Reinforcement learning for maintenance decision-making of multi-state component systems with imperfect maintenance

Van-Thai NGUYEN, Phuc DO, Alexandre VOISIN and Benoit IUNG

Université de Lorraine, CRAN, UMR CNRS 7039, Campus Sciences, BP 70239, Vandoeuvre-les-Nancy, 54506, France. E-mail: van-thai.nguyen, phuc.do, alexandre.voisin, benoit.iung@univ-lorraine.fr

In this paper we propose an artificial intelligence (AI) based framework for maintenance decision-making and optimization of multi-state component systems with imperfect maintenance. Our proposed framework consists of two main phases. The first aims at constructing artificial neural network (ANN) based predictors to forecast system's reliability and maintenance cost. The second refers to the use of deep reinforcement learning (DRL) algorithms to optimize maintenance policy which can deal with large scale applications. Numerical results show that ANN is suitable to reliability, maintenance cost forecasting and DRL is a potentially powerful tool for maintenance decision-making and optimization.

Keywords: Deep reinforcement learning, maintenance decision-making, multi-state component system, imperfect maintenance, optimization.

1. Introduction

Condition-based maintenance (CBM) refers to maintenance policy that makes decisions based on system's health information. Thanks to recent advances in sensing technology, CBM nowadays becomes sophisticated in maintenance planning for industrial system, especially for multi-state component system (MSCS) which is defined as the multi-component system where the deteriorating process of each component can be discretized into a finite set of states ranging from "as good as new" to failure [Karabağ et al. (2020); Do et al. (2019); Yousefi et al. (2020)].

CBM scheduling for MSCS is a challenging problem due to the dependencies between components in terms of economy, stochasticity and structure [Nicolai and Dekker (2008)]. Among these dependencies, economic dependence, whereby the joint maintenance of several components is cheaper (positive dependence) or more expensive (negative dependence), is usually considered to allow opportunistic and group maintenance to be carried out to reduce total cost.

In addition, scaling algorithms to deal with large scale system is another issue for maintenance decision-making (MDM) of MSCS. However, the combination of reinforcement learning (RL) and deep learning creating a new field called DRL seem to the answer for this problem. Particularly, Zhang and Si (2020) minimized maintenance cost for multi-component system with dependent competing risks. DRL is also employed to optimize structure maintenance policy for the long-span cable-stayed bridge with 263 components in [Wei et al. (2020)]. More recently, preventive maintenance policy for general serial production line with intermediate buffers using DRL has been studied in [Huang et al. (2020)] and the learned policy surprisingly showed that opportunistic and group maintenance are occasionally conducted.

In DRL, one of the most important things is how to properly construct an environment with which an agent can interact to find the best policy. However, almost all papers using DRL for MDM optimization assume that some parts of the environment such as maintenance cost model, system structure to be known at hand making them less practical. Therefore, we propose in this paper an AI based framework that can learn the system's characteristics from historical maintenance data and can tackle with large scale MDM problem. The proposed framework consists of two primary phases. The first aims at creating ANN based predictors to forecast reliability and maintenance cost of the system. The second refers to the use DRL algorithms to find optimal maintenance policy.

The rest of the paper is organized as following. Section 2 is devoted to the general description of the system, collected data and problem statement formulation. ANN based predictors for system's reliability, maintenance cost and DRL based maintenance policy optimization process are presented in section 3. The simulation results of proposed framework are depicted and analyzed in section 4. Finally, the last section presents the conclusion and the future work.

Proceedings of the 31st European Safety and Reliability Conference
Edited by Bruno Castanier, Marko Cepin, David Bigaud, and Christophe Berenguer
Copyright © ESREL 2021.*Published by* Research Publishing, Singapore.
ISBN: 978-981-18-2016-8; doi:10.3850/978-981-18-2016-8_304-cd

2. Assumptions and system description

2.1. *Multi-state component systems and its maintenance operations*

We consider a complex system being composed of N non-identical components which are redundantly constructed and suffer from stochastic deterioration processes that are independent to each other. The system is monitored periodically at time $T_k = k.\Delta_T$ $(k = 0, 1, 2, ...)$ where Δ_T is inter-inspection time [Alaswad and Xiang (2017)]. In addition, each component has $m_i + 1$ discrete health condition states including new state, $m_i - 1$ degraded states and failure state. By denoting x_k^i as the state of i^{th} component at time T_k, the component's state can be expressed as:

$$x_k^i = \begin{cases} 0, & \text{if component } i \text{ is new} \\ j, & \text{if component } i \text{ is in degraded state } j \\ m_i, & \text{if component } i \text{ fails} \end{cases} \tag{1}$$

Furthermore, the degradation between two successive inspection times of a component is assumed to obey a discrete Markov chain, however, whose transition matrix is unknown.

Both corrective maintenance (CM) and preventive maintenance (PM) strategy are considered for all components of MSCS. In addition, it is supposed that each component can be individually repaired and maintenance action is only executed at inspection time. Besides, it is also assumed that both perfect and imperfect maintenance are performed for CM and PM. Specifically, perfect maintenance restores the component to new state with the cost of c^r. Conversely, imperfect maintenance originated from several causes such as the lack of spare parts, human resources [Pham and Wang (1996)] implies that the component's state after maintenance is somewhere between the state before maintenance and ”as good as new” [Do and Bérenguer (2012)]. The cost of imperfect maintenance is denoted as c^{ip} which is smaller than c^r.

It should be noted that maintenance of MSCS in reality usually benefits from single or multiple setup costs which can be saved if a group of components is maintained simultaneously due to the economic dependence between them [Nicolai and Dekker (2008)]. In addition, the system downtime caused by the failure of critical components is also an opportunity for the maintenance of other components.

2.2. *Data description and maintenance optimization problem*

In this work, we assume that the historical maintenance operation for aforementioned MSCS is collected in a dataset which has the form $\{s_k, s'_k, c_k^m, h_k\}$ [Do et al. (2019)] described as belows.

- $s_k = \left[x_k^1 \; x_k^2 \; x_k^3 \; \dots \; x_k^N\right]$ is the system's state before maintenance at time T_k.
- $s'_k = \left[x_k^{1'} \; x_k^{2'} \; x_k^{3'} \; \dots \; x_k^{N'}\right]$ is the system's state after maintenance at time T_k. It should be noted that the maintenance duration is small and then can be neglected.
- c_k^m is the system's maintenance cost at time T_k.
- h_k is the system's reliability after maintenance up to time step T_{k+1}.

In reality, this dataset is evidently not complete due to two main reasons. Firstly, several states are rare or just appeared in some specific conditions making them difficult to be recorded. Secondly, it is impossible to have the dataset consisting of all cases of the pair $\{s_k, s'_k\}$ because the number of such pairs increases exponentially when the number of components as well as the number of states per component increases as depicted in table 1. In particular, for the system described in previous section, the complete dataset has a total of $\prod_{i=1}^{N} \frac{(m_i+1)(m_i+2)}{2}$ elements.

Table 1. Size of full dataset corresponding to the number of components and the number of states per component of MSCS.

N	m_i	Size of full dataset
3	1	27
3	2	216
3	3	1000
3	4	3375
5	1	243
5	2	7776
5	3	100000
5	4	759375
10	1	59049
10	2	60466176
10	3	10000000000
10	4	576650390625

Large state space lead to large imperfect maintenance action space causing difficulties for classical maintenance methods, see for instance [Nicolai and Dekker (2008); Zhang and Si (2020); Kuhnle et al. (2019); Skordilis and Moghaddass (2020); Liu et al. (2020)].

Therefore, the question arising here is how to design a method that can benefit from the historical data above to find optimal maintenance policy taking into consideration imperfect maintenance actions and can cope with large scale problem. To overcome this issue, we propose in this paper an artificial intelligence (AI) based framework for MDM to deal with large MSCS. The details of this framework are presented in the following.

3. AI based framework for maintenance decision-making

The proposed framework consists of two main phases as shown in figure 1. The first refers to training process of maintenance cost and system's reliability predictor. The second is involved in constructing a model of the system that employs trained ANNs from the first phase and in training the DRL agent to optimize maintenance policy by letting it interact with the constructed model. The details of this framework are presented in the following sections.

3.1. *Reliability and maintenance cost model learning*

Traditional reliability prediction methods can only be applied to small MSCS due to difficulties in describing the distribution of time to failure for the huge one. Similarly, the complex structure and the complicated interdependence of real industrial systems also cause the issues for maintenance cost forecasting.

In order to overcome these challenges, ANN is chosen as predictor in the proposed framework because it can be applied to large scale problem [Do et al. (2019)] thanks to the fast development in parallel computational hardware such as graphics and tensor processing unit and software (PyTorch, TensorFlow) dedicated for ANN reducing the time for training and testing.

Specifically, two fully connected feed forward ANNs are used to predict maintenance cost and system's reliability separately. The network's structure is illustrated in figure 2. In particular, the input for each ANN is an array consisting of state before maintenance s_k and the state after maintenance s'_k, and the output is maintenance cost c^m_k or system's reliability h_k. Therefore, we can denote ANN-based predictors for reliability and maintenance cost prediction as $f^r(s_k, s'_k)$ and $f^c(s_k, s'_k)$, respectively.

It should be noted that the hyper parameters for training ANNs such as the number of epochs, hidden layers and the number of neurons per each hidden layer ... need to be carefully chosen to avoid the case of underfitting and overfitting. In addition, ADAM algorithm [Kingma and Ba (2014)] which is an extension of stochastic gradient descent, is highly recommended as the optimizer for prediction task due to its accuracy and time-efficiency according to Do et al. (2019).

After training, ANNs are used in the following section to construct the environment with which DRL agent can interact to optimize maintenance policy.

3.2. *DRL based maintenance decision optimization*

In this second phase of the proposed framework, DRL is used for MDM optimization. Specifically, the environment construction procedure and agent training process are presented in section 3.2.1 and section 3.2.2, respectively.

3.2.1. *Environment construction*

The environment also known as the simulator, is modeled by a Markov decision process (MDP) which is a tuple of five components: state space, action space, transition function, reward function and discount factor. The formal definition of these components is presented in the following.

State space The state space $\mathcal{S}$ is a set covering all possible states of the system. In particular, the system's health state at time T_k is defined as $s_k = \left[x^1_k \; x^2_k \; \dots \; x^N_k\right]$ where x^i_k, $i = \overline{1, N}$, is the degradation level of i^{th} component at that time.

Action space The action space $\mathcal{A}$ is a set of all possible actions. The action chosen at time T_k is denoted as $a_k = \left[a^1_k \; a^2_k \; \dots \; a^N_k\right]$ in which a^i_k, $i = \overline{1, N}$, is the component's action. In this work, deterministic maintenance action is investigated which means that state after maintenance of a component can be transitioned to a specific state between state before maintenance and "as good as new" by the following:

$$x^{i'}_k = x^i_k - a^i_k \tag{2}$$

where $x^{i'}_k$ is the state after maintenance of component i at time T_k. Therefore, the value of a^i_k belongs to the set $\{0, 1, \dots, m_i\}$ and should be understood flexibly. For instance, $a^i_k = 2$ corresponds to perfect maintenance if the current state is 2 or basically is understood as moving the component to state 1 if current state is 3 as implementing imperfect maintenance action. In addition, we can notice that some actions do not exist for a given component's state. For example, if component is in state 2 and the action number 3 and 4 are then not feasible. Hence, these actions should be avoided.

Reward function The reward function R gives a measure of action's quality at a specific state.

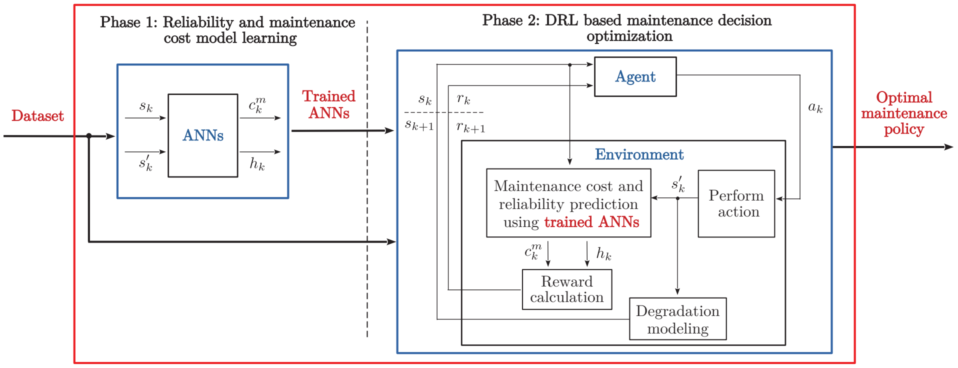
Phase 1: Reliability and maintenance cost model learning
Phase 2: DRL based maintenance decision optimization
Dataset
ANNs
s_k
s'_k
c^m_k
h_k
Trained ANNs
s_k
r_k
s_{k+1}
r_{k+1}
Agent
a_k
Environment
Maintenance cost and reliability prediction using trained ANNs
s'_k
Perform action
c^m_k
h_k
Reward calculation
Degradation modeling
Optimal maintenance policy

Fig. 1. Illustration of artificial intelligence based framework for maintenance decision-making

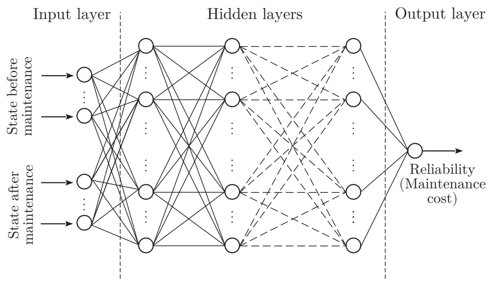

Fig. 2. ANN architecture for maintenance cost and reliability predictor

In this work, the objective of reward function is to minimize total cost c_k. Therefore, reward r_k at time T_k is defined as follows.

$$r_k = -c_k \tag{3}$$

Specifically, c_k includes cost c^m_k predicted by ANN and some extra downtime costs at time T_k. In particular, if the failure of a component causing the failure of the system, downtime cost c^f is then added to total cost. Additionally, a potential downtime cost after maintenance proportional to h_k is also considered. Therefore, total cost c_k is computed according to the below equation.

$$\begin{aligned} c_k &= c^m_k + (1-h_k).c^f + I^f_k.c^f \\ &= f^c(s_k, s'_k) + (1 - f^r(s_k, s'_k)).c^f + I^f_k.c^f \end{aligned} \tag{4}$$

where I^f_k is equal to one if the system is not functioning at time T_k or is equal to zero otherwise. Although the system's structure is unknown, we can still infer the system's failure via its reliability by using the predictor. In particular, if $f^r(s_k, s_k)$ is less than a small positive number τ, the system then fails at inspection time.

It should be noted that in the case of choosing infeasible actions at a given state, c_k is set to be very high.

Discount factor The last element in MDP is discount factor $\gamma \in [0, 1)$ used to guarantee the convergence of infinite cumulative reward as well as to control the relative impact of future reward [Zhang and Si (2020)].

3.2.2. *Agent training process*

The training process is implemented by letting the agent interact with the environment. Specifically, at inspection time T_k, the agent observes state before maintenance s_k. Based on this observation, the agent chooses an action to interact with the environment. After that, the system transitions to state after maintenance s'_k and then degrades to next state s_{k+1} at time T_{k+1}. The reward at time T_k is computed using 3.

The objective of training process is to optimize policy $\pi(s_k, a_k) = \Pr(A_k = a_k \mid S_k = s_k)$ describing the conditional probability of chosen action $A_k = a_k$ given state $S_k = s_k$ in terms of maximizing the action-value function defined as $q(s_k, a_k) = \mathbb{E}[G_k \mid s_k, a_k]$ where $G_k = R_k + \gamma R_{k+1} + \gamma^2 R_{k+2} + \ldots$ (note that the uppercase character denotes random variable and the lowercase one denotes its values). The theory of RL specifies that optimal policy π^* can be obtained from optimal action-value function denoted as $q^*(s_k, a_k) = \max_\pi q(s_k, a_k)$ by using the following equation.

$$\pi^*(a_k) = \underset{a}{\operatorname{argmax}}\, q^*(s_k, a) \tag{5}$$

Therefore, the problem of computing optimal action-value function is a primary concern in many different RL algorithms. Tabular methods such as Q-learning have one major drawback that it cannot be applied for applications with large scale of state-action space because these require a lot of memory. Hence, these approaches are evidently not suitable for optimizing maintenance policy of huge MSCS.

Fortunately, DRL provides a powerful framework to tackle with this challenge. In particular, DRL algorithms use ANN to approximate action-value function which is much more flexible than table-based approaches [Zhang and Si (2020)]. Deep Q network (DQN) is the first DRL algorithm which can solve real world problem at human level [Mnih et al. (2015)]. However, the performance of DQN in some situations is very poor due to large overestimation of action values. In order to overcome this issue, Double DQN (DDQN) is proposed which is based on the theory of double estimators [Van Hasselt et al. (2016)]. Due to the advantage of DDQN, it is chosen to find optimal maintenance policy in the proposed framework.

In particular, DDQN parameterizes action-value function using two ANNs which are policy and target network denoted as $q(s, a, w)$ and $\hat{q}(s, a, w_p)$, respectively. Policy network is used for selecting action given the current state and is updated frequently. In contrast, target network are copied from policy network after every N_{target} steps and aims at computing target y_k for updating policy network following below equation.

$$y_k \leftarrow r_k + \gamma \hat{q}(s_{k+1}, \underset{a}{\operatorname{argmax}}\, q(s_{k+1}, a, w), w_p) \tag{6}$$

It should be noted that policy network is updated using data from experience replay buffer. This kind of memory plays a vital role in all DRL algorithms, which allows not only to gain the i.i.d. assumption required by gradient descent based algorithms but also to exploit the rare experiences to update deep neural network more than one time [Schaul et al. (2015)]. Specifically, at each time step, the experience (s_k, a_k, r_k, s_{k+1}) is stored in the buffer that has fixed length of N_b and functions following first-in first-out mechanism.

4. Numerical studies

This section aims at illustrating how the proposed framework is applied to optimize maintenance policy for the system described in section 2.1 with $N = 5$ and $m_i = 4$. The system's structure is shown in figure 3. As stated in section 2.2, only historical dataset of system's state before and after maintenance with corresponding reliability and maintenance cost is provided. The visualization of this data set is depicted in [Do et al. (2019)].

The detail description of data generation process and the system's parameters are given in Appendix A.

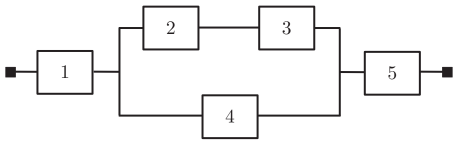

Fig. 3. Five-component system

4.1. *Reliability and maintenance cost model learning*

The dataset is divided into two different sets which are training and evaluation set according to split ratio of 4:1. The hyper parameters for training process are presented in table 2. Mean squared loss is used to evaluate the performance of the models. The obtained results are illustrated in the figure 4 and 5 showing the convergence of mean squared loss for reliability and maintenance cost predictor on both training and validation set.

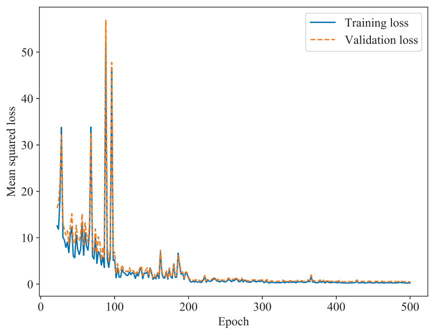

Fig. 4. The convergence of training and validation loss for maintenance cost predictor

4.2. *Maintenance optimization*

In this section, DDQN algorithm is implemented to optimize maintenance decision. In particular, policy and target network are composed of three fully connected layers which have 128, 128 and 32 neurons, respectively. The number of inputs is equal to N. The size of output layer is equal to the size of action space which is 3125. The other hyper parameters related to training process are presented in table 3.

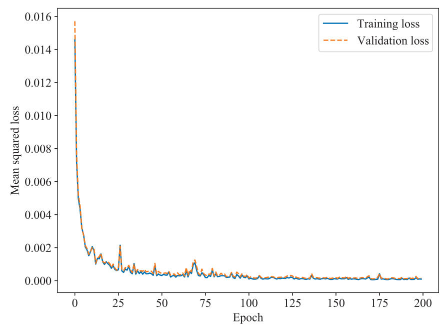

Fig. 5. The convergence of training and validation loss for reliability predictor

Table 2. Hyper parameters for training ANN based predictors.

	$f^c(s_k, s_k')$	$f^r(s_k, s_k')$
Number of epochs	500	200
Hidden layers	100, 100, 50	30, 50
Initial learning rate	0.01	0.005
Batch size	32	32
Activation function	ReLU	ReLU

Note: Learning rate is multiplied with the factor of 0.8 after every 50 epochs.

Table 3. Hyper parameters for training DDQN agent.

N_{train}	N_b	N_{mini}	N_{tagert}	α	γ
2×10^6	5×10^5	128	25000	0.001	0.99

In order to supervise training process, after every 5×10^3 steps, the latest policy network $q(s, a, w)$ is employed to interact with the environment 10^4 times and we then observe the average of total cost per step. The numerical experiment is deployed using PyTorch framework on a desktop computer with 3.91 GHz CPU, 64 GB RAM and 2 GB dedicated GPU memory. The total training time is about 5.18 hours.

The convergence of average cost during training process is illustrated in figure 6. In this picture, we can notice that the policy starts to converge since step 10×10^5 with the corresponding average is about 181.8.

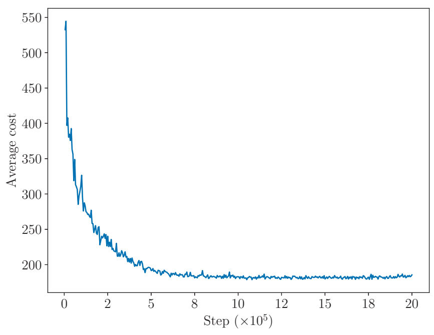

Fig. 6. Average cost in the case of deterministic maintenance action

4.3. *Comparison with traditional maintenance approach*

For large MSCS, DRL algorithms can be used for MDM optimization with deterministic actions while it is difficult for traditional approaches that optimize predefined preventive maintenance thresholds. Therefore, in order to make the comparison between the proposed framework and classical maintenance methods, we adjust maintenance action to have random quality. In particular, there exits three possible actions for each component which are ”do nothing”, ”replacement” and ”imperfect maintenance”. ”Doing nothing” means that no maintenance is performed. If ”replacement” action is carried out, state after maintenance is as good as new. Finally, if imperfect maintenance is executed, state after maintenance is distributed uniformly in the interval from new state to state before maintenance.

The traditional maintenance policy used in this section is presented in [Do and Bérenguer (2012)]. The detail maintenance operation for component i is described in the following:

- If $x_k^i = m_i$, the component is in failed state. Therefore, ”replacement” action is immediately carried out.
- If $l_i \le x_k^i < m_i$ where l_i is preventive maintenance threshold, the system is still operational but badly. Hence, ”imperfect maintenance” is implemented.
- If $x_k^i < l_i$, the system is functioning well. Accordingly, ”do nothing” action is chosen.

The grid-based search algorithm is employed to optimize preventive maintenance thresholds and the total time for searching is about 0.84 hours. The optimal control limit vector is $[2\ 3\ 3\ 1\ 2]$ and the corresponding average cost is 254.6.

The DDQN agent is trained through 20×10^4 steps and the convergence of average cost during training process is illustrated in figure 7. The total time for training is about 0.92 hours and the minimal value is 194.5 that is smaller ($1 - 194.5/254.6 = 23.6\%$) than the one obtained by traditional approach.

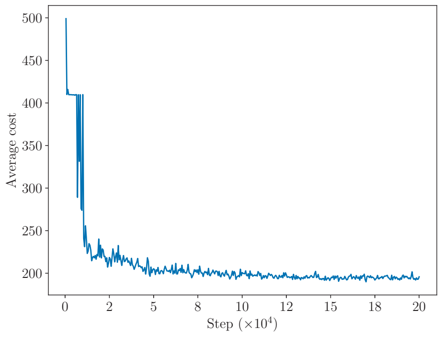

Fig. 7. Average cost in the case of random quality maintenance action

5. Conclusions

In this work, an AI based framework for MDM optimization is proposed which consists of two main phases. The first aims at constructing ANN based predictors for system's reliability and maintenance cost. The second refers to the use of DRL algorithms to optimize maintenance decision which can deal with large scale applications. The obtained results show that ANN is suitable to the reliability, maintenance cost forecasting and DRL is a potentially powerful framework for MDM optimization considering the effects of imperfect maintenance.

Our feature work will focus on the estimation of time-dependent transition matrix using ANN and the use of multi-agent DRL for optimizing maintenance policy.

Acknowledgement

This work has been conducted within the framework of AI-PROFICIENT project, funded by the European Union's Horizon 2020 research and innovation program (grant agreement No 957391).

Appendix A. Data generation

The system considered in section 4 is composed of 5 non-identical components as shown in figure 3. The degradation transition matrices of the components are given as belows.

$$P_1 = \begin{bmatrix} 0.3 & 0.3 & 0.2 & 0.15 & 0.05 \\ 0 & 0.2 & 0.3 & 0.3 & 0.2 \\ 0 & 0 & 0.3 & 0.4 & 0.3 \\ 0 & 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0.1 & 0.3 & 0.3 & 0.2 & 0.1 \\ 0 & 0.1 & 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0.3 & 0.3 & 0.4 \\ 0 & 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 0.2 & 0.2 & 0.3 & 0.3 & 0 \\ 0 & 0.1 & 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0.2 & 0.3 & 0.5 \\ 0 & 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_4 = \begin{bmatrix} 0.25 & 0.3 & 0.2 & 0.2 & 0.05 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 \\ 0 & 0 & 0.2 & 0.3 & 0.5 \\ 0 & 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_5 = \begin{bmatrix} 0.25 & 0.2 & 0.3 & 0.2 & 0.05 \\ 0 & 0.15 & 0.2 & 0.4 & 0.25 \\ 0 & 0 & 0.2 & 0.4 & 0.4 \\ 0 & 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The other system's parameters are given in the table 4.

Table 4. Parameters of 5-component system.

N	m_i	c^r	c^s	c^c	c^f	β
5	4	50	20	30	200	2

The maintenance cost c_k^m at time T_k is calculated by using the formula in equation A.1.

$$c_k^m = \mathbb{I}_k^s . c^s + \sum_{i=1}^{5} c_i^{mc} + \sum_{i=1}^{5} c^{ins} \tag{A.1}$$

where:

- c^s, c^{ins} are setup cost and inspection cost of one component, respectively.
- $\mathbb{I}_k^s = \begin{cases} 1, \text{if maintenance is carried out} \\ 0, \text{if maintenacne is not carried out} \end{cases}$
- c_i^{mc} is maintenance cost of i^{th} component at time T_k. To simplify the notation, we get rid of the subscription i in all formulas. Therefore, if no maintenance action is implemented, then $c^{mc} = 0$. If perfect maintenance is carried out, c^{mc} is then equal

to replacement cost c^r. If imperfect maintenance is implemented, c^{mc} is calculated following the equation below [Do and Bérenguer (2012)].

$$c^{mc} = c^{ip} = c^r.(u_k)^{\beta} \tag{A.2}$$

where:

- c^r is constant for replacement cost
- $u_k = \frac{x_k - x'_k}{x_k}$ where x_k and x'_k are state before and after maintenance of the component, respectively.
- β is real positive number representing the imperfect maintenance characteristics of the component.

Furthermore, the system's reliability h_k describes the survival probability from the time after maintenance to the next time step is the function of all component's reliability and is calculated following equation below.

$$h_k = h_k^1. \left[1 - \left(1 - h_k^2.h_k^3\right).\left(1 - h_k^4\right)\right].h_k^5 \tag{A.3}$$

where $h_k^i = 1 - P_i(x_k^i, m_i)$ is the probability that component i survives until next schedule inspection.

References

Alaswad, S. and Y. Xiang (2017). A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering & System Safety 157*, 54–63.

Do, P. and C. Bérenguer (2012). Condition-based maintenance with imperfect preventive repairs for a deteriorating production system. *Quality and Reliability Engineering International 28*(6), 624–633.

Do, P., B. Iung, and C. Cavalcante (2019). Reliability and maintenance cost forecasting for systems with multistate components using artificial neural networks. In *2019 4th International Conference on System Reliability and Safety (ICSRS)*, pp. 181–185. IEEE.

Huang, J., Q. Chang, and J. Arinez (2020). Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications 160*, 113701.

Karabağ, O., A. S. Eruguz, and R. Basten (2020). Integrated optimization of maintenance interventions and spare part selection for a partially observable multi-component system. *Reliability Engineering & System Safety 200*, 106955.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kuhnle, A., J. Jakubik, and G. Lanza (2019). Reinforcement learning for opportunistic maintenance optimization. *Production Engineering 13*(1), 33–41.

Liu, Y., Y. Chen, and T. Jiang (2020). Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach. *European Journal of Operational Research 283*(1), 166–181.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. (2015). Human-level control through deep reinforcement learning. *nature 518*(7540), 529–533.

Nicolai, R. P. and R. Dekker (2008). Optimal maintenance of multi-component systems: a review. *Complex system maintenance handbook*, 263–286.

Pham, H. and H. Wang (1996). Imperfect maintenance. *European journal of operational research 94*(3), 425–438.

Schaul, T., J. Quan, I. Antonoglou, and D. Silver (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Skordilis, E. and R. Moghaddass (2020). A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics. *Computers & Industrial Engineering 147*, 106600.

Van Hasselt, H., A. Guez, and D. Silver (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 30.

Wei, S., Y. Bao, and H. Li (2020). Optimal policy for structure maintenance: A deep reinforcement learning framework. *Structural Safety 83*, 101906.

Yousefi, N., S. Tsianikas, and D. W. Coit (2020). Reinforcement learning for dynamic condition-based maintenance of a system with individually repairable components. *Quality Engineering 32*(3), 388–408.

Zhang, N. and W. Si (2020). Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks. *Reliability Engineering & System Safety 203*, 107094.