TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
de Pater, I., & Mitici, M. (2027). Multi-stage stochastic programming for predictive maintenance scheduling with probabilistic Remaining Useful Life prognostics. *Reliability Engineering and System Safety*, *277*, Article 113184. https://doi.org/10.1016/j.ress.2026.113184

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Reliability Engineering and System Safety 277 (2027) 113184



Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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



Multi-stage stochastic programming for predictive maintenance scheduling with probabilistic Remaining Useful Life prognostics

Ingeborg de Pater [a,*], Mihaela Mitici [b]

[a] Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, HS 2926 Delft, The Netherlands
[b] Faculty of Science, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

ARTICLE INFO

Dataset link: https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/

Keywords:
Predictive maintenance scheduling
Remaining Useful Life
Multi-stage stochastic program
Nested Benders decomposition

ABSTRACT

Predictive maintenance for modern machines requires a dynamic process of decision making, where updated information about the health of machines steers the maintenance decisions. For this, it is important to consider the uncertainty associated with the health of the machines. In this paper, we propose a multi-stage stochastic program to schedule maintenance for machines that are continuously monitored by sensors. First, we model the uncertainty associated with the health of the machines by means of probabilistic Remaining Useful Life (RUL) prognostics. With this, the proposed stochastic program schedules maintenance for multiple machines sharing a limited capacity and spare parts. These machines can only be maintained if these resources are available. For failed machines, this may require rescheduling the maintenance of other machines. To reduce the computational time, we formulate the multi-stage stochastic program such that the constraint matrix is totally unimodular, and we can therefore solve it using nested Benders decomposition. We apply our approach to a case study on aircraft engines, planning maintenance 12 weeks ahead. We simulate the maintenance schedule over a period of 10 years. Based on this case study, the results show that our approach reduces the expected costs by 1.4% compared to a non-stochastic approach.

1. Introduction

Predictive maintenance is a promising new maintenance strategy, enabled by the abundance of sensors that monitor the health of modern systems [1]. In predictive maintenance, the data of these sensors are used to predict the Remaining Useful Life (RUL, time left until failure) of a system. The premise of predictive maintenance is that with accurate RUL prognostics, systems can be maintained just before their failure, reducing the number of failures and the maintenance costs compared to traditional maintenance approaches [2]. Most studies on predictive maintenance focus on developing RUL prognostic models. The second step of integrating the RUL prognostics into the maintenance schedule, called predictive maintenance scheduling, has received comparatively less attention. However, predictive maintenance scheduling is a complex task that faces several challenges.

The first challenge in predictive maintenance scheduling stems from the uncertainty of the RUL prognostics. Due to this uncertainty, maintaining systems only when the predicted RUL is near zero may lead to more failures than in traditional maintenance approaches. Literature accounts for this by instead starting a maintenance action when the predicted RUL falls below a non-zero threshold. For instance, in [3–5], a system is immediately replaced if the predicted RUL at a maintenance moment falls below the time to the next maintenance moment plus the mean RUL prediction error. This mean error provides some buffer time to prevent failures. Thresholds on the predicted RUL are also used to order spare parts. In [6], system replacements are scheduled when the predicted RUL falls below a threshold, where the threshold similarly consists of a fixed value plus the mean RUL prediction error. Last, in [7], a system replacement is scheduled when the predicted RUL falls below a threshold that is optimized using linear programming and a genetic algorithm. Although these studies reduce the maintenance costs and the number of failures, they only consider one identical threshold for the maintenance of all systems. The main limitation of this approach is therefore that the individual uncertainty of a single RUL prediction, that may vary between systems and over time, is not explicitly considered. This may lead to a replacement time that is too early for some systems, wasting their useful life, while it is too late for other systems, risking a failure.

To address this limitation, other studies instead consider probabilistic RUL prognostics as input to the maintenance scheduling model, which explicitly quantify the uncertainty of each individual RUL prediction. Probabilistic RUL prognostic models for instance estimate a Probability Density Function (PDF) or a confidence interval of the RUL.

* Corresponding author.
E-mail address: i.i.depater@tudelft.nl (I. de Pater).

https://doi.org/10.1016/j.ress.2026.113184
Received 10 March 2026; Received in revised form 15 July 2026; Accepted 16 July 2026
Available online 22 July 2026
0951-8320/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

For instance, in [8], the optimal replacement time of an aircraft engine is determined by minimizing the expected costs. The predicted PDF of the RUL is used as input in a deep reinforcement learning algorithm. In [9,10], the authors identify both the optimal time to replace a system and the optimal ordering time for a new spare part. In [10], the expected costs are minimized, while the multi-objective optimization model in [9] minimizes the expected maintenance costs while also maximizing the expected reliability and availability. Finally, the authors of [11] jointly optimize the maintenance and the production schedule for a single machine, based on the predicted failure probabilities within upcoming time windows. In this study, we also address the challenge of uncertain RUL prognostics by using probabilistic RUL prognostics as input to the maintenance scheduling model.

A limitation of the aforementioned studies, however, is that they schedule maintenance for only one single system at the time. However, a second challenge in predictive maintenance scheduling is that, usually, a fleet of multiple systems must be maintained together. These systems often compete for shared limited resources, such as spare parts and manpower. Other studies therefore schedule predictive maintenance for multiple systems. For example, in [12,13], linear programming is employed to schedule maintenance for a fleet of aircraft engines. In both studies, a maximum capacity constraint is imposed, such that only a limited number of systems can be maintained at the same time. In [14], linear programming is similarly used to plan the replacements of cooling units. Here, the maintenance is linked through both a limited amount of spare cooling units and a limited capacity. Last, in [15], deep reinforcement learning is used to jointly optimize the production and maintenance schedule of multiple machines. Here, the maintenance of the multiple machines is linked through the production requirements. To address this second challenge, we also consider the maintenance scheduling for multiple systems under limited resources in this study.

In predictive maintenance, maintenance for a system is often scheduled when the predicted probability of failure is small, but non-zero [12]. As a consequence, system failures still occur. A third challenge in predictive maintenance scheduling therefore concerns the realistic handling of these failures. The studies discussed so far assume that failed systems can always be maintained immediately, regardless of the constraints imposed on the maintenance schedule. For instance, in [3,4,10], both the maintenance and spare parts ordering for a single system are optimized. In these three studies, it is assumed that a corrective replacement can immediately be performed upon a failure, even if no spare part is available. This incurs a high out-of-stock cost. In [12,13], the maintenance schedule for multiple systems is optimized, subject to a capacity constraint. If a system fails, it is assumed that it can always be replaced immediately, disregarding the capacity constraint.

In reality, however, it is unrealistic to assume that failed systems can always be maintained immediately. For instance, the maintenance of other systems may have to be rescheduled to free up resources for maintaining the failed system. Ignoring such maintenance scheduling constraints in the case of system failures may have two consequences. First, the cost of failure might be underestimated. For instance, a failed system that cannot immediately be maintained leads to more downtime, while the rescheduling of maintenance for other systems might lead to a higher probability of an additional failure. Second, the maintenance schedule becomes less robust, since maintenance decisions that mitigate the consequences of a failure are not explicitly optimized. For instance, if a failed system can only be replaced when a spare is available, it might be beneficial to order spare parts relatively early or to adjust the maintenance time of other systems to ensure a spare is always available. The main innovation of this paper is therefore to consider a realistic failure handling process, where failed systems might not be maintained immediately due to the constraints on the maintenance schedule.

In [16], the constraints on the maintenance schedule are also considered when systems fail, as in this paper. The authors combine a POMDP (Partially Observable Markov Decision Process) with deep reinforcement learning to schedule corrective, preventive and predictive maintenance tasks for a fleet of aircraft. By using a (PO)MDP, potential failures and the corresponding updates to the maintenance schedule are integrated in the optimization. In the POMDP, the authors define three states for the health of a system (healthy, degraded and failed), based on the predicted RUL. While this discretization reduces the state space of the MDP and thus enables solving the model in a reasonable time, it also allows for less granular maintenance decisions compared to directly considering the probabilistic RUL prognostics. Moreover, the solution becomes dependent on the formulation of the states.

To address these limitations, we instead propose to use stochastic programming to optimize the predictive maintenance schedule, using probabilistic RUL prognostics as input. With stochastic programming, we do not need to discretize the probabilistic RUL prognostics, thus allowing for more granular maintenance decisions. Specifically, we schedule maintenance for multiple systems under a capacity constraint and with a shared stock of spare parts. If a system fails, the maintenance of the other systems may need to be rescheduled to free up resources to maintain the failed system. This limited capacity and number of spares introduce long-term dependencies in the maintenance schedule. We consider these dependencies by formulating a stochastic program with multiple stages. This stochastic program jointly optimizes the initial maintenance schedule and its potential future updates in response to failures. In this way, the consequences of a potential failure can already be mitigated, enhancing the robustness of the final solution.

Stochastic programming has previously been applied in literature to schedule condition-based, rather than predictive, maintenance. In condition-based maintenance, the degradation of a system is explicitly modeled using statistical processes, rather than estimating the RUL solely based on sensor data. For instance, in [17], inspections are used to reveal the health state of a component in a multi-component system, where the evolution of degradation between inspections is uncertain. In [18,19], the maintenance optimization is integrated with the production planning for a plant. Here, the production decisions influence the uncertain degradation process, and consequently the probability of failure. In contrast to our approach, the papers above schedule maintenance for only a single system, and do not consider data-driven RUL prognostics. Moreover, these papers assume that failed systems can immediately be maintained, and use stochastic programming to handle other sources of uncertainty instead.

We apply the proposed approach to a case study with aircraft engines. First, we obtain probabilistic RUL prognostics for these engines with a Convolutional Neural Network with Monte Carlo dropout. We then formulate the multi-stage stochastic program that optimizes the maintenance schedule for the next 12 weeks/stages. Because we formulate our constraint matrix such that it is totally unimodular, we can solve the stochastic program using nested Benders decomposition. We subsequently simulate a long-term maintenance schedule for a period of 10 years with a rolling horizon approach. To analyze the benefits of jointly optimizing the initial maintenance schedule and its potential future updates, we compare our stochastic approach with a non-stochastic approach that does not consider the future updates. Our stochastic approach reduces the maintenance costs by 1.4% compared to this non-stochastic version.

In summary, our work provides the following novel contributions:

- We use a multi-stage stochastic program that optimizes a predictive maintenance schedule for multiple systems sharing limited resources. This stochastic program is able to integrate data-driven, probabilistic RUL prognostics.
- The multi-stage stochastic program anticipates failures by explicitly optimizing maintenance decisions that mitigate the consequences of a failure. To the best of our knowledge, we are the first to use a stochastic program for this realistic failure handling. This allows us to directly consider the probabilistic RUL prognostics without discretization.

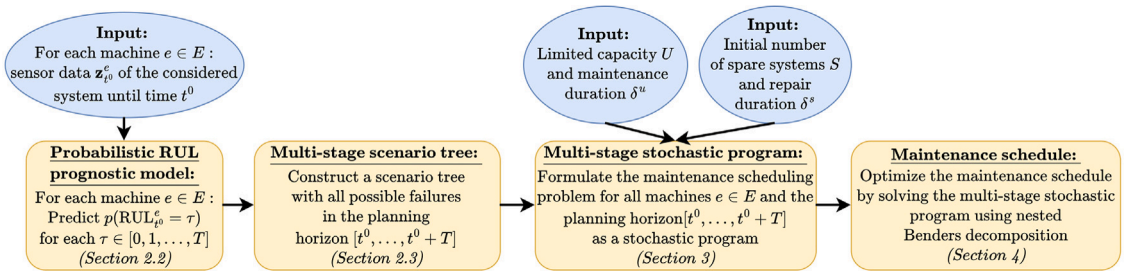


Fig. 1. An overview of the predictive maintenance scheduling process.

- We solve the multi-stage stochastic program efficiently with nested Benders decomposition by formulating the constraint matrix such that it is totally unimodular.

In the remainder of this work, we first introduce the considered predictive maintenance scheduling problem in Section 2. We then formulate the corresponding multi-stage stochastic program in Section 3, and introduce nested Benders decomposition to solve this problem in Section 4. Last, we apply this method to a case study with aircraft engines in Section 5, and perform a sensitivity analysis in Section 6. The conclusions are provided in Section 7.

2. Problem formulation — predictive maintenance scheduling

In this section, we describe a discrete-time predictive maintenance scheduling problem for a set of machines, denoted by E. These machines can be, for instance, trucks, aircraft or industrial machines. We only schedule maintenance for a single system within each machine. For instance, we could schedule maintenance for only the engine of an aircraft or truck, without considering other systems. We schedule maintenance for these machines within a planning window $[t^0, t^0 + 1, \ldots, t^0 + T]$, i.e., from time-step t^0 to time-step $t^0 + T$. An overview of the proposed approach is given in Fig. 1.

2.1. Maintenance process with a limited capacity and a limited number of spares

We maintain a machine $e \in E$ by replacing the considered system with a spare from the stock. This maintenance action takes δ^u time-steps, e.g., if maintenance for a machine starts at time-step t, it is finished at time-step $t + \delta^u$. Due to a limited capacity, at most U machines can be maintained simultaneously. In the beginning of the considered planning window, a stock with S spare systems is available. Once a system is replaced, it is repaired in an external facility. After repair, the system is added as spare to the stock and becomes available for (re)installation in any machine $e \in E$. The repair process takes δ^s time-steps, e.g., if maintenance for a machine starts at time-step t, its replaced system becomes available as spare at time-step $t + \delta^s$. This process is illustrated in Fig. 2.

We consider three costs components in the maintenance process. First, we assume that a failure of the considered system in machine $e \in E$ leads to an immediate failure of the entire machine. Upon such a failure, a large failure penalty c_f is incurred due to unexpected operational disturbances. Second, maintaining a machine costs c_m. Last, each time-step a machine remains failed without being maintained, a downtime cost c_d is incurred.

2.2. Data-driven probabilistic RUL prognostics

Let $\mathbf{z}^e_{t^0}$ denote a multivariate vector of sensor measurements recorded for the considered system in machine $e \in E$, such as the temperature, pressure or vibration. These measurements are recorded from its installation in the machine up to the initial time-step t^0 of the planning window. With these measurements, we estimate the Remaining Useful Life (RUL) of the considered system in machine e with a probabilistic approach. Let $p(\text{RUL}^e_{t^0} = \tau), \tau \geq 0$ denote the probability that the RUL of the considered system in machine e at time t^0, $\text{RUL}^e_{t^0}$, equals τ. In other words, $p(\text{RUL}^e_{t^0} = \tau)$ denotes the probability that the considered system in machine e fails at time $t^0 + \tau$.

2.3. Scenario tree for multi-stage maintenance scheduling

We model the maintenance scheduling problem with a discrete-time multi-stage scenario tree. At each time-step t within the planning window, we observe for each machine $e \in E$ whether it has failed. In this model, we only consider machine failures due to a failure of the considered system. These potential failures define the nodes of the scenario tree.

Fig. 3 shows an example of a scenario tree with two machines $E = \{1, 2\}$ and three time-steps $[t^0, t^0 + 1, t^0 + 2]$. Let N denote the set of all nodes in the multi-stage scenario tree, and let $E^n_{\text{fail}} \subseteq E$ denote the set of machines that have failed prior to, or at, node n. We then define $E^n_{\text{new}} \subseteq E^n_{\text{fail}}$ as the subset of all new failures, i.e., the machines that fail at node n itself. For each machine $e \in E^n_{\text{fail}}$, we also record the time-step $t^{n,e}_{\text{fail}}$ at which machine e failed at node n. For instance, for node 7 in Fig. 3, $E^7_{\text{fail}} = \{1, 2\}$ with $t^{7,1}_{\text{fail}} = 1$ and $t^{7,2}_{\text{fail}} = 2$. We assume that each machine fails at most once in the considered planning window.

We denote the subset of nodes that belong to time-step t with $N_t \subseteq N$, while we denote the time-step of a node $n \in N$ by t^n. At the initial time-step t^0, the tree begins with a single root node 0. Let a_n denote the direct ancestor node of a node $n \in N \setminus \{0\}$, and let W^n denote the direct successor nodes for any node $n \in N \setminus N_{t^0+T}$. The leaf nodes at the last time-step $t^0 + T$ do not have any successors. To calculate the transition probabilities between nodes, we use conditional probabilities. Specifically, let $p'(\text{RUL}^e_{t^0} = \tau)$ denote the conditional probability that the considered system in machine $e \in E$ fails at time-step $t^0 + \tau$, given that it has not failed before that time, i.e.,

$$p'(\text{RUL}^e_{t^0} = \tau) = p(\text{RUL}^e_{t^0} = \tau | \text{RUL}^e_{t^0} \geq \tau) = \frac{p(\text{RUL}^e_{t^0} = \tau \cap \text{RUL}^e_{t^0} \geq \tau)}{p(\text{RUL}^e_{t^0} \geq \tau)}$$
$$= \frac{p(\text{RUL}^e_{t^0} = \tau)}{1 - \sum_{d=0}^{\tau-1} p(\text{RUL}^e_{t^0} = d)}. \quad (1)$$

From the conditional failure probabilities, the transition probability q_{nl} from node n to a successor node $l \in W^n$ is:

$$q_{nl} = \prod_{e \in E^l_{\text{new}}} p'(\text{RUL}^e_{t^0} = t^l) \prod_{e \in E \setminus E^l_{\text{fail}}} \left(1 - p'(\text{RUL}^e_{t^0} = t^l)\right). \quad (2)$$

2.4. Dynamic updates of the proposed maintenance schedule

At each node $n \in N$, a maintenance schedule is proposed. All maintenance tasks scheduled at time t^n, i.e., at the time of node n, are immediately executed. All maintenance tasks scheduled at later time-steps, however, can still be rescheduled at the successor nodes of n, based on newly observed failures. This is illustrated in Fig. 4.

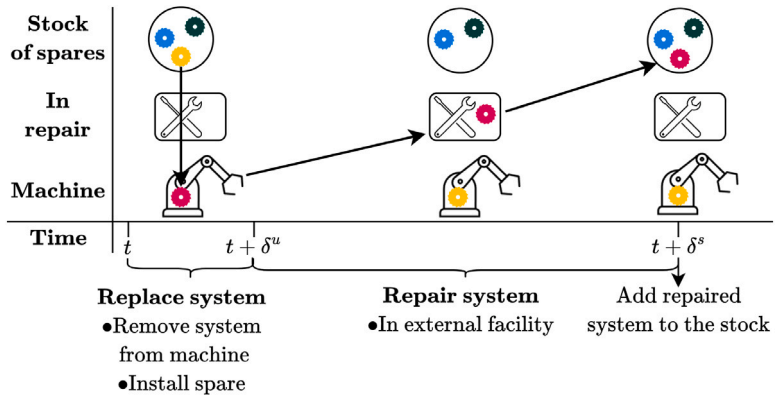


Fig. 2. An illustration of the maintenance process.

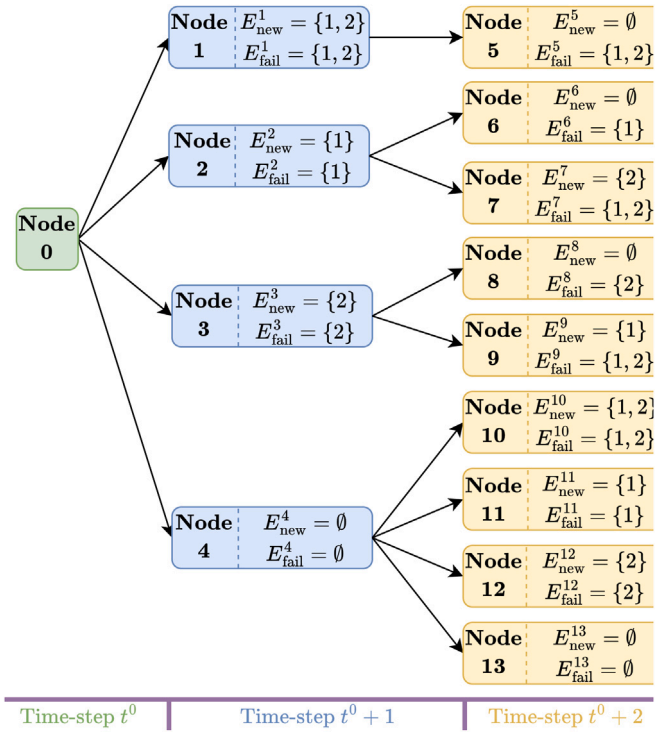


Fig. 3. Example of a scenario tree for two machines $E=\{1,2\}$ and three time-steps $[t^0, t^0+1, t^0+2]$.

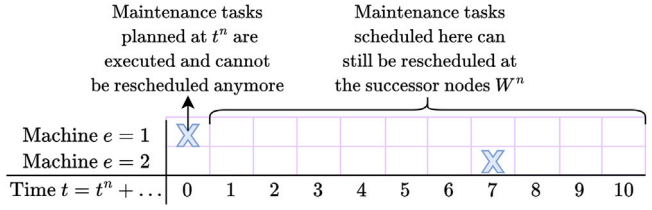


Fig. 4. An example of a proposed maintenance schedule at node n, time t^n.

Fig. 5 shows an update of a proposed maintenance schedule by the insertion and cancellation of a maintenance task. In this example, we schedule at the root node (time t^0) to maintain machine $e=1$ at time t^0+6. At a successor node l, time $t^l=t^0+1$, we observe that machine 1 fails. At this successor node, we might therefore reschedule the maintenance task to occur immediately. In this case, we first *cancel* the maintenance task scheduled at time t^0+6, and subsequently *insert* a new maintenance task at time t^0+1. Since constantly changing the maintenance schedule is undesirable, we define that canceling a scheduled maintenance task costs $c_k > 0$, while inserting a new maintenance task costs $c_i > 0$.

3. Multi-stage stochastic program to model the predictive maintenance scheduling problem

In this section, we formulate the optimization of the predictive maintenance schedule with a multi-stage stochastic linear program. Specifically, we formulate for each node $n \in N$ a linear program that optimizes the proposed maintenance schedule at this node. We introduce the decision variables (Section 3.1), the constraints (Section 3.2) and the objective function (Section 3.3) of this linear program.

3.1. Decision variables of the linear program at a node $n \in N$

The main decision variable x^n_{et} denotes at node $n \in N$, at which time-step $t \in [t^n, \ldots, t^0+T]$ we plan to maintain machine $e \in E$:

$$x^n_{et} = \begin{cases} 1, & \text{At node } n, \text{ we plan to maintain machine } e \in E \text{ at time } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{3}$$

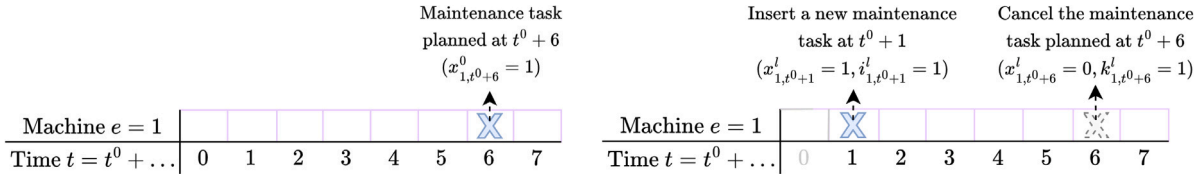
At each node $n \in N \setminus \{0\}$, we introduce auxiliary variables that keep track of updates to the maintenance schedule at time-step $t \in [t^n, \ldots, t^0+T]$. Specifically, we track the insertion of new maintenance tasks and the cancellations of previously scheduled maintenance tasks:

$$i^n_{et} = \begin{cases} 1, & \text{At node } n, \text{ a new maintenance task for machine} \\ & e \in E \text{ is scheduled at time } t, \text{i.e., } x^n_{et}=1 \\ & \text{and } x^{a_n}_{et}=0. \\ 0, & \text{Otherwise.} \end{cases} \tag{4}$$

$$k^n_{et} = \begin{cases} 1, & \text{At node } n, \text{ a previously scheduled maintenance} \\ & \text{task of machine } e \in E \text{ at time } t \text{ is} \\ & \text{canceled, i.e., } x^{a_n}_{et}=1 \text{ and } x^n_{et}=0. \\ 0, & \text{Otherwise.} \end{cases} \tag{5}$$

Recall that a_n denotes the ancestor node of node n. These variables are illustrated in Fig. 5. We also define an auxiliary variable indicating whether machine $e \in E$ has already been maintained prior to node $n \in N$:

$$m^n_e = \begin{cases} 1, & \text{Machine } e \in E \text{ has been maintained prior to node } n. \\ 0, & \text{Otherwise.} \end{cases} \tag{6}$$



(a) Maintenance schedule proposed at root node 0, time t^0: A maintenance task for machine 1 is scheduled at time-step t^0+6.

(b) Maintenance schedule proposed at a successor node l, time $t^l = t^0+1$. We update the maintenance schedule proposed at the root node, since machine 1 fails.

Fig. 5. An example of updating the proposed maintenance schedule by the cancellation and insertion of a maintenance task.

Last, we define for each node $n \in N$ two auxiliary variables, within $t \in [t^n, \ldots, t^0+T]$, that track the number of spare systems and capacity:

s^n_t =Number of available spare systems in stock at node n at time t, (7)

u^n_t =Available capacity at node n at time t. (8)

3.2. Constraints of the linear program at a node $n \in N$

3.2.1. Constraints for the insertion and cancellation of maintenance tasks

For any node $n \in N \setminus \{0\}$, we add the following constraints to define the auxiliary variables that keep track of the insertion (Eq. (4)) and cancellation (Eq. (5)) of maintenance tasks:

$$x^n_{et} - i^n_{et} \leq x^{a_n}_{et} \qquad \forall e \in E, \quad \forall t \in \left[t^n, \ldots, t^0+T\right] \tag{9}$$

$$k^n_{et} + x^n_{et} \geq x^{a_n}_{et} \qquad \forall e \in E, \quad \forall t \in \left[t^n, \ldots, t^0+T\right]. \tag{10}$$

For instance, we enforce that the insertion variable $i^n_{et} = 1$ if a new maintenance task for machine e at time t is scheduled at node n that was not scheduled yet at the ancestor node a_n, i.e., if $x^n_{et} = 1$ and $x^{a_n}_{et} = 0$. These constraints are only sufficient to define i^n_{et} and k^n_{et} if the costs c_i and c_k associated with inserting and canceling a maintenance task, respectively, are strictly positive.

3.2.2. Constraints for tracking maintained machines

For any node $n \in N$, we add the following constraint to keep track of maintained machines:

$$m^n_e = m^{a_n}_e + x^{a_n}_{et^{a_n}} \qquad \forall e \in E \tag{11}$$

$$x^n_{et} + m^n_e \leq 1 \qquad \forall e \in E, \forall t \in [t^n, \ldots, t^0+T]. \tag{12}$$

Eq. (11) defines the variable m^n_e, which indicates if machine e has already been maintained prior to node n. This is the case if (i) it already has been maintained prior to the ancestor node a_n ($m^{a_n}_e = 1$) or (ii) if it is maintained at the ancestor node a_n itself ($x^{a_n}_{et^{a_n}} = 1$). At the root node, the right-hand side of Eq. (11) is 0 since no machine has been maintained yet. We assume that a machine, once maintained, will not fail anymore within the considered planning window. Eq. (12) therefore ensures that each machine is maintained at most once.

3.2.3. Constraints to model the stock of spares

For any node $n \in N$, we add the following constraints to model the variable s^n_t, denoting the available spare systems at node n and time t:

$$s^n_t + \sum_{e \in E} x^n_{et^n} = s^{a_n}_t \qquad \forall t \in [t^n, \ldots, \min(t^n+\delta^s-1, t^0+T)] \tag{13}$$

$$s^n_t = S \qquad \forall t \in [t^n+\delta^s, \ldots, t^0+T] \tag{14}$$

$$\sum_{e \in E} \sum_{t'=\max(t^n, t-\delta^s+1)}^{t} x^n_{et'} \leq s^{a_n}_t \qquad \forall t \in [t^n, \ldots, t^0+T] \tag{15}$$

Eqs. (13) and (14) define the number of spares available in stock per time-step. Only maintenance tasks scheduled at node n itself (i.e., at time t^n) are considered, since maintenance tasks scheduled at later time-steps may still be canceled. When we maintain a machine e at node n, we immediately use a spare system at time t^n. However, the replaced system only becomes available as spare after δ^s time-steps (see Fig. 2). Consequently, there is one spare system less available from time t^n to time $t^n+\delta^s$ (Eq. (13)). At time $t^n+\delta^s$, however, all replaced systems are repaired and returned to the stock, restoring the number of spares to the initial number of available spares S (Eq. (14)). We add the same constraints to the root node 0, where we set the right-hand side of Eq. (13) equal to the initial number of spares S.

Eq. (15) ensures the feasibility of the proposed maintenance schedule at node n by enforcing that a spare is available for each scheduled maintenance task. Specifically, it imposes that the number of maintenance tasks scheduled within each time-window with a length of δ^s time-steps does not exceed the number of available spares in this window.

3.2.4. Constraints to model the capacity

For any node $n \in N$, we add the following constraints to model the variable u^n_t, denoting the available capacity at node n and time t:

$$u^n_t + \sum_{e \in E} x^n_{et^n} = u^{a_n}_t \qquad \forall t \in [t^n, \ldots, \min(t^n+\delta^u-1, t^0+T)] \tag{16}$$

$$u^n_t = U \qquad \forall t \in [t^n+\delta^u, \ldots, t^0+T] \tag{17}$$

$$\sum_{e \in E} \sum_{t'=\max(t^n, t-\delta^u+1)}^{t} x^n_{et'} \leq u^{a_n}_t \qquad \forall t \in [t^n, \ldots, t^0+T]. \tag{18}$$

These constraints are similar to the constraints that model the spares in Section 3.2.3. Specifically, constraint (16) and (17) define the remaining capacity, where U denotes the initial capacity. Eq. (18) ensures that the proposed maintenance schedule does not exceed the available capacity. At the root node 0, the right-hand side of constraints (16) and (18) equals the initial capacity U.

3.2.5. Bounds of the decision variables

Last, we impose for each node $n \in N$ the following boundaries on the decision variables:

$$s^n_t \in [0, 1, \ldots, S], u^n_t \in [0, 1, \ldots, U] \qquad \forall t \in [t^n, \ldots, t^0+T] \tag{19}$$

$$x^n_{et}, i^n_{et}, k^n_{et}, m^n_e \in \{0, 1\} \qquad \forall e \in E, \forall t \in [t^n, \ldots, t^0+T] \tag{20}$$

Here, the root node 0 does not contain the variables i^n_{et} and k^n_{et}.

3.3. Objective function of the linear program at a node $n \in N$

We minimize the expected costs over the expected lifetime of the systems in the machines. Let $A_e(t)$ denote the age of the considered system in machine $e \in E$ at time t. Let y^n denote the set of decision variables of the linear program at node $n \in N$. Let $v^n(y^{a_n})$ denote the objective function of node $n \in N \setminus \{0\}$:

$$v^n(y^{a_n}) = \sum_{e \in E^n_{\text{new}}} \left(1 - m^n_e\right) \frac{c_f}{A_e(t^n)} + \sum_{e \in E^n_{\text{fail}}} \left(1 - m^n_e - x^n_{et^n}\right) \frac{c_d}{A_e\left(t^{n,e}_{\text{fail}}\right)}$$

$$+ x^n_{et^n} \frac{c_m}{A_e\left(t^{n,e}_{\text{fail}}\right)} \tag{21}$$

$$+ \sum_{e \in E \setminus E^n_{\text{fail}}} x^n_{et^n} \frac{c_m}{A_e(t^n)} + \sum_{e \in E} \sum_{t=t^n}^{t^0+T} \left(c_i i^n_{et} + c_k k^n_{et}\right) + \sum_{l \in W^n} q_{nl} v^l(y^n).$$

The first term of this objective function contains the costs of the new failures, where c_f denotes the failure penalty. Here, we do not incur any failure costs for machines maintained prior to node n, i.e., if $m^n_e = 1$. The second term of the objective contains (i) the downtime costs of machines that remain failed without being maintained (which costs c_d per time-unit) and (ii) the costs c_m of maintaining machines. The third term represents the costs c_m of maintaining machines that have not failed yet. For node n, we only include the maintenance costs for machines maintained at node n itself, i.e., at time t^n. However, the costs of maintenance tasks scheduled at later time-steps are implicitly included through the future expected costs. We scale these three cost terms by the age of the considered system in machine e, at the moment of the maintenance task or failure time.

The fourth term of the objective reflects the costs of canceling (c_k) and inserting (c_i) maintenance tasks. When updating the maintenance schedule for machine e, the maintenance time of the machine, and thus the age of the considered system at the moment of maintenance, may still change. We therefore do not scale these costs by the age of the considered system. The last term contains the expected future costs at node n, which is obtained by summing the costs $v^l(y^n)$ at all successor nodes $l \in W^n$ of node n, scaled by the transition probability q_{nl}. These costs are excluded in the objective function of the leaf nodes at time $t^0 + T$. The objective function at the root node is the same as for any node $n \in N \setminus N_{t^0+T}$, except that it does not include the costs for canceling and inserting maintenance tasks.

3.3.1. Objective function at the leaf nodes

We aim to maintain each selected machine within the considered planning window. Otherwise, we could simply leave the machine out of the maintenance schedule optimization. At the same time, however, we want to ensure that our model is feasible. We therefore add the following term to the objective function of all leaf nodes n, giving a large penalty M if a machine is not maintained within the considered time window:

$$\sum_{e \in E} M \cdot \left(1 - m^n_e - x^n_{e,t^0+T}\right). \tag{22}$$

4. Solution method for the stochastic program

4.1. Totally unimodular constraint matrix

Our stochastic program contains a mix of binary and integer variables. If the constraint matrix of an integer program is totally unimodular, and if the right-hand side coefficients are integral, then every extreme point of the feasible region is integral [20]. In this case, the optimal solution of the relaxation of the integer linear program (found with the simplex method), is thus integer. We have formulated our constraint matrix of the linear program at any node $n \in N$ such that it is totally unimodular. We show this in the Appendix with the consecutive ones property [21]. We also show in the Appendix that in the optimal solution of the stochastic program, the right-hand side of each linear program is integer. When we solve the linear relaxation of our integer program with the simplex method, the optimal solution will thus be integer.

To ensure that the constraint matrix is totally unimodular, it is feasible to schedule multiple maintenance tasks for a single machine. However, in constraint (12), we do restrict that at most one maintenance task is executed for each machine. In general, it is not desirable to plan multiple maintenance tasks and subsequently cancel them. To prevent this, the cancellation costs c_k must be larger than the insertion costs c_i ($c_k > c_i$).

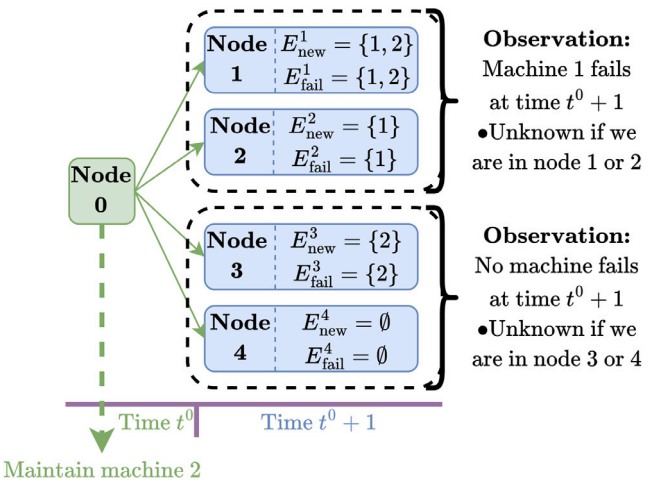


Fig. 6. An example of the observed failure times with the endogenous uncertainty in the considered problem.

4.2. Endogenous uncertainty

The scenario tree is constructed based on the uncertain failure times of the considered systems. However, the actual failure time of a system is only *observed* if *no* maintenance is performed on the machine in which it is installed. For instance, in Fig. 6, we maintain machine 2 at the root node, time t^0. Due to this maintenance task, the actual failure time of machine 2 will not be observed anymore. If we therefore observe at time $t^0 + 1$ that machine 1 fails, we do not know if we are in node 1 or in node 2, i.e., node 1 and 2 are indistinguishable. This is called endogenous uncertainty [22,23]. After maintaining a machine, we assume that the probability of failure is zero at all subsequent time-steps in the planning window.

The optimal solution of two indistinguishable nodes, such as node 1 and 2 in our example, must be identical. In our problem, this property arises automatically from the assumption that we can maintain a machine at most once in the considered planning window. Consequently, at node n, it is only feasible to cancel all scheduled maintenance tasks and not schedule any new maintenance tasks for a machine e that is already maintained prior to node n. Therefore, the only feasible values for the main decision variables x^n_{et} and the corresponding auxiliary variables for any machine maintained prior to node $n \in N$ are fixed. Since only the failure time of a maintained machine might not be observed, the optimal solution at any node is independent of the unobserved failure times. The optimal solution for indistinguishable nodes is thus identical.

4.3. Nested benders decomposition

Our problem is sufficiently expensive and complete, since the objective value of any node $n \in N$ cannot go to $-\infty$ or ∞. We thus solve the relaxation of the multi-stage stochastic program using nested Benders decomposition, also called the nested L-shaped method. We shortly recap this solution method in the context of our problem here, but we refer the interested reader to [24,25] for more details.

The schematic version of the objective function at node $n \in N$ is denoted as:

$$C^n + (c^n)' y^n + Q^n(y^n), \tag{23}$$

where $'$ denotes the transpose and $Q^n(y^n)$ denotes the future expected costs:

$$Q^n(y^n) = \sum_{l \in W^n} q_{nl} v^l(y^n), \tag{24}$$

where W^n denote the successor nodes of node n, q_{nl} denotes the transition probability from node n to node $l \in W^n$, and $v^l(y^n)$ denotes the costs at node $l \in W^n$. Moreover, C^n denotes the constant term and $(c^n)'y^n$ contains all remaining terms of the objective function.

The difficult term in a stochastic program are the expected future costs $Q^n(y^n)$. In the nested Benders decomposition, $Q^n(y^n)$ is therefore replaced by a variable θ_n for each node $n \in N \setminus N_{t^0+T}$. We let the variable θ^n converge to $Q^n(y^n)$ by iteratively adding constraints on θ^n, called optimality cuts, to the linear program. To start, we add the constraint $\theta^n \geq 0$ to each node $n \in N \setminus N_{t^0+T}$, where 0 is a trivial lower bound for the expected future costs. At each node $n \in N$, we always have a feasible solution, since we can always cancel all scheduled maintenance tasks and plan no new maintenance tasks. It is thus not necessary to derive feasibility cuts.

We move through the scenario tree using the "Fast-Forward-Fast-Backward" (FFFB) heuristic [25]. We start in the forward mode and solve the linear program of the root node, of each node at time $t^0 + 1$, etc., until we have solved the linear program of each leaf node. We then switch to the backward mode, and use the solution of the leaf nodes to derive a new optimality cut for each node at time t^0+T-1, then for each node at time $t^0 + T - 2$, etc., until we have added a new optimality cut to the root node. We then switch back to the forward mode and repeat the procedure. We stop once we have found an optimal solution. We count each switch in direction as a new iteration.

Let $\hat{\pi}^{l,i}$ denote the optimal dual variables of the linear program at a successor node $l \in W^n$ in iteration i. The optimality cut that we add to node n is:

$$\theta^n \geq \sum_{l \in W^n} q_{nl} \left(C^l + \left(\hat{\pi}^{l,i}\right)^T \left(h^l - T^l y^n\right)\right), \tag{25}$$

where $h^l - T^l y^n$ denotes the right-hand side coefficients of node l, while C^l denotes the constant term in the objective function of node l. It is only useful to add this cut if the optimal value of θ^n at iteration $i-1$, $\hat{\theta}^{n,i-1}$, does not fulfill this constraint yet. Here, we use an optimality tolerance of ϵ, where ϵ is a very small number. We thus only add the cut if $\hat{\theta}^{n,i-1} + \epsilon$ is strictly smaller than the value of the cut in iteration i. We find an optimal solution, given the optimality tolerance, if we move backward from the leaf nodes to the root node without adding any optimality cuts.

By adding optimality cuts to the linear programs, the constraint matrices are not totally unimodular anymore. The solutions we find at each node during the FFFB heuristic are thus not necessarily integer. However, with the FFFB heuristic, we converge to the optimal solution. If there is one unique optimal solution, this optimal solution is integer [26], with a tolerance of ϵ. If there are multiple optimal solutions, then the obtained optimal solution is not necessarily integer [26]. In this case, we can solve our problem several times, while restricting a non-integer variable to a different value each time (branch-and-bound). However, we expect that this is highly unlikely for the considered problem, and it indeed does not occur in the case study.

The optimality cut in Eq. (25) is derived with the unicut strategy, where at most one optimality cut is added to a node n per iteration. Another strategy to derive optimality cuts is the multicut strategy [27], where we add one variable θ^n_l to the linear program at node n for each successor node $l \in W^n$. This variable represents the expected future costs at the successor node l only. The objective at node n becomes to minimize $C^n + (c^n)'y^n + \sum_{l \in W^n} \theta^n_l$. As in Eq. (24), C^n denotes the constant term, while $(c^n)'y^n$ contains all remaining terms of the objective function of node $n \in N$. We add an optimality cut on θ^n_l to node n for each successor node $l \in W^n$ when going backward:

$$\theta^n_l \geq q_{nl} \left(C^l + (\hat{\pi}^{l,i})^T (h^l - T^l y^n)\right). \tag{26}$$

Here, as in Eq. (25), $(h^l - T^l y^n)$ denotes the right-hand side coefficients of node l, $\hat{\pi}^{l,i}$ denotes the optimal dual variables in iteration i at successor node $l \in W^n$, and C^l denotes the constant term of the objective function of node l. We again only add this cut if it is not yet fulfilled in the current solution, with a tolerance of ϵ.

Table 1
C-MAPSS datasets for turbofan engines from [28,29].

	FD001	FD002	FD003	FD004
Training instances	100	260	100	249
Testing instances	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

5. Case study — Maintenance schedule of aircraft engines using probabilistic RUL prognostics

In this section, we illustrate the predictive maintenance scheduling model with a case study for engines, where each engine (the considered system) is installed in an aircraft (the machine). We consider discrete time-steps of one week. We first explain the probabilistic RUL prognostic model in Section 5.1, and introduce the parameters and benchmarks of the stochastic model in Sections 5.2 and 5.3, respectively. We then optimize the maintenance schedule for one time window in Section 5.4, while we analyze the long-term impact of our approach by simulating the maintenance schedule for a period of 10 years in Section 5.5.

5.1. Probabilistic Remaining Useful Life (RUL) prognostics for aircraft engines

In this case study, we employ the aircraft engines in the C-MAPSS dataset [28,29]. In this section, we shortly discuss how we predict the PDF of the RUL of these engines (i.e., the probabilistic RUL prognostics) following [12]. The C-MAPSS dataset contains the simulated measurements of 21 sensors. For each engine, one measurement per sensor per flight is available. The data of the C-MAPSS dataset is divided into four subsets with different operating conditions and fault modes, named FD001, FD002, FD003 and FD004 (see Table 1). Each subset is in turn divided into a training and a test set.

To predict the RUL, we use the Convolutional Neural Network (CNN) introduced in [12], consisting of five convolutional layers and two fully connected layers. We use the same architecture, sensor selection, hyperparameters and optimization tools as in Section 3.1 of [12]. In the test phase, we predict the PDF of the RUL using Monte Carlo dropout with a dropout rate of 0.5 [39]. With this, we generate 5000 RUL predictions per flight in the test set, with which we construct the PDF. The obtained RUL predictions, however, are in flights instead of in weeks. We transform the RUL predictions in flights to RUL predictions in weeks by assuming that each aircraft performs 10 flights per week.

Table 2 shows the Root Mean Square Error (RMSE) [2] with the mean RUL predictions of the test engines. The RMSE is between 12.3 to 16.0 flights, with an average of 13.7 flights. As shown in Table 2, this is comparable to the RUL predictions in recent, state-of-the-art papers. Improving existing RUL prediction models, while quantifying the uncertainty, falls outside the scope of this paper.

Table 3 shows several metrics constructed using either the mean RUL prediction or the probabilistic RUL prognostics of the test engines. The Continuous Ranked Probability Score (CRPS) evaluates both the accuracy and the sharpness of a single probabilistic RUL prognostic, and is seen as a probabilistic generalization of the absolute error. The CRPS is low if the estimated RUL distribution is both accurate, i.e., centered around the true RUL, and sharp, i.e., with a low variance [40]. Overall, the mean CRPS is lower than the Mean Absolute Error (MAE) for all test sets. Subset FD002 and FD004, which have a higher RMSE and MAE than subset FD001 and FD003, also have a higher mean CRPS.

Table 3 and Fig. 7 additionally show the calibration of the probabilistic RUL prognostics. For the confidence intervals up to roughly 80%, the probabilistic RUL predictions are well calibrated or slightly underconfident. At the confidence intervals of 90%, 95% and 99%, however, the probabilistic RUL predictions are slightly overconfident

Table 2
RMSE (in flights) of the mean RUL predictions for the test engines of the four subsets of C-MAPSS, for our approach and other recent, state-of-the-art approaches.

Approach	FD001	FD002	FD003	FD004	Mean
Without uncertainty					
I-Transformer [30] (2024)	11.4	13.2	11.8	15.2	12.9
MambAtt [31] (2026)	11.5	13.2	11.6	13.3	12.4
Graph neural network [32] (2026)	11.2	13.0	11.5	13.3	12.3
MSMT-FDA [33] (2026)	11.9	15.6	11.2	17.8	14.1
With uncertainty					
Gaussian dropout - Transformer [34] (2026)	11.8	13.9	11.1	16.7	13.4
Gaussian dropout - Graph neural network [34] (2026)	11.7	15.7	11.3	16.5	13.8
Active learning [35] (2026)	10.7	14.7	11.0	15.1	12.9
Physics-guided [36] (2026)	11.5	n.a.	n.a.	12.9	n.a.
Bayesian transformer [37] (2024)	12.1	11.5	10.2	13.9	11.9
STAGNN-Dir [38] (2026)	11.9	13.4	11.7	15.3	13.1
Proposed approach	12.4	13.9	12.3	16.0	13.7

Table 3
Metrics for the RUL prognostics of the test engines of the four subsets of C-MAPSS. The RMSE and MAE are calculated with the mean RUL prediction, while the CRPS, the ECE and the confidence intervals are calculated with the probabilistic RUL prognostics.

Metric		FD001	FD002	FD003	FD004	All test instances
Overall metrics						
	RMSE (flights)	12.4	13.9	12.3	16.0	14.3
	MAE (flights)	9.4	10.3	9.6	11.7	10.6
	Mean CRPS (flights)	6.9	7.7	6.9	8.4	7.7
	ECE	0.036	0.029	0.032	0.022	0.020
Confidence intervals (C.I.)						
50% C.I.	Observed confidence	0.55	0.53	0.54	0.52	0.53
	Mean width (flights)	16.1	15.6	16.6	16.7	16.2
	Mean width (weeks)	1.6	1.6	1.7	1.7	1.6
90% C.I.	Observed confidence	0.91	0.86	0.92	0.84	0.87
	Mean width (flights)	39.1	36.7	40.2	39.7	38.6
	Mean width (weeks)	3.9	3.7	4.0	4.0	3.9
95% C.I.	Observed confidence	0.95	0.88	0.96	0.91	0.91
	Mean width (flights)	46.2	43.3	47.6	46.9	45.6
	Mean width (weeks)	4.6	4.3	4.8	4.7	4.6
99% C.I.	Observed confidence	0.97	0.97	0.98	0.95	0.96
	Mean width (flights)	59.7	56.2	61.9	60.7	59.1
	Mean width (weeks)	6.0	5.6	6.2	6.1	5.9

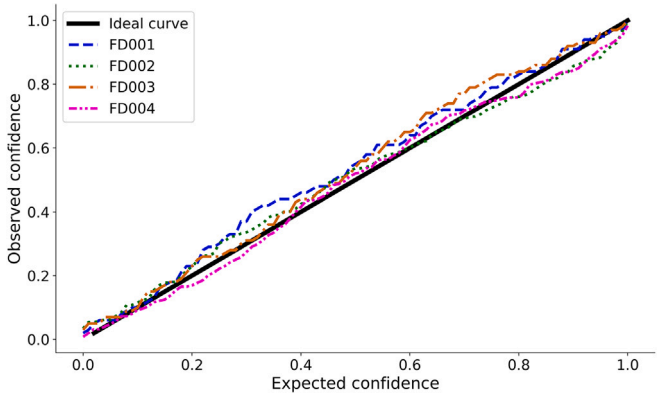


Fig. 7. A calibration plot of the probabilistic RUL prognostics for the test engines of the four subsets of C-MAPSS.

instead. Moreover, these confidence intervals are quite wide. For instance, for the 99% confidence interval, the mean width ranges from 5.6 (FD002) to 6.2 (FD003) weeks. However, the overall expected calibration error (ECE) equals only 0.020, showing that the predicted probabilistic RUL prognostics are well calibrated. The model therefore produces suitable RUL predictions for analyzing our maintenance planning approach.

5.2. Parameters of the maintenance schedule

An overview of the parameters of the maintenance scheduling model is provided in Table 4. To account for the long-term dependencies between maintenance, we construct a maintenance schedule for the next 12 weeks (3 months), from week 0 to week $T = 11$.

We estimate that the costs of maintaining the aircraft engine based on the RUL prognostic, without considering major overhauls or the replacement of Life Limited Parts (LLPs), equals one million euros, based on [41–43]. Based on [44], we assume that the downtime costs per hour are 20,000 euros, which equals 480,000 euros per day. We assume that if we experience a failure, we have one day of downtime, even if the aircraft is immediately maintained. An additional downtime period of 6 days is concurred if the aircraft engine remains failed for a week without maintenance. Last, the average lifetime of a training engine in the C-MAPSS dataset is 227 flights, while in reality, the time between engine maintenance is around 5000 to 12,000 flights [41,45]. We therefore scale all costs by a factor of 50 and round them.

With this, we estimate that the costs of maintaining an aircraft by replacing its engine are c_m = 20,000 euros, that a failure incurs a penalty of c_f = 10,000 euros, and that we have a downtime costs of c_d = 50,000 euros for each week we do not maintain the aircraft while its engine has failed. Last, we also consider a small penalty of c_i = 50 and c_k = 100 for inserting or canceling a maintenance task, respectively. We set $c_k > c_i$, to ensure that at most one maintenance task per aircraft is scheduled in the optimal solution. c_i and c_k are relatively

Table 4
Overview of the considered parameters.

Parameter	Description	Value
Problem parameters		
T	Last week of the maintenance schedule	11
δ^u	Duration of maintaining an aircraft (weeks)	2
U	Capacity (number of aircraft)	1
δ^s	Duration of externally repairing an engine (weeks)	3
S	Initial number of spare engines	2
Costs (euros)		
c_m	Cost of maintaining an aircraft	20,000
c_f	Penalty for failure	10,000
c_d	Downtime cost per week an aircraft remains failed without maintenance	50,000
c_i	Cost of inserting a new maintenance task in the schedule	50
c_k	Cost of canceling a scheduled maintenance task	100
Parameters of the solution method		
ϵ	Optimality tolerance	10^{-5}
M	Penalty for not maintaining an aircraft	10^6

small, since these are the only costs not scaled by the age of the engine at the maintenance moment.

We implement the multi-stage stochastic program in Python using Gurobi version 10, on a computer with 16 GB of RAM memory and 10 Intel i7 CPU cores (13th generation).

5.3. Benchmarks for the stochastic program

We evaluate the stochastic approach by comparing it with two benchmarks. The first benchmark assumes that we have perfect information, i.e., that we have a perfect RUL prognostic model that always predicts the true RUL without any errors or uncertainty. This perfect information benchmark is equivalent with the wait-and-see problem. Let z^{perfect} denote the optimal objective value of the perfect information benchmark, while z^{stoch} denotes the optimal objective value of the multi-stage stochastic program. We then define the Expected Value of Perfect Information (EVPI) as [24]:

$$\text{EVPI} = 100 \cdot \frac{z^{\text{stoch}} - z^{\text{perfect}}}{z^{\text{perfect}}}. \tag{27}$$

The EVPI constitutes the potential costs savings that could be achieved by further improving the RUL prognostic model.

The second benchmark is a non-stochastic version of our problem. Specifically, in the proposed approach we anticipate failures by explicitly optimizing the maintenance decisions that mitigate the consequences of a failure. The non-stochastic benchmark, in contrast, only optimizes the initial maintenance schedule and does not anticipate failures by also optimizing the maintenance decisions in case of a failure. The optimal objective value of this non-stochastic benchmark, denoted by $z^{\text{non-stoch}}$, provides an upper bound on the optimal objective value of the stochastic solution. We therefore define the Value of the Stochastic Solution (VSS) as [24]:

$$\text{VSS} = 100 \cdot \frac{z^{\text{non-stoch}} - z^{\text{stoch}}}{z^{\text{stoch}}}. \tag{28}$$

The VSS shows the expected cost savings by using the proposed approach instead of the non-stochastic benchmark.

This non-stochastic benchmark is similar to the approach employed for predictive maintenance scheduling for a fleet of aircraft engines with probabilistic RUL prognostics in [12,13]. Also in these studies, the expected costs of the predictive maintenance schedule are minimized using a linear program. Similar to our proposed non-stochastic benchmark, they optimize the initial maintenance schedule without jointly optimizing the potential future updates of this schedule in response to failures. In both [12,13], it is instead assumed that a failed engine can immediately be replaced. In contrast, we assume in both the stochastic approach and the non-stochastic benchmark that failed engines cannot be immediately replaced, as the maintenance scheduling constraints still have to be fulfilled.

5.4. Maintenance scheduling — One time window

In this section, we optimize the maintenance schedule for one time window only. This provides initial insights into the potential benefits and computational requirements of our approach.

5.4.1. Results — One time window

Table 5 shows the results when scheduling maintenance for an increasing number of aircraft within a single time window. We iteratively add an aircraft with an engine for which the mean RUL is smaller than T, to ensure that it is optimal to schedule maintenance for this aircraft within the considered time window. For these case studies, the VSS ranges from 2.6% for two aircraft to 0.8% for five aircraft, i.e., the expected costs decrease by 0.8 to 2.6 percent by solving the stochastic problem rather than the non-stochastic benchmark. The VSS is considerably lower when considering 4 or 5 aircraft. This might be because considering more aircraft gives less flexibility to schedule and subsequently reschedule maintenance, given that we do not increase the capacity or the number of spares. For instance, with the considered repair capacity, we can schedule maintenance for at most 6 aircraft within the considered planning window. When scheduling maintenance for 5 aircraft, there is thus a very limited flexibility to reschedule the maintenance of engines in response to failures. The EVPI ranges from 12.3% for a single aircraft to 10.2% for four aircraft, showing that the potential cost savings that could be realized by improving the RUL prognostic model are very high. However, it is unlikely that a model that predicts the exact failure time without any uncertainty or errors can be developed due to the aleatoric uncertainty in the degradation process and noisy data.

Fig. 8 shows the maintenance planning, as made at the root node, for both the stochastic approach and the non-stochastic benchmark, when considering 5 engines. In the stochastic approach, we on average use the engines for a longer time by planning to maintain them later. This is optimal since we can move the maintenance for an engine forward in case of a failure, by rescheduling the maintenance for the other engines. This longer usage reduces the expected costs.

5.4.2. Computational time — one time window

In this section, we compare the computational time of three different solution methods. The first method solves the large-scale deterministic equivalent (LDE) of the problem. The LDE is a single integer program in which all constraints and objective functions of all nodes are aggregated. The second method is nested Benders decomposition with the unicut and multicut strategy (see Section 4.3). Last, we consider the "Benders - Grouping" method, where we apply Benders decomposition, while we aggregate the integer programs of the nodes from the first five weeks together into a single large integer program [25]. As this destroys

Table 5
The results of the proposed maintenance scheduling model for one time window.

Number of aircraft	Selected test engines	Objective value			VSS (%)	EVPI (%)
		Stochastic z^{stoch}	Non-stochastic $z^{non\text{-}stoch}$	Perfect information $z^{perfect}$		
1	1-1	1594.6	1634.9	1419.5	2.5	12.3
2	1-1, 1-3	2891.9	2967.8	2582.5	2.6	12.0
3	1-1, 1-3, 1-4	4138.6	4235.7	3721.7	2.3	11.2
4	1-1, 1-3, 1-4, 1-5	5378.2	5424.7	4881.5	0.9	10.2
5	1-1, 1-3, 1-4, 1-5, 1-6	6873.7	6932.1	6194.1	0.8	11.0

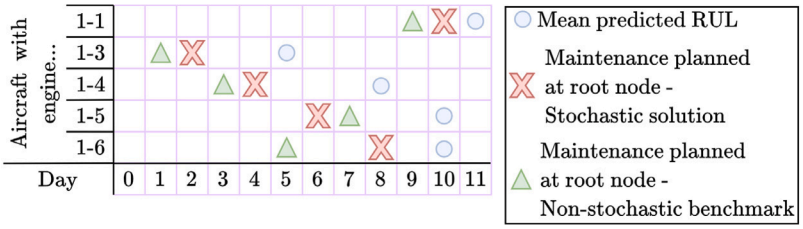


Fig. 8. The maintenance planning, as made at the root node, for the stochastic problem and the non-stochastic benchmark, when considering 5 engines.

Table 6
The number of nodes in the scenario tree and the computational time required for solving the stochastic program, for the cases in Table 5, to optimality. We stop solving the problem if the optimal solution has not yet be found after one hour, denoted by >60. The shortest time per instance is denoted in bold.

Number of aircraft	Number of nodes	Computational time (min)				
		LDE	Benders		Benders - Grouping	
			Unicut	Multicut	Unicut	Multicut
2	221	**0.0**	0.1	0.1	**0.0**	**0.0**
3	1743	0.5	0.9	1.1	**0.1**	0.2
4	9667	19.0	12.2	13.8	2.4	**2.1**
5	57 681	>60	>60	>60	**40.9**	55.3

the totally unimodularity, we impose that the decision variables of this first large integer program are integer, instead of considering the linear relaxation. Benders decomposition still works in this case, as it was originally developed for solving first-stage (master) integer programs [24].

Table 6 shows the size of the scenario tree and computational time required to solve the cases of Table 5 to optimality. As the number of aircraft increases, the number of nodes grows exponentially, leading to an exponential growth of the computational time as well. For the Benders strategy, there is no clear benefit between employing the unicut or the multicut strategy. Though the multicut strategy generally reduces the number of iterations necessary to solve the problem to optimality, each iteration takes a longer time due to the large number of cuts added to each node in each iteration.

In general, the Benders method with grouping the integer programs of the first five weeks together requires the least computational time. When considering four aircraft, this method required only 2.4 (unicut) to 2.1 (multicut) minutes to solve the problem to optimality. However, when considering five aircraft, it already takes 40.9 (unicut) to 55.3 (multicut) minutes to solve the problem to optimality. Moreover, both the LDE and the Benders strategy without grouping did not succeed in solving the problem with 5 aircraft to optimality within one hour.

Aircraft engines are constructed such that they rarely need maintenance [41]. Even for airlines with a fleet of several dozen aircraft, we thus expect that only few engines will fail within the considered planning window. For this specific application, the computational time is thus acceptable. However, for other applications, large airlines or when extending the scope of the problem, alternative solution methods will be necessary.

5.5. Long-term simulation of the maintenance schedule

In this section, we simulate the maintenance schedule over a period of 10 years using 200 Monte Carlo simulations for three aircraft. At the beginning of each week, we use the RUL prognostics of the engines to optimize the maintenance schedule. After each week, we employ the CNN to update the RUL prognostics based on the new sensor measurements, and subsequently update the maintenance schedule. Since the updates of the RUL prognostics are highly varying if the true RUL is large, we only schedule maintenance for an aircraft if the mean RUL prediction of its engine is less than or equal to 7 weeks. Once an engine is replaced, we randomly sample a new engine from the set of engines for its aircraft. We refer the interested reader to [12] for further information on Monte Carlo simulation in this context.

At the root node, i.e., at the current time in the simulation, we incorporate the maintenance decisions from the previous weeks as well. This requires the following changes to the stochastic program:

- For the root node, we update constraints (13), (15), (16) and (18) to incorporate the number of spares and capacity remaining after the maintenance tasks performed in the previous weeks.
- For the root node, we add the variables, constraints and costs that regulate the insertion and cancellation of maintenance tasks (Eqs. (4), (5), (9), and (10)) to the linear program as well. We do not consider these costs the first time we schedule maintenance for an aircraft.
- The costs of inserting and canceling maintenance tasks at the root node still are $c_k = 100$ and $c_i = 50$ respectively. However, we set the costs of inserting and canceling maintenance tasks at the other nodes at $c_k = 120$ and $c_i = 70$, to ensure that maintenance tasks are rescheduled at the root node rather than at later nodes in the tree.
- Within the linear program, we use a relatively small value for the cancellation and insertion costs as these costs are not scaled by the age of an engine. To calculate the total cost of the simulation, we instead use a cancellation cost of 1000 and an insertion cost of 500.

We compare the maintenance schedule using both the stochastic program, the perfect information benchmark and the non-stochastic benchmark. To ensure a fair comparison, we update the RUL prognostics and maintenance schedule each week for all three methods.

In Section 5.1, we present the RUL prognostics of the test engines from the C-MAPSS dataset. The measurements of these test engines stop at some point before failure, at which moment we predict the

Table 7
Results of 200 simulations for the proposed stochastic approach and the non-stochastic benchmark. Here, 95% C.I. denotes the 95% confidence interval. Δ is bold if the difference is significant at a level of 5%.

	Stochastic		Non-stochastic		Difference	
	Mean	95% C.I.	Mean	95% C.I.	Δ(%)	95% C.I. Δ(%)
Maintenance costs ($\cdot 10^6$)	1.563	[1.555, 1.570]	1.584	[1.576, 1.592]	**1.4**	[1.2, 1.5]
Number of maintenance tasks	75.9	[75.5, 76.2]	77.0	[76.6, 77.3]	**1.4**	[1.3, 1.6]
Number of inserted tasks	27.3	[26.4, 28.3]	27.2	[26.2, 28.2]	0.9	[−1.7, 3.5]
Number of canceled tasks	27.3	[26.4, 28.3]	27.2	[26.2, 28.2]	0.9	[−1.7, 3.5]
Age of engine at maintenance	18.50	[18.40, 18.60]	18.35	[18.25, 18.45]	**−0.8**	[−1.0, −0.5]
Number of failures	0.41	[0.32, 0.49]	0.33	[0.24, 0.41]	−19.5[a]	[−0.16, 0.00][b]
Number of days an engine remains failed	0.01	[0.00, 0.02]	0.02	[0.00, 0.04]	100[a]	[−0.00, 0.02][b]

[a] As we cannot calculate Δ_i for all $i \in \{1, 2, \dots, n\}$, Δ is instead calculated with the mean PIs.
[b] As we cannot calculate the confidence interval of Δ, this is instead the confidence interval of $\text{PI}^i_{\text{non-stoch}} - \text{PI}^i_{\text{stoch}}$.

RUL and plan maintenance in Section 5.4. However, to update the RUL prognostics over time, we need measurements until the failure of the engine, and we thus cannot use the test engines. We therefore follow [12] and select 80% of the engines of each C-MAPSS training set to train the CNN. We subsequently predict the RUL after each flight for the remaining 20% of the training engines. We use these 20% of the training engines to perform the simulation of the maintenance schedule. For more information, see [12].

5.5.1. Results — long-term simulation of the maintenance schedule

Table 7 shows the mean and the 95% confidence interval of the mean for several performance indicators (PIs) when simulating the maintenance schedule over a 10 year period for three aircraft. These PIs are given for both the stochastic method and the non-stochastic benchmark. Since we randomly select the same engines per simulation in the stochastic and non-stochastic simulation, our simulations are paired. We therefore calculate the mean difference Δ, in percent, between the PIs as follows:

$$\Delta_i = 100 \cdot \frac{\text{PI}^i_{\text{non-stoch}} - \text{PI}^i_{\text{stoch}}}{\text{PI}^i_{\text{stoch}}}, \tag{29}$$

$$\Delta = \frac{1}{n} \sum_{i=1}^{n} \Delta_i \tag{30}$$

where $\text{PI}^i_{\text{non-stoch}}$ and $\text{PI}^i_{\text{stoch}}$ denote the PI of simulation i of the non-stochastic benchmark and the stochastic approach, respectively. Δ_i denotes the difference, in percent, between these PIs of simulation i, and $n = 200$ denotes the number of simulations. With this, we calculate the 95% confidence interval (C.I.) of Δ with:

$$95\%\ \text{C.I. } \Delta = \Delta \pm 1.96 \frac{s_\Delta}{\sqrt{n}}, \tag{31}$$

where s_Δ denotes the sample standard deviation of Δ_i over all simulations $i \in \{1, 2, \dots, n\}$, while 1.96 is the critical value of the standard normal distribution for a two-sided 95% confidence interval. For the number of failures and the number of days an engine remains failed, we cannot calculate Δ_i for all simulations i, as $\text{PI}^i_{\text{stoch}}$ is zero for some simulations. We therefore instead calculate Δ as follows:

$$\Delta = 100 \cdot \frac{\overline{\text{PI}}_{\text{non-stoch}} - \overline{\text{PI}}_{\text{stoch}}}{\overline{\text{PI}}_{\text{stoch}}}, \tag{32}$$

where $\overline{\text{PI}}_{\text{non-stoch}}$ and $\overline{\text{PI}}_{\text{stoch}}$ denote the mean of the PI for the non-stochastic and stochastic solution, respectively. Moreover, we then provide the confidence interval of $\text{PI}^i_{\text{non-stoch}} - \text{PI}^i_{\text{stoch}}$. The difference between the mean of the stochastic and non-stochastic approach is statistically significant, at a level of 5%, if the confidence interval of Δ, or the confidence interval of $\text{PI}^i_{\text{non-stoch}} - \text{PI}^i_{\text{stoch}}$, does not contain 0.

In the solution of the proposed approach, engines are replaced at a significantly higher age than in the non-stochastic benchmark. This leads to a significant decrease of 1.4% in the number of maintenance tasks, which in turn leads to a significant reduction in the average maintenance costs by 1.4%, i.e., the VSS is 1.4% (see Table 7). The higher age at replacement also leads to a slightly higher number of failures in the stochastic version, though this difference is not significant. In aviation, aircraft maintenance, repair and overhaul (MRO) accounts for approximately 10% of an airline's operating costs, with on average 2.9 million US dollars spent per aircraft per year [46]. Around 40% of these costs stem from engine maintenance [46]. Therefore, even a 1.4% decrease in the engine maintenance costs still constitutes a considerable cost savings.

Table 8 shows the results with the perfect information benchmark, i.e., if we would have a prognostic model that predicts the RUL without any errors or uncertainty. We calculate Δ_i as:

$$\Delta_i = 100 \cdot \frac{\text{PI}^i_{\text{stoch}} - \text{PI}^i_{\text{perfect}}}{\text{PI}^i_{\text{perfect}}}, \tag{33}$$

where $\text{PI}^i_{\text{perfect}}$ denotes the PI of simulation i for the perfect information benchmark. We subsequently calculate the confidence interval of Δ as in Eq. (31). With perfect RUL prognostics, the engines would on average be replaced at a significantly older age, which leads to a significant reduction in the number of maintenance tasks. Compared to the perfect information benchmark, the number of maintenance tasks increases with 8.0% in the stochastic solution. Moreover, due to the absence of uncertainty, the number of inserted and canceled tasks and the number of failures in the perfect information benchmark is significantly lower as well. This leads to a significant difference of 10.6% in the maintenance costs. The potential costs savings that could be achieved by improving the RUL prognostic model are thus very high.

6. Sensitivity analysis

Both maintenance costs and problem parameters, such as the maintenance duration, are difficult to estimate. In this Section, we therefore perform a sensitivity analysis with respect to the main costs parameters (Section 6.1) and the main problem parameters (Section 6.2). As before, we simulate the maintenance schedule for a period of 10 years for three aircraft using 200 Monte Carlo simulations for each set of parameters. Moreover, for each set of parameters, we compute the VSS, i.e., the difference in costs between the proposed approach and the non-stochastic benchmark. In addition, we also compute the EVPI, i.e., the difference in costs between the proposed approach and the perfect information benchmark.

6.1. Sensitivity analysis — maintenance costs

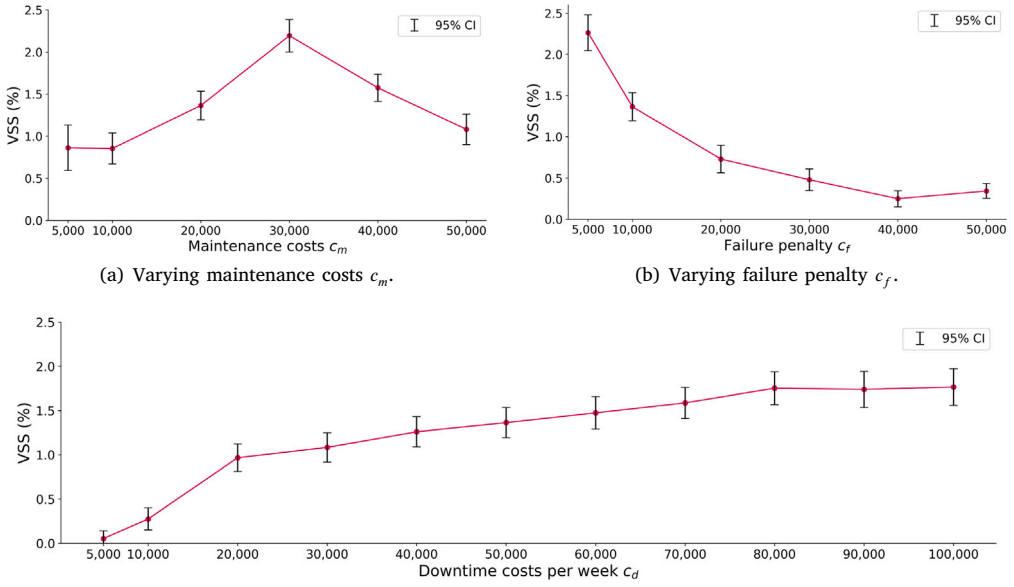
In this Section, we perform a sensitivity analysis with respect to the maintenance costs. Fig. 9 shows the VSS if we vary one cost parameter (either c_m, c_f, or c_d) at the time. The results show that although the exact VSS depends on the considered costs parameters, our proposed approach does outperform the non-stochastic benchmark for all considered costs. Fig. 10 shows the corresponding EVPI.

Fig. 9(a) shows the VSS when changing the costs c_m of maintaining an aircraft. The VSS is highest, at 2.2%, when the maintenance costs

Table 8
Results of 200 simulations for the stochastic approach and the perfect information benchmark. Here, 95% C.I. denotes the 95% confidence interval. Δ is bold if the difference is significant at a level of 5%.

	Stochastic		Perfect information		Difference	
	Mean	95% C.I.	Mean	95% C.I.	Δ(%)	95% C.I. Δ(%)
Maintenance costs ($\cdot 10^6$)	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**10.6**	[10.4, 10.9]
Number of maintenance tasks	75.9	[75.5, 76.2]	70.3	[69.9, 70.6]	**8.0**	[7.7, 8.2]
Number of inserted tasks	27.3	[26.4, 28.3]	3.1	[2.8, 3.4]	**780.6**[a]	[23.2, 25.2][b]
Number of canceled tasks	27.3	[26.4, 28.3]	3.1	[2.8, 3.4]	**780.6**[a]	[23.2, 25.2][b]
Age of engine at maintenance	18.50	[18.40, 18.60]	19.37	[19.25, 19.48]	**−4.4**	[−4.8, −4.1]
Number of failures	0.41	[0.32, 0.49]	0.19	[0.13, 0.24]	**115.8**[a]	[0.12, 0.32][b]
Number of days an engine remains failed	0.01	[0.00, 0.02]	0.01	[0.00, 0.02]	0.0[a]	[−0.01, 0.01][b]

[a] As we cannot calculate Δ_i for all $i \in \{1, 2, \dots, n\}$, Δ is instead calculated with the mean PIs.
[b] As we cannot calculate the confidence interval of Δ, this is instead the confidence interval of $\text{PI}^i_{\text{stoch}} - \text{PI}^i_{\text{perfect}}$.



(a) Varying maintenance costs c_m.

(b) Varying failure penalty c_f.

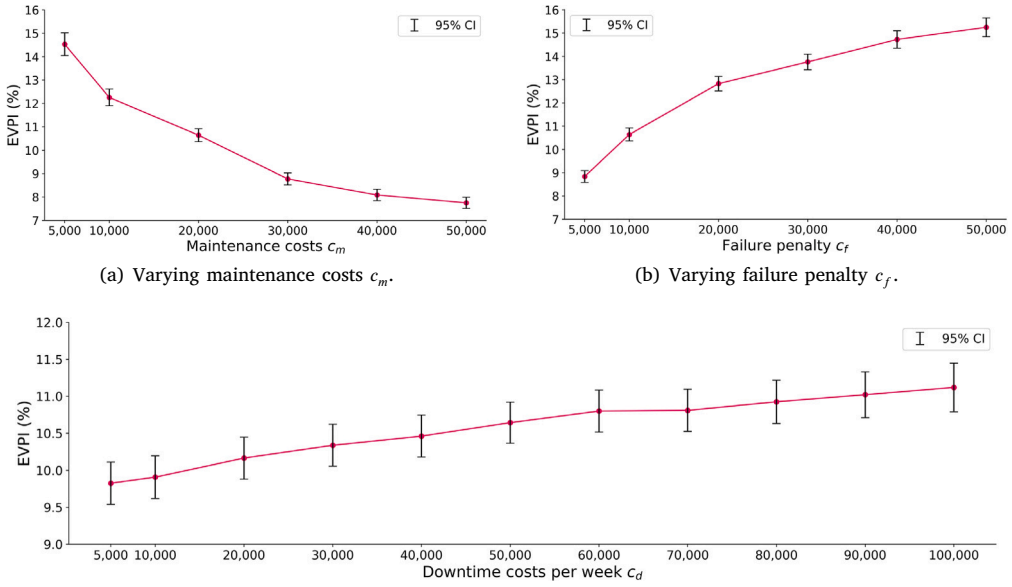
(c) Varying downtime costs per week c_d.

Fig. 9. The VSS when one of the cost parameters is varied. The remaining (cost) parameters are not changed, and are given in Table 4. The vertical black bars denote the 95% confidence interval of the mean VSS.

equal $c_m = 30{,}000$ euros. If the maintenance costs decrease below 30,000 euros, engines are replaced earlier in both the stochastic and the non-stochastic solution, i.e., engines are replaced at a younger age. This is because replacing an engine is relatively cheap. Consequently, the difference between the average age of the engine at maintenance of the stochastic and non-stochastic solution becomes smaller as well. Therefore, the difference in the number of maintenance tasks, and thus the VSS, decreases. Vice versa, if the maintenance costs increase above 30,000 euros, the difference in the number of maintenance tasks increases. *In absolute numbers,* the costs savings with the stochastic solution therefore still increase when the maintenance costs increase above 30,000 euros. The VSS *in percent*, however, decreases. This is because the total maintenance costs of the stochastic approach, which are the denominator in the calculation of Δ_i (see Eq. (29)), increase. In contrast, the EVPI (see Fig. 10(a)) consistently decreases as c_m increases, as most costs stem directly from performing maintenance tasks.

Fig. 9(b) shows the VSS, and Fig. 10(b) shows the EVPI, when changing the failure penalty c_f. Table 9 additionally shows the main PIs for a changing failure penalty. If the failure penalty increases, engines are replaced earlier in both the stochastic and the non-stochastic solution to decrease the number of failures. This gives, on average, a younger age of the engine at maintenance (see Table 9). Moreover, the difference between the average age of the engine at maintenance of the stochastic and non-stochastic solution decreases as well. This is because both methods aim to avoid failures if the failure penalty is very high, and thus replace engines based on the predicted lower bound of the RUL. Consequently, the difference in the number of maintenance tasks decreases as well (see Table 9), and therefore, the VSS decreases. In contrast, the EVPI increases when the failure penalty increases. This is because the difference in the number of maintenance tasks between the stochastic method and the perfect information benchmark further increases, as engines are replaced at a younger age in the stochastic approach.

Last, Fig. 9(c) shows the VSS, and Fig. 10(c) the EVPI, if the costs c_d of remaining failed for one week without maintenance varies. Both the VSS and the EVPI consistently increase with an increasing c_d, though this increase is not always significant. The highest VSS of 1.8% is reached when $c_d = 100{,}000$. This is as expected: Since the non-stochastic benchmark does not consider rescheduling maintenance in response to failures, maintenance is usually scheduled early, i.e., when the engine is still young, to avoid the high costs of remaining failed. This leads to an increase in the number of maintenance tasks for the non-stochastic benchmark when c_d increases. The number of maintenance tasks for the stochastic solution increases as well, but less. This subsequently gives an increase in the VSS if c_d increases.



(a) Varying maintenance costs c_m.

(b) Varying failure penalty c_f.

(c) Varying downtime costs per week c_d.

Fig. 10. The EVPI when one of the cost parameters is varied. The remaining (cost) parameters are not changed, and are given in Table 4. The vertical black bars denote the 95% confidence interval of the mean EVPI.

Table 9
Results of 200 simulations for the proposed stochastic approach and the non-stochastic benchmark, for a varying failure penalty c_f. The results are bold if the difference between the stochastic approach and the non-stochastic benchmark is significant at a level of 5%.

		Failure penalty c_f (·1000)					
		5	10	20	30	40	50
Maintenance costs (·10^6)	Stochastic	**1.537**	**1.563**	**1.595**	**1.609**	**1.625**	**1.633**
	Non-stochastic	**1.572**	**1.584**	**1.606**	**1.617**	**1.629**	**1.639**
Number of maintenance tasks	Stochastic	**74.2**	**75.9**	**77.4**	**78.2**	**78.9**	**79.3**
	Non-stochastic	**76.4**	**77.0**	**77.9**	**78.5**	**79.0**	**79.4**
Age of engine at maintenance	Stochastic	**18.70**	**18.50**	**18.28**	**18.16**	18.08	18.04
	Non-stochastic	**18.45**	**18.35**	**18.21**	**18.12**	18.07	18.03
Number of failures	Stochastic	**4.05**	0.41	0.19	0.10	0.05	0.05
	Non-stochastic	**0.45**	0.33	0.16	0.10	0.05	0.04

6.2. Sensitivity analysis — model problem parameters

In this section, we analyze the sensitivity of the results when varying the main problem parameters, specifically the capacity U, maintenance duration δ^u, the number of spare engines S and the external repair duration δ^s. Table 10 shows the VSS and EVPI when we vary one problem parameter at the time. The results show that the benefits of the proposed approach, compared to the non-stochastic benchmark, depends on the problem parameters. Our proposed approach, however, outperforms the non-stochastic benchmark for all considered parameter values.

The VSS and the EVPI increase if there is more flexibility to reschedule maintenance in response to failures (see also Section 5.4.1). For instance, if the capacity U and the number of spare engines S increase, or if the maintenance duration δ^u decreases, there are more possibilities to reschedule maintenance in case of a failure. This leads to an increase of both the VSS and the EVPI. This indicates that it could be beneficial for airlines to invest in increased flexibility, for instance by increasing the number of spare engines or the capacity.

The VSS and EVPI, however, are constant for varying durations of externally repairing an engine δ^s. This indicates that the maintenance planning is mainly constrained by the limited capacity U, the maintenance duration δ^u and the initial number of spare engines S, and only to a lesser extent by δ^s. Similarly, the VSS and EVPI do not change if we increase the capacity U from 2 to 3, or if we increase the number of spare engines S from 2 to 3. Also in this case, this indicates that the maintenance planning is mainly constrained by the other problem parameters.

7. Conclusion

In this paper, we address three key challenges in predictive maintenance scheduling. First, we optimize the maintenance schedule using probabilistic, rather than deterministic, RUL prognostics. Second, we schedule maintenance for multiple systems with a limited maintenance capacity and a limited stock of spare parts. Last, failed systems cannot always be immediately maintained due to the limited resources: A spare part must be available, and other scheduled maintenance tasks may need to be postponed to free up capacity. We address these three challenges by formulating a multi-stage stochastic program that jointly optimizes the initial maintenance schedule and its future updates in response to system failures. The constraint matrix of this stochastic program is totally unimodular. We can therefore efficiently solve the stochastic program using nested Benders decomposition.

We apply our method to a case study with aircraft engines, where we plan maintenance for up to five aircraft for a period of 12 weeks (12 stages). With a rolling horizon approach, we simulate the maintenance schedule for a period of 10 years for three aircraft. We compare the

Table 10
The mean expected maintenance costs, the VSS and the EVPI, including the 95% confidence interval (95% C.I.), when one of the problem parameters is varied. The remaining parameters are not changed, and are given in Table 4. The VSS and EVPI are bold if the difference is significant at a level of 5%.

Value	Maintenance costs ($\cdot 10^6$)						Difference			
	Non-stochastic		Stochastic		Perfect information		VSS	95% C.I	EVPI	95% C.I
	Mean	95% C.I.	Mean	95% C.I.	Mean	95% C.I.				
Capacity U										
1	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]
2	1.573	[1.566, 1.581]	1.545	[1.538, 1.552]	1.395	[1.388, 1.401]	**1.9**	[1.7, 2.1]	**10.8**	[10.5, 11.0]
3	1.573	[1.566, 1.581]	1.545	[1.538, 1.552]	1.395	[1.388, 1.401]	**1.9**	[1.7, 2.1]	**10.8**	[10.5, 11.0]
Maintenance duration δ^u										
1	1.576	[1.568, 1.583]	1.546	[1.539, 1.553]	1.397	[1.391, 1.404]	**1.9**	[1.7, 2.1]	**10.7**	[10.4, 10.9]
2	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]
3	1.625	[1.615, 1.635]	1.618	[1.607, 1.629]	1.520	[1.505, 1.535]	**0.5**	[0.1, 0.9]	**6.7**	[5.9, 7.4]
Spare engines S										
1	1.625	[1.615, 1.635]	1.609	[1.599, 1.620]	1.491	[1.478, 1.504]	**1.0**	[0.6, 1.4]	**8.1**	[7.5, 8.7]
2	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]
3	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]
External repair duration δ^s										
1	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]
2	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]
3	1.584	[1.576, 1.592]	1.563	[1.555, 1.570]	1.413	[1.406, 1.420]	**1.4**	[1.2, 1.5]	**10.6**	[10.4, 10.9]

results of this simulation with a non-stochastic benchmark and a perfect information benchmark. Using our proposed approach instead of the non-stochastic benchmark significantly lowers the average costs by 1.4%. The perfect information benchmark shows that the maintenance costs can be reduced by up to 10.6% if the RUL prognostic model is further improved. Last, the sensitivity analysis shows that our approach outperforms the non-stochastic benchmark for all considered sets of parameter values.

The main disadvantage of this approach is that the computational time exponentially increases with the number of considered systems, which limits its applicability to small-scale problems. For future work, we therefore aim to decrease the computational time of the stochastic approach by developing an alternative method to solve the predictive maintenance scheduling problem, for instance by employing reinforcement learning or heuristics. At the same time, we plan to extend the scope of the problem, for instance by considering multiple components and systems and by integrating other operational activities in the maintenance scheduling as well. We aim to analyze such an extended approach on a different, large, preferable real-world, dataset.

Last, we aim to extend the uncertainty considered in this paper. First, we aim to incorporate various sources of uncertainty, such as uncertainty in the available workforce. Moreover, we aim to use a more advanced RUL prognostic model with more advanced uncertainty quantification methods, such as methods that distinguish between aleatoric and epistemic uncertainty, and develop an approach that can handle different descriptions of the uncertainty, such as a confidence interval of the RUL.

CRediT authorship contribution statement

Ingeborg de Pater: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Mihaela Mitici:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Funding acquisition, Conceptualization.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Totally unimodular constraint matrix and integer right-hand side

A.1. Totally unimodular constraint matrix with the consecutive ones property

In this appendix, we show that the constraint matrix at any node $n \in N$ is totally unimodular using the consecutive ones property [21]. A matrix fulfills the consecutive ones property if (i) all entries are either 0 or +1 and (ii) for each row, the ones appear consecutively. Moreover, when adding a unit vector as row/column to a totally unimodular matrix, and when multiplying a row/column in a totally unimodular matrix with minus one, the matrix remains totally unimodular [20].

Fig. A.11 shows a schematic overview of the constraint matrix for any node $n \in N \setminus \{0\}$. We divide this constraint matrix in four blocks:

- **Block 1:** Block 1 in Fig. A.11 contains the coefficients belonging to the x^n_{et} variables, and to all constraints except the canceling constraints (Eq. (10)), the constraints with the bounds of the variables (Eqs. (19), (20)) and the constraint with the definition of m^n_e (Eq. (11)). The first column contains the coefficients of $x^n_{1t^n}$, where 1 denotes the first considered machine, until the $|E|$th column contains the coefficients of $x^n_{|E|t^n}$, where $|E|$ denotes the last machine. We then continue with the coefficients of $x^n_{1(t^n+1)}$ in the next column, until the last column of block 1 contains the coefficients of $x^n_{|E|(t^0+T)}$ (see Fig. A.11). With this ordering of the variables, the matrix in block 1 fulfills the consecutive ones property [21], and is thus totally unimodular:
  - In constraints (15) and (18), we iteratively sum the x^n_{et} variables over all systems and over a subset of consecutive weeks. In constraints (13) and (16), we sum the $x^n_{et^n}$ variables over all systems. Due to the ordering of the variables, the ones in the rows of these constraints appear consecutively.

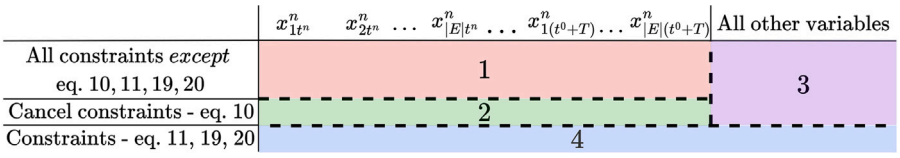


Fig. A.11. Overview of the constraint matrix for any node $n \in N \setminus \{0\}$.

– Constraints Eqs. (9) and (12) contain only one x^n_{et} variable per constraint, each time with a coefficient of one. Constraints Eqs. (11), (14) and (17) do not contain the variables x^n_{et}. The rows of these constraints in block 1 thus contain a single one, and only zeros beside, or only zeros.

- **Block 2:** The rows in block 2 contain the coefficients for the x^n_{et} variables of the canceling constraints (Eq. (10)). Each row, with the coefficients of the x^n_{et} variables, of a canceling constraint contains a single -1, and only zeros beside. The matrix in block 1 and 2 is thus totally unimodular.
- **Block 3:** Block 3 contains the columns with the coefficients belonging to all other variables. There is no specific ordering of these variables. Each variable, except the x^n_{et} variables, is present in exactly one constraint, with a coefficient of either -1 or 1. The column with the coefficients belonging to any variable (except the x^n_{et} variables) thus contains a single -1 or 1 element, and only zeroes beside. The matrix in block 1, 2 and 3 is thus totally unimodular.
- **Block 4:** Block 4 contains the constraint regarding the bounds of the variables (Eqs. (19) and (20)) and the definition of m^n_e (Eq. (11)). These rows contain a single $+1$ element.

Our constraint matrix is thus totally unimodular for any node $n \in N \setminus \{0\}$. The constraint matrix of the root node is a submatrix of the constraint matrix in Fig. A.11, without the cancellation and inserting variables, and without constraints (9) and (10). The constraint matrix at the root node is thus also totally unimodular.

A.2. Integer right-hand side with induction

We now show by induction that in the optimal solution (found with the simplex method), the right-hand side of the integer program at each node $n \in N$ is integer.

Base step for the root node. The right-hand side of the linear program of the root node consists of 0 (Eq. (11)), 1 (Eq. (12)), S, (Eqs. (13), (14), and (15)), and U (Eqs. (16), (17) and (18)). The right-hand side of the linear program of the root node is thus integer, while the constraint matrix is totally unimodular. The optimal solution (when using the simplex method) of the relaxation of the integer program is thus integer.

Induction step. Assume that for any time $t' \leq t$, with $t \in [t^0, t^0+1, \ldots, t^0+T-1]$, the right-hand side of the integer program at any node $n \in N_{t'}$ is integer. The optimal solution of such a node n is than also integer when found by the simplex method.

Consider a node $l \in N$ at a time $t^l = t+1$. The right-hand side of the linear program consists of integers (1, S and U), together with a linear combination (with coefficients of -1 and 1 only) of the variables of the ancestor node. By the induction step, the optimal solution of the ancestor node is integer. This means that the right-hand side of the linear program of node l is integer, and that the optimal solution, when found by the simplex method, is integer as well.

Data availability

The data is publicly available at https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/.

References

[1] Fink O, Wang Q, Svensen M, Dersin P, Lee W-J, Ducoffe M. Potential, challenges and future directions for deep learning in prognostics and health management applications. Eng Appl Artif Intell 2020;92:103678.
[2] Lei Y, Li N, Guo L, Li N, Yan T, Lin J. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. Mech Syst Signal Process 2018;104:799–834.
[3] Wang L, Zhu Z, Zhao X. Dynamic predictive maintenance strategy for system Remaining Useful Life prediction via deep learning ensemble method. Reliab Eng Syst Saf 2024;245:110012.
[4] Zhu J, Li B, Zhu Z, Zhao X. Dynamic predictive maintenance strategy for the multi-state system based on remaining life prediction. Reliab Eng Syst Saf 2025;111289.
[5] Chen C, Zhu ZH, Shi J, Lu N, Jiang B. Dynamic predictive maintenance scheduling using deep learning ensemble for system health prognostics. IEEE Sensors J 2021;21(23):26878–91.
[6] Wang L, Zhao X, Pham H. Novel formulations and metaheuristic algorithms for predictive maintenance of aircraft engines with Remaining Useful Life prediction. Reliab Eng Syst Saf 2025;261:111064.
[7] de Pater I, Reijns A, Mitici M. Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. Reliab Eng Syst Saf 2022;221:108341.
[8] Lee J, Mitici M. Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics. Reliab Eng Syst Saf 2023;230:108908.
[9] Dong E, Zhan X, Yan H, Tan S, Bai Y, Wang R, Cheng Z. A data-driven intelligent predictive maintenance decision framework for mechanical systems integrating transformer and kernel density estimation. Comput Ind Eng 2025;201:110868.
[10] Zhuang L, Xu A, Wang X-L. A prognostic driven predictive maintenance framework based on Bayesian deep learning. Reliab Eng Syst Saf 2023;234:109181.
[11] Shoorkand HD, Nourelfath M, Hajji A. A hybrid CNN-LSTM model for joint optimization of production and imperfect predictive maintenance planning. Reliab Eng Syst Saf 2024;241:109707.
[12] Mitici M, de Pater I, Barros A, Zeng Z. Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. Reliab Eng Syst Saf 2023;234:109199.
[13] Zeng J, Liang Z. A dynamic predictive maintenance approach using probabilistic deep learning for a fleet of multi-component systems. Reliab Eng Syst Saf 2023;109456.
[14] de Pater I, Mitici M. Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. Reliab Eng Syst Saf 2021;214:107761.
[15] Ding C, Qiao F, Wang D, Liu J. Adaptive real-time scheduling for production and maintenance: Integrating RUL prediction with multi-agent deep reinforcement learning. Reliab Eng Syst Saf 2025;111394.
[16] Tseremoglou I, Santos BF. Condition-based maintenance scheduling of an aircraft fleet under partial observability: A deep reinforcement learning approach. Reliab Eng Syst Saf 2024;241:109582.
[17] Zhu Z, Xiang Y. Condition-based maintenance for multi-component systems: Modeling, structural properties, and algorithms. IISE Trans 2021;53(1):88–100.
[18] Leo E, Engell S. Condition-based maintenance optimization via stochastic programming with endogenous uncertainty. Comput Chem Eng 2022;156:107550.
[19] Leo E, Engell S. Handling Type-I and Type-II endogenous uncertainties in simultaneous production planning and condition-based maintenance optimization in continuous production. Comput Chem Eng 2023;174:108227.
[20] Wolsey LA, Nemhauser GL. Integer and combinatorial optimization. vol. 1, John Wiley & Sons; 1988.
[21] Fulkerson D, Gross O. Incidence matrices and interval graphs. Pacific J Math 1965;15(3):835–55.

[22] Goel V, Grossmann IE. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. Comput Chem Eng 2004;28(8):1409–29.
[23] Gupta V, Grossmann IE. Solution strategies for multistage stochastic programming with endogenous uncertainties. Comput Chem Eng 2011;35(11):2235–47.
[24] Haneveld WKK, Van der Vlerk MH, Romeijnders W. Stochastic programming: Modeling decision problems under uncertainty. Springer Nature; 2019.
[25] Murphy J. Benders, nested benders and stochastic programming: An intuitive introduction. 2013, arXiv preprint arXiv:1312.3158.
[26] Kong N, Schaefer AJ, Ahmed S. Totally unimodular stochastic programs. Math Program 2013;138(1–2):1–13.
[27] Birge JR, Louveaux FV. A multicut algorithm for two-stage stochastic linear programs. European J Oper Res 1988;34(3):384–92.
[28] Saxena A, Goebel K. Turbofan engine degradation simulation data set. 2008, p. 878–87, NASA Ames Prognostics Data Repository.
[29] Saxena A, Goebel K, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 international conference on prognostics and health management. IEEE; 2008, p. 1–9.
[30] Zhou Z, Long Z, Wang R, Bai M, Liu J, Yu D. An aircraft engine Remaining Useful Life prediction method based on predictive vector angle minimization and feature fusion gate improved transformer model. J Manuf Syst 2024;76:567–84.
[31] Han J, Kwon Y, Yoon H. Mamba-attention: A self-supervised framework for efficient Remaining Useful Life prediction. Reliab Eng Syst Saf 2025;111492.
[32] Ren Y, Meng Z, Li J. Spatial-temporal graph hybrid neural network for Remaining Useful Life prediction of aero-engines. Reliab Eng Syst Saf 2026;112433.
[33] Qin M, Xu G, Chen X, Li W. Multi-sensor spatial and multiscale temporal transformer guided by frequency-domain attention for aeroengine Remaining Useful Life prediction. Reliab Eng Syst Saf 2026;112535.
[34] Xu W, Gao S, Wang Z, Zio E. A novel uncertainty-aware framework for Remaining Useful Life prediction integrating uncertainty quantification and calibration. Reliab Eng Syst Saf 2026;112763.
[35] Chen Y, Xiao S, Zhao X. Active learning with dynamic exploitation-exploration balance for aircraft engine Remaining Useful Life prediction. Reliab Eng Syst Saf 2026;112612.
[36] Tian L, Gao Y, Li X, Gao L. Physics-guided neural process with adaptive learning for uncertainty quantification of aero-engine Remaining Useful Life. Comput Ind Eng 2026;111877.
[37] Xiang F, Zhang Y, Zhang S, Wang Z, Qiu L, Choi J-H. Bayesian gated-transformer model for risk-aware prediction of aero-engine Remaining Useful Life. Expert Syst Appl 2024;238:121859.
[38] He J, Hu J, Wang H. Spatio-temporal adaptive graph neural network for Remaining Useful Life prediction with uncertainty estimation. Neurocomputing 2026;133739.
[39] Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: International conference on machine learning. PMLR; 2016, p. 1050–9.
[40] de Pater I, Mitici M. Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. In: PHM society European conference. vol. 7, 2022, p. 96–109.
[41] Ackert S. Engine maintenance concepts for financiers. Aircr Monit 2011;2:1–43.
[42] Justin CY, Mavris DN. Aircraft and engine economic evaluation for fleet renewal decision-making and maintenance contract valuation. Proc Inst Mech Eng G 2015;229(11):2051–65.
[43] Cai J, Li X, Chen X. Optimization of aeroengine shop visit decisions based on Remaining Useful Life and stochastic repair time. Math Probl Eng 2016;2016(1):5648360.
[44] Alomar I, Nikita D. Managing operational efficiency and reducing aircraft downtime by optimization of aircraft on-ground (AOG) processes for air operator. Appl Sci 2025;15(9):5129.
[45] Stojiljković B, Vasov L, Čokorilo O. Turbofan engine shop visit management. In: The seventh international conference transport and logistics. 2019, p. 121–6.
[46] Markou C, Cros G. Airline maintenance cost executive commentary FY2022 data Public Version. Maintenance Cost Technical Group. IATA; 2024.