European Journal of Operational Research 283 (2020) 166–181

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

# Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach

Yu Liu[a,b,*], Yiming Chen[a], Tao Jiang[a]

[a] School of Mechanical and Electrical Engineering, University Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China
[b] Center for System Reliability and Safety, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China

ARTICLE INFO

*Article history:*
Received 18 May 2019
Accepted 31 October 2019
Available online 7 November 2019

*Keywords:*
Maintenance
Dynamic selective maintenance
Deep reinforcement learning
Imperfect maintenance
Multi-state system

ABSTRACT

Selective maintenance, which aims to choose a subset of feasible maintenance actions to be performed for a repairable system with limited maintenance resources, has been extensively studied over the past decade. Most of the reported works on selective maintenance have been dedicated to maximizing the success of a single future mission. Cases of multiple consecutive missions, which are oftentimes encountered in engineering practices, have been rarely investigated to date. In this paper, a new selective maintenance optimization for multi-state systems that can execute multiple consecutive missions over a finite horizon is developed. The selective maintenance strategy can be dynamically optimized to maximize the expected number of future mission successes whenever the states and effective ages of the components become known at the end of the last mission. The dynamic optimization problem, which accounts for imperfect maintenance, is formulated as a discrete-time finite-horizon Markov decision process with a mixed integer-discrete-continuous state space. Based on the framework of actor-critic algorithms, a customized deep reinforcement learning method is put forth to overcome the "curse of dimensionality" and mitigate the uncountable state space. In our proposed method, a postprocess is developed for the actor to search the optimal maintenance actions in a large-scale discrete action space, whereas the techniques of the experience replay and the target network are utilized to facilitate the agent training. The performance of the proposed method is examined by an illustrative example and an engineering example of a coal transportation system.



## 1. Introduction

Maintenance activities have been pervasively implemented in industry practices to restore aged systems to a better condition, to restore system performance, and to prolong systems' residual lives (Nourelfath, Fitouhi & Machani, 2010; Yang, Ye, Lee, Yang & Peng, 2019). In some engineering applications, systems are intended to execute a sequence of missions with a finite break between two adjacent missions. Maintenance actions can be performed within a break to recover components from a worse/failure state to a better state to ensure the success of the next mission. However, due to limited maintenance resources, such as budget, time, manpower, and repair facilities, performing all the desirable maintenance actions in a break is oftentimes impossible. Alternatively, a subset of maintenance actions that can maximize the success of the ensuing missions are selected from all the options and performed before the start of the next mission. This strategy is known as selective maintenance (Cassady, Pohl & Murdock, 2001).

The earliest study on selective maintenance can be tracked back to Rice, Cassady and Nachlas (1998) in which a selective maintenance model for parallel-series systems with identical components was developed. Following the framework proposed in Rice et al. (1998), many new selective maintenance optimization models and solutions have been intensively explored over the past few decades from various angles. For example, Cassady, Murdock and Pohl (2001) introduced a set of variations of selective maintenance models with distinct objective functions and constraints. A general selective maintenance was proposed by Cassady et al. (2001), in which the lifetime of components conforms to the Weibull distribution and one of three optional maintenance actions, i.e., minimal repair, corrective replacement, and preventive replacement, can be performed for components. The selective maintenance strategy was implemented for machine lines to reduce maintenance costs and failure losses (Zhu, Liu, Shao, Liu & Deng, 2011).

* Corresponding author at: No. 2006, Xiyuan Avenue, West High-Tech Zone, Chengdu, Sichuan 611731, PR China.
*E-mail address:* yuliu@uestc.edu.cn (Y. Liu).

https://doi.org/10.1016/j.ejor.2019.10.049
0377-2217/

Maaroufi, Chelbi and Rezg (2013) studied a selective maintenance strategy for systems subject to propagated failures with global effects and failure isolation phenomena. A selective maintenance model for a fleet of systems was investigated by Schneider and Cassady (2015). By taking into account the stochastic mission durations and break duration, a new selective maintenance model was presented by Khatab, Aghezzaf, Djelloul and Sari (2017). Three heuristic methods were introduced by Lust, Roux and Riane (2009) to facilitate the optimization of a selective maintenance model with many components.

Nevertheless, these preceding works assumed that both a system and its components are in binary-state, i.e., either working perfectly or failing completely. In engineering practices, some systems, such as manufacturing systems (Nourelfath et al., 2010) and networked systems (Ramirez-Marquez & Coit, 2005), exhibit multi-state characteristics; namely, they possess multiple distinguished performance capacities or damage severities between the perfect working and completely failed states. These systems are termed as multi-state systems. By using multi-state system models, the deterioration behavior of engineered systems can be characterized in a more detailed manner than that of traditional binary-state system models. In the context of multi-state systems, many new selective maintenance models have been developed over the past few years. For example, Liu and Huang (2010) put forth a new selective maintenance model for multi-state systems, and each binary-capacitated component in the system can be imperfectly repaired to a state somewhere between "as good as new" and "as bad as old" states during a break. The selective maintenance model for multi-state systems consisting of multi-state components was investigated by Pandey, Zuo and Moghaddass (2013). Three selective maintenance models, taking account of the economic dependence (Dao, Zuo & Pandey, 2014), stochastic dependence (Dao & Zuo, 2016), and structural dependence (Dao & Zuo, 2017) among components, were developed for multi-state systems. As the uncertainties associated with the durations of breaks and maintenance actions may produce significant impacts on the selective maintenance effectiveness, sequence planning for the selective maintenance of multi-state systems was recently studied by Liu, Chen and Jiang (2018).

Notably, all these aforementioned models for selective maintenance only maximized the success of the next mission and cannot be used for cases where more than one consecutive mission is to be executed in order. For multiple consecutive missions, Iyoob, Cassady and Pohl (2006) developed a selective maintenance model for systems performing a sequence of identical missions with the same durations of breaks between the adjacent missions, and a discrete-time Markov chain was utilized to evaluate the long-run average cost per mission. Based on the premise that failed components can be minimally repaired immediately during a mission, Khatab, Aghezzaf and Claver (2015) and Pandey, Zuo and Moghaddass (2016) studied a selective maintenance strategy for systems executing multiple missions. Such an assumption of minimal repairs upon failures may not be valid in real-world systems, as maintenance actions can only be performed during breaks between two adjacent missions in the general framework of selective maintenance. With the assumption that a binary-state series-parallel system is composed of identical components and that lifetime of the components complies with the exponential distribution, the selective maintenance optimization for a system over a finite horizon was formulated as a stochastic dynamic programming by Maillart, Cassady, Rainwater and Schneider (2009). In their study, the failed components to be replaced in each break are dynamically identified at the end of the last mission. The approximate dynamic programming algorithm was used by Ahadi (2018) to address the selective maintenance optimization raised in Maillart et al. (2009) when the number of components in a series-parallel system is extremely large. However, the assumptions of identical components and exponentially distributed lifetimes block the implementation of selective maintenance for many real-world engineering systems. Even though the reliability assessment and optimization of phased-mission systems, which can experience several phases in a mission, have been extensively investigated in the literature (Levitin, Finkelstein & Dai, 2017; Wang, Peng & Xing, 2018, 2020), the system studied in most of these works was unrepairable and the system will stop functioning for the remaining phases if it fails in any one of preceding phases.

To further advance the state-of-the-art of selective maintenance optimization and facilitate its implementation to broader applications, a new dynamic selective maintenance model for multi-state systems over a finite horizon is put forth hereinafter, in which both the assumptions of the identical components and exponential lifetime distribution in the earlier literature are released. Multiple maintenance actions, including "doing nothing", minimal repair, imperfect maintenance, and replacement, are available to be chosen for each component, and they can only be performed in-between two adjacent missions. However, due to limited maintenance resources in each break, the selective maintenance strategy has to be dynamically optimized at the end of each mission based on the states and effective ages of all the components in a system to ensure the maximum expected number of the successes of future missions. The resulting stochastic dynamic programming is formulated as a discrete-time finite-horizon Markov decision process with a mixed integer-discrete-continuous state space. To deal with the "curse of dimensionality" and the uncountable state space, a customized deep reinforcement learning (DRL) approach is developed based on the framework of actor-critic algorithms (Sutton & Barto, 2018). The critic, which evaluates the effectiveness of maintenance actions, is mimicked by an artificial neural network named the Q-network, whereas a postprocess is used in the actor to identify the optimal maintenance actions in a large-scale discrete action space. The two techniques, specifically the experience replay and the target network, of the deep Q-network (DQN), are utilized to improve the robustness of the customized DRL approach. The performance of the proposed method is examined by an illustrative example and an engineering example of a coal transportation system.

The remainder of this paper is implemented out as follows. Section 2 gives the problem description and the basic assumptions in our study. The states and effective ages of the components at the end of the last mission, along with the probability of a system successfully completing the next mission, are evaluated in Section 3, and the dynamic selective maintenance optimization is formulated as a discrete-state finite-horizon Markov decision process. A customized DRL algorithm is put forth in Section 4 to resolve the resulting optimization problem. Two illustrative examples are presented in Section 5 to examine the effectiveness of the proposed method. Section 6 details conclusions and some future studies.

## 2. Problem description and model assumptions

### 2.1. Selective maintenance of multi-state systems

In many military and industrial environments, systems are planned to execute multiple missions with a break in-between two adjacent missions. Maintenance actions can only be performed during the break to restore the aged system to a better condition and thus complete as many future missions as possible. For example, a truck is scheduled to finish a sequence of transportation tasks with a break between two adjacent tasks, whereas an aircraft is intended to complete multiple flights with a stop between two consecutive flights. However, due to limited maintenance resources, only a subset of components, rather than all of the

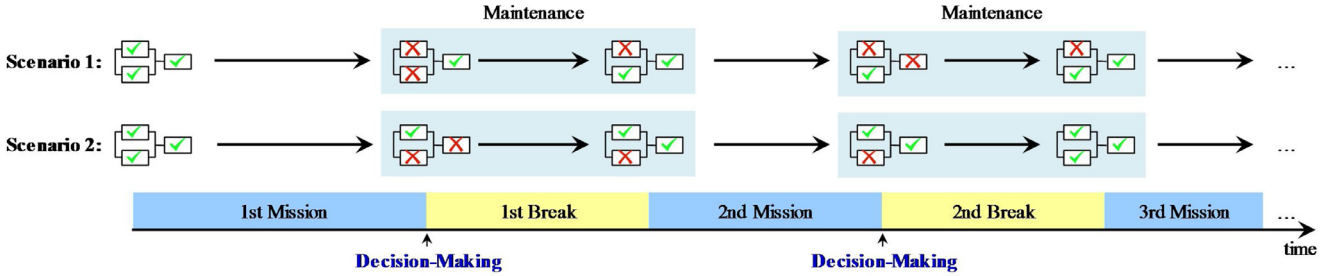


**Fig. 1.** Selective maintenance for a system that can execute multiple consecutive missions.

components in a system, can be recovered to their better conditions. Due to the randomness of components' failures in a mission, the state of each component at the end of each mission varies from one system to another as shown in Fig. 1. The selective maintenance strategy has to be dynamically optimized based on the health status of all the components in a specific individual system to ensure the maximum number of the successes of future missions for the system.

Without loss of generality, the specific selective maintenance problem and the multi-state systems' characteristics are defined as follows:

(1) A multi-state system is composed of $M$ statistically independent binary-capacitated components. The performance capacity of component $l$ ($l \in \{1, 2, \cdots, M\}$) at any time instant $t$, denoted as $G_l(t)$, equates $g_{l,1}$ and $g_{l,0}$ for the operational state (State 1) and the failure state (State 0), respectively. Specifically, the performance capacity of components in the failure state is zero, specifically, $g_{l,0}$=0.

(2) The performance capacity of the entire system at any time instant $t$, denoted as $G(t)$, can be determined by the performance capacities of all the components and the structure function $\phi(\cdot)$ of the system. For instance, one has $G(t)=\phi(G_1(t), G_2(t)) = \min\{G_1(t), G_2(t)\}$ if two components in a flow transmission system are connected in series, whereas $G(t)=\phi(G_1(t), G_2(t)) = G_1(t) + G_2(t)$ is used for a parallel connection. Several existing tools, such as the universal generating function (UGF) (Levitin, 2005), multi-valued decision diagram (Amari, Xing, Shrestha, Akers & Trivedi, 2010; Mo, Xing, Zhong & Zhang, 2016), and Bayesian network (Jiang & Liu, 2017), can efficiently enumerate all the possible combinations of components' states with respect to each system performance capacity. The system can, therefore, exhibit a finite number of states, denoted as $\mathbf{G} = \{g_1, g_2, \ldots, g_{N_s}\}$, distinguished by the system performance capacity, where $g_1$ and $g_{N_s}$ are the worst and best states, respectively, and $N_s$ is the number of the possible states. At any particular time instant $t$, $G(t) \in \mathbf{G}=\{g_1, g_2, \ldots, g_{N_s}\}$.

(3) The system is intended to execute a sequence of $K$ missions, and the duration of the $k$th mission is denoted as $Z_k$, where $k \in \{1, 2, \ldots, K\}$. The system configuration remains unchanged throughout all the missions. Additionally, even if a proceeding mission fails, the system can be recovered to the functioning state in a break by executing maintenance actions.

(4) The failure time of each component can conform to an arbitrary distribution. The state of component $l$ at the beginning of the $k$th mission is denoted as a binary variable $X_{l,k}$, where:

$$X_{l,k} = \begin{cases} 1 & \text{if component } l \text{ is functioning} \\ 0 & \text{if component } l \text{ is failed} \end{cases}. \quad (1)$$

The state of component $l$ at the end of the $k$th mission is denoted as a binary variable $Y_{l,k}$, where:

$$Y_{l,k} = \begin{cases} 1 & \text{if component } l \text{ is functioning} \\ 0 & \text{if component } l \text{ is failed} \end{cases}. \quad (2)$$

Hence, the states of all the components in a system at the beginning and end of the $k$th mission are represented by a vector $\mathbf{X}_k = (X_{1,k}, X_{2,k}, \ldots, X_{M,k})$ and a vector $\mathbf{Y}_k = (Y_{1,k}, Y_{2,k}, \ldots, Y_{M,k})$, respectively. At the beginning of the first mission, all the components are in brand new condition and one has $\mathbf{X}_1 = (1, 1, \ldots, 1)$.

(5) The $k$th mission is successfully completed if the system performance capacity at the end of the $k$th mission is greater than a prespecified demand, denoted as $W_k$, which could be either a constant or a random quantity in engineering reality (Ekin, 2018; Sloan, 2004). In the case that the prespecified demand of the $k$th mission is a random variable with $H$ possible values, the $h$th possible value of the demand of the $k$th mission is denoted as $w_{h,k}$.

(6) Several optional maintenance actions, including "doing nothing", minimal repair, imperfect maintenance, and replacement, can be chosen for each failed or aged component. However, due to limited maintenance budget and time in each break, denoted as $C_{\text{lim}}$ and $T_{\text{lim}}$, respectively, only a subset of components can be maintained in each break. In our study, it is assumed that the maintenance time and budget of each break are pre-specified as reported in Iyoob et al. (2006), Maillart et al. (2009), and Ahadi (2018). For example, production lines can, oftentimes, only be maintained at the weekend, and maintenance budget for each weekend has to be pre-specified at the beginning of a month.

### 2.2. Imperfect maintenance model

During each break, maintenance actions can be performed to recover the condition of a failed or aged component in a system to ensure the success of future missions. Based on the efficiency, maintenance actions can be categorized into three categories, i.e., perfect repair/maintenance (including replacement), minimal repair/maintenance, and imperfect repair/maintenance (Pham & Wang, 1996). Apart from the minimal repair and replacement that restore a component to "as bad as old" and "as good as new" conditions, respectively, many maintenance actions in engineering practices can be viewed as imperfect maintenance with which a failed/aged component can only be restored to somewhere between the aforementioned two extremes (Pham & Wang, 1996). A higher quality of maintenance action, of course, consumes a greater amount of maintenance resources (cost and time). Without loss of generality, in this study, "doing nothing" can be viewed as one of the optional maintenance actions for a component.

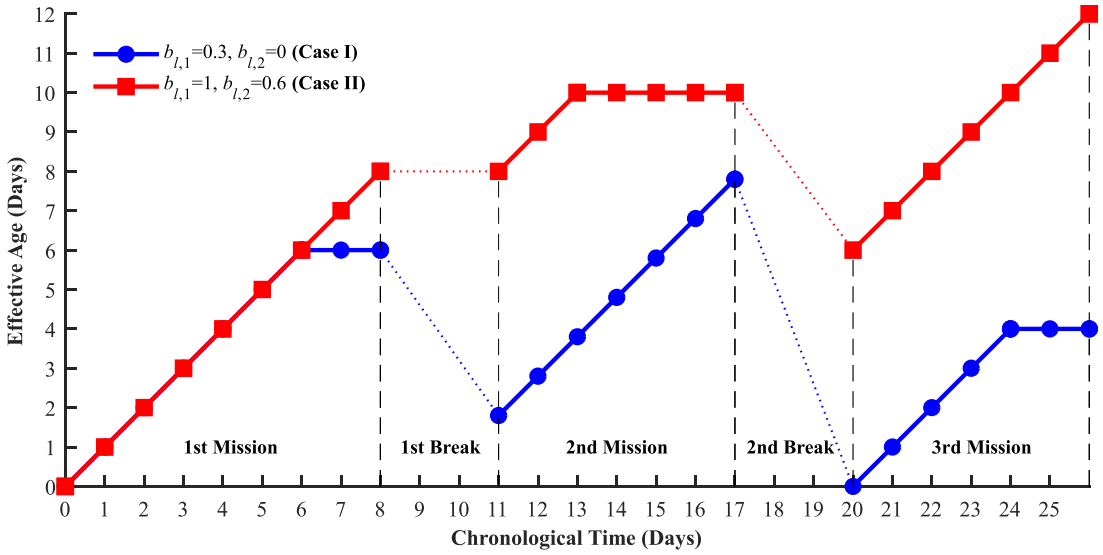


**Fig. 2.** The chronological time vs. the effective age for the Kijima type II model.

Suppose that $N_L$+1 optional maintenance actions are available for each component. The selected maintenance action for component $l$ in the $k$th break, i.e., the break between the $k$th mission and the $(k+1)$st mission, is denoted as $a_{l,k}$ ($a_{l,k} \in \{0, 1, \cdots, N_L\}$), and a greater value of $a_{l,k}$ represents a maintenance action with a higher efficiency. $a_{l,k} = N_L$ and $a_{l,k} = 0$ represent the replacement and "doing nothing", respectively. Specifically, for a failed component, $a_{l,k} = 1$ signifies a minimal repair that merely recovers a failed component back to its functioning state, whereas $a_{l,k} = 0$ corresponds to "doing nothing", namely, the component is left in its failure state.

The maintenance cost associated with maintenance action $a_{l,k}$ for component $l$ in the $k$th break is given by the following:

$$C_{l,k} = \begin{cases} 0 & a_{l,k} = 0 \\ c_l^0 + c_l(a_{l,k}, Y_{l,k}) & a_{l,k} = 1, 2, \cdots, N_L \end{cases}, \tag{3}$$

where $c_l^0$ is the fixed maintenance cost of component $l$, such as the disassembly and assembly costs, the setup cost, and the cost of organizing maintenance personnel. $c_l(a_{l,k}, Y_{l,k})$ denotes the variable maintenance cost that is determined by the selected maintenance action $a_{l,k}$ and the state of component $l$ at the end of the $k$th mission. Specifically, $c_l(a_{l,k}, 1)$ and $c_l(a_{l,k}, 0)$ correspond to preventive maintenance and corrective maintenance for a failed and functioning component $l$, respectively. The total maintenance cost to be consumed in the $k$th break, denoted as $C_k$, can be evaluated by the following:

$$C_k = \sum_{l=1}^{M} C_{l,k}. \tag{4}$$

In the same manner, the maintenance time of component $l$ can be defined as follows:

$$T_{l,k} = \begin{cases} 0 & a_{l,k} = 0 \\ t_l^0 + t_l(a_{l,k}, Y_{l,k}) & a_{l,k} = 1, 2, \cdots, N_L \end{cases}, \tag{5}$$

where $t_l^0$ is the fixed maintenance time of component $l$, whereas $t_l(a_{l,k}, Y_{l,k})$ is the variable maintenance time associated with the selected maintenance action and the state of component $l$ at the end of the $k$th mission. The total maintenance time to be spent in the $k$th break, denoted as $T_k$, is a function of the respective maintenance time of each component, and it can be given, in a general form, as:

$$T_k = f(T_{1,k}, T_{2,k}, \cdots, T_{M,k}). \tag{6}$$

In the case that the selected maintenance actions have to be executed in a sequential manner, the total maintenance time is the summation of the respective maintenance times of all the components. If all the selected maintenance actions can be executed independently in a parallel fashion, the total maintenance time will equate to the longest maintenance time among all the individually selected maintenance actions.

Many imperfect maintenance models have been developed to characterize the imperfection of maintenance actions (Doyen & Gaudoin, 2004; Kijima, 1989, 1988; Pham & Wang, 1996; Shen, Cui & Ma, 2019; Si & Yang, 2016; Wu & Zuo, 2010). In this study, the Kijima type II model is specifically used to reflect the imperfection of maintenance actions via the age reduction mechanism (Dijoux, Fouladirad & Nguyen, 2016; Kijima, 1989). The Kijima type II model assumes that the condition of a component is closely associated with its effective age, which may not be necessarily the same as the calendar age. The effective age of component $l$ at the beginning and end of the $k$th mission are denoted as $A_{l,k}$ and $B_{l,k}$, respectively, and based on the Kijima type II model, one has the following:

$$A_{l,k+1} = b_{l,k} B_{l,k}, \tag{7}$$

where $b_{l,k} (0 \le b_{l,k} \le 1)$ is the age reduction factor, representing the efficiency of a selected maintenance action for component $l$ at the end of the $k$th mission. A smaller value of $b_{l,k}$ indicates a superior maintenance efficiency. The relationship between the chronological time and effective age of two components is given in Fig. 2 to exemplify the principle of the Kijima type II model.

$\mathbf{B}_k = (B_{1,k}, B_{2,k}, \ldots, B_{M,k})$ denotes the effective ages of all the components at the end of the $k$th mission and $\mathbf{a}_k = (a_{1,k}, a_{2,k}, \ldots, a_{M,k})$ denotes the selected maintenance actions for all the components in the $k$th break. The states and effective ages of all the components at the beginning of the $(k+1)$st mission, denoted as $\mathbf{X}_{k+1} = (X_{1,k+1}, X_{2,k+1}, \ldots, X_{M,k+1})$ and $\mathbf{A}_{k+1} = (A_{1,k+1}, A_{2,k+1}, \ldots, A_{M,k+1})$, respectively, can, therefore, be completely determined by $\mathbf{a}_k$, $\mathbf{B}_k$, and $\mathbf{Y}_k$.

## 3. Dynamic selective maintenance decision model

To achieve the maximum number of successes for future missions, maintenance actions in each break should be dynamically selected based on the states and effective ages of all the components in a system. By identifying the maintenance actions for each combination of the states and effective ages of all the components in each break, a "maintenance policy" (also called a policy function), denoted as $\pi$ (i.e., $\mathbf{a}_k = \pi(\mathbf{Y}_k, \mathbf{B}_k, k)$), is determined. In this section, the state and effective age of a component at the end of a mission are deduced in Section 3.1, and the probability of a system successfully completing a mission is evaluated via the universal generating function (UGF) in Section 3.2. The resulting discrete-time finite-horizon Markov decision process is formulated together with its Bellman equation in Section 3.3.

### 3.1. The states and effective ages of components at the end of a mission

Based on the state and effective age of component $l$ at the beginning of the $k$th mission, the state and effective age of the component at the end of the $k$th mission can be evaluated by the following two cases.

**Case 1:** $X_{l,k} = 0$

Component $l$ is in the failure state at the beginning of the $k$th mission, and the state of the component remains unchanged during the $k$th mission, i.e., $Y_{l,k} = X_{l,k} = 0$; meanwhile, the effective age of the component is suspended, that is, $B_{l,k} = A_{l,k}$.

**Case 2:** $X_{l,k} = 1$

Component $l$ is functioning at the beginning of the $k$th mission. Let $F_l(t)$ and $f_l(t)$ denote the cumulative distribution function and the probability density function of the failure time of component $l$, respectively. The survival probability of component $l$ at the end of the $k$th mission, i.e., the probability of component $l$ functioning at the end of the $k$th mission, is equal to the following:

$$r_{l,k} = 1 - \frac{F_l(A_{l,k} + Z_k) - F_l(A_{l,k})}{1 - F_l(A_{l,k})} = \frac{1 - F_l(A_{l,k} + Z_k)}{1 - F_l(A_{l,k})}. \tag{8}$$

In this case, the state and effective age of the component at the end of the $k$th mission comprise one of the following two cases.

**Case 2.1:** $X_{l,k} = Y_{l,k} = 1$

Component $l$ remains functioning at the end of the $k$th mission. The increment of the effective age of component $l$ is equal to the duration of the $k$th mission. Hence, the effective age of component $l$ at the end of the $k$th mission equates as follows:

$$B_{l,k} = A_{l,k} + Z_k. \tag{9}$$

**Case 2.2:** $X_{l,k} = 1$ **and** $Y_{l,k} = 0$

Component $l$ is functioning at the beginning of the $k$th mission, but it fails within the $k$th mission. The operating time of component $l$, denoted as $T_{l,k}$, in the $k$th mission is a random variable with the probability density function formulated as follows:

$$f_{T_{l,k}}(t) = \frac{f_l(A_{l,k} + t)}{F_l(A_{l,k} + Z_k) - F_l(A_{l,k})}. \tag{10}$$

where $T_{l,k} \in (0, Z_k)$. $f_{T_{l,k}}(t)$ is, essentially, the conditional probability density function of $f_l(t)$ on the condition that component $l$ fails within the time interval of $(A_{l,k}, A_{l,k} + Z_k)$. The effective age of component $l$ at the end of the $k$th mission can therefore be given by the following:

$$B_{l,k} = A_{l,k} + T_{l,k}. \tag{11}$$

$p_{l,X_{l,k},Y_{l,k}}$ denotes the probability of component $l$ sojourning in state $Y_{l,k}$ at the end of the $k$th mission, while being in state $X_{l,k}$ at the beginning of the $k$th mission. The state transition probability of component $l$ at the end of the $k$th mission is given by the following:

$$p_{l,X_{l,k},Y_{l,k}} = \Pr\{Y_{l,k} | X_{l,k}\} = \begin{cases} 1 & X_{l,k} = 0, Y_{l,k} = 0 \\ r_{l,k} & X_{l,k} = 1, Y_{l,k} = 1 \\ 1 - r_{l,k} & X_{l,k} = 1, Y_{l,k} = 0 \\ 0 & X_{l,k} = 0, Y_{l,k} = 1 \end{cases}. \tag{12}$$

The effective age of component $l$ at the end of the $k$th mission is one of the following three cases:

$$B_{l,k} = \begin{cases} A_{l,k} & X_{l,k} = 0, Y_{l,k} = 0 \\ A_{l,k} + Z_k & X_{l,k} = 1, Y_{l,k} = 1 \\ A_{l,k} + T_{l,k} & X_{l,k} = 1, Y_{l,k} = 0 \end{cases}. \tag{13}$$

As seen from these derivations, the states and effective ages of all the components at the end of the $k$th mission, i.e., $\mathbf{Y}_k$ and $\mathbf{B}_k$, can be completely determined by the states and effective ages of all the components at the beginning of the $k$th mission, i.e., $\mathbf{X}_k$ and $\mathbf{A}_k$. The transition of the state and effective age of a component is, therefore, independent of its history.

### 3.2. Evaluating the probability of a system successfully completing a mission

In this study, a mission is successfully completed by a system if the system performance capacity at the end of the mission is not less than a prespecified demand. The probability of a system successfully completing the $k$th mission, denoted as $R_k$, can therefore be evaluated by the state distribution of the system at the end of the mission. The universal generating function (UGF) method (Ushakov, 1986), as a computationally efficient tool for evaluating the state distribution of multi-state systems (Levitin & Lisnianski, 2000), is utilized to estimate the state distribution of a system at the end of a mission by the state distribution of all the components.

The UGF is a polynomial form representation of the probability mass function of a discrete random variable. As the state of each component is a discrete random variable, the UGF of component $l$ at the end of the $k$th mission, denoted as $u_{l,k}$, is written as follows:

$$u_{l,k} = \sum_{Y_{l,k}=0}^{1} p_{l,Y_{l,k}} \cdot z^{g_{l,Y_{l,k}}}. \tag{14}$$

where $p_{l,Y_{l,k}}$ is the probability of component $l$ sojourning in state $Y_{l,k}$ at the end of the $k$th mission. Given the state and effective age of component $l$ at the beginning of the $k$th mission, i.e., $X_{l,k}$ and $A_{l,k}$, one has $p_{l,Y_{l,k}} = p_{l,X_{l,k},Y_{l,k}}$, where $p_{l,X_{l,k},Y_{l,k}}$ is computed by Eq. (12). Let $p_{s,i,k}$ denote the probability of the system performance capacity being equal to $g_i$ at the end of the $k$th mission, and the UGF of the system at the end of the $k$th mission, denoted as $U_{s,k}$, can be evaluated by the following:

$$\begin{aligned} U_{s,k} &= \sum_{i=1}^{N_s} p_{s,i,k} z^{g_i} \\ &= \sum_{Y_{1,k}=0}^{1} \sum_{Y_{2,k}=0}^{1} \cdots \sum_{Y_{M,k}=0}^{1} \left( \prod_{l=1}^{M} p_{l,Y_{l,k}} z^{\phi(g_{1,Y_{1,k}}, g_{2,Y_{2,k}}, \cdots, g_{M,Y_{M,k}})} \right), \end{aligned} \tag{15}$$

where $\phi(g_{1,Y_{1,k}}, g_{2,Y_{2,k}}, \ldots, g_{M,Y_{M,k}})$ is the structure function of the system.

Hence, the probability of a system successfully completing the $k$th mission can be written as follows:

$$R_k = \sum_{i=1}^{N_s} p_{s,i,k} \cdot 1(g_i \geq W_k), \tag{16}$$

where $1(\cdot)$ is an indictor function that is equal to one if $g_i \geq W_k$ is true or zero otherwise. In the case that the prespecified demand of the $k$th mission is a random variable with $H$ possible values, the probability of a system successfully completing the $k$th mission can be given by the following:

$$R_k = \sum_{h=1}^{H} \Pr\{W_k = w_{h,k}\} \cdot \sum_{i=1}^{N_s} p_{s,i,k} \cdot 1(g_i \geq w_{h,k}). \tag{17}$$

where $w_{h,k}$ is the $h$th possible value of the demand of the $k$th mission.

The expected number of the successes of future missions, denoted as $N$, can be evaluated by the following:

$$N = \sum_{k=1}^{K} R_k. \tag{18}$$

It is worth noting that the weight of each mission is assumed to be identical in this study. If each mission possesses a different weight, the expected weighted number of the successes of future missions can be evaluated by $K \cdot \sum_{k=1}^{K} \alpha_k R_k$, where $\alpha_k$ is the weight of the $k$th mission and $\sum_{k=1}^{K} \alpha_k = 1$.

The probability of a system successfully completing the $k$th mission is completely determined by the states and effective ages of all the components at the beginning of the $k$th mission. The probability $R_k$ can therefore be represented as a function of $\mathbf{X}_k$ and $\mathbf{A}_k$, i.e., $R_k = R(\mathbf{X}_k, \mathbf{A}_k)$. Referring to Section 2.2 that $\mathbf{X}_k$ and $\mathbf{A}_k$ are completely determined by $\mathbf{a}_{k-1}$, $\mathbf{B}_{k-1}$, and $\mathbf{Y}_{k-1}$, the probability $R_k$ can be further represented as $R_k = R(\mathbf{Y}_{k-1}, \mathbf{B}_{k-1}, \mathbf{a}_{k-1})$.

### 3.3. Markov decision process and Bellman equation

Maximizing the expected number of the successes of future missions, i.e., Eq. (18), with the maintenance budget and limited duration of breaks, is a stochastic dynamic programming; it can be viewed as a Markov decision process (MDP) since the transitions of the states and effective ages of all the components in a system are independent of their histories. The state space, the action space, and the reward of the MDP are detailed as follows.

**State Space**: The state space of the MDP consists of all the possible combinations of the states and effective ages of components in a system at the end of a mission. The state of a component is a binary variable, whereas the effective age of a component is a continuous variable. Hence, the MDP has a mixed integer-discrete-continuous state space that is uncountable.

**Action Space**: In the $k$ th break, the action space of the MDP, denoted as $\mathbf{S}_k^{\text{act}}$ ($k \in \{1, 2, \ldots, K-1\}$), is the set of feasible maintenance actions for all the components, and $\mathbf{S}_k^{\text{act}}$ can be written as follows:

$$\mathbf{S}_k^{\text{act}} = \left\{ \mathbf{a}_k \middle| \sum_{l=1}^{M} C_{l,k} \leq C_{\text{lim}}, \sum_{l=1}^{M} T_{l,k} \leq T_{\text{lim}} \right\}. \tag{19}$$

**Reward**: In the $k$ th break, the reward of the MDP is the probability of the system successfully completing the $(k+1)$ mission, i.e., $R_{k+1}$.

Given the states and the effective ages of all the components in a system at the end of the $k$th mission, i.e., $\mathbf{Y}_k$ and $\mathbf{B}_k$, the maximum expected number of the successes of the future remaining $K-k$ missions is a value function of the MDP, denoted as $V(\mathbf{Y}_k, \mathbf{B}_k, k)$. If $k = K-1$, that is, only one future mission is left, the value of $V(\mathbf{Y}_{K-1}, \mathbf{B}_{K-1}, K-1)$ is equivalent to the maximum probability of the system successfully completing the last mission and can be computed by the following:

$$V(\mathbf{Y}_{K-1}, \mathbf{B}_{K-1}, K-1) = \max_{\mathbf{a}_{K-1} \in \mathbf{S}_{K-1}^{\text{act}}} R_K = \max_{\mathbf{a}_{K-1} \in \mathbf{S}_{K-1}^{\text{act}}} R(\mathbf{Y}_{K-1}, \mathbf{B}_{K-1}, \mathbf{a}_{K-1}). \tag{20}$$

$\mathbf{S}_k^{\text{F}} = \{l | X_{l,k} = 1,\ Y_{l,k} = 0\}$ denotes the set of components that fail within the $k$th mission, where $\mathbf{S}_k^{\text{F}} \subseteq \{1, 2, \cdots, M\}$. $M_k$ denotes the number of components in $\mathbf{S}_k^{\text{F}}$. Hence, if more than one future mission is left, the Bellman equation of the MDP can be formulated as follows:

$$\begin{aligned} V(\mathbf{Y}_k, \mathbf{B}_k, k) &= \max_{\pi} E\left[ \sum_{n=k+1}^{K} R_n | \mathbf{Y}_k, \mathbf{B}_k \right] \\ &= \max_{\mathbf{a}_k \in \mathbf{S}_k^{\text{act}}} \left\{ R(\mathbf{Y}_k, \mathbf{B}_k, \mathbf{a}_k) + \left[ \sum_{Y_{1,k+1}=0}^{1} \sum_{Y_{2,k+1}=0}^{1} \cdots \sum_{Y_{M,k+1}=0}^{1} \right. \right. \\ &\quad \left( \prod_{l=1}^{M} p_{l,X_{l,k+1},Y_{l,k+1}} \int_0^{Z_k} \int_0^{Z_k} \cdots \int_0^{Z_k} \prod_{m=1}^{M_{k+1}} f_{T_{l_m,k+1}}(t_{l_m,k+1}) \right. \\ &\quad \left. \left. \left. \cdot V(\mathbf{Y}_{k+1}, \mathbf{B}_{k+1}, k+1) dt_{l_1,k+1} dt_{l_2,k+1} \cdots dt_{l_{M_{k+1}},k+1} \right) \right] \right\}, \end{aligned} \tag{21}$$

where $l_m \in \mathbf{S}_{k+1}^{F}$ is the $m$th component in $\mathbf{S}_{k+1}^{F}$; $t_{l_m,k+1}$ is the operation time of component $l_m$ during the $(k+1)$st mission, i.e., $t_{l_m,k+1} = B_{l_m,k+1} - A_{l_m,k+1}$. For a brand new system, the maximum expected number of the successes of future missions can be solved by the following:

$$\begin{aligned} N^* &= V(\mathbf{X}_1, \mathbf{A}_1, 0) = \max_{\pi} E\left[ \sum_{n=1}^{K} R_n | \mathbf{X}_1, \mathbf{A}_1 \right] \\ &= R(\mathbf{X}_1, \mathbf{A}_1) + \left[ \sum_{Y_{1,1}=0}^{1} \sum_{Y_{2,1}=0}^{1} \cdots \sum_{Y_{M,1}=0}^{1} \right. \\ &\quad \left( \prod_{l=1}^{M} p_{l,X_{l,1},Y_{l,1}} \cdot \int_0^{Z_1} \int_0^{Z_1} \cdots \int_0^{Z_1} \prod_{m=1}^{M_1} f_{T_{l_m,1}}(t_{l_m,1}) \right. \\ &\quad \left. \left. \cdot V(\mathbf{Y}_1, \mathbf{B}_1, 1) dt_{l_1,1} dt_{l_2,1} \cdots dt_{l_{M_1},1} \right) \right] \Bigg\}. \end{aligned} \tag{22}$$

Once the states, the effective ages, and the selected maintenance actions of all the components in a system at the end of the $k$th mission, i.e., $\mathbf{Y}_k$, $\mathbf{B}_k$, and $\mathbf{a}_k$, are given, the maximum expected number of the successes of the future remaining $K-k$ missions is a "Q-function" (also called "state-action" value function or "action-value" function) of the MDP, denoted as $Q^*(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k)$. This function can be formulated as follows:

$$\begin{aligned} Q^*(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k) &= R_{k+1} + \max_{\pi} E\left[ \sum_{n=k+2}^{K} R_n | \mathbf{Y}_k, \mathbf{B}_k, \mathbf{a}_k \right] \\ &= R(\mathbf{Y}_k, \mathbf{B}_k, \mathbf{a}_k) + \max_{\pi} E\left[ \sum_{n=k+2}^{K} R_n | \mathbf{Y}_k, \mathbf{B}_k, \mathbf{a}_k \right]. \end{aligned} \tag{23}$$

The relationship between the value function and the Q-function can be written as follows:

$$V(\mathbf{Y}_k, \mathbf{B}_k, k) = \max_{\mathbf{a}_k \in \mathbf{S}_k^{\text{act}}} Q^*(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k). \tag{24}$$

The value function can be solved by the Q-function. On the other hand, the Q-function can be computed by the value function via the Bellman equation. Based on the Q-function and Bellman

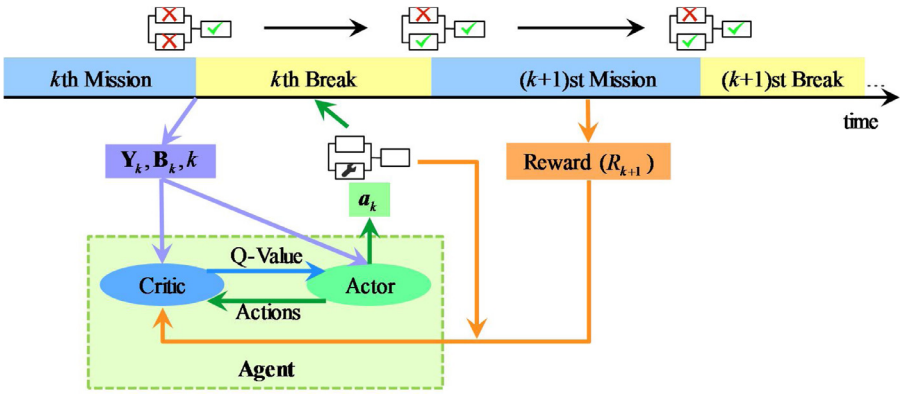


**Fig. 3.** An illustration of the actor-critic framework.

equation, the optimal selective maintenance strategy can be found by the following:

$$\pi^*(\mathbf{Y}_k, \mathbf{B}_k, k) = \underset{\mathbf{a}_k \in \mathbf{S}_k^{\text{act}}}{argmax}\, Q^*(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k). \tag{25}$$

## 4. The proposed deep reinforcement learning algorithm

The discrete-state finite-horizon Markov decision process can be resolved by the dynamic programming or heuristic algorithms when its state and action spaces are quite small and enumerable. However, in this study, the mixed integer-discrete-continuous state space of the MDP is uncountable. Furthermore, due to the "curse of dimensionality", the state and action spaces of the MDP increase exponentially with respect to the number of components in a system. For example, without the constraints of maintenance resources, the number of possible maintenance actions is $(N_L+1)^M$ for each combination of the states and the effective ages of all the components. Hence, the resulting optimization problem cannot be readily resolved by the traditional methods (Andriotis & Papakonstantinou, 2019), such as the dynamic programming, genetic algorithm, and tabu search.

As an alternative, the deep reinforcement learning (DRL) (Sutton & Barto, 2018) is a promising tool to overcome systematic problems in a computationally efficient manner. The DRL algorithms have been studied in recent years to resolve complicated MDP problems, particularly in the field of artificial intelligence (Dulac-Arnold et al., 2015; Lillicrap et al., 2015; Mnih et al., 2015; Silver et al., 2014; Sutton & Barto, 2018). In the DRL algorithms, an agent derives efficient representations of the environment from high-dimensional inputs and uses them to generalize past experiences to new situations (Mnih et al., 2015). Hence, the agent must discover how to interact with a dynamic environment to maximize its expected cumulative rewards (Schmidhuber, 2015). Specifically, the DRL algorithm defeated humans in numerous games such as Atari 2600 games, Go, chess, and shogi (Mnih et al., 2015; Silver et al., 2018). The DRL algorithm was recently used to facilitate engineering systems management (Andriotis & Papakonstantinou, 2019).

To resolve our specific optimization problem, a customized DRL algorithm is put forth in this study to resolve the MDP with a mixed integer-discrete-continuous state space and overcome the "curse of dimensionality" for both the state space and action space. In the proposed DRL algorithm, an agent will repeatedly execute the $K$ missions with the same initial conditions. Based on the framework of the actor-critic algorithms, a critic will judge the effectiveness of the selected maintenance actions by evaluating the maximum expected number of the successes of future missions for a given condition, i.e., evaluating the value of the Q-function. For simplicity, the value of the Q-function estimated by the critic is denoted as the "Q-value". Based on the states and effective ages of all the components, an actor, as a reference to the learned policy, will select maintenance actions for all the components with the constraints of maintenance resources for the maximum Q-value. By performing the maintenance actions selected by the actor, the agent can estimate the states and effective ages of all the components at the end of the mission along with the reward, i.e., the probability of the system successfully completing the next mission. As the iteration evolves, the actor adjusts its parameters for a higher Q-value from the critic, whereas the critic changes its parameters based on the rewards from the selected maintenance actions to accurately judge the effectiveness of the selected maintenance actions. An illustration of the customized DRL algorithm is given in Fig. 3.

Inspired by the Wolpertinger architecture (Dulac-Arnold et al., 2015), which can search actions in a large discrete action space, a postprocess is utilized in the actor. The two major techniques, namely, the experience replay and the target network of the deep Q-network (DQN) (Mnih et al., 2015), are used to train the networks in the customized DRL algorithm. Both the $\varepsilon$-greedy policy (Mnih et al., 2015) and the noise (Lillicrap et al., 2015) are utilized for exploration and exploitation. The basic elements of the customized DRL algorithm are elaborated in the ensuing sections.

### 4.1. Critic: Q-Network

In the actor-critic framework, a critic will evaluate the effectiveness of the actions selected by an actor. In this study, based on the states and effective ages of all the components in a system, the critic evaluates the effectiveness of the selected maintenance actions in each break by evaluating the maximum expected number of the successes of future missions, i.e., the Q-function. To overcome the "curse of dimensionality" of the state space, the critic is represented by a multi-layer artificial neural network to approximate the Q-function, i.e., $Q(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k) \approx Q^*(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k)$, where $Q(\cdot)$ is the artificial neural network with parameters $\theta_Q$. The artificial neural network is the so-called Q-network, and its structure is delineated in Fig. 4.

As shown in Fig. 4, at the end of the $k$th mission, the inputs of the Q-network consist of the states and effective ages of all the components, the number of completed missions, and the selected maintenance actions, i.e., $\mathbf{Y}_k$, $\mathbf{B}_k$, $k$, and $\mathbf{a}_k$. The output of the Q-network is the Q-value, representing the effectiveness of the selected maintenance actions. The nonlinearity rectifier, i.e., the rectified linear unit (ReLU), serves as the activation function of the Q-network.

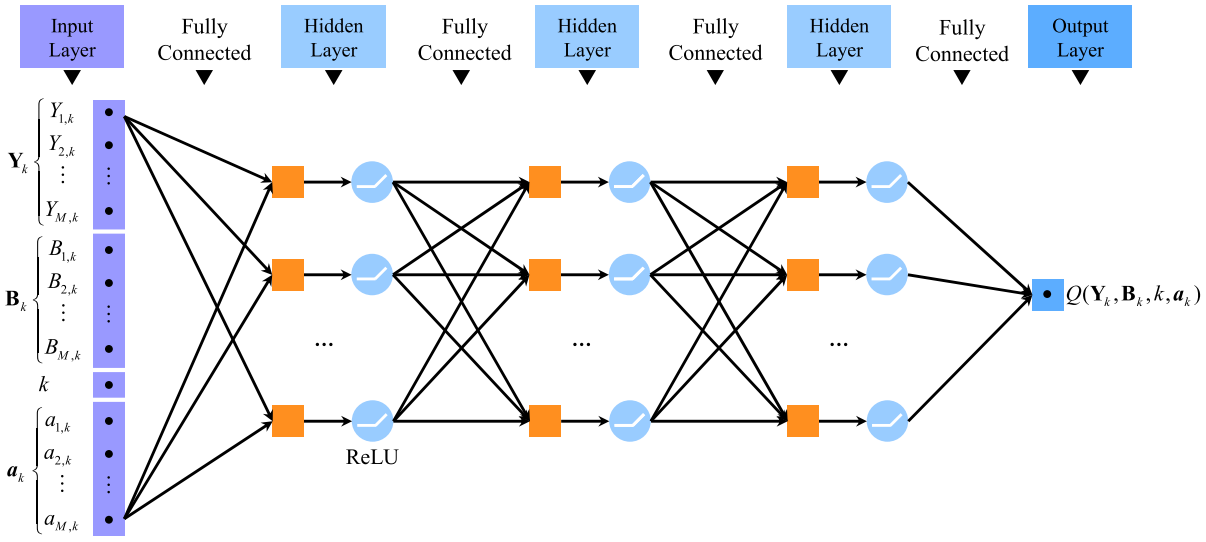


**Fig. 4.** The structure of the Q-network of the customized DRL algorithm.

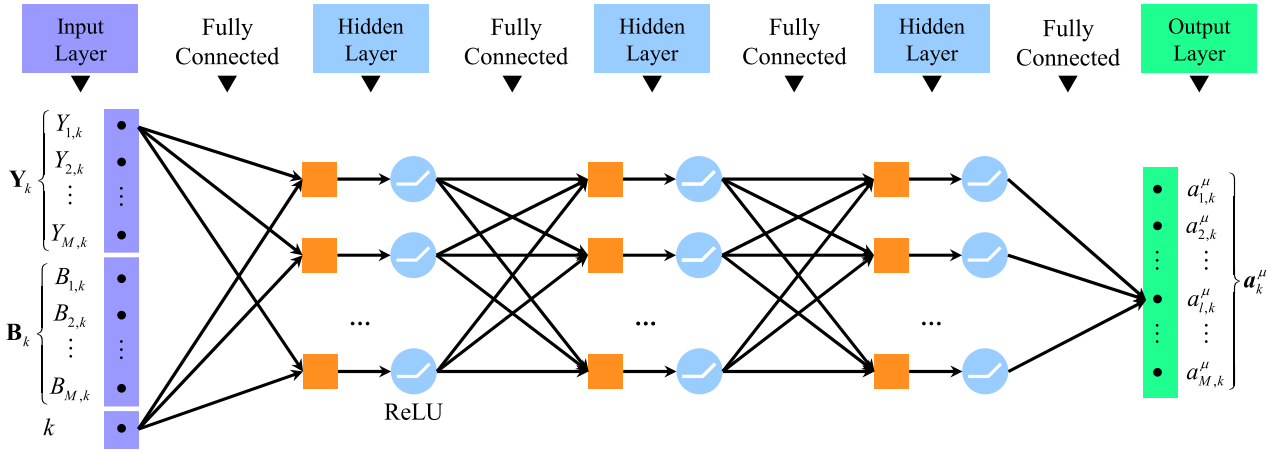


**Fig. 5.** The structure of the actor network of the customized DRL algorithm.

### 4.2. Actor: actor network and postprocess

In the actor-critic framework, an actor will select an action based on a specific situation. Identifying the optimal action via the Q-function is important for the DRL algorithm. In the problems with a small-scale action space, the optimal action can be selected by enumerating all the possible actions as reported in many existing approximate dynamic programming (ADP) and reinforcement learning (RL) algorithms (Sutton & Barto, 2018). However, enumerating all the possible actions is computationally unaffordable for large-scale problems. Likewise, in lieu of an exhaustive enumeration, a multi-layer artificial neural network, namely, the actor network, is also constructed to select the maintenance actions in this study. $\mu(\cdot)$ denotes the actor network and $\theta_\mu$ represents its parameters. The configuration of the actor network is presented in Fig. 5, where each hidden layer is followed by a nonlinearity rectifier.

The inputs of the actor network are composed of the states and effective ages of all the components at the end of the $k$th mission and the number of completed missions, i.e., $\mathbf{Y}_k$, $\mathbf{B}_k$, and $k$. The outputs of the actor network, denoted as $\mathbf{a}_k^\mu$, are the selected maintenance actions, namely, $\mathbf{a}_k^\mu = \mu(\mathbf{Y}_k, \mathbf{B}_k, k) = (a_{1,k}^\mu, a_{2,k}^\mu, \cdots, a_{M,k}^\mu)$, where $a_{l,k}^\mu$ is the maintenance action for component $l$ in the $k$th break. Each output of the actor network, i.e., $a_{l,k}^\mu$, is a continuous value and has to be converted into a discrete value to represent all the optional maintenance actions for component $l$. To deal with this problem and search the optimal maintenance actions for all the components in the large discrete action space, a postprocess, inspired by the Wolpertinger architecture developed by Dulac-Arnold et al. (2015), is used in this study. The illustration of the postprocess for a two-component case is delineated in Fig. 6, where the dark dots represent the optional maintenance actions for the two components and the purple dot is an output from the actor network, i.e., $\mathbf{a}_k^\mu$. All the optional maintenance actions neighboring to $\mathbf{a}_k^\mu$ can be iteratively identified by the postprocess, and the action with the maximum Q-value will be selected as the optimal maintenance actions for the two components. The detailed procedures are as follows.

*Step 1:* Produce a proto-action. By putting the states and effective ages of all the components and the number of completed missions, i.e., $\mathbf{Y}_k$, $\mathbf{B}_k$, and $k$, into the actor network, a continuous output vector, i.e., $\mathbf{a}_k^\mu = \mu(\mathbf{Y}_k, \mathbf{B}_k, k)$, can be obtained. Consequently, $\mathbf{a}_k^\mu$ can be normalized to an integer vector by rounding down each element in $\mathbf{a}_k^\mu$. The normalized maintenance actions for all the components are denoted as a proto-action, denoted as $\mathbf{a}_k^{\mathrm{p}}$.

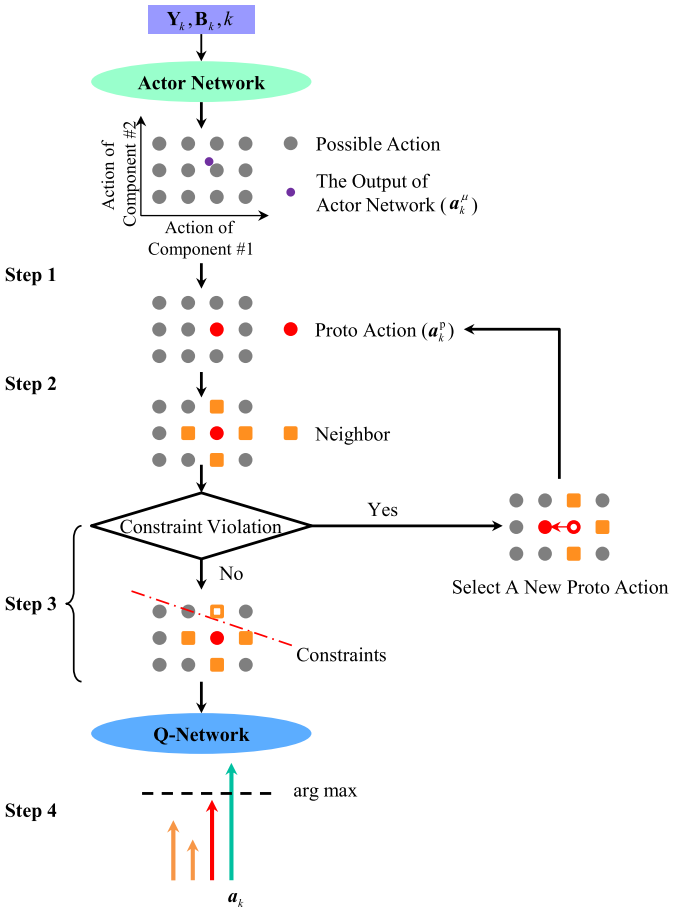


**Fig. 6.** Illustration of the postprocess for the actor.

*Step 2:* Find all the neighboring solutions to the proto action $\mathbf{a}_k^{\mathrm{p}}$. All the optional maintenance actions with the 2-norm distance to the proto action $\mathbf{a}_k^{\mathrm{p}}$ being less than a prespecified value of $L$ will be identified as the neighbors of the proto action $\mathbf{a}_k^{\mathrm{p}}$. A greater value of $L$ implies a larger local search space and a longer time consumption.

*Step 3:* Select the feasible neighboring solutions complying with all the constraints of maintenance resources. Evaluate the maintenance cost and time of each neighboring solution by Eqs. (4) and (6). If all of the neighboring solutions violate the constraints, the neighboring solution with the minimal maintenance cost and time among all the neighbors will be chosen as a new proto-action, and the iteration goes to Step 2 for finding all the neighbors to the new proto-action. Steps 2 and 3 will be repeatedly executed until at least one of the neighbors of the proto action satisfies all the constraints. The illustration of selecting the feasible neighbors around the proto action is delineated in Fig. 7, where the dark dots represent the optional maintenance actions for all the components. The neighbors violating the constraints will be eliminated during the search process.

*Step 4:* Identify the optimal maintenance actions. Based on $\mathbf{Y}_k$, $\mathbf{B}_k$, and $k$, the Q-values of all the feasible neighboring solutions around the proto action can be evaluated by the Q-network of the critic. The solution with the maximum Q-value will be chosen by the actor as the optimal maintenance actions for all the components.

The process of the actor selecting the maintenance actions for all the components can be represented by the policy function $\pi(\mathbf{Y}_k, \mathbf{B}_k, k|\mu, Q) = \mathbf{a}_k$, where $\mu$ and $Q$ represent the actor network and Q-network, respectively.

### 4.3. Agent training: experience replay and target network

In this study, a multi-state system is intended to execute $K$ consecutive missions. The customized DRL algorithm iteratively simulates the processes of executing missions and maintenance decision-making until reaching a prespecified maximum number of iterations $I_{\max}$. In each iteration, the agent sequentially selects maintenance actions for all the breaks. The detailed simulation procedure of the agent in the $k$th break is as follows: First, given the states and effective ages of all the components at the end of the $k$th mission, i.e., $\mathbf{Y}_k$ and $\mathbf{B}_k$, the agent executes the selected maintenance actions, i.e., $\mathbf{a}_k$, for all the components. Second, the probability of the repaired system successfully completing the $(k+1)$st mission, i.e., $R_{k+1}$, can be evaluated by Eq. (17). Third, the states and effective ages of all the components, i.e., $\mathbf{Y}_{k+1}$ and $\mathbf{B}_{k+1}$, at the end of the $(k+1)$st mission can be randomly simulated. After executing a simulation iteration, a transition realization denoted as $(\mathbf{Y}_k, \mathbf{B}_k, k, \mathbf{a}_k, R_{k+1}, \mathbf{Y}_{k+1}, \mathbf{B}_{k+1})$ can be recorded.

The agent can be iteratively trained by a number of recorded transitions, i.e., $N_D$ records, stored in a memory, denoted as $D$. The memory will be overwritten by the $N_D$ most recent transition records. However, each individual record in the memory $D$ has strong correlation with one another, and such correlations have an adverse impact on the agent training. To break the correlations among records, as in the DQN (Mnih et al., 2015), the experience replay technique is utilized in our study. Based on the experience replay technique, a minibatch with the size of $N_m (N_m < N_D)$ is used to update the Q-network and actor network by randomly sampling from the memory $D$.

In addition to the experience replay technique, the target network technique is also used to enhance the robustness of the proposed DRL algorithm. After every $C$ iterations of the foregoing simulation of the customized DRL algorithm, the Q-network and the actor network are cloned to obtain a target Q-network and a target actor network, respectively, and the two target networks are used to generate the target values for the update of the Q-network and the actor network in each iteration. $C$ is the frequency for the target network update. The target Q-network is denoted as $\hat{Q}(\cdot)$ with parameters $\hat{\theta}_Q$, whereas the target actor network is denoted as $\hat{\mu}(\cdot)$ with parameters $\hat{\theta}_\mu$.

Through the experience replay and target network techniques, the agent can iteratively update its Q-network and actor network by the transition records of the minibatch. The $j$th record in the minibatch is denoted as $(\mathbf{Y}_{k_j,j}, \mathbf{B}_{k_j,j}, k_j, \mathbf{a}_{k_j,j}, R_{k_j+1,j}, \mathbf{Y}_{k_j+1,j}, \mathbf{B}_{k_j+1,j})$, and the target value of the Q-network, denoted as $y_j$, can be computed by the following:

$$y_j = \begin{cases} R_{k_j+1,j} & k_j = K-1 \\ R_{k_j+1,j} + \hat{Q}(\mathbf{Y}_{k_j+1,j}, \mathbf{B}_{k_j+1,j}, k_j+1, \\ \quad \pi(\mathbf{Y}_{k_j+1,j}, \mathbf{B}_{k_j+1,j}, k_j+1|\hat{\mu}, \hat{Q})) & k_j < K-1 \end{cases}, \tag{26}$$

where the target value of the Q-network is evaluated by the two target networks, i.e., the target Q-network and target actor network. Because the maximum expected number of the successes of future remaining $K-k_j-1$ missions falls in the range of $[0, K-k_j-1]$, Eq. (26) can be further written as follows:

$$y_j = \begin{cases} R_{k_j+1,j} & k_j = K-1 \\ R_{k_j+1,j} + \max\{0, min\{K-k_j-1, q_j\}\} & k_j < K-1 \end{cases}, \tag{27}$$

where $q_j = \hat{Q}(\mathbf{Y}_{k_j+1,j}, \mathbf{B}_{k_j+1,j}, k_j+1, \pi(\mathbf{Y}_{k_j+1,j}, \mathbf{B}_{k_j+1,j}, k_j+1|\hat{\mu}, \hat{Q}))$. The parameters of the Q-network can be updated by the following:

$$\theta_Q = \arg\min_{\theta_Q} \sum_{j=1}^{N_m} (y_j - Q(\mathbf{Y}_{k_j,j}, \mathbf{B}_{k_j,j}, k_j, \mathbf{a}_{k_j,j}))^2. \tag{28}$$

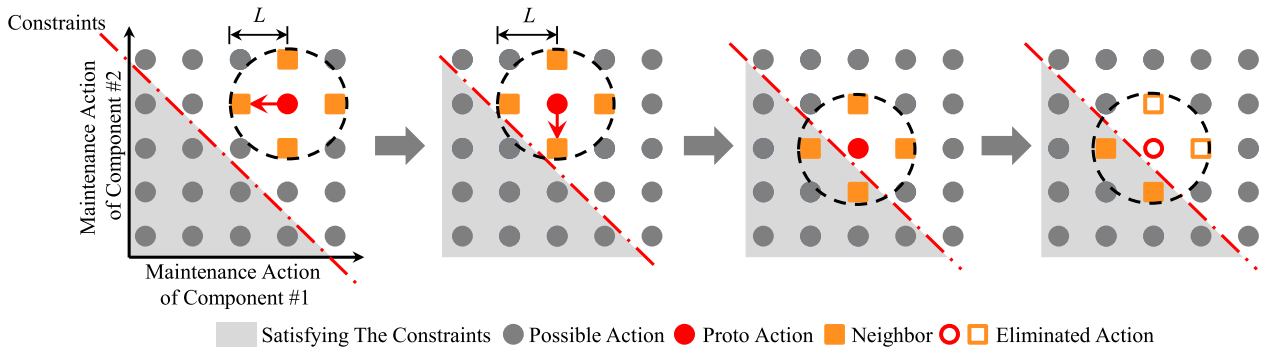


**Fig. 7.** Illustration of searching for feasible neighboring solutions.

In this study, the Levenberg-Marquardt algorithm (Hagan & Menhaj, 1994), which executes several iterative steps to resolve nonlinear least squares problems, is implemented to resolve Eq. (28). To avoid overfitting, the Levenberg–Marquardt algorithm only executes one step in Eq. (28) with respect to $\theta_Q$.

In the same fashion, the target value of the actor network of the $j$th record in the minibatch, denoted as $\hat{\mathbf{a}}_{k_j,j}$, is generated by the Q-network and target actor network, and $\hat{\mathbf{a}}_{k_j,j}$ can be, therefore, written as follows:

$$\hat{\mathbf{a}}_{k_j,j} = \pi\left(\mathbf{Y}_{k_j,j}, \mathbf{B}_{k_j,j}, k_j | \hat{\mu}, Q\right). \tag{29}$$

The parameters of the actor network can be updated by the following:

$$\theta_\mu = \arg\min_{\theta_\mu} \sum_{j=1}^{N_m} \left\| \hat{\mathbf{a}}_{k_j,j} - \mu\left(\mathbf{Y}_{k_j,j}, \mathbf{B}_{k_j,j}, k_j\right) \right\|_2, \tag{30}$$

where $\|\cdot\|_2$ is the 2-norm of a vector. The Levenberg–Marquardt algorithm carries out one step on Eq. (30) with respect to $\theta_\mu$ to avoid overfitting.

The Q-network and the actor network are iteratively updated by Eqs. (26)–(30), whereas after every $C$ iterations, the target Q-network and target actor network are updated by the following:

$$\begin{cases} \hat{\theta}_Q = \theta_Q \\ \hat{\theta}_\mu = \theta_\mu \end{cases}. \tag{31}$$

### 4.4. Exploration and exploitation

The "exploration-exploitation dilemma" is a crucial problem of the customized DRL algorithm. Exploring too many possible maintenance actions results in computational inefficiency, and the algorithm may fall into the local optimum due to excessive exploitation. In this study, a $\varepsilon$-greedy policy and noise are introduced to address these phenomena.

At the beginning of the customized DRL algorithm, the Q-network and actor network are randomly initialized, and the two target networks are initialized by Eq. (31). In the first 10% of iterations, the agent randomly performs maintenance actions for all the components to explore the action space. In the remaining iterations, the agent executes maintenance actions with a $\varepsilon$-greedy policy, namely, randomly performing maintenance actions for all the components with a probability of $\varepsilon$ or executing maintenance actions selected by the actor with a probability of $1-\varepsilon$.

In addition to the $\varepsilon$-greedy policy, for exploration, noise denoted as a vector of $\mathbf{N}_\pi = (N_{\pi,1}, N_{\pi,2}, \ldots, N_{\pi,l}, \ldots, N_{\pi,M})$ is included by the actor to select maintenance actions, where $N_{\pi,l}$ is noise to be added to the selected maintenance action for component $l$. Hence, the maintenance actions for all the components in the $k$th break becomes $\pi\left(\mathbf{Y}_k, \mathbf{B}_k, k | Q, \mu\right) + \mathbf{N}_\pi$. In this study, noise is generated from the normal distribution as follows:

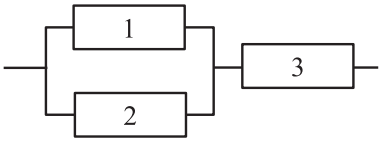


**Fig. 8.** Configuration of a three-component system.

$$N_{\pi,l} \sim \frac{I_{\max} - I}{I_{\max}} \cdot N(0, \sigma^2), \tag{32}$$

where $I$ is the index of iterations; $N(0,\sigma^2)$is a normally distributed random number with a mean of zero and a variance of $\sigma^2$. As the iteration evolves, the magnitude of the noise will gradually decrease.

In summary, the pseudocode of the customized DRL algorithm for our specific dynamic selective maintenance problem is provided in the Appendix.

## 5. Illustrative examples

In this section, two illustrative examples are presented to demonstrate the effectiveness of the proposed method. A three-component flow transmission system with the exact solution is first designed to examine the accuracy of the proposed DRL algorithm, whereas a coal transportation system is presented as the second illustrative example to demonstrate the implementation of the proposed method to large-scale systems.

### 5.1. A three-component system

The configuration of a three-component flow transmission system is delineated in Fig. 8. Components #1 and #2 are connected in parallel, whereas the two components are connected with Component #3 in series. The structure function of the system is defined as: $G(t) = \min\{G_1(t) + G_2(t), G_3(t)\}$.

Following the studies reported in Liu and Huang (2010) and Pandey, Zuo, Moghaddass and Tiwari (2013), if $a_{l,k} > 0$, the variable maintenance cost and time associated with a particular maintenance action $a_{l,k}$ are respectively formulated as follows:

$$c_l(a_{l,k}, Y_{l,k}) = \begin{cases} c_l^{\mathrm{rf}} \cdot \dfrac{a_{l,k}-1}{N_L - 1} & Y_{l,k} = 0 \\ c_l^{\mathrm{rp}} \cdot \dfrac{a_{l,k}}{N_L} & Y_{l,k} = 1 \end{cases}, \tag{33}$$

$$t_l(a_{l,k}, Y_{l,k}) = \begin{cases} t_l^{\mathrm{rf}} \cdot \dfrac{a_{l,k}-1}{N_L - 1} & Y_{l,k} = 0 \\ t_l^{\mathrm{rp}} \cdot \dfrac{a_{l,k}}{N_L} & Y_{l,k} = 1 \end{cases}. \tag{34}$$

**Table 1**
The parameter settings of each component, where the performance capacity is in tons/hour, time is in days, and costs are in $10^3$ US dollars.

| Component ID | $g_l$ | $\beta_l$ | $\eta_l$ | $m_l^{\mathrm{p}}$ | $c_l^{\mathrm{rp}}$ | $t_l^{\mathrm{rp}}$ | $m_l^{\mathrm{f}}$ | $c_l^{\mathrm{rf}}$ | $t_l^{\mathrm{rf}}$ | $c_l^0$ | $t_l^0$ | $A_{l,1}$ | $X_{l,1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 55 | 1.5 | 25 | 2.5 | 10 | 0.5 | 2.5 | 15 | 0.5 | 2 | 0.1 | 0 | 1 |
| 2 | 80 | 2.4 | 38 | 2.2 | 10 | 0.5 | 2.0 | 15 | 0.5 | 2 | 0.1 | 0 | 1 |
| 3 | 90 | 2.6 | 40 | 2.2 | 26 | 0.8 | 3.2 | 40 | 1 | 5 | 0.3 | 0 | 1 |

The efficiency of a particular maintenance action $a_{l,k}$ in the $k$th break is quantified by defining the age reduction factor $b_{l,k}$ of the maintenance action as a function of the variable maintenance cost of component $l$, and it is formulated as follows:

$$b_{l,k} = \begin{cases} 1 - \left( \dfrac{c_l(a_{l,k}, 0)}{c_l^{\mathrm{rf}}} \right)^{\frac{1}{m_l^{\mathrm{f}}}} & Y_{l,k} = 0 \\ 1 - \left( \dfrac{c_l(a_{l,k}, 1)}{c_l^{\mathrm{rp}}} \right)^{\frac{1}{m_l^{\mathrm{p}}}} & Y_{l,k} = 1 \end{cases}, \tag{35}$$

where $c_l^{\mathrm{rf}}$ and $c_l^{\mathrm{rp}}$ are the corrective and preventive replacement costs for a failed component and a functioning component, respectively, namely, $c_l^{\mathrm{rf}} = c_l(N_L, 0)$ and $c_l^{\mathrm{rp}} = c_l(N_L, 1)$. $m_l^{\mathrm{f}}$ $(m_l^{\mathrm{f}} > 0)$ and $m_l^{\mathrm{p}}$ $(m_l^{\mathrm{p}} > 0)$ are two characteristic parameters that can precisely determine the relationship between the variable maintenance cost and the age reduction factor of a failed component $l$ and a functioning component $l$, respectively; and these parameters can be inferred by historical data and/or experts' judgements as mentioned in Doyen and Gaudoin (2004)), Gasmi, Love and Kahle (2003), and Liu, Huang and Zhang (2012). In this example, all the selected maintenance actions are executed sequentially, hence the total maintenance time to be spent in the $k$th break, i.e., $T_k$, is given by:

$$T_k = f(T_{1,k}, T_{2,k}, \cdots, T_{M,k}) = \sum_{l=1}^{M} T_{l,k}. \tag{36}$$

The studied system is in the brand new condition and is intended to execute three missions with a maintenance budget $C_{\mathrm{lim}}$ =50×$10^3$ US dollars and a limited duration of $T_{\mathrm{lim}}$ =3 days for each break. The pre-specified demand of each mission is set at 50 tons/hour, and the duration of each mission is set at $Z_k$=10 days ($k = 1, 2, 3$). In this case, three optional maintenance actions are available to each component, i.e., $N_L$ =2. The failure time of component $l$ is assumed to conform to the Weibull distribution with the shape parameter $\beta_l$ and the scale parameter $\eta_l$. All the parameter settings for each component, including the performance capacity, the maintenance cost, the parameters of the failure time distribution, the initial effective ages and states of all the components are tabulated in Table 1.

The dynamic selective maintenance optimization problem is resolved by the proposed DRL algorithm with the aim of maximizing the expected number of the successes of the three consecutive missions. Four artificial neural networks, namely, the Q-network, target Q-network, actor network, and target actor network, are constructed, each of which has three hidden layers and each hidden layer consists of ten neurons. The distance $L$ in the postprocess is set at 1. The size of the memory is set at $N_D$=200, and the size of the minibatch is set at $N_m$=32. The parameter of the $\varepsilon$-greedy policy is set at $\varepsilon$=0.01. The noise is set to be normally distributed with the distribution of $N_{\pi,l} \sim (I_{\max} - I)/I_{\max} \cdot N(0, 0.5^2)$. The maximum number of iterations for agent training is set at $I_{\max}$=1000, and all the target networks will be dynamically updated with every ten iterations.

**Table 2**
The comparison between the dynamic programming and the proposed deep reinforcement learning method, where the runtime is in seconds.

| Methods | $N$ | Relative Error | Runtime |
|---|---|---|---|
| Dynamic Programming | 2.8856 | – | $1.1989 \times 10^4$ |
| The Proposed DRL Algorithm | 2.8886 | 0.10% | $1.1160 \times 10^3$ |

The runtime of training neural networks is $1.1160 \times 10^3$ seconds on a PC with an Intel Core (TM) i5-4590 3.30GHz CPU and 8 G RAM. The expected number of successes of the three missions is 2.8886 as evaluated by the Q-network, whereas the average number of successful missions is 2.8837 from a Monte Carlo simulation with $10^6$ samples. The Q-network can therefore obtain an accurate estimation of the expected number of the successes of the three missions.

To further examine the effectiveness of the proposed DRL algorithm, the dynamic programming algorithm is implemented to solve the problem and serves as the benchmark for comparison. As the dynamic programming algorithm can only tackle with a discrete state space, the effective age of each component is discretized with an interval of 0.5 days. The expected number of successes of the three missions of the dynamic programming algorithm and the proposed deep reinforcement learning method are presented in Table 2. As presented in Table 2, for the dynamic programming algorithm, the maximum expected number of successes of the three missions is 2.8856, which is close to the result from the proposed DRL algorithm. However, the runtime of the dynamic programming algorithm is $1.1989 \times 10^4$ seconds, which is much greater than the runtime of the proposed DRL algorithm. On the other hand, the computational time and the memory space consumption of the dynamic programming algorithm will significantly increase with respect to the number of discretized effective ages.

The maximum expected number of successes of the three missions varies for both the maintenance budget $C_{\mathrm{lim}}$ and the duration $T_{\mathrm{lim}}$ of each break. By changing the maintenance budget from 0 to $80 \times 10^3$ US dollars and the break duration from 0 to 3 days, the maximum expected number of the successes of the three missions was found by the proposed DRL algorithm, as shown in Fig. 9. The figure shows that with the increase of both the maintenance budget and the break duration, the maximum expected number of successes of the three missions gradually approaches a value of 2.8935, which is three times of the probability of a brand new system successfully completing a mission.

### 5.2. A multi-state coal transportation system

A multi-state coal transportation system introduced in Liu and Huang (2010) is illustrated here to examine the effectiveness of the proposed method for a large-scale system. The system is composed of five subsystems as shown in Fig. 10. Feeder #1, consisting of three components, loads coal from the bin to Conveyor #1. Conveyor #1 has two components for transferring the coal to the stacker reclaimer, which has three components. Feeder

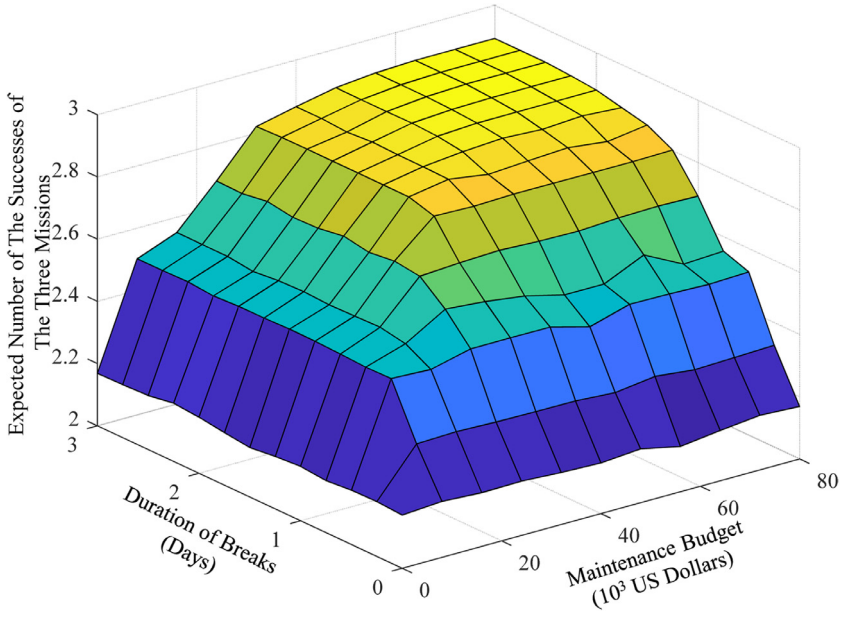


**Fig. 9.** The expected number of successes of the three missions vs. the maintenance budget and the break durations.

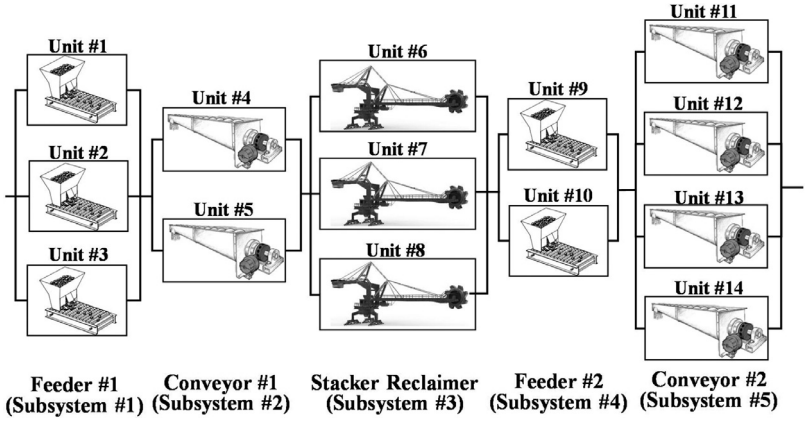


**Fig. 10.** The configuration of a multi-state coal transportation system.

#2 is composed of two components and can load coal from the stacker reclaimer to the Conveyor #2, which has four components. The failure time of each component conforms to the Weibull distribution with shape parameter $\beta_l$ and scale parameter $\eta_l$ ($l \in \{1, 2, \ldots, 14\}$). The parameters for all the components follow the settings reported in Liu and Huang (2010) and are tabulated in Table 3. The system is in brand new condition and is intended to execute five consecutive missions with a break in-between two adjacent missions. The maintenance budget and the duration of each break are set at $200 \times 10^3$ US dollars and 3 days, respectively. The duration of each mission is set at $Z_k$ =10 days ($k = 1, 2, \ldots, 5$). During each break, one of the eight optional maintenance actions, i.e., $N_L$=7, from replacement down to "doing nothing", can be selected for each component. The maintenance cost and time associated with each optional maintenance action are given by Eqs. (33) and (34), whereas the age reduction factor of each maintenance action is formulated by Eq. (35). All the selected maintenance actions are conducted sequentially and the total maintenance time to be spent in the $k$th break can be obtained by Eq. (36).

Based on the structure function of the coal transportation system, the performance capacity of the entire system can be determined by the performance capacities of all the components as follows:

$$\begin{aligned} G(t) = \min\{&G_1(t) + G_2(t) + G_3(t), G_4(t) + G_5(t), \\ &G_6(t) + G_7(t) + G_8(t), G_9(t) + G_{10}(t), \\ &G_{11}(t) + G_{12}(t) + G_{13}(t) + G_{14}(t)\}. \end{aligned} \tag{37}$$

The demand of each mission is a random variable with the probability distribution as shown in Table 4.

Without taking into account the constraints of the maintenance budget and the duration of each break, the size of the action space is $8^{14} = 4.3980 \times 10^{12}$, whereas the state space is composed by $2^{14} = 16384$ possible combinations of the states of all the components and the uncountable combinations of the effective ages of all the components. In this case, the dynamic programming is inapplicable due to the huge action and state spaces.

**Table 3**
The parameter settings of each component, where the performance capacity is in tons/hour, time is in days, and costs are in $10^3$ US dollars.

| ID | $g_l$ | $\beta_l$ | $\eta_l$ | $m_l^{\mathrm{p}}$ | $c_l^{\mathrm{rp}}$ | $t_l^{\mathrm{rp}}$ | $m_l^{\mathrm{f}}$ | $c_l^{\mathrm{rf}}$ | $t_l^{\mathrm{rf}}$ | $c_l^0$ | $t_l^0$ | $A_{l,1}$ | $X_{l,1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 55 | 1.5 | 25 | 2.5 | 15 | 0.13 | 2.5 | 25 | 0.25 | 3 | 0.03 | 0 | 1 |
| 2 | 80 | 2.4 | 38 | 2.2 | 20 | 0.20 | 2.0 | 32 | 0.31 | 4 | 0.03 | 0 | 1 |
| 3 | 120 | 1.6 | 28 | 2.6 | 25 | 0.20 | 3.0 | 35 | 0.33 | 3 | 0.03 | 0 | 1 |
| 4 | 90 | 2.6 | 40 | 2.2 | 20 | 0.12 | 3.2 | 35 | 0.32 | 5 | 0.04 | 0 | 1 |
| 5 | 145 | 1.8 | 28 | 1.8 | 25 | 0.21 | 4.0 | 34 | 0.34 | 2 | 0.02 | 0 | 1 |
| 6 | 70 | 2.4 | 34 | 2.4 | 15 | 0.14 | 3.2 | 20 | 0.19 | 3 | 0.03 | 0 | 1 |
| 7 | 95 | 2.5 | 26 | 2.8 | 24 | 0.20 | 3.0 | 30 | 0.27 | 6 | 0.05 | 0 | 1 |
| 8 | 80 | 2.0 | 28 | 2.3 | 20 | 0.17 | 2.8 | 35 | 0.31 | 5 | 0.05 | 0 | 1 |
| 9 | 95 | 1.2 | 26 | 2.0 | 18 | 0.18 | 2.5 | 28 | 0.26 | 3 | 0.04 | 0 | 1 |
| 10 | 130 | 1.4 | 35 | 2.5 | 20 | 0.20 | 2.8 | 35 | 0.32 | 6 | 0.05 | 0 | 1 |
| 11 | 50 | 2.8 | 40 | 3.2 | 22 | 0.21 | 3.0 | 32 | 0.31 | 7 | 0.07 | 0 | 1 |
| 12 | 75 | 1.5 | 35 | 2.6 | 25 | 0.23 | 2.2 | 35 | 0.33 | 4 | 0.04 | 0 | 1 |
| 13 | 85 | 2.4 | 30 | 2.8 | 18 | 0.16 | 2.8 | 36 | 0.35 | 6 | 0.06 | 0 | 1 |
| 14 | 95 | 2.2 | 45 | 2.2 | 15 | 0.14 | 2.6 | 38 | 0.35 | 3 | 0.05 | 0 | 1 |

**Table 4**
Mission demand.

| Demand(ton/hour) | 120 | 90 | 60 | 30 | 10 |
|---|---|---|---|---|---|
| Probability | 0.1 | 0.25 | 0.35 | 0.2 | 0.1 |

Alternatively, through the proposed DRL algorithm, the "curse of dimensionality" can be overcome. The parameters settings for the proposed DRL algorithm are set at: the memory size $N_D$=200, the minibatch size $N_m$=32, the maximum number of iterations for agent training $I_{\max}$=5000, the frequency of updating the target network $C$=50, the $\varepsilon$-greedy parameter $\varepsilon$=0.01, and the noise $N_{\pi,l} \sim (I_{\max} - I)/I_{\max} \cdot N(0, 0.5^2)$. Each of the four artificial neural networks, i.e., the Q-network, target Q-network, actor network, and target actor network, has three hidden layers, each of which consists of ten neurons. The distance $L$ in the postprocess is set at 1. As shown in Fig. 11, the maximum expected number of successes of the five missions estimated by the Q-network and target Q-network in the proposed DRL algorithm evolves iteratively. At the

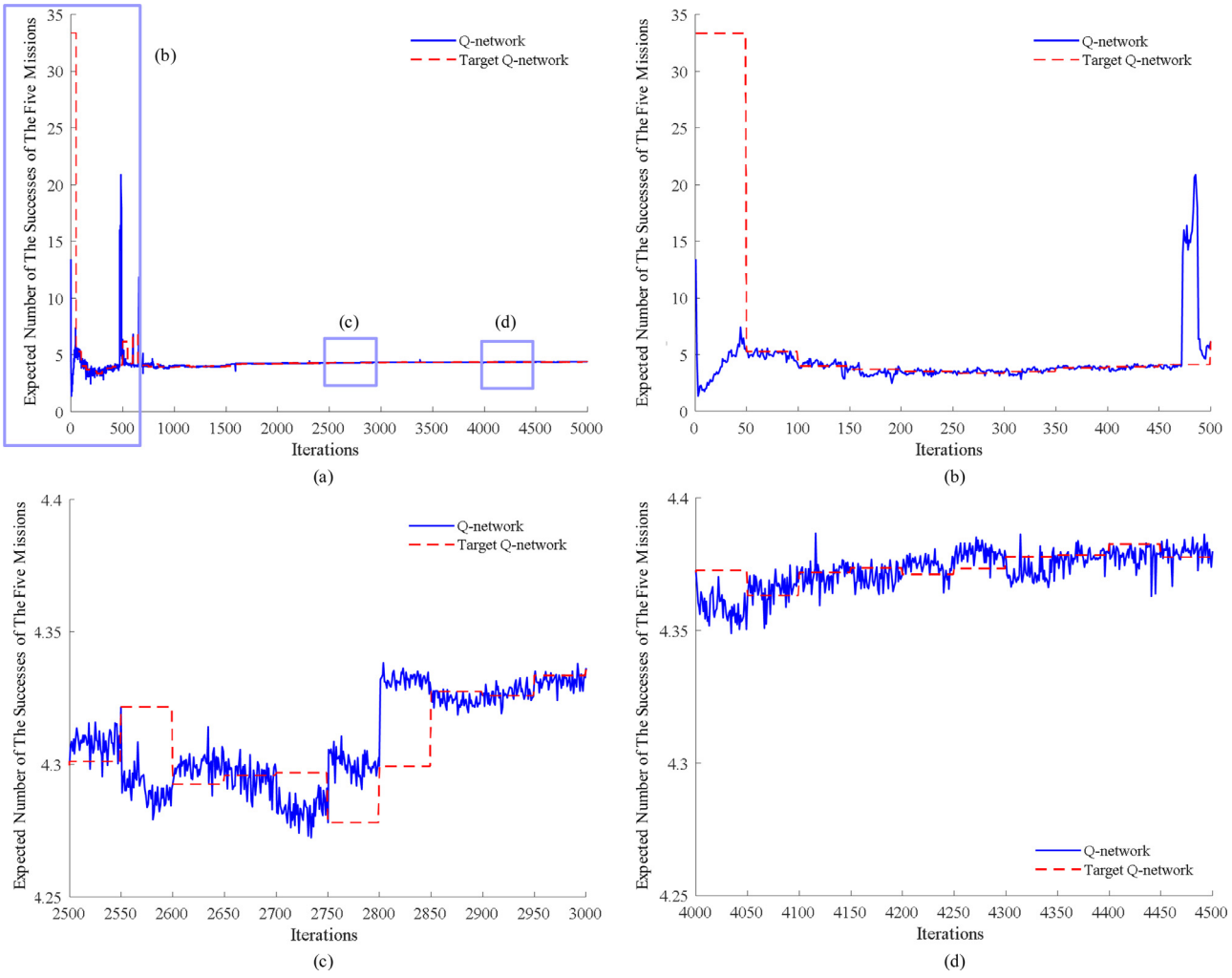


**Fig. 11.** The training process of the proposed DRL algorithm. (a) The training process of all the iterations; (b) The training process of the first 500 iterations; (c) The training process of the 2500th to 3000th iterations; (d) The training process of the 4000th to 4500th iterations.

beginning of the first 500 iterations, the agent randomly explored all the possible maintenance actions, and the convergence of the values of the Q-network and target Q-network was relatively slow as seen in Fig. 11(b). However, after the first 2500 iterations, the values of the Q-network and target Q-network gradually converged as shown in Fig. 11(c) and Fig. 11(d).

The estimated maximum expected number of successes of the five missions is 4.3859 when the iteration reaches $I_{\max}$, and the runtime is $6.8581 \times 10^3$ seconds on a PC with an Intel Core (TM) i5-4590 3.30 GHz CPU and 8 G RAM. By using the Monte Carlo simulation with $10^6$ samples, the agent can successfully complete 4.3721 missions on average. The absolute error of the Q-network with respect to the result from the Monte Carlo simulation is 0.0138, with the relative error being less than 0.3%.

The effectiveness of taking into account the imperfect maintenance is illustrated by comparing the results from the case where only three optimal maintenance actions, i.e., "doing nothing", minimal repair, and replacement, can be selected. The maximum expected number of the successes of the five missions for this case was resolved by the proposed DRL algorithm, and it is only 3.3817 which is much inferior to the case of imperfect maintenance.

## 6. Conclusions and future works

In this paper, a new dynamic selective maintenance optimization for multi-state systems that can execute multiple consecutive missions over a finite horizon was developed. To maximize the successes of future missions, the aged or failed components can be maintained/repaired during the break in-between two adjacent missions, and multiple optional maintenance actions can be chosen for each component. The resulting stochastic dynamic programming was formulated as a discrete-time finite-horizon Markov decision process with a mixed integer-discrete-continuous state space. Based on the framework of actor-critic algorithms, the DRL algorithm was customized to overcome the "curse of dimensionality" and cope with the uncountable state space, and two multilayer artificial neural networks were constructed for the critic and actor, respectively. A postprocess was proposed for the actor to select the optimal maintenance actions for all the components in a large discrete action space subject to the constraints of maintenance resources. The experience replay and target network were implemented to enhance the robustness of the customized DRL algorithm. As demonstrated in the two illustrative examples, the proposed DRL approach can dynamically identify the optimal maintenance actions for all the components in a computationally efficient manner.

It is noteworthy that there are several challenges to be addressed in our future work. Firstly, in this study, the duration of each individual mission, maintenance action, and break are assumed to be deterministic. Such an assumption will be released to accommodate the stochastic case (Khatab et al., 2017; Liu et al., 2018). Nevertheless, if the durations of breaks and maintenance actions are stochastic, the sequence of maintenance actions will produce a significant impact on the success of the next mission (Liu et al., 2018). On the other hand, the uncertainty associated with the durations of missions will also post a challenge in terms of evaluating the probability of a system successfully completing a mission. Secondly, the system configuration is assumed to be unchanged throughout all the missions. In some engineering cases, the system configuration and/or components' failure behaviors may vary from one mission to another. Thirdly, as a preliminary study, the maintenance budget for each break is assumed to be pre-specified. However, in some engineering applications, the maintenance budget could be shared among all the breaks, and the maintenance budget for each break may be, therefore, distinct. A more efficient maintenance strategy could be identified by taking account of the shared total maintenance budget. Lastly, in some engineering situations, a multi-state system can keep operating even if the system demand cannot meet the demand (Levitin, Xing & Huang, 2019). In such a circumstance, the reward of the MDP should be associated with the expected unsupplied demand to reflect the performance of a repaired system in future missions.

## Acknowledgement

The authors greatly acknowledge grant support from the National Natural Science Foundation of China under contract number 71771039 and 71922006.

## Appendix

Fig. 12.

$I_{\max}$ : {The maximum number of iterations}
$I$ : {The iteration index}
$N_D$ : {The size of memory}
$N_m$ : {The size of minibatch}
$K$ : {The number of missions}
$C$ : {The frequency of target network updating}
Randomly initialize the Q-network and actor network with parameters $\theta_Q$ and $\theta_\mu$;
Initialize the two target networks $\hat{\theta}_Q = \theta_Q$ and $\hat{\theta}_\mu = \theta_\mu$;
Initialize the memory and minibatch;
**For** $I = 1$ to $I_{\max}$
    Initialize the states and effective ages of all the components at the beginning of the first mission, i.e., $\mathbf{X}_1$ and $\mathbf{A}_1$;
    Execute the first mission and obtain the states and effective ages of all the components at the end of the first mission, i.e., $\mathbf{Y}_1$ and $\mathbf{B}_1$;
    **For** $k = 1$ to $K - 1$
        **If** $I \le 0.1 I_{\max}$
            Randomly select maintenance actions;
        **Else**
            Select maintenance actions $\pi(\mathbf{Y}_k, \mathbf{B}_k, k \mid Q, \mu) + \mathbf{N}_\pi$ with the probability of $1 - \varepsilon$, or randomly select maintenance actions with the probability of $\varepsilon$;
            Normalize the selected maintenance actions;
        **End if**
        $R_{k+1} = R(\mathbf{Y}_k, \mathbf{B}_k, \boldsymbol{a}_k)$;
        Simulate the states and effective ages of all the components at the end of the $(k+1)$st mission, i.e., $\mathbf{Y}_{k+1}$ and $\mathbf{B}_{k+1}$;
        Store the transition record $(\mathbf{Y}_k, \mathbf{B}_k, k, \boldsymbol{a}_k, R_{k+1}, \mathbf{Y}_{k+1}, \mathbf{B}_{k+1})$ in memory $D$;
    **End for**
    Sample a random minibatch of $N_m$ transition records from memory $D$;
    Calculate the target value $y_j$ of the Q-network by Eq. (27);
    Execute the Levenberg-Marquardt algorithm for one step on Eq.(28) with respect to the Q-network parameters $\theta_Q$;
    Calculate the target value of the actor network by Eq. (29);
    Execute the Levenberg-Marquardt algorithm for one step on Eq. (30) with respect to the actor network parameters $\theta_\mu$;
    Update the two target networks for every $C$ iterations: $\hat{\theta}_Q = \theta_Q$ and $\hat{\theta}_\mu = \theta_\mu$;
**End for**
Output the target Q-network and target actor network.

**Fig. 12.** The pseudocode of the customized DRL algorithm.

## References


Ahadi, K. (2018). *Optimization methods for maintaining complex systems*. Fayetteville: University of Arkansas.

Amari, S. V., Xing, L., Shrestha, A., Akers, J., & Trivedi, K. S. (2010). Performability analysis of multistate computing systems using multivalued decision diagrams. *IEEE Transactions on Computers, 59*(10), 1419–1433.

Andriotis, C. P., & Papakonstantinou, K. G. (2019). Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety, 191*, 106483.

Cassady, C. R., Murdock, W. P., & Pohl, E. A. (2001). Selective maintenance for support equipment involving multiple maintenance actions. *European Journal of Operational Research, 129*(2), 252–258.

Cassady, C. R., Pohl, E. A., & Murdock, W. P. (2001). Selective maintenance modeling for industrial systems. *Journal of Quality in Maintenance Engineering, 7*(2), 104–117.

Dao, C. D., & Zuo, M. J. (2016). Selective maintenance for multistate series systems with s-dependent components. *IEEE Transactions on Reliability, 65*(2), 525–539.

Dao, C. D., & Zuo, M. J. (2017). Selective maintenance of multi-state systems with structural dependence. *Reliability Engineering & System Safety, 159*, 184–195.

Dao, C. D., Zuo, M. J., & Pandey, M. (2014). Selective maintenance for multi-state series-parallel systems under economic dependence. *Reliability Engineering & System Safety, 121*, 240–249.

Dijoux, Y., Fouladirad, M., & Nguyen, D. T. (2016). Statistical inference for imperfect maintenance models with missing data. *Reliability Engineering & System Safety, 154*, 84–96.

Doyen, L., & Gaudoin, O. (2004). Classes of imperfect repair models based on reduction of failure intensity or virtual age. *Reliability Engineering & System Safety, 84*(1), 45–56.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J. et al. (2015). Deep reinforcement learning in large discrete action spaces. arXiv:1512.07679.

Ekin, T. (2018). Integrated maintenance and production planning with endogenous uncertain yield. *Reliability Engineering & System Safety, 179*, 52–61.
Gasmi, S., Love, C. E., & Kahle, W. (2003). A general repair, proportional-hazards, framework to model complex repairable systems. *IEEE Transactions on Reliability, 52*(1), 26–32.
Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks, 5*(6), 989–993.
Iyoob, I. M., Cassady, C. R., & Pohl, E. A. (2006). Establishing maintenance resource levels using selective maintenance. *The Engineering Economist, 51*(2), 99–114.
Jiang, T., & Liu, Y. (2017). Parameter inference for non-repairable multi-state system reliability models by multi-level observation sequences. *Reliability Engineering & System Safety, 166*, 3–15.
Khatab, A., Aghezzaf, E. H., & Claver, D. (2015). Maintenance optimization of series–parallel systems operating missions with scheduled breaks. In Proceedings *of* the international conference on microelectronics.
Khatab, A., Aghezzaf, E. H., Djelloul, I., & Sari, Z. (2017). Selective maintenance optimization for systems operating missions and scheduled breaks with stochastic durations. *Journal of Manufacturing Systems, 43*, 168–177.
Kijima, M. (1989). Some results for repairable systems with general repair. *Journal of Applied Probability, 26*(1), 89–102.
Kijima, M., Morimura, H., & Suzuki, Y. (1988). Periodical replacement problem without assuming minimal repair. *European Journal of Operational Research, 37*(2), 194–203.
Levitin, G. (2005). *The universal generating function in reliability analysis and optimization*. London: Springer.
Levitin, G., Finkelstein, M., & Dai, Y. (2017). Redundancy optimization for series-parallel phased mission systems exposed to random shocks. *Reliability Engineering & System Safety, 167*, 554–560.
Levitin, G., & Lisnianski, A. (2000). Optimization of imperfect preventive maintenance for multi-state systems. *Reliability Engineering & System Safety, 67*(2), 193–203.
Levitin, G., Xing, L., & Huang, H. Z. (2019). Dynamic availability and performance deficiency of common bus systems with imperfectly repairable components. *Reliability Engineering & System Safety, 189*, 58–66.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y. et al. (2015). Continuous control with deep reinforcement learning. arXiv:1509.02971
Liu, Y., Chen, Y., & Jiang, T. (2018). On sequence planning for selective maintenance of multi-state systems under stochastic maintenance durations. *European Journal of Operational Research, 268*(1), 113–127.
Liu, Y., & Huang, H. Z. (2010). Optimal selective maintenance strategy for multi-state systems under imperfect maintenance. *IEEE Transactions on Reliability, 59*(2), 356–367.
Liu, Y., Huang, H. Z., & Zhang, X. (2012). A data-driven approach to selecting imperfect maintenance models. *IEEE Transactions on Reliability, 61*(1), 101–112.
Lust, T., Roux, O., & Riane, F. (2009). Exact and heuristic methods for the selective maintenance problem. *European Journal of Operational Research, 197*(3), 1166–1177.
Maaroufi, G., Chelbi, A., & Rezg, N. (2013). Optimal selective renewal policy for systems subject to propagated failures with global effect and failure isolation phenomena. *Reliability Engineering & System Safety, 114*, 61–70.
Maillart, L. M., Cassady, C. R., Rainwater, C., & Schneider, K. (2009). Selective maintenance decision-making over extended planning horizons. *IEEE Transactions on Reliability, 58*(3), 462–469.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.
Mo, Y., Xing, L., Zhong, F., & Zhang, Z. (2016). Reliability evaluation of network systems with dependent propagated failures using decision diagrams. *IEEE Transactions on Dependable and Secure Computing, 13*(6), 672–683.
Nourelfath, M., Fitouhi, M. C., & Machani, M. (2010). An integrated model for production and preventive maintenance planning in multi-state systems. *IEEE Transactions on Reliability, 59*(3), 496–506.
Pandey, M., Zuo, M. J., & Moghaddass, R. (2013). Selective maintenance modeling for a multistate system with multistate components under imperfect maintenance. *IIE Transactions, 45*(11), 1221–1234.
Pandey, M., Zuo, M. J., & Moghaddass, R. (2016). Selective maintenance scheduling over a finite planning horizon. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 230*(2), 162–177.
Pandey, M., Zuo, M. J., Moghaddass, R., & Tiwari, M. K. (2013). Selective maintenance for binary systems under imperfect repair. *Reliability Engineering & System Safety, 113*, 42–51.
Pham, H., & Wang, H. (1996). Imperfect maintenance. *European Journal of Operational Research, 94*(3), 425–438.
Ramirez-Marquez, J. E., & Coit, D. W. (2005). A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability. *Reliability Engineering & System Safety, 87*(2), 253–264.
Rice, W. F., Cassady, C. R., & Nachlas, J. A. (1998). Optimal maintenance plans under limited maintenance time. In Proceedings *of* the industrial engineering research conference.
Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85–117.
Schneider, K., & Cassady, C. R. (2015). Evaluation and comparison of alternative fleet-level selective maintenance models. *Reliability Engineering & System Safety, 134*, 178–187.
Shen, J., Cui, L., & Ma, Y. (2019). Availability and optimal maintenance policy for systems degrading in dynamic environments. *European Journal of Operational Research, 276*(1), 133–143.
Si, W., & Yang, Q. (2016). A generalized mixed effect Kijima model and application in optimal maintenance planning. *IEEE Transactions on Reliability, 65*(3), 1551–1561.
Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science (New York, N.Y.), 362*(6419), 1140–1144.
Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In Proceedings *of* the International Conference on Machine Learning.
Sloan, T. W. (2004). A periodic review production and maintenance model with random demand, deteriorating equipment, and binomial yield. *Journal of the Operational Research Society, 55*(6), 647–656.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed). Cambridge, MA: MIT Press.
Ushakov, I. (1986). A universal generating function. *Soviet Journal of Computer and Systems Sciences, 24*(5), 118–129.
Wang, C., Xing, L., Amari, S. V., & Tang, B. (2020). Efficient reliability analysis of dynamic *k*-out-of-*n* heterogeneous phased-mission systems. *Reliability Engineering & System Safety, 193*, 106586.
Wang, G., Peng, R., & Xing, L. (2018). Reliability evaluation of unrepairable *k*-out-of-*n*: *G* systems with phased-mission requirements based on record values. *Reliability Engineering & System Safety, 178*, 191–197.
Wu, S., & Zuo, M. J. (2010). Linear and nonlinear preventive maintenance models. *IEEE Transactions on Reliability, 59*(1), 242–249.
Yang, L., Ye, Z. S., Lee, C. G., Yang, S. F., & Peng, R. (2019). A two-phase preventive maintenance policy considering imperfect repair and postponed replacement. *European Journal of Operational Research, 274*(3), 966–977.
Zhu, H., Liu, F., Shao, X., Liu, Q., & Deng, Y. (2011). A cost-based selective maintenance decision-making method for machining line. *Quality and Reliability Engineering International, 27*(2), 191–201.