compared to a new one, such as the production rate (e.g., units of a product produced per minute). A machine with a higher virtual age operates less efficiently resulting in a lower performance level and reduced production rate. Moreover, in the present study, we also consider that a machine may fail. When a machine experiences a failure, it halts production, requiring immediate action. We assume that minimal repairs are carried out by on-site technicians who can quickly restore the machine to a functional state. However, as observed in previous studies, minimal repairs do not alter the machine's virtual age nor reduce the likelihood of another failure. To model both virtual age and failures, we use degradation states, represented by a Markovian model that allows for the modeling of stochastic degradation events.

In other words, the machine is not constrained to having an exactly known virtual age throughout the entire time horizon. Failure and performance level are integrated into our model as expected penalty costs associated with the machine's degradation state. The expected failure cost is calculated considering the minimal repair cost and the failure probability, while the performance level cost accounts for the losses in the performance of a machine compared to a new one. To summarize, our machine reliability model, represented by the degradation state, encompasses the machine's performance level and failure probability through associated penalty costs. Our model allows for stochastic degradation events.

We consider a set $\mathcal{M}$ of geographically dispersed machines and a depot δ. $\forall (i_1, i_2) \in (\mathcal{M} \cup \{\delta\})^2, d_{i_1 i_2}$ is the known travel time between machines i_1 and i_2 or between machines and the depot. The time horizon is discretized into a set $\mathcal{T} = \{1, 2, \dots, |\mathcal{T}|\}$ of periods. The state of a machine is represented by a given discrete-time Markovian model over the horizon of time. The set $\mathcal{K} = \{1, 2, \dots, |\mathcal{K}|\}$ represents the different degradation states of the machines, with State 1 being the good-as-new state, State $|\mathcal{K}|$ being the worst state, and the other states being intermediate states that represent different levels of degradation such that, $\forall (k_1, k_2) \in \mathcal{K}^2$, if $k_1 < k_2$ then k_1 is a less-degraded state than k_2. $\forall (k_1, k_2) \in \mathcal{K}^2$, $p_{k_1 k_2}$ is the probability that a machine in state k_1 will degrade to state k_2 between two consecutive time periods. i.e., a machine in state k_1 at the end of a period $t \in \mathcal{T} \mid t < |\mathcal{T}|$ transitions into state k_2 at the beginning of period $t + 1$. Note that, if $k_1 > k_2, p_{k_1 k_2} = 0$ because a machine cannot reduce its degradation by itself and $\forall k_1 \in \mathcal{K}$, $\sum_{k_2 \in \mathcal{K}} p_{k_1 k_2} = 1$. For each machine $i \in \mathcal{M}$, the initial probability to be in a degradation state $k \in \mathcal{K}$ before the first period is given by wa_{i0}^{k}.

We denote as $\mathcal{N}$ the set of technicians available to perform the MOs. Each technician $j \in \mathcal{N}$ has a skill level c_j such that technician j can only operate on a machine if its degradation state $k \in \mathcal{K}$ is less than or equal to the technician's skill level, i.e., $c_j \geq k$. During each period, each technician executes his or her route and all their assigned MOs. The technicians are all based at the depot δ and each technician starts and ends their route at the depot.

In our problem, the MOs have a *target state*, that is, the degradation state that the given machine should have after the MO is completed. Upon arrival at the machine location, technicians may face three different situations: (i) the machine degradation state exceeds the technician skill level, (ii) the machine degradation state is less than or equal to the target state of the planned MO, or (iii) the machine degradation state exceeds the target state but remains within the technician's skill level. In cases (i) and (ii), no MO is performed and the machine remains in its current degradation state. In case (iii), the technician executes the requisite MO to bring the machine to the target state. Section 2.1 illustrates the three possible situations. The operating time and operating cost of each MO depend on the current state of the machine and the target state. We introduce $r_{k_1 k_2}$ and $o_{k_1 k_2}$ $\forall (k_1, k_2) \in \mathcal{K}^2 \mid k_1 > k_2$ to denote the costs and times needed to take a machine from degradation state k_1 to k_2. $\forall (k_1, k_2, k_3) \in \mathcal{K}^3 \mid k_1 > k_2 > k_3$, $r_{k_1 k_2} + r_{k_2 k_3} = r_{k_1 k_3}$ and $o_{k_1 k_2} + o_{k_2 k_3} = o_{k_1 k_3}$. During each period, no more than one MO can be performed on each machine.

For the sake of simplicity, we use the tuple (k, i, j, t), $k \in \mathcal{K}$, $i \in \mathcal{M}$, $j \in \mathcal{N}$, $t \in \mathcal{T} \mid k < c_j$ to denote an MO planned on machine i for technician j during period t with a target state k.

The duration of the technicians' routes is limited by time constraints. Each technician $j \in \mathcal{N}$ has an available working time E for each period $t \in \mathcal{T}$. This is the time available for performing MOs and traveling between machine locations. It is worth noting that E can be expressed in different time units (e.g., a day or a week). As it stands, for a solution to our problem to be feasible, the duration of each technician route for each period must be shorter than E. Therefore, when planning an MO on a machine, we must always consider the worst-case MO time, that is, the time needed to bring the machine from the degradation state matching the technician's highest skill level to the target level. For example, if technician $j \in \mathcal{N}$ has a planned MO with a target state $k \in \mathcal{K} \mid k < c_j$, the solution has to consider the operating time $o_{c_j k}$.

We denote by Ctr the cost per traveling time unit, and by C_k the penalty cost incurred if a machine finishes a period in state $k \in \mathcal{K}$. Typically, these costs increase with the degradation of the machine, starting low (may be equal to 0) for state 1 and reaching a high and prohibitive value for the worst state $|\mathcal{K}|$. More formally, $\forall (k_1, k_2) \in \mathcal{K}^2 | k_1 < k_2, C_{k_1} \leq C_{k_2}$. As mentioned previously, these costs encompass the failure costs and the performance level costs.

The problem is to generate a maintenance plan with minimal cost. The maintenance plan is made of a selection of MOs and the routes to execute those MOs. The objective function is to minimize the sum of the expected costs of the MOs, the traveling costs, and the sum of the expected penalty costs. In summary, the problem consists in selecting the MOs to execute in each period (i.e., the machines and target states), assigning each MO to a technician, and generating the routes for the technicians.

Table 2 summarizes the notations. The following assumptions are made:

- Minimal repairs are performed whenever a failure is detected by a low-level local technician
- The exact degradation state of a machine is only known when a technician arrives on the machine
- The degradation of a machine remains constant during the execution of the maintenance plan, i.e., within a period, if no maintenance is performed
- The skill levels are progressive such that the set of MOs that can be performed by a lower-skilled technician is included in the set of MOs that can be performed by a higher-skilled technician
- A technician arriving on a machine with a degradation state outside his/her skill cannot perform an MO

2.1. Degradation states and transitions: An illustrative example

This section briefly presents three illustrations (Figs. 1, 2, and 3) representing the evolution of the state of a machine over several periods. The example uses six degradation levels. Fig. 1 shows the transitions between the states of one machine over three periods without maintenance. The machine degrades over time and, when a new period begins, it either maintains its degradation state or degrades into a worse state. In this example, the initial state of the machine is State 1.

Fig. 2 presents the evolution of the degradation state of a machine over three periods but includes planned MOs executed by a technician with a skill level $c_j = 4$. In the figure, the terms "t_1 before maintenance", "t_2 before maintenance", and "t_3 before maintenance" refer to the beginning of periods t_1, t_2, and t_3, and the terms "t_1", "t_2", and "t_3" refer to the end of these periods. All the states surpassing the technician skill level are depicted in gray.

We consider a machine that is initially good as new in State 1 at the beginning of the horizon (i.e., $wa_{i0}^{1} = 1$ with i being the machine). The first MO is planned in period t_1, having State 1 as the target state; in other words, the goal is to return the machine to the good-as-new state. No MO is planned in t_2 (the state of the machine is the same at the beginning and the end of the period). Finally, an MO is planned in t_3 having State 2 as the target state.

Fig. 2 shows the expected states and transitions for period t_1. Before the MO, the probabilities of all the states are computed by considering the degradation over time from the initial state of the machine, thus using the transition probabilities from State 1. During t_1, the technician will travel to the machine to execute an MO. However, if the machine is in States 5 or 6, the technician does not have the required skill level to intervene, so the degradation state remains the same until the end of period t_1. Similarly, if the machine is already in State 1, the technician

Table 2
Summary of notation.

Sets	
$\mathcal{M}$	Set of machines
$\mathcal{N}$	Set of technicians
$\mathcal{T}$	Set of periods
$\mathcal{K}$	Set of degradation states
Constants	
$p_{k_1k_2}$	Transition probability from state k_1 to k_2
$d_{i_1i_2}$	Travel time between i_1 and i_2
$o_{k_1k_2}$	Time needed to change the state of a machine from k_1 to k_2
$r_{k_1k_2}$	Cost of an operation to change the state of a machine from k_1 to k_2
wa^k_{i0}	Probability that machine i is in state k at the beginning of the time horizon
c_j	Skills level of technician j
Ctr	Traveling cost per unit time
C_k	Penalty cost for state k
E	Time available for each technician and for each period
Indexes	
δ	Depot
i	Machine index
t	Period index
k	State index
j	Technician index

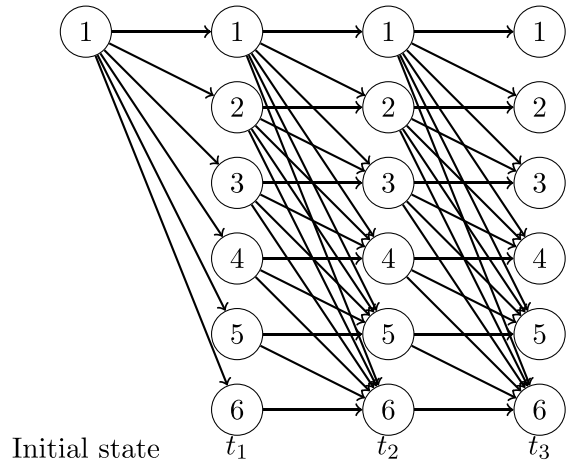


Fig. 1. Illustration of degradation states of a machine and their possible transitions.

does not need to intervene. Conversely, if the machine is in States 2, 3, or 4, the technician can and will maintain the machine, bringing it to the target state (i.e., State 1). The probability of each state at the end of period t_1 is therefore computed by considering these transitions and their probabilities, and we can see, for example, that states 2, 3, and 4 are not possible and that the probability of State 1 at the end of the period is the sum of the probabilities of states 1, 2, 3, and 4 at the beginning of the period. The expected penalty cost to be paid for period t_1 is the sum, for all k in $\mathcal{K}$, of the probability of state k times C_k.

Next, no maintenance is planned in period t_2 and the probability of each state is the same at the beginning and at the end of the period. It only depends on the probabilities at the end of period t_1 and on the known transition probabilities.

Finally, one MO is planned in period t_3. States 5 and 6 have similar transitions as in t_1 (i.e., if the machine is in such states at the beginning of the period, the technician cannot intervene). The difference here is that the MO's target state is State 2 and if the machine is in a better state at the beginning of period t_2 (State 1), it remains in that state.

Fig. 3 depicts an illustrative example of the routing and maintenance decisions on an instance with 6 machines. This example considers three time periods and two technicians with varying skill levels. More specifically, Technician 1 is more skilled than Technician 2. To simulate the degradation of machines, each machine is associated with a Markovian model, akin to the representation in Fig. 2. However, for the sake of clarity, and without loss of generality, we limit this example to three degradation states: good-as-new (state 1), minor degradation (state 2), and major degradation (state 3). The associated degradation probabilities are: 20% from state 1 to state 2, 10% from state 1 to state 3, and 30% from state 2 to state 3. All the machines have the same initial probabilities for the degradation states. Probabilities of each state at the end of each period and for each machine are given in the figure. In this scenario, Technician 1 can execute maintenance operations on machines across all degradation states. On the other hand, Technician 2 faces limitations and is unable to address major degradation (i.e., state 3). The illustrated solution considers that Technician 1 performs MOs transitioning the machine from major degradation to minor degradation and Technician 2 from minor degradation to good-as-new. In the schedule table (bottom of the figure), (mi, kj) indicates that a MO is planned on machine $i \in \mathcal{M}$ with a goal degradation state of $j \in \mathcal{K}$. D indicates the depot where the technician starts. This figure allows us to illustrate several aspects of the problem and its solutions:

1. Only Technician 1 is skilled enough to repair a major degradation. Consequently, his/her available time is crucial to decreasing the high penalty costs of highly degraded states (i.e., major degradation costs). This motivates the execution of short MOs that shift machines from major degradation to minor degradation. Thereafter, Technician 2 consistently performs operations on these machines during the following periods, restoring them to a good-as-new state. For example, Technician 1's MOs on machines 3 and 4 during Period 2 result in a null probability of major degradation, leaving them either in a minor degradation or good-as-new condition. In Period 3, Technician 2 performs MOs on machines 3 and 4, significantly increasing the likelihood of them returning to a good-as-new state. It is important to note, however, that there is still a possibility of major degradation, as the machines may have deteriorated between periods 2 and 3. In summary, two MOs are performed to fully restore a machine from a major degradation state to good-as-new, this is due to the limited availability of time for Technician 1, who focuses on addressing the complex issues with the machines leaving the simpler maintenance tasks to the Technician 2. It means that there is an incentive to perform imperfect MOs, and not only full-repair maintenance, i.e., the target state is not always the good-as-new state.

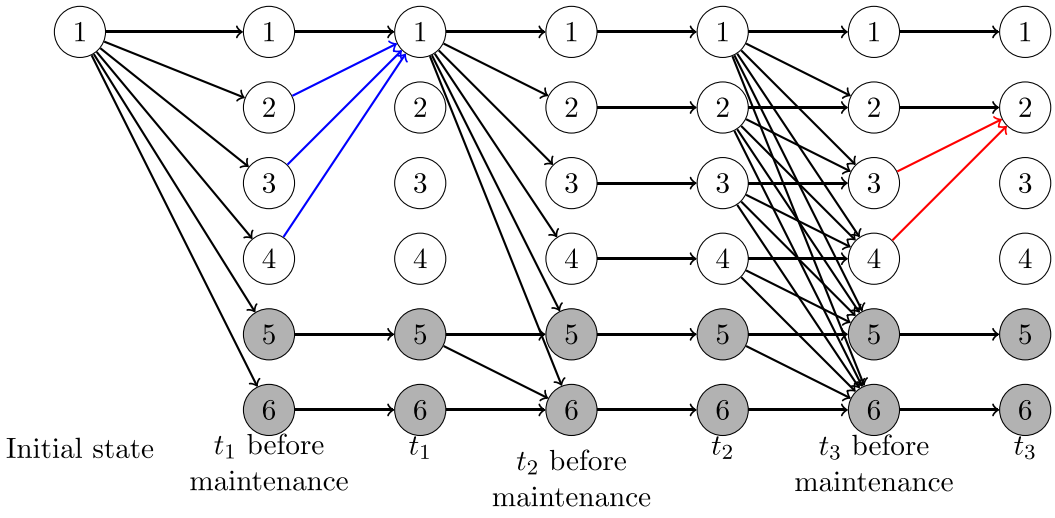


Fig. 2. Illustration of states and transitions with planned MOs.

2. In the first period, Technician 2 is engaged in numerous minor MOs on the machines, shifting them, if possible, from a minor degradation state to the good-as-new state. The good-as-new state is less likely to transition to a major degradation state in the next periods thus reducing the overall expected penalty and maintenance costs. In summary, the technician performs various types of preventive MOs and does not only intervene in the last degradation state.
3. There is no incentive to visit all machines in every period consistently. This can be seen in Period 2 where multiple machines are not maintained. The reasons are that (i) Technician 1 does not have enough available time to maintain more than two machines and (ii) the probability of minor degradation is low, rendering the intervention of Technician 2 not cost-efficient. It should be noted that logistical costs might dissuade a technician from moving from the depot, particularly when such costs outweigh the expected costs of the degradation in the system.

2.2. A mixed integer programming model

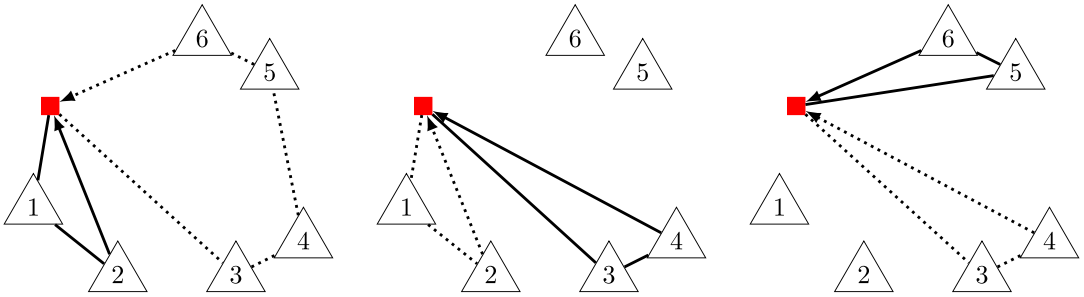
This section introduces a mixed integer programming formulation for the problem. The formulation is given by Model 1

This model uses the following decision variables:

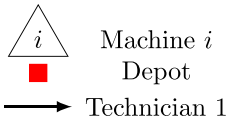
- x^k_{ijt}, $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$, $\forall j \in \mathcal{N}$, $\forall t \in \mathcal{T}, | \ k < c_j$ equals 1 if the MO (k, i, j, t) is planned, 0 otherwise. In other words, x^k_{ijt} equals 1 if machine i has a maintenance planned in period t for technician j with a target state k.
- $y^{i_1 i_2}_{jt}$, $\forall (i_1, i_2) \in (\mathcal{M} \cup \{\delta\})^2 \mid i_1 \neq i_2$, $\forall j \in \mathcal{N}$, $\forall t \in \mathcal{T}$ equals 1 if technician j moves from i_1 to i_2 at period t, zero otherwise.
- $y^{\delta\delta}_{jt}$, $\forall j \in \mathcal{N}$, $\forall t \in \mathcal{T}$ equals 1 if technician j stays at the depot during period t (i.e., does not execute MOs), zero otherwise.
- z_{it}, $\forall i \in \mathcal{M}$, $\forall t \in \mathcal{T}$ is the expected cost of the MO planned on machine i at period t. The variable takes the value of zero if no MO is planned. The value is computed by using the probabilities of the states of i.
- wa^k_{it}, $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$, $\forall t \in \mathcal{T}$ is the probability that machine i is in state k at the end of period t (after an MO is performed).
- wb^k_{it}, $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$, $\forall t \in \mathcal{T}$ is the probability that machine i is in state k at the beginning of period t (before an MO is performed).
- u_{ijt}, $\forall i \in \mathcal{M}$, $\forall j \in \mathcal{N}$, $\forall t \in \mathcal{T}$ is a dummy variable for the sub-tour elimination constraints, as in Ref. Miller et al. (1960).

The constraints are the following:

- **Constraint (1)** is the objective function. The first part sums the traveling times and multiplies them by the traveling cost Ctr. The second part sums the expected MO costs and the expected penalty costs. $\forall i \in \mathcal{M}, \forall t \in \mathcal{T}, \forall k \in \mathcal{K}, wa^k_{it}$ is the probability that machine i is in state k at the end of period t, thus the probability of paying C_k for i at t.
- **Constraints (2)** fix the values of wb^k_{it}, $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$, $\forall t \in \mathcal{T}$ (probabilities at the start of a period) by taking the transition probabilities and the probabilities either at the end of the previous period, or the initial probabilities (i.e., wa^k_{i0}, $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$).
- **Constraints (3)–(5)** are used to fix the probabilities at the end of a period (wa^k_{it}, $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$, $\forall t \in \mathcal{T}$). With **Constraints (3)**, the probability of a state at the end of a period equals the probability at the beginning ($wa^k_{it} = wb^k_{it}, k \in \mathcal{K}, i \in \mathcal{M}, t \in \mathcal{T}$) if no MO is planned (i.e., for $k_2 \in \mathcal{K}, j \in \mathcal{N}$, $x^{k_2}_{i,j,t} = 0 \ \forall c_j \geq k > k_2$). **Constraints (4)** are used for the other cases and fix the probability of the target state of an MO to the sum of the probabilities of the states on which the technician can intervene. For example, in Fig. 1, at period t_3, **Constraints (3)** are used for the probabilities of State 1 (better than the maintenance planned) and States 5 and 6 (outside the skills level of the technician). Still, in Fig. 1, **Constraints (4)** fix the probability of State 2 at period t_3 by summing the probabilities before maintenance of States 2, 3, and 4. The probabilities of states 3 and 4 are not restrained by **Constraints (3)** and **Constraints (4)** and are fixed at zero by **Constraints (5)**.
- **Constraints (6)** compute the expected operation costs (z_{it}, $\forall i \in \mathcal{M}$, $\forall t \in \mathcal{T}$). The expected cost of an MO is simply the sum of the probability of each state $k \in \mathcal{K}$ multiplied by the cost of the MO needed to bring the state from k to the target state. No cost is incurred if no MO is performed. Constant M is a large positive value. In our implementation, it corresponds to the smallest possible $r_{|\mathcal{K}|k_1}$, which leads to the highest possible expected cost of maintenance (i.e., the machine is in the worst degradation state).
- **Constraints (7)** ensure that there is at the most 1 planned MO per machine per period.
- **Constraints (8)** restrain the working time of a technician during a period. The first part sums the worst-case operating times of the MOs and the second part sums the traveling times. Thereafter, it is referred to as the duration constraint.



(a) Time period 1 (b) Time period 2 (c) Time period 3



Machine	1 & 2			3 & 4			5 & 6		
State	1	2	3	1	2	3	1	2	3
Period 1	20%	80%	0%	60%	0%	40%	60%	0%	40%
Period 2	74%	0%	26%	42%	58%	0%	42%	12%	46%
Period 3	51.8%	14.8%	33.4%	78.4%	0%	21.6%	29.4%	70.6%	0%

(d) Probabilities of the degradation states at the end of each time period

Technician	Period 1	Period 2	Period 3
Technician 1	$D \to (m1,k2) \to (m2,k2) \to D$	$D \to (m3,k2) \to (m4,k2) \to D$	$D \to (m5,k2) \to (m6,k2) \to D$
Technician 2	$D \to (m3,k1) \to (m4,k1) \to (m5,k1) \to (m6,k1) \to D$	$D \to (m1,k1) \to (m2,k1) \to D$	$D \to (m3,k1) \to (m4,k1) \to D$

(e) Schedule of MO activities

Fig. 3. Illustration of a solution.

- **Constraints (9) and (10)** force a maintained machine to be on the route of its technician: the technician arrives and leaves the machine once.
- **Constraints (11) and (12)** force a technician to leave and enter the depot at most once during the period.
- **Constraints (13)** are the sub-tour elimination constraints, as in Ref. Miller et al. (1960).
- **Constraints (14)–(20)** define the nature of the decision variables.

A quick inspection of the model shows that the complexity of the problem increases rapidly with the number of periods, technicians, and machines. The number of constraints and variables are both $O(|\mathcal{M}| \times (|\mathcal{M}||\mathcal{N}||\mathcal{T}| + |\mathcal{K}||\mathcal{N}||\mathcal{T}|))$.

3. A heuristic solution approach

In this section, we introduce a single-pass heuristic that combines components that heavily rely on the specific structure of the problem and components that are directly borrowed or adapted from solution methods for related problems. The resulting method can be used as a stand-alone procedure or be embedded into more sophisticated solution schemes.

3.1. Overall algorithm

This section presents an overview of the algorithm introduced in our work. The algorithm starts with an empty solution and adds a new MO to the current solution at each iteration. The MO to be added is selected according to the utilities. The algorithm stops when it is no longer possible to incorporate MOs into a feasible solution. The current solution is frequently improved by local search operators. Algorithm 1 presents the overall method and a more comprehensive illustration of the utility mechanism is given in Appendix A. In this algorithm, the variable $tours$ is the set of tours (or routes) of all technicians for all periods. It is initialized by the $Initialize\_Tours(problem)$ function that returns for all of them an empty tour. $allMOs$ is the set of all planned MOs, and $varMO$ is the MO selected to be added to the solution. $allMOs$ is initialized by the $Initialize\_MOs(problem)$ function, which returns no MO on the time horizon. The utilities are computed by the $Compute\_Utilities(tours, allMOs)$ function, as explained in Section 3.2. Function $Select\_MO(utilities)$ returns an MO, selected as explained in Section 3.3. Function $Local\_Search\_3\_condition(v_3, tours, varMO)$ returns true if it is impossible to add any MO (i.e., there is not enough time available) or if adding $varMO$ will be the v-th consecutive addition to the solution that degrades the objective function (i.e., that increases the objective function). Functions $Add\_To\_Tour(tours, varMO)$ and $Add\_To\_Mos(allMOs, varMO)$ simply add the MO $varMO$ to the set of tours $tours$ and the set of MOs $allMOs$. $Solution(tours, allMOs)$ is the solution defined by $tours$ (the routes of the technicians) and $allMOs$ (the MOs performed). The solution is retained if its objective function returns a value that is lower than that of the best solution found so far. $LS1(tours, varMO)$ is a function that uses the first LS for the technician and the period corresponding to the last added MO $varMO$, and $LS1All(tours)$ is the first LS executed for each technician and each period.

3.2. Computing the utilities

First of all, the utility of an MO (k,i,j,t) is only computed if there is no MO planned for machine i in period t. The utilities are always computed considering the current solution (i.e., a set of feasible MOs planned over the entire time horizon, and the routes of the technicians).

The utility of an MO is intended to reflect the traveling cost, the impact on the state of the machine (for the current period and the subsequent periods), and the expected operation cost. Let us denote as U_{kijt}, $k \in \mathcal{K}$, $i \in \mathcal{M}$, $j \in \mathcal{N}$, $t \in \mathcal{T} \mid k < c_j$ the utility of an MO executed on machine i by technician j to bring the machine to state k at period t.

The traveling cost for an MO is approximated by the best insertion cost of the machine into the current technician route. Let R_{jt} represent the route of technician j at period t in the current solution; in other

$$\text{Minimize}\left(\sum_{\forall t\in\mathcal{T}}\sum_{\forall j\in\mathcal{N}}\sum_{\forall(i_1,i_2)\in(\mathcal{M}\cup\{\delta\})^2,i_1\neq i_2} y_{jt}^{i_1i_2}d_{i_1i_2}\text{Ctr}\right)+\left(\sum_{\forall t\in\mathcal{T}}\sum_{\forall i\in\mathcal{M}}\left(z_{it}+\sum_{\forall k\in\mathcal{K}} wa_{it}^{k}C_k\right)\right) \tag{1}$$

$$wb_{it}^{k_1} = \sum_{\forall k_2\in\mathcal{K},k_2\leq k_1} wa_{it-1}^{k_2}p_{k_2k_1} \qquad \forall k_1\in\mathcal{K},\ \forall t\in\mathcal{T},\ \forall i\in\mathcal{M} \tag{2}$$

$$wa_{it}^{k_1} \geq -\Big(\sum_{\forall j\in\mathcal{N}|k_1\leq c_j}\ \sum_{\forall k_2\in\mathcal{K}|k_2<k_1} x_{ijt}^{k_2}\Big)+wb_{it}^{k_1} \qquad \forall k_1\in\mathcal{K},\ \forall t\in\mathcal{T},\ \forall i\in\mathcal{M} \tag{3}$$

$$wa_{it}^{k_1} \geq -(1-x_{ijt}^{k_1})+\sum_{\forall k_2\in\mathcal{K}|k_1\leq k_2\leq c_j} wb_{it}^{k_2} \qquad \forall t\in\mathcal{T},\ \forall i\in\mathcal{M},j\in\mathcal{N},\ \forall k_1\in\mathcal{K}\mid k_1<c_j \tag{4}$$

$$\sum_{k\in\mathcal{K}} wa_{it}^{k} = 1 \qquad \forall t\in\mathcal{T},\ \forall i\in\mathcal{M} \tag{5}$$

$$z_{it} \geq -M(1-x_{ijt}^{k_1})+\sum_{\forall k_2\in\mathcal{K}|k_1\leq k_2\leq c_j} wb_{it}^{k_2}r_{k_2k_1} \qquad \forall t\in\mathcal{T},\ \forall i\in\mathcal{M},\forall j\in\mathcal{N},\ \forall k_1\in\mathcal{K}\mid k_1<c_j \tag{6}$$

$$\sum_{\forall j\in\mathcal{N}}\sum_{\forall k\in\mathcal{K}|k<c_j} x_{ijt}^{k} \leq 1 \qquad \forall t\in\mathcal{T},\ \forall i\in\mathcal{M} \tag{7}$$

$$\left(\sum_{\forall i\in\mathcal{M}}\sum_{\forall k\in\mathcal{K}|k<c_j} x_{ijt}^{k}o_{c_jk}\right)+\left(\sum_{\forall(i_1,i_2)\in(\mathcal{M}\cup\{\delta\})^2,i_1\neq i_2} y_{jt}^{i_1i_2}d_{i_1i_2}\right)\leq E \qquad \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{8}$$

$$\sum_{\forall i_2\in\mathcal{M}\cup\{\delta\}|i_1\neq i_2} y_{jt}^{i_1i_2} = \sum_{\forall k\in\mathcal{K}|k<c_j} x_{i_1jt}^{k} \qquad \forall i_1\in\mathcal{M},\ \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{9}$$

$$\sum_{\forall i_2\in\mathcal{M}\cup\{\delta\}|i_1\neq i_2} y_{jt}^{i_2i_1} = \sum_{\forall k\in\mathcal{K}|k<c_j} x_{i_1jt}^{k} \qquad \forall i_1\in\mathcal{M},\ \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{10}$$

$$\sum_{\forall i\in\mathcal{M}\cup\{\delta\}} y_{jt}^{i\delta} = 1 \qquad \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{11}$$

$$\sum_{\forall i\in\mathcal{M}\cup\{\delta\}} y_{jt}^{\delta i} = 1 \qquad \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{12}$$

$$u_{i_1jt}-u_{i_2jt}+(|\mathcal{M}|+1)y_{jt}^{i_1i_2} \leq |\mathcal{M}| \qquad \forall j\in\mathcal{N},\ \forall t\in\mathcal{T},\forall(i_1,i_2)\in(\mathcal{M})^2,i_1\neq i_2 \tag{13}$$

$$x_{ijt}^{k} \in \{0,1\} \qquad \forall j\in\mathcal{N},\ \forall k\in\mathcal{K}\mid k<c_j,\forall i\in\mathcal{M},\ \forall t\in\mathcal{T} \tag{14}$$

$$y_{jt}^{i_2i_1} \in \{0,1\} \quad \forall(i_1,i_2)\in(\mathcal{M}\cup\{\delta\})^2\mid i_1\neq i_2,\ \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{15}$$

$$y_{jt}^{\delta\delta} \in \{0,1\} \qquad \forall j\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{16}$$

$$z_{it} \in \mathbb{R}^{+} \qquad \forall i\in\mathcal{M},\ \forall t\in\mathcal{T} \tag{17}$$

$$wa_{it}^{k} \in [0,1] \qquad \forall k\in\mathcal{K},\ \forall i\in\mathcal{M},\ \forall t\in\mathcal{T} \tag{18}$$

$$wb_{it}^{k} \in [0,1] \qquad \forall k\in\mathcal{K},\ \forall i\in\mathcal{M},\ \forall t\in\mathcal{T} \tag{19}$$

$$u_{ijt} \in [1,|\mathcal{M}|] \qquad \forall j\in\mathcal{N},\ \forall i\in\mathcal{M},\ \forall t\in\mathcal{T} \tag{20}$$

Model 1. Mathematical formulation of the problem.

words, R_{jt} is a set of pairs $(a,b)\in(\mathcal{M}\cup\{\delta\})^2,\ a\neq b$ such that, in the current solution, j has an MO planned on a immediately followed by an MO planned on b, and j is expected to travel from a to b between the two MOs. The impact on the traveling costs of the insertion of an MO (k,i,j,t) is therefore computed by using

$$U1_{kijt} = \min_{(a,b)\in R_{jt}} (d_{ai}+d_{ib}-d_{ab})\text{Ctr}. \tag{21}$$

Estimating the impact of an MO on the state of the machine requires computing the probability of transitioning to each degradation state i of the machine for each period $t'\in\mathcal{T}\mid t'\geq t$ if the MO is added. The cost saved corresponds to the diminishing penalty costs and the diminishing expected operational costs of the MOs already planned.

The computation of $U2_{kijt}$ is explained in detail in Algorithm 2. The function names are self-explanatory, describing their purposes: penaltyCost(S,i,t) calculates the expected penalty cost of machine i at period t in solution S, maintenanceCost(S,i,j,t) calculates the expected maintenance cost on machine i by technician j at period t in solution S and addMO(S,k,i,j,t) calculates the solution obtained by adding the MO (k,i,j,t) to the solution S.

Thereafter, the savings generated by adding MO (k,i,j,t) into the current solution are estimated by

$$U_{kijt} = U2_{kijt}-U1_{kijt}. \tag{22}$$

In other words, the savings correspond to the changes in the expected penalty costs and the expected operations costs minus the travel costs. If this value is greater than zero, then adding the corresponding MO to the current solution, with the best insertion of the machine in the route, is guaranteed to generate savings and thereby contribute to the minimization of the objective. The utilities are updated every time the current solution changes (i.e., every time an MO is selected and added to a route).

Note that, when technician j has insufficient time available in period t, it is possible that a given MO cannot be added to the solution. If we consider a best-insertion in the route of the technician, the

Algorithm 1: Metaheuristic

```
tours ← Initialize_Tours(problem)
allMOs ← Initialize_MOs(problem)
bestSolution ← Solution(tours, allMOs)
utilities ← Compute_Utilities(tours, allMOs)
varMO ← Select_MO(utilities)
if Local_Search_3_condition(ν3, tours, varMO) then
    (tours, allMOs) ← LS3(tours, allMOs)
    varMO ← Select_MO(utilities)
while Is_Feasible(tours, varMO) do
    tours ← Add_To_Tour(tours, varMO)
    allMOs ← Add_To_MOs(allMOs, varMO)
    utilities ← Compute_Utilities(tours, allMOs)
    varMO ← Select_MO(utilities)
    if Local_Search_2_condition(ν2, tours, allMOs) then
        tours ← LS2(varMO)
    if Local_Search_1_condition(ν1, tours, allMOs) then
        tours ← LS1(tours, varMO)
    if Solution(tours, allMOs) < bestSolution then
        bestSolution ← Solution(tours, allMOs)
if ν1 > 0 then
    for j ∈ 𝒩, t ∈ 𝒯 do
        tours ← LS1All(tours)
```

Algorithm 2: Computation of $U2_{kijt}$ for given (k,i,j,t) and a current solution S

```
U2_kijt ← 0
currentPenaltyCost ← 0
currentMaintenaceCost ← 0
for t' ∈ 𝒯 do
    currentPenaltyCost ← currentPenaltyCost + penaltyCost(S, i, t)
    currentMaintenanceCost ←
      currentMaintenanceCost + maintenanceCost(S, i, j, t)
S' ← addMO(S, k, i, j, t)
newPenaltyCost ← 0
newMaintenaceCost ← 0
for t' ∈ 𝒯 do
    newPenaltyCost ← newPenaltyCost + penaltyCost(S', i, t)
    newMaintenanceCost ←
      newMaintenanceCost + maintenanceCost(S', i, j, t)
U2_kijt ← (currentMaintenanceCost + currentPenaltyCost) −
 (newMaintenanceCost + newPenaltyCost)
```

infeasibility can be detected by

$$U3_{kijt} = \min_{(a,b)\in R_{jt}} (d_{ai} + d_{ib} - d_{ab}) + o_{c_j k}. \tag{23}$$

If this value is greater than the remaining available time in period t for technician j in the current solution, the MO cannot be added with the best-insertion method. In this case, we forbid the method to select such an MO.

3.3. Selection of a maintenance operation

This subsection discusses how the proposed method iteratively selects the MO to be added to the current solution. First of all, as mentioned earlier, we forbid the addition of an unfeasible MO. The MO (k,i,j,t), $k \in \mathcal{K}$, $i \in \mathcal{M}$, $j \in \mathcal{N}$, $t \in \mathcal{T} \mid k < c_j$ selected is randomly picked between all the feasible MOs with the greatest utility. Two or more MOs may have the same utility, which is often the case when several machines are on a single production site and start their time horizon in the same initial state. In such a case, our heuristic randomly selects an MO from those with the maximum utility.

Once selected, the MO is inserted into the route of technician j at period t and at the position obtained with the best-insertion method.

3.4. Local search operators

3.4.1. Local search operator 1: Traveling salesman problem solved heuristically

The first local search (LS) operator (LS1) aims to find shorter routes for the technicians in the different periods, without changing the machines visited or their planned MOs. Given that finding the shortest route for each technician in each period corresponds to solving a traveling salesman problem (TSP) (Flood, 1956), we based this LS operator on the Concorde TSP solver (Applegate et al., 2006), which first uses heuristics (as a Lin–Kernighan heuristic (Lin and Kernighan, 1973)) to compute bounds on the objective function and then uses a branch and bound algorithm. Our method stops before the branch and bound algorithm and returns the best solution found by the heuristics.

For technician $j \in \mathcal{N}$ and period $t \in \mathcal{T}$, this LS operator is called every ν_1 MO additions to the route of j at t. For instance, if $\nu_1 = 10$, the LS operator is called every time 10 MOs have been added to the route of technician $j \in \mathcal{N}$ at period $t \in \mathcal{T}$. The method calls this operator on every route making up the final solution at the end of its execution.

3.4.2. Local search operator 2: Swap and transfer

In this second LS operator (LS2), we use the two methods Swap and Transfer, both of which aim to reassign MOs to different technicians [e.g., for MO (k,i,j,t), $k \in \mathcal{K}$, $i \in \mathcal{M}$, $j \in \mathcal{N}$, $t \in \mathcal{T} \mid k < c_j$ in the solution, j changes to $j' \in \mathcal{N}$, $j \neq j'$] either by swapping two MOs in the routes of two technicians or by transferring one MO from the route of one technician to the route of another technician. In these moves, we do not seek to change the probabilities of the different degradation states of the machines, so we restrict both components (Swap and Transfer) to only apply to pairs of technicians with the same skill levels.

The LS operator is called each time ν_2 MOs have been added to the current solution, where ν_2 is a parameter (e.g., $\nu_2 = 1$ means that the LS is called after each MO selection and insertion).

The LS operator falls in the variable neighborhood descent (Mladenović and Hansen, 1997) category with the Swap neighborhood scheme first applied in each period. Next, the Transfer scheme is also applied in each period. Both schemes are restricted to swaps or transfers involving the technician j given as a parameter and corresponding to the last MO added to the solution before the call of the LS. This is a first improvement method however, within these two approaches, neighborhoods that yield no successful outcomes (i.e., technicians with no beneficial swap identified) are not revisited at a later time, even if the current solution undergoes modifications.

3.4.2.1. Swap. For the first neighborhood, Swap takes as parameters technician $j_1 \in \mathcal{N}$ and period $t \in \mathcal{T}$. As the name suggests, this neighborhood swaps two MOs from the routes of different technicians. As mentioned before, it impacts the traveling costs but does not change the probabilities of the degradation states of the machines.

Our implementation works as follows: For each technician $j_2 \in \mathcal{N}$ such that $c_{j_1} = c_{j_2}$, we test all possible swaps. $\forall (a_1,b_1,c_1) \in (\mathcal{M} \cap \{\delta\})^3$ with $(a_1,b_1),(b_1,c_1) \in R_{j_1 t}$ ($R_{j_1 t}$ being the route of j_1 at t, see Section 3.2) and $\forall (a_2,b_2,c_2) \in (\mathcal{M} \cap \{\delta\})^3$ with $(a_2,b_2),(b_2,c_2) \in R_{j_2 t}$. If $d_{a_1b_1} + d_{b_1c_1} + d_{a_2b_2} + d_{b_2c_2} \geq d_{a_1b_2} + d_{b_2c_1} + d_{a_2b_1} + d_{b_1c_2}$, then we swap b_1 and b_2: (a_1,b_1) and (b_1,c_1) are removed from $R_{j_1 t}$ and replaced by (a_1,b_2) and (b_2,c_1). Therefore, the MO on b_1 will now be performed by j_2 and the MO on b_2 by j_1. Similarly, in j_2's route, (a_2,b_2) and (b_2,c_2) are replaced by (a_2,b_1) and (b_1,c_2). Note that this swap is done if and only if the swap leads to a route that satisfies the duration constraint [Constraints (8)].

3.4.2.2. Transfer. This neighborhood takes as parameters a technician $j_1 \in \mathcal{N}$ and a period $t \in \mathcal{T}$. Similar to the Swap neighborhood, Transfer impacts the traveling costs but does not change the probabilities of the degradation states of the machines. For each machine maintained, it transfers the corresponding MO to another technician if doing so produces a saving. The MO is removed from the route of technician j_1 (without changing the visiting order of the remaining MOs) and inserted into the best possible position in the route of a second technician $j_2 \in \mathcal{N}$.

The neighborhood works as follows: For each $j_2 \in \mathcal{N}$ such that $c_{j_1} = c_{j_2}$ and $j_1 \neq j_2$, we test all possible transfers; namely, $\forall (a_1, b_1, c_1) \in (\mathcal{M} \cap \{\delta\})^3$ with (a_1, b_1), $(b_1, c_1) \in R_{j_1 t}$ and $\forall (a_2, c_2) \in (\mathcal{M} \cap \{\delta\})^2$ with $(a_2, c_2) \in R_{j_2 t}$. If $d_{a_1 b_1} + d_{b_1 c_1} + d_{a_2 c_2} \geq d_{a_1 c_1} + d_{a_2 b_1} + d_{b_1 c_2}$ then the MO on b_1 is transferred to technician j_2 [i.e., (a_1, b_1) and (b_1, c_1) are removed from $R_{j_1 t}$ and replaced by (a_1, c_1), and (a_2, b_2) is removed from $R_{j_1 t}$ and replaced by (a_2, b_1) and (b_1, b_2)]. The neighborhood is explored following a first-improvement strategy.

3.4.3. Local search operator 3: Slot opener

The idea behind this LS operator (LS3) is to *compress* the duration of the routes making up the current solution to make room to accommodate more MOs. This goal is achieved by lowering the target level of one or more MOs included in the current solution. Note that compressing the route does not necessarily improve the solution but most often degrades it. The travel cost of the route does not change after the route is compressed but the two other costs do change. On the one hand, the operating costs decrease because at least one MO will have a higher target level. On the other hand, the expected penalty costs increase because a machine will have an overall higher degradation since the MO is reducing less the degradation. Since the penalty costs usually are significantly larger than the operating costs, the overall solution tends to degrade after a route compression. However, route compression is called only when an enhancing MO cannot be added to the solution and its changes will free sufficient time for a single enhancing MO.

For each period $t \in \mathcal{T}$, we try to modify the current solution to make room for one enhancing MO. This approach constitutes a first-improvement method for each period and stops at the first modification. The method considers all MOs (k, i, j, t) $\forall k \in \mathcal{K}$, $\forall i \in \mathcal{M}$, $\forall j \in \mathcal{N} \mid k < c_j$ that could not be included in the route R_{jt} (insufficient time), and that have positive utilities ($U_{kijt} > 0$).

Let (k, i, j, t), $k \in \mathcal{K}$, $i \in \mathcal{M}$, $j \in \mathcal{N} \mid k < c_j$ be such an MO, and let M_{jt} be the set of pairs $(i', k') \in \mathcal{M} \times \mathcal{K} \mid k' < c_j$ such that j has an MO planned on machine i' in the current solution at $t \in \mathcal{T}$ with target state k'. We try to modify the target state k' of each $(i', k') \in M_{jt}$ to insert MO (k, i, j, t).

Let Θ be the time needed to insert MO (k, i, j, t) in the route of technician j at period t. It is computed by using the formula

$$\Theta = E - \left(\sum_{\forall (i',k') \in M_{jt}} o_{c_j k'} + \sum_{(a,b) \in R_{jt}} d_{ab} \right) - U3_{kijt}. \tag{24}$$

Let $\Psi_{k'' i' t}$ $\forall (i', k') \in M_{jt}$ $\forall k'' \in \mathcal{K} \mid c_j > k'' \geq k'$ be the time saved if MO (k', i', j, t) is dropped and replaced by MO (k'', i', j, t):

$$\Psi_{k'' i' t} = o_{c_j k'} - o_{c_j k''}. \tag{25}$$

Finally, let $\mu_{k'' i' t}$ $\forall (i', k') \in M_{jt}$, $\forall k'' \in \mathcal{K} \mid c_j > k'' \geq k'$ be the increase in the cost if (k', i', j, t) is dropped and replaced by MO (k'', i', j, t). It is computed similarly to $U2$ (see Section 3.2).

The sub-problem here is then to find new target states such that this minimizes the total cost increase and frees enough time to accommodate the MO (k, i, j, t). The model of this sub-problem is presented in Model 2

$\forall (i', k') \in M_{jt}$, $\forall k'' \in \mathcal{K} \mid c_j > k'' \geq k'$, $v_{k'' i' t}$ is a decision variable equal to 1 if, in the solution returned, the target state for the maintenance on machine i' is k'' or zero otherwise. Constraint (27) ensures that a feasible solution frees sufficient time to accommodate the MO. Constraints (28) impose that a new target state is found for each machine in the route (note that this target state can equal the previous one). If no feasible solution is found, it is not possible to add MO (k, i, j, t) to the route by only changing the target states. If the value of the objective function of the returned solution is less than the utility of (k, i, j, t), changing the target states and inserting the MO improves the current solution.

Since the model above is solved several times during the execution of our heuristic, solving it exactly becomes a daunting task. Alternatively, we solve it approximatively by using a heuristic method. The idea is simply to change the target states of the MO one by one until sufficient space is freed to accommodate the MO being inserted. $\forall (i', k') \in M_{jt}$, we compute the maximal ratio $U4_{i'k'} = \max_{\forall k'' \in \mathcal{K} \mid c_j \leq k' > k'} \frac{\Psi_{k'' i' t}}{\mu_{k'' i' t}}$ between freed time and cost increase for the different possible changes in the target states. Afterward, the new target states of the machines are changed one by one, in decreasing order of $U4_{i'k'}$, $\forall (i', k') \in M_{jt}$. The new target state for $(i', k') \in M_{jt}$, if changed, is $\operatorname{argmax}_{\forall k'' \in \mathcal{K}, c_j > k'' > k'} \frac{\Psi_{k'' i' t}}{\mu_{k'' i' t}}$. We iteratively change the MO one by one in decreasing order of utility until sufficient time is freed for the MO (k, i, j, t). Once again, if the cost increase exceeds the gains of the MO tested, no change is made and this MO is discarded.

Algorithm 3 summarizes the operation of this LS procedure. $NoMaintenance(currentSolution, i, t)$ is a function returning true if $currentSolution$ has no MO planned on i at period t, otherwise it returns false. The function $DoesNotFit$ with parameters $(currentSolution, i, j, t)$ returns true if the best-insertion method cannot add an MO (k, i, j, t) $\forall k \in \mathcal{K} \mid k < c_j$ in $currentSolution$. The function $SolveModel2$ with the parameters $(currentSolution, k, i, j, t)$ solves Model 2 for the MO (k, i, j, t) using the heuristic presented above. It returns the solution obtained (new target states) and the objective value (total cost increase).

Algorithm 3: Slot opener

```
notFound ← true
for t ∈ T and notFound do
    for j ∈ N and notFound do
        for i ∈ M and notFound do
            if NoMaintenance(currentSolution, i, t) and
             DoesNotFit(currentSolution, i, j, t) then
                bestSolution ← currentSolution
                for k ∈ K|k ≤ c_j do
                    (solution, objValue) ←
                     SolveModel2(currentSolution, k, i, j, t)
                    if Feasible(solution) and objValue < U_kijt then
                        bestSolution ← solution
                        notFound ← false
                currentSolution ← bestSolution
```

This LS operator is called in two steps of the heuristic. The first is whenever it is impossible to insert an additional MO into a route making up the current solution. The second is when the heuristic has no utility greater than zero in the past ν_3 consecutive MO additions to the solution (ν_3 is a parameter). This means that the heuristic method does not improve the current solution in ν_3 consecutive steps (MO additions).

4. Experimentation

We designed and undertook experiments to test the quality and scalability of the proposed heuristic. We analyze the solutions returned to draw insights from the modeling choices made. Several research questions guide our experimentation:

- **RQ1:** How should the method be parametrized to obtain a trade-off between running time and solution quality?
- **RQ2:** How good are the solutions produced?
- **RQ3:** How does the method scale with respect to problem size (more machines, technicians, or periods)?

$$\text{Minimize} \sum_{(i',k')\in M_{jt}} \sum_{k''\in\mathcal{K}|c_j>k''\geq k'} \mu_{k''i't} v_{k''i't} \tag{26}$$

$$\sum_{(i',k')\in M_{jt}} \sum_{k''\in\mathcal{K}|c_j>k''\geq k'} \Psi_{k''i't} v_{k''i't} \geq \Theta \tag{27}$$

$$\sum_{k''\in\mathcal{K}|c_j>k''\geq k'} v_{k''i't} = 1 \qquad \forall (i',k') \in M_{jt} \tag{28}$$

$$v_{k''i't} \in \{0,1\} \qquad \forall (i',k') \in M_{jt},\ \forall k'' \in \mathcal{K} \mid c_j > k'' \geq k' \tag{29}$$

Model 2. New targets with minimal losses.

- **RQ4:** How is the distribution of expected costs (between expected penalty cost, expected MO cost, and traveling cost) impacted by machines grouped into production sites?
- **RQ5:** How valuable is a look-ahead approach (i.e., that considers many time periods instead of just the current period)?
- **RQ6:** What are the managerial implications, i.e., how valuable are technicians and their skill levels?
- **RQ7:** How valuable is the possibility to perform different types of MO?

To answer each question, we run the proposed method on a dedicated set of instances generated by using the generator presented in the next section.

All the experiments reported in this section were run on the Béluga supercomputer from Calcul Québec.

4.1. Instance generation

This section describes the method used to generate the instances. The following are the main input parameters of the methods:

- T is the number of periods;
- m is the number of machines;
- L and l are the length and width of the two-dimensional (2D) rectangle area considered, respectively, and the coordinates of each machine $i \in M$ are randomly selected from a uniform distribution inside this zone;
- n is the number of technicians;
- $C^k, \forall k \in \mathcal{K}$ are the penalty costs at the end of a period;
- Ctr is the cost of traveling per time/distance unit;
- E is the available time per technician per period.

In all our instances we use six degradation states ($\{1,2,3,4,5,6\}$, State 6 being the worst state and State 1 the good-as-new state). The skill of a technician is represented by the highest degradation state c_j on which he or she can intervene. This state is randomly picked from the set $\{2,3,4,5,6\}$ (no need to consider a technician who can only intervene on State 1), except for one technician with a skill level fixed at $|K|$ (to be sure that at least one technician can intervene on all the degradation states).

The transition matrix $\mathcal{P}$ is produced as follows. The idea is to respect two principles: $p_{k_1k_3} < p_{k_2k_3}$ and $p_{k_1k_2} > p_{k_1k_3}\ \forall k_1 < k_2 < k_3$, $(k_1,k_2,k_3) \in \mathcal{K}^3$. This is done by using Algorithm 4.

Initialization() serves to initialize the values of $p_{0k}\ \forall k \in \mathcal{K}$ to 0 and p_{00} to 1, and the function $rand(x)$ returns a random value in the range $[0,x)$. This algorithm produces random but realistic probabilities. It generates higher probabilities for a machine to remain in the same degradation state or to slightly degrade (e.g., from State 1 to State 2) than for a machine to undergo greater degradation (e.g., from State 1 to State 6). The initial state of each machine is considered to be the good-as-new state (State 1), which means that for the first period, the probability of each state $k \in \mathcal{K}$ for each machine is p_{1k}.

Algorithm 4: Computation of the probabilities

```
P ← Initialization()
for k1 ∈ K do
    h ← p_{(k1−1)(k1−1)}
    for k2 ∈ K | k2 ≥ k1, k2 < |K| do
        r ← h/2 + rand(h/4)
        p_{k1 k2} ← p_{(k1−1)k2} + r
        h ← h − r
    p_{k1|K|} ← p_{(k1−1)|K|} + h
```

The costs of the MOs and their operating times are computed by using the same principles. The operating times $\forall k_1 > k_2 > k_3$, $(k_1,k_2,k_3) \in \mathcal{K}^3$ must satisfy the following two conditions: (i) $o_{k_1k_2} < o_{k_1k_3}$ (i.e., an more complex operation takes more time) and (ii) $o_{k_1k_2} + o_{k_2k_3} = o_{k_1k_3}$. The same principles are considered for the costs. The generator uses randomness and takes two parameters as input, obase and rbase, controlling the possible values generated and thus their impact on the instance. For example, a higher value of rbase implies that the generator will produce higher maintenance costs overall. This approach is described in Algorithm 5.

Algorithm 5: Computation of costs and operation times

```
h_o ← 0  h_r ← 0
for k ∈ K do
    o_{k1} ← h_o + obase × rand(k)
    r_{k1} ← h_r + rbase × rand(k)
    h_o ← o_{k1}
    h_r ← r_{k1}
for k1 ∈ {2,3,4,5,6} do
    for k2 ∈ K, k2 > k1 do
        o_{k2 k1} ← o_{k2 1} − o_{k1 1}
        r_{k2 k1} ← r_{k2 1} − r_{k1 1}
```

In the first loop, the algorithm computes the costs and operation times to return a machine to the good-as-new state from all other degradation states. The computation should reflect the fact that increasing the degradation between two periods increases both costs and operation time (i.e., we guarantee that $o_{k_11} \leq o_{k_21}\ \forall (k_1,k_2) \in \mathcal{K}^2,\ k_2 > k_1$). The second loop simply computes the remaining costs and operation times for each pair of degradation states different from State 1. No randomness appears in this second loop because it has to respect the two principles (i) and (ii) mentioned above.

The cost of traveling, the penalty costs, and the maximum time available per period are all input parameters. Although to remain general, the model proposed in our work considers a penalty cost C_k for each degradation state $k \in \mathcal{K}$, in the present experimentation we consider a non-zero cost only for the worst state. With such a cost, we chose to model the most practical examples where the penalty cost of a functioning state (inducted by the performance level cost and minimal repair expected cost) is negligible compared to the cost of a failure that

Table 3
Parameters to generate an instance.

Parameter	Notation	Default value
Number of technicians	n	10
Number of machines	m	150
Number of periods	T	20
Size of area	$l \times L$	300 × 300
Penalty cost of the failure state during a period	$C_{\|K\|}$	10 000
Penalty cost of a state $k \in \mathcal{K} \mid k < \|K\|$	C_k	0
Cost of traveling	Ctr	1
Available time per period per technician	E	1500
Parameter 1 used to compute operating times	obase	20
Parameter 2 used to compute operating costs	rbase	20

Table 4
Results without local search.

Average value of objective function	1 791 102
Average CPU time (s)	45.25

cannot be addressed by minimal repair, i.e., no possibility to repair it by a local technician. Such failures are modeled with state $|\mathcal{K}|$, now referred to as the failure state. With $C_k = 0, \forall k \in \mathcal{K} - \{|\mathcal{K}|\}$, we can model such problems with a high incentive to avoid such failures. Note that results for generic instances are discussed in Section 4.3 and it demonstrates that results for this specific case can be generalized. Table 3 summarizes the parameters used to generate an instance and their default values. In other words, in the experiments discussed below, the parameters used to generate the instances are fixed to their default values unless otherwise specified. We consider these values to produce both realistic and challenging instances.

4.2. RQ1: How should the method be parametrized?

A higher LS call frequency is likely to improve the solution returned, although it is more than likely to add extra running time. This experiment analyzes how to determine the proper default values for the frequency parameters to achieve a good trade-off between running time and quality.

A set of fifty instances was generated with the default parameters. Each instance was solved thirteen times by our method (once for each parameter tested), each time with different LS-frequency parameters. Recall that three types of LS operators exist and their frequencies are controlled by the parameters $\{\nu_1, \nu_2, \nu_3\}$, which correspond to the numbers of MOs added (in a route for LS1, in the solution for LS2, and the solution without improvement for LS3) between two calls of an LS operator. For each of these parameters, we tested the values $\{1, 10, 20, 50\}$. However, when testing one parameter, the other LS operators are not executed (except for the third LS, see Section 3.4.3, which is called even when MOs can no longer be added to the solution). For instance, when testing the impact of ν_3, the TSP (LS1) and Swap-Transfer LS (LS2) operators are never executed. Tables 5–7 present the results for ν_1, ν_2, and ν_3, respectively. Each table gives the average value of the objective function of the returned solutions and the average CPU times in seconds. Table 4 gives the average value of the objective function and the average running time when all LS operators are inactive (note, however, that the third LS operator is called even when the solution being built can no longer accept an MO). This last version is labeled as the default version in the remainder of this section. We also report in the row labeled "Average gap" the average gap in the objective function compared with the results of the default version. In other words, the average value of $1 - \frac{f}{f_d}$ with f_d being the value of the objective function found by the default version and f being the one found with the corresponding LS operator.

As expected, the method produces better solutions when at least one active LS procedure exists. When the LS procedures are called often (lower values for ν_1, ν_2, or ν_3), the average quality of the solution increases. The average running time has the opposite behavior: it always increases with more frequent calls of the LS procedure. The only exception to these patterns is the second LS procedure (Swap + Transfer) between $\nu_2 = 20$ and $\nu_2 = 50$.

In almost the entire experiment, the running time does not increase above twice the running time of the default version, except for the third LS operator (slot opener, see Section 3.4.3) and $\nu_3 = 1$, where the average running time is almost tenfold greater than the average default running time. Thus, we recommend running the heuristic with intermediate values of ν_3 (e.g., 10) if CPU time is a concern. The results also suggest that the call frequency of the Swap + Transfer LS operator has the least impact on the results.

To conclude with these first experiments, each LS procedure helps the method deliver better solutions in exchange for a moderate increase in running time. Based on the results of this study, the values recommended for the LS frequency parameters are $\nu_1 = 1$, $\nu_2 = 1$, and $\nu_3 = 10$. Consequently, these are considered as the fixed default values in the remainder of the experiments.

We now study the impact of simultaneously activating the three LS procedures. To this end, we ran an additional experiment re-using the same set of instances. In this experiment, we ran the proposed method four additional times, with different combinations of our LS operators. As mentioned earlier, the parameters ν_1, ν_2, and ν_3 now have default values and do not change. The first run uses only TSP (LS1) and Swap + Transfer (LS2). The second run uses TSP (LS1) and Slot opener (LS3). The third run uses Swap + Transfer (LS2) and Slot opener (LS3). Finally, the last run uses all LS procedures. Table 8 presents the results.

These results show that running the heuristic with virtually any combination of LS operators leads to better solutions than running it with the best single-operator configuration, except for the LS3 which outperformed LS1 combined with LS2 by a small margin. The results also show that the CPU times increase when using combinations of LS procedures rather than only a single LS procedure. Based on these observations, we set our heuristic default configuration to that of exploiting the three LS operators with a frequency of $\nu_1 = 1$, $\nu_2 = 1$, and $\nu_3 = 10$.

4.3. RQ2: How good are the solutions produced?

In this experiment, we focus on the quality of the solutions. In particular, we propose to compare the best solutions returned by the proposed method with those returned by CPLEX running on the model introduced in Section 2. The latter are either optimal or the best integer solution found within a given time limit. We also include in the comparison the best lower bounds delivered by the solver for every instance. To find the best possible benchmarks for our experiment, we ran CPLEX in two modes, stand-alone and warm-start, with the best solution found by our heuristic. A warm start usually (but not always) accelerates the solution process, allowing the solver to find better solutions within the same time limits. The two solutions (stand-alone and warm-start) are reported for CPLEX in our tables. Note that the warm-start solution is, by definition, at least as good as the solution reported by our heuristic.

Table 5
Results with different values of ν_1.

ν_1	1	10	20	50
Average value of objective function	1 766 696	1 776 939	1 783 119	1 786 969
Average CPU time (s)	86.64	61.04	59.40	58.11
Average gap	1.36%	0.79%	0.45%	0.23%

Table 6
Results with different values of ν_2.

ν_2	1	10	20	50
Average value of objective function	1 766 343	1 780 237	1 784 969	1 787 722
Average CPU time (s)	48.73	47.16	46.64	46.71
Average gap	1.38%	0.61%	0.34%	0.19%

Table 7
Results with different values of ν_3.

ν_3	1	10	20	50
Average value of objective function	1 741 687	1 749 107	1 754 348	1 764 942
Average CPU time (s)	381.80	83.37	67.02	56.99
Average gap	2.76%	2.34%	2.05%	1.46%

Table 8
Results with different combinations of LS procedures.

Local searches	LS1 & LS2	LS1 & LS3	LS2 & LS3	LS1 & LS2 & LS3
Average value of objective function	1 742 380	1 719 633	1 722 626	1 695 080
Average CPU time (s)	86.29	103.64	85.18	116.97
Average gap	2.72%	3.99%	3.82%	5.36%

Table 9
Instance parameters tested for the RQ2 experiment.

Parameter	Set 1	Set 2	Set 3
Number of technicians	1	2	3
Number of machines	50	50	50
Number of periods	3	2	1
Degradation state	6	6	6

Table 10
Results for different sets of RQ2 experiment.

	Gap Set 1	Gap Set 2	Gap Set 3
CPLEX stand-alone	−10.88%	−4.15%	1.21%
CPLEX warm-start	5.22%	3.31%	2.96%
CPLEX best lower bound	27.20%	23.66%	15.46%

In preliminary experiments, we ran CPLEX with medium-size instances (150 machines, 10 technicians, and 20 periods). However, CPLEX had difficulty finding a first feasible solution and then a lower objective value (numerous iterations without improvement). Additionally, it quickly ran out of allowed memory and the process was killed without producing a good solution (i.e., the objective value of the best feasible solution was an order of magnitude greater than the objective value of our heuristic). We used our preliminary experiment to find the size of the instances for this experiment such that CPLEX does not run out of allowed memory within the budgeted time and produces good solutions. We used three sets of instances generated by using the parameters reported in Table 9, with each set containing fifty instances. For each instance, we ran CPLEX in the two modes (i.e., stand-alone and warm-start) with a time limit of three hours.

The results are presented in Table 10. The values represent the average gap between the objective function o_1 returned by the proposed method and (i) the best objective value o_2 returned by CPLEX in the stand-alone mode, (ii) the best objective value o_3 returned by CPLEX in the warm-start mode, and (iii) the best lower bound o_4 found by CPLEX in the two modes (stand-alone and warm-start). In other words, the first value in a given column is the average value of $1-\frac{o_2}{o_1}$, the second is the average value of $1-\frac{o_3}{o_1}$, and the third is the average value of $1-\frac{o_4}{o_1}$. Note that, for all these instances, the average running time of the proposed method is around 0.4 s.

In these results, with a budgeted time of three hours and two different runs, the improvement on the objective value returned by CPLEX in the warm-start mode is around 3.83% of our solution. This is a small improvement, especially considering the running-time difference and this approach is only possible for small instances because CPLEX does not scale well for our problem. Conversely, the lower bound returned by CPLEX at the end of the budgeted time is, on average, approximately 22.1% better than our solution. Note that CPLEX returns the confirmed optimal solutions in only two instances, with the proposed method being 1.38% and 2.90% from the optimal solutions.

These results show that CPLEX in stand-alone mode produces, on average, an objective value 4.61% worse than our solution. With the first set, which has more periods, the difference is even greater, with the gap being around 10%. Surprisingly, CPLEX in stand-alone mode works better on average with the third set of instances because this set has only one period and is similar to the vehicle routing problem. Given only one period, there is no consideration for the degradation state (except for the penalty cost of the failure state, $|\mathcal{K}|$, for period zero) and thus no selection between different target states. The pre-processing of CPLEX removed many constraints and decision variables.

In these experiments, the solutions returned by CPLEX stand-alone mode are worse than those returned by the proposed method. In addition, an average gap of 8.44% appears between CPLEX in stand-alone mode and CPLEX in warm-start mode. The gap is an average of 16.1%, in particular on the first set with more periods, which shows that CPLEX warm-started by our solution produces much better solutions.

As mentioned in Section 4.1, the instances generated are special cases of our problem as we do consider only penalty costs for the last degradation state. Additional experiments were run to check if the conclusion drawn from this experiment could be generalized, with different penalty costs considered for each state. We generated fifty instances for the different sets of parameters from Table 9, with the penalty costs: $c_1 = 0, c_2 = 250, c_3 = 500, c_4 = 1000, c_=2000, c_6 = 4000$. Note that CPLEX ran out of memory in a few instances of parameter set 1 and parameter set 2, and we decided to discard these instances

Table 11
Results for different sets of RQ2 experiment, with penalty costs for each state.

	Gap Set 1	Gap Set 2
CPLEX stand-alone	−1.24%	−1.05%
CPLEX warm-start	0.10%	0.10%
CPLEX best lower bound	11.15%	17.81%

and replace them with new instances until each one of these sets of parameters has fifty instances with results from CPLEX. Unfortunately, set 3 has too many out-of-memory instances (44 out of 50 instances) and we decided not to report their results. Similarly to Table 10, we obtain Table 11.

In these results, we observe similar results than in Table 10, that is CPLEX in stand-alone does not manage to match our results and CPLEX warm-start does only improve the results by a small margin. Although the gaps are narrowing, demonstrating that the penalty costs have an impact on the problem and its complexity, it is still allowing us to draw the same conclusion, i.e., our method is performing well and provides efficient solutions. Note that the computation times are on the same scale as the previous experiment, with less than a second for our method and 3 h of computation time for CPLEX.

4.4. RQ3: How does the method scale with increasing parameters?

To answer to this question, we ran our heuristic on 18 sets of instances generated with a wide range of parameters and analyzed the computational performance. The experiment was divided into three sub-experiments, each one testing a different parameter: number of technicians, number of machines, and number of periods.

4.4.1. Number of technicians

In this sub-experiment, we report the running time of the proposed method on different instances generated with the default parameters but with a varying number of technicians. In particular, we tested our method on instances with $\{5, 10, 15, 20, 30, 50\}$ technicians. For each value, a set of fifty instances was generated. The average running times, in seconds, are reported in Table 12.

As expected, these results show that the number of technicians correlates directly and positively with the running time of the method. However, the increase seems to scale well and even increases less upon a higher number of technicians. The running time per technician always decreases as the number of technicians grows. These results were expected because the number of technicians linearly affects the number of potential MOs and thus affects the computational effort made during the utility updates. Swap + Transfer is the only component of the method that does not scale linearly with the number of technicians. However, as discussed in Section 4.2, this component does not significantly affect the overall execution time of the method. One final observation is that, with more technicians, the heuristic executes more iterations. Because more total working time is available, more MOs can be inserted into the solution. Nonetheless, the data collected in this experiment suggest that the increase in the number of iterations does not necessarily translate into exorbitant CPU times.

4.4.2. Number of machines

In this sub-experiment, we report the running time of the proposed method on different instances generated with the default parameters but with a varying number of machines. In particular, we tested our method on instances with $\{50, 100, 150, 200, 300, 400\}$ machines. For each value, a set of fifty instances was generated. The average running times, in seconds, are reported in Table 13.

The results suggest that the running time of the proposed method depends strongly on the number of machines, and a greater number of machines in an instance translates into longer routes (in terms of the number of planned MOs). Because the geographical area remains unchanged, machines tend to be closer together in instances with a large number of machines, thereby allowing technicians to accommodate more visits per period. The increase in route lengths implies more iterations of our algorithm, more complex utilities to compute (i.e., more complex best-insertions), and a concomitant increase in the running time of the LS procedures. Additionally, more utilities are computed in each iteration. As a result of all this overhead, the CPU time increases sharply with the number of machines. However, the results show that, even in large instances with 400 machines, the CPU times remain reasonable for industrial applications (e.g., 926.53 s, or about 15 min).

4.4.3. Number of periods

In this last sub-experiment, we report the average running time of our method on instances generated with all the default parameters except for the number of periods, which varies between the values in the set $\{5, 10, 15, 20, 30, 50\}$. For each value, we generated a set of fifty instances. The average running times, in seconds, are reported in Table 14. As with the two previous experiments, we report here the execution time per period (i.e., the CPU time divided by the number of periods).

Not surprisingly, increasing the number of periods increases the running time of the method, with the increment being nonlinear in the number of periods. This result is expected because more utilities (with higher computational complexity) are computed in each iteration of our algorithm, and the algorithm goes through more iterations. However, the worst average running time remains acceptable and the method scales well with the number of periods.

To conclude, the number of periods, technicians, and machines directly affect the running time of the proposed method. The number of technicians has a positive, linear relationship with CPU time, making it the parameter with the least significant impact. Conversely, the other two parameters (the number of periods and the number of machines) have a positive, nonlinear relationship with the execution time, making their impact more pronounced. However, the heuristic remains quite fast and scales very well with the size of the problem.

4.5. RQ4: What is the impact of machines grouped into production sites?

In this experiment, we study how machine clustering affects the quality and structure of the solutions. Toward this goal, we assume that the set $\mathcal{M}$ is divided into disjoint subsets of machines such that each subset is a production site. All pairs of machines where both machines belong to the same site have a negligible travel time that can be ignored. To generate the instances for this experiment, we slightly modify our generator, with the new version taking as a parameter the number of production sites. The generator then randomly generates the coordinates of each of the sites in the 2D area ($l \times L$) and assigns each machine to one of the production sites, ensuring that every site ends up with the same number of machines. For example, for an instance with 150 machines and 10 sites, each site would have 15 assigned machines. The rest of the generation process remains untouched. Note that the depot δ remains randomly located, thus, even when considering a single site, a travel cost must still be paid (for travel between the depot and the unique production site). In all instances generated, we keep the default parameters and thus always consider 150 machines but with different numbers of production sites. We generated instances with $\{1, 10, 30, 50, 150\}$ sites and, for each number of sites, generated a set of fifty instances. Note that instances with 150 sites (with 1 machine per site) correspond to instances where machines are not clustered. We report in Table 15 the average objective value, average traveling cost, expected operation cost, expected penalty cost, average CPU time (in seconds), and average number of MOs planned in a solution.

The results show that the geographical dispersion of the machines strongly affects the average objective value, the costs, and the shape of

Table 12
Results for different numbers of technicians for RQ3 experiment.

Number of technicians	5	10	15	20	30	50
CPU time (s)	61.42	116.80	148.65	193.08	279.44	402.27
Time per technician	12.2840	11.6800	9.910	9.6540	9.3147	8.0454

Table 13
Results for different numbers of machines for RQ3 experiment.

Number of machines	50	100	150	200	300	400
CPU time (s)	10.73	47.89	125.07	224.92	543.91	926.53
Time per machine	0.2146	0.4749	0.8338	1.1246	1.8130	2.3163

Table 14
Results for different numbers of periods for RQ3 experiment.

Number of periods	5	10	15	20	30	50
CPU time (s)	19.76	45.69	77.74	119.48	225.18	528.55
Time per period	3.9520	4.5690	5.1827	5.9700	7.5060	10.5710

Table 15
Results with machines grouped.

Number of sites	Objective value	Traveling cost	Operation cost	Penalty cost	CPU time (s)	No. of MOs
1	1 356 644	51 446	25 396	1 279 802	527.07	2839
10	1 453 386	71 755	25 344	1 356 287	381.10	2808
30	1 536 519	83 860	25 374	1 427 286	217.19	2754
50	1 570 872	89 586	25 383	1 455 903	129.00	2689
150	1 675 907	103 278	25 541	1 547 088	120.04	2567

the solutions. The traveling time increases when the machines are dispersed into different sites. A direct consequence of the increase in travel time is a reduction in the number of MOs that can be accommodated in the planning. However, a less intuitive observation concerns the operating costs: because fewer MOs are planned, one would expect this cost to go down, whereas the results show that it remains fairly stable. One plausible explanation is that the algorithm tries to compensate for the higher travel costs with lower expected penalty costs by selecting MOs with lower target levels. Therefore, although the solution has fewer MOs, each MO is (on average) more expensive, leading to a similar overall operating cost. Finally, the running time of the proposed method decreases substantially when the number of production sites increases. This behavior is explained by the number of MOs: because the solutions to instances with a larger number of sites typically include fewer MOs, the heuristic requires fewer iterations. However, even instances with only one site require little CPU time and fit perfectly to industrial applications.

4.6. RQ5: How valuable is a look-ahead approach

For this research question, we investigate the benefits of a look-ahead approach: how many savings are we making with an MO decision process that considers the many upcoming periods instead of just the current period? For example, a company is using each Monday morning an algorithm that plan the MOs performed for the remaining days of the week. If the algorithm is a myopic approach, it is only considering the current week during the decision process. Whereas, if the algorithm is a look-ahead process, it is considering all the upcoming weeks (e.g., the next four months) to plan the MOs. It is expected that a look-ahead approach is better than a myopic approach, but the objective of our research question is to quantify how much by proposing an alternative solving algorithm for our problem, more intuitive but based on a myopic behavior. In this experiment, we investigate a myopic approach called the Period-Per-Period Process (referred to as 4P in the rest of this section) which, as its name indicates, solves iteratively the problem for each period $t \in \mathcal{T}$ (from period 1 to period $|\mathcal{T}|$). When solving period t, it does not consider future periods ($\forall t' > t$) and the MOs computed for the previous periods ($\forall t'' < t$) are fixed. Each period is solved using CPLEX. However, it is clear that, since 4P is not considering future periods, only *minimal MOs* are decided (i.e., MOs with goal state $|\mathcal{K}| - 1$,) that put the machines at the limit of the failure state to avoid its prohibitive penalty costs while having limited maintenance costs. It is not efficient since it leads to a need to maintain very regularly the machines, almost at each period. We try to overcome this aspect by investigating an additional strategy for the 4P: only MOs aiming for the as-good-as-new state (State 1) are considered. At each period, an intervention on a degraded or failed machine consists of a complete replacement. We call this variant the 4P variant, as opposed to the 4P classic which was presented earlier. The 4P classic builds solutions with minimal MOs and the 4P variant with full replacements. In Table 16, we briefly report this preliminary experiment with a comparison between our method, 4P classic, and 4P variant. For this experiment, we generate a set of fifty instances with a limited size (50 machines, 6 degradation states, 3 technicians, and 10 periods) to avoid a long or impossible computation during the CPLEX calls for the 4P and 4P variants. Because of its myopic behavior, the initial states of the machines are an important factor for 4P as it tends to not plan any MO for the first periods of an initially well-maintained system (all machines start as good as new). Indeed, for the first periods, the traveling costs and maintenance costs are far greater than the possible penalty costs since it only considers the failure in the current period and not its impact on the next periods. Therefore, on the opposite of Section 4, we investigate $|\mathcal{K}| = 6$ sets of instances. Each set $k \in \mathcal{K}$ corresponds to all the machines being in State k at the beginning of the time horizon. A budget time is given for each 4P (classic or variant) to solve an instance, with 15 min allotted per period for the CPLEX solving. For each set of instances, we report the average value in percent of the solution from our method. i.e., $\frac{f_{4P}}{f_S}$ with f_{4P} the objective function value of the solution from 4P (either classic or variant) and f_S the objective function value of our solution

These results quantify the important savings made by a look-ahead approach over a myopic approach. In our instances, the myopic approaches (both 4Ps) are between 2.76 and 6.85 times worse than the look-ahead approach. It means that, in the worst case of our instances, a look-ahead method still saves $\frac{2}{3}$ of the costs. These results justify any investment made in look-ahead mechanics. Furthermore, this highlights

Table 16
Preliminary experiments result.

Initial State of all the machines	1	2	3	4	5	6
4P classic	685%	748%	737%	608%	450%	428%
4P variant	287%	356%	344%	361%	276%	301%

Table 17
Average objective value for different technicians with high level of skill ratio.

Percent of technicians with high level of skill	0	10	20	30	40	50
Average gap from best solution	5 997 101	3 542 556	2 050 993	1 240 227	807 251	538 395
Average gap from best solution (in %)	2373%	1317%	719%	424%	275%	186%
Percent of technicians with high level of skill	50	60	70	80	90	100
Average gap from best solution	538 395	358 175	234 997	144 438	68 447	0
Average gap from best solution (in %)	186%	126%	84%	54%	26%	0%

the benefits of considering multiple levels of repair, rather than only minimal repairs or full replacements as in the 4P methods.

4.7. RQ6: What are the managerial implications, i.e., how valuable are technicians and their skill levels?

This research question explores the managerial recommendations that can be extracted from the solutions returned from our problem. In our model, we treat the number of technicians and their skills as fixed parameters and not as part of the decision-making process. However, an analysis of the overall cost of the system in our solutions can assist decision-makers in making skill level-related decisions. It should be noted that skill levels play a crucial role in our problem, particularly because we are considering condition-based maintenance. In this context, the MOs performed depends on both the current state of the machine, revealed by the technician when he/she arrives on the machine and the skill level of the technician. For instance, a technician with lower skills may frequently arrive at machines without the capability to perform necessary maintenance operations. Thereafter, lower skill levels may have a high negative impact on costs.

In this section, we conduct experiments as follows: we adjust the instance generator such that several scenarios are considered for each instance created. Each scenario has a certain percentage, denoted as p, of the technicians with a high level of skill, allowing them to perform maintenance regardless of the machine's degradation state, i.e., $j \in \mathcal{N}$ has a high-level skill if $c_j = |\mathcal{K}|$. Other technicians have randomly assigned skill levels within the range of $[2, \mathcal{K} - 1]$. We test various values of $p \in \{0\%, 10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%\}$. The objective is to study how impacting is the pool of technicians on our costs.

We report in Table 17, for each value of p, the average deviation between the objective value found for p and the best objective value found. Note that, as expected, the best objective value is always found for $p = 100\%$.

The findings from our analysis reveal a significant cost decrease from having a team composed of technicians with a high level of skill. Reported results highlight the impact of adding more technicians with a high level of skill depending on the current pool of technicians. For example, we observe a substantial objective value gap between scenarios with low percentages of technicians with high levels of skill and those with higher percentages. This gap narrows as the proportion of technicians with a high level of skill increases, although it remains relatively significant even at a high level, such as $p = 90\%$. The observed losses for instances that involve less-skilled technicians should be considered in the context of the managerial expected costs (e.g., training cost) which are not modeled in the present work, however, it is evident that there is a continual incentive to maintain a technician team with overall high skill levels.

4.8. RQ7: How valuable is the possibility to perform different types of MO

In this research, we incorporate the possibility of performing imperfect MOs. This means that we allow for situations where a machine remains in a degraded state after an MO is performed, as opposed to always achieving a full repair, leading to the good-as-new state. In this section, our focus is on highlighting the potential cost savings that can be achieved by considering imperfect repairs, especially when dealing with geographically dispersed machines. Indeed, while it might seem intuitive to always opt for heavy MOs that restore a machine to a good-as-new state to avoid additional routing costs for revisits, our study reveals that this may still not be the most efficient approach. This is due to factors like limited time per maintenance period and variations in technician skills. For instance, we discuss a scenario in Section 2.1 where high-skilled technicians handle the major maintenance tasks to bring a machine from a highly degraded state to a moderate one, and then low-skilled technicians complete the process, bringing the machine to a good-as-new state (e.g., the first technicians take the machine from state 6 to state 3, and the second technician takes it from state 3 to state 1). To emphasize the cost savings achieved by our model, we conduct a comparative analysis. We run our solving method twice on a set of fifty instances. The first run, labeled $R1$, represents our traditional approach with no modifications. The second run, labeled $R2$, incorporates a slight alteration in the solving method, which restricts the selection of MOs to those aiming for a good-as-new degradation state. It is important to note that, even with this restriction, different types of MOs are considered, as we do conditional-based MOs. For example, even if a good-as-new state is always planned, the MO carried depends on the state of the machine: there is a different MO performed to transition from state 3 to state 1 than from state 6 to state 1.

The results have shown an average decrease of 8.95% in the objective function from $R1$ to $R2$. This indicates that relying solely on good-as-new maintenance, although initially intuitive, does not yield optimal solutions and falls significantly short of achieving optimality. Including the concept of imperfect maintenance, operations result in considerable cost savings, confirming the decisions made in our modeling approach.

5. Overall conclusion on the experimentation

In our experiments, we aimed to address several research questions that focused on two key aspects:

(i) Method Performance: We assessed the performance of our method in terms of solution quality and computation time. The results indicated that our method's computation times are suitable for the intended applications, with an average runtime of just a few minutes. Furthermore, it demonstrated excellent scalability across all instances, even with a high number of technicians, periods, and machines. Additionally, we evaluated the efficiency of our solutions through comparisons with CPLEX. CPLEX encounters scalability issues, running out of

memory when dealing with instances of non-small sizes. Consequently, we were restricted to comparison with these smaller instances. These comparisons revealed that CPLEX could only marginally improve the objective value, and this improvement was noticeable only when starting from our solutions, after several hours of computation, and on small instances.

(ii) Insights from Solution: Our results were designed to draw insights into the maintenance strategies required for a production system with geographically dispersed machines. First of all, short-term and myopic, it was found to be detrimental to the system, whereas a look-ahead approach brings major savings to the system. Second, having a range of MO options beyond full replacement and minimal repair, specifically with imperfect repairs, proved to be cost-effective. Finally, when considering imperfect maintenance, it becomes evident that there is a significant impact on the skill levels of the technicians. There is a strong incentive to minimize the presence of low-skilled technicians who primarily handle only small imperfect maintenance or minor full repairs, in favor of having a greater number of highly skilled technicians.

6. Conclusion

This work proposes a problem that couples maintenance selection with technician routing. In this problem, the technicians are based at a depot and displace themselves to execute MOs on machines. The intrinsic difficulty of the problem is that the MOs are defined by their target degradation levels. Thus, an MO that leaves the machine in a less degraded state is more costly and consumes more technician time but reduces the penalty costs associated with the virtual age of the machine in future periods. The problem also considers the constraints of technician skill level and route duration. We model the problem as a mixed-integer program and solve it by using an LS-based heuristic. The method builds a solution by sequentially adding MOs to the solution, with the selection of the MOs guided by a carefully crafted utility formula. The method relies on three LS operators to improve the partial (and final) solution. Through a set of experiments, we study the efficiency of the proposed method and conclude that (i) the running time scales well with the instance size, and (ii) the results are competitive.

This work constitutes the first contribution to the maintenance routing problem with selective MOs. Many extensions are possible in terms of modeling the degradation (e.g., continuous degradation or degradation states representing different types of fault, such as electrical or mechanical fault), restoring the machines, and optimizing the routing. Furthermore, in this study, we model the overall reliability of the system as a cost that we are aiming to minimize, alongside the logistical costs. Nevertheless, there are applications where it may not be feasible or could be viewed as an oversimplification to assign a cost to the degradation state of a machine. An alternative approach to tackle the reliability issue is to address it as an objective to be optimized (e.g., minimizing the worst virtual age of any machine). In such scenarios, we are dealing with a multiobjective problem: the minimization of logistical costs and the optimization of system reliability. Also worth investigating is the dynamic aspect of the problem, whereby random data are dynamically revealed and a real-time adaptation of technician routes may bring additional savings. In the present work, the technicians do not perform any MO when they arrive on machines that are in better degradation states than the goal state of the MOs planned. In a more realistic application, we may consider that the operation time of the MO canceled could be reused to perform a different MO (on this or another machine). This last point makes it possible to combine the current concerns for using monitoring data to optimize preventive maintenance: the so-called predictive maintenance.

CRediT authorship contribution statement

Florian Delavernhe: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization. **Bruno Castanier:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Christelle Guéret:** Writing – review & editing, Writing – original draft, Project administration, Methodology, Funding acquisition, Conceptualization. **Jorge E. Mendoza:** Writing – review & editing, Writing – original draft, Resources, Conceptualization.

Data availability

Data will be made available on request.

Acknowledgments

This work was financially supported by RFI Atlanstic 2020 (Région Pays de la Loire), France and the Canada First Research Excellence Fund through IVADO. This research was enabled in part by support provided by Calcul Québec and the Digital Research Alliance of Canada (alliancecan.ca).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cor.2024.106667.

References

Al-Najjar, B., 2007. The lack of maintenance and not maintenance which costs: A model to describe and quantify the impact of vibration-based maintenance on company's business. Int. J. Prod. Econ. 107 (1), 260–273.
Aoudia, M., Belmokhtar, O., Zwingelstein, G., 2008. Economic impact of maintenance management ineffectiveness of an oil and gas company. J. Qual. Maint. Eng.
Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J., 2006. The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton.
Blakeley, F., Argüello, B., Cao, B., Hall, W., Knolmajer, J., 2003. Optimizing periodic maintenance operations for schindler elevator corporation. Interfaces 33 (1), 67–79.
Dai, L., Stålhane, M., Utne, I.B., 2015. Routing and scheduling of maintenance fleet for offshore wind farms. Wind Eng. 39 (1), 15–30.
Flood, M.M., 1956. The traveling-salesman problem. Oper. Res. 4 (1), 61–75.
Gonzalo, A.P., Benmessaoud, T., Entezami, M., Márquez, F.P.G., 2022. Optimal maintenance management of offshore wind turbines by minimizing the costs. Sustain. Energy Technol. Assess. 52, 102230.
Irawan, C.A., Ouelhadj, D., Bakken Sperstad, I., Jones, D., 2023. A combined tactical and operational framework for maintenance scheduling and routing in offshore wind farms. J. Oper. Res. Soc. 74 (10), 2241–2260.
Irawan, C.A., Ouelhadj, D., Jones, D., Stålhane, M., Sperstad, I.B., 2017. Optimisation of maintenance routing and scheduling for offshore wind farms. European J. Oper. Res. 256 (1), 76–89.
Jafar-Zanjani, H., Zandieh, M., Sharifi, M., 2022. Robust and resilient joint periodic maintenance planning and scheduling in a multi-factory network under uncertainty: A case study. Reliab. Eng. Syst. Saf. 217, 108113.
Jia, C., Zhang, C., 2020. Joint optimization of maintenance planning and workforce routing for a geographically distributed networked infrastructure. IISE Trans. 52 (7), 732–750.
Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21 (2), 498–516.
López-Santana, E., Akhavan-Tabatabaei, R., Dieulle, L., Labadie, N., Medaglia, A.L., 2016. On the combined maintenance and routing optimization problem. Reliab. Eng. Syst. Saf. 145, 199–214.
López-Santana, E., Méndez, G., Franco, C., 2023. On the multi-period combined maintenance and routing optimisation problem. Int. J. Prod. Res. 61 (23), 8265–8290.
Matt, D.T., Rauch, E., Dallasega, P., 2015. Trends towards distributed manufacturing systems and modern forms for their design. Procedia CIRP 33, 185–190.
Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer programming formulation of traveling salesman problems. J. ACM 7 (4), 326–329.
Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.
Nguyen, T.A.T., Chou, S.-Y., 2019. Improved maintenance optimization of offshore wind systems considering effects of government subsidies, lost production and discounted cost model. Energy 187, 115909.

Nguyen, H.S.H., Do, P., Vu, H.-C., Iung, B., 2019. Dynamic maintenance grouping and routing for geographically dispersed production systems. Reliab. Eng. Syst. Saf. 185, 392–404.

O'Neil, R., Khatab, A., Diallo, C., Venkatadri, U., 2023. Optimal joint maintenance and orienteering strategy for complex mission-oriented systems: A case study in offshore wind energy. Comput. Oper. Res. 149, 106020.

Rashidnejad, M., Ebrahimnejad, S., Safari, J., 2018. A bi-objective model of preventive maintenance planning in distributed systems considering vehicle routing problem. Comput. Ind. Eng. 120, 360–381.

Schrotenboer, A.H., uit het Broek, M.A., Jargalsaikhan, B., Roodbergen, K.J., 2018. Coordinating technician allocation and maintenance routing for offshore wind farms. Comput. Oper. Res. 98, 185–197.

Si, G., Xia, T., Gebraeel, N., Wang, D., Pan, E., Xi, L., 2022. A reliability-and-cost-based framework to optimize maintenance planning and diverse-skilled technician routing for geographically distributed systems. Reliab. Eng. Syst. Saf. 226, 108652.

Si, G., Xia, T., Zhang, K., Wang, D., Pan, E., Xi, L., 2021. Technician collaboration and routing optimization in global maintenance scheduling for multi-center service networks. IEEE Trans. Autom. Sci. Eng. 19 (3), 1542–1554.

Snyder, B., Kaiser, M.J., 2009. Ecological and economic cost-benefit analysis of offshore wind energy. Renew. Energy 34 (6), 1567–1578.

Srai, J.S., Kumar, M., Graham, G., Phillips, W., Tooze, J., Ford, S., Beecher, P., Raj, B., Gregory, M., Tiwari, M.K., et al., 2016. Distributed manufacturing: scope, challenges and opportunities. Int. J. Prod. Res. 54 (23), 6917–6935.

Stålhane, M., Vefsnmo, H., Halvorsen-Weare, E.E., Hvattum, L.M., Nonås, L.M., 2016. Vessel fleet optimization for maintenance operations at offshore wind farms under uncertainty. Energy Procedia 94, 357–366.

Urbani, M., Brunelli, M., Punkka, A., 2023. An approach for bi-objective maintenance scheduling on a networked system with limited resources. European J. Oper. Res. 305 (1), 101–113.

Zhang, J., Zhao, X., Song, Y., Qiu, Q., 2022. Joint optimization of condition-based maintenance and spares inventory for a series–parallel system with two failure modes. Comput. Ind. Eng. 168, 108094.

Zhao, X., Deng, Q., Liu, X., Zhang, L., Wu, S., Jiang, C., 2022a. Integrated scheduling of distributed service resources for complex equipment considering multiple on-site MRO tasks. Int. J. Prod. Res. 60 (10), 3219–3236.

Zhao, X., He, Z., Wu, Y., Qiu, Q., 2022b. Joint optimization of condition-based performance control and maintenance policies for mission-critical systems. Reliab. Eng. Syst. Saf. 226, 108655.