J Intell Manuf (2019) 30:147–161
https://doi.org/10.1007/s10845-016-1237-7

Optimal preventive maintenance policy based on reinforcement learning of a fleet of military trucks

Stephane R. A. Barde[1] · **Soumaya Yacout**[2] · **Hayong Shin**[1]

Received: 14 November 2015 / Accepted: 10 June 2016 / Published online: 18 June 2016

Abstract In this paper, we model preventive maintenance strategies for equipment composed of multi-non-identical components which have different time-to-failure probability distribution, by using a Markov decision process (MDP). The originality of this paper resides in the fact that a Monte Carlo reinforcement learning (MCRL) approach is used to find the optimal policy for each different strategy. The approach is applied to an already existing published application which deals with a fleet of military trucks. The fleet consists of a group of similar trucks that are composed of non-identical components. The problem is formulated as a MDP and solved by a MCRL technique. The advantage of this modeling technique when compared to the published one is that there is no need to estimate the main parameters of the model, for example the estimation of the transition probabilities. These parameters are treated as variables and they are found by the modeling technique, while searching for the optimal solution. Moreover, the technique is not bounded by any explicit mathematical formula, and it converges to the optimal solution whereas the previous model optimizes the replacement policy of each component separately, which leads to a local optimization. The results show that by using the reinforcement learning approach, we are able of getting a 36.44 % better solution that is less downtime.

✉ Soumaya Yacout
soumaya.yacout@polymtl.ca

Stephane R. A. Barde
sbace@kaist.ac.kr

Hayong Shin
hyshin@kaist.ac.kr

[1] Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea

[2] Ecole Polytechnique de Montreal, Montreal, QC, Canada

Keywords Preventive maintenance · Opportunistic maintenance · Markov decision process · Reinforcement learning

Introduction

Reinforcement learning is a computational approach where a learning agent (a virtual decision-maker) interacts with an uncertain environment in order to achieve a specific goal. The virtual agent learns what action to take for each situation it encounters in order to maximize a cumulative reward, which represents the agent objective. The objective of reinforcement learning is to find an optimal strategy defined as a mapping from situation (state) to actions for control problems. This is the focus of this paper. To achieve this objective, the agent discovers which actions yield the most reward, by trying all the possible actions without knowing in advance what actions should be taken. Another important aspect of how the agent chooses the actions is that it takes into consideration the consequences of the actions on the reward, and it extends them over the time periods. In some periods it will choose to sacrifice some present reward in order to gain more afterwards. This leads to an optimization over a finite or infinite horizon as opposed to optimization over each period of time. To solve the optimization problems, the concept of exploitation and exploration has been addressed intensively in the literature (Wang et al. 2013). The reinforcement learning crucial element is the trade-off between exploitation and exploration. Exploration consists of the agent trying all the possible actions at least once in order to make better actions' selection in the future, whereas exploitation consists of the agent using its current knowledge to obtain a reward. Finally, the reinforcement problem is highly related to the formalism of Markov decision processes (MDPs) and dynamic pro-

gramming. This machine learning approach adds to MDPs a focus on approximation and incomplete information (Sutton and Andrew 1998). Incomplete information in this context means that we do not know the one-step transition matrix of the MDP. Hence, an optimal policy cannot be finding by using dynamic programming due to the incomplete information, which is the transition matrix of the problem. The suggested model-free reinforcement learning algorithms overcome this issue through the exploration, that is the interaction between the agent and the simulation of the environment. For this reason, the preventive maintenance problem is formulated as an MDP, and it is solved by using reinforcement learning technique instead of using the dynamic programming concepts. This later suffers from the 'curse of dimensionality' and the 'curse of modeling'. The curse of dimensionality comes from much longer computational time and much larger memory space needed, as the state space of the problem increases. The curse of modeling comes from the need to estimate the transition probabilities which is often difficult to estimate, especially when the state space is large (Powell 2007). Reinforcement learning algorithms are capable to overcome these two curses of dynamic programming. They have the capability to solve very large MDPs without the knowledge of the transition probabilities. They have been successfully applied in some domains such as the board games (for example Othello chess and game of Go) and in robotics control (Sutton and Andrew 1998). For example, Monte Carlo Tree Search based reinforcement learning algorithm was successfully applied in the game of Go. It is a very challenging problem since a 9×9 Go has $3^{9\times 9} \approx 10^{38}$ of distinct board positions. Yet, it was successfully learned and was able to defeat a professional level human player in 2009 who won at least one major tournament before (Gelly et al. 2012).

Despite the strength and advantages of the reinforcement learning approach, it has not been applied intensively in solving industrial problems. It was used successfully in scheduling problems in an uncertain environment. Tuncel et al. (2014) successfully solved the disassembly line balancing problem by applying Monte Carlo reinforcement learning. Their goal consisted of assigning disassembly operations to workstations to build a feasible sequence of disassembly tasks such that the minimum number of workstations is achieved as well as the variation of idle time among workstation is minimized (Tuncel et al. 2014). Das and Sudeep (1999) applied it for finding the optimal preventive maintenance policy in a production inventory system which produces a single product type to satisfy an external demand and the inventory is maintained following an (S,s) policy. Similarly, Gosavi (2004) developed a new reinforcement learning algorithm for solving Semi-Markov Decision Processes (SMDPs) in the context of long-run average cost and apply it to the same problem described above. Finally, Wang et al. (2014) applied multi-agent reinforcement learning in order to find the optimal policy for a flow line system consisting of two series machines with an intermediate finite buffer between them. To our knowledge, there is no paper that tries to find the optimal replacement policy that minimize the total downtime for an equipment composed of multi-non-identical components by applying reinforcement learning, which is the application problem presented in this paper.

The objective of this paper is to obtain the optimal replacement policy that minimize the total downtime for an equipment composed of multi-non-identical components which are neither in series nor in parallel, and which have all different random time-to-failure (Steven 2001; Jardine and Tsang 2013). In this paper, three different preventive maintenance strategies are formulated by using MDP. The optimal policies are found by using the on-policy first-visit Monte Carlo control algorithm for ε-soft policy (Sutton and Andrew 1998). The three strategies are: a classical maintenance, where each component is replaced at a specified constant time, the second one differs from the previous one by adding a scheduled overhaul where all the components are replaced, and the third one differs from the first one by adding the notion of neighborhood where the components are organized into groups and if a component fails or is replaced preventively, then all other components belonging to the same group are replaced. This is an opportunistic strategy which is based on the assumption that it is sometimes better to replace a component that didn't fail but its neighbor has failed, than to wait for each component in this neighborhood to fail. This situation usually happens when the time of replacing separately each component in the neighbor is much higher than the time to replace a group of neighboring component, and the cost of a component is usually less than the cost of stopping the equipment. The modeling approach that is presented in this paper is evaluated by comparing its results to those obtained when using traditional simulation technique used by Abdel Haleem and Yacout (1998), this model is called the reference. For the purpose of comparison, the traditional Monte Carlo simulation approach is first used, and the parameters of the time-to-failure probability distribution of each component, which follow different Weibull probability distributions are estimated based on the real data of times to failure. A Monte Carlo reinforcement learning (MCRL) approach is then presented to model the same problem, and the results are compared. In the next sections, the formulation of the different preventive maintenance strategies by using the MDP and the traditional Monte Carlo simulation model is presented. The Monte Carlo reinforcement learning (MCRL) algorithm applied to the same problem is then introduced. In "Model description" section, we present the evaluation method, which consists of a simulation of a fleet of military Truck. Finally, we show the numerical results comparing our algorithm's performance to the referred one.

Model description

Problem description

The problem can be described as equipment that has multiple non-identical components in a general structure. These components are replaced several times during the life time of the equipment. The downtime is defined as the non-productive time, which is the time that the system is not operational due to failure or preventive action. The cost is not relevant to make the replacement decisions, but the downtime is to be minimized, in other words the optimal replacement decisions are made by minimizing the non-operational time of the system. The replacement strategies of interest have the following assumptions:

1. The equipment is composed of eight statistically independent components.
2. The time to replace a failed component is longer than the time of replacing it preventively. In addition, the time to replace the whole system or a group of neighbor components is less than the sum of times to replace each component separately. This assumption is relevant to the third and fourth strategies that are presented in the next section.
3. There are replacement opportunities at fixed intervals, for example when a planned overhaul takes place. This is relevant to the fourth strategy.

The paper presents four different strategies where the first one is a corrective maintenance, whereas the other three are preventive. The four strategies are described as follows:

Strategy I: It is based on corrective maintenance where every component is replaced at failure.
Strategy II: It is based on preventive maintenance where every component is replaced at failure and at replacement intervals T_i for each component i. In other words, every component is replaced at failure if it occurs before T_i; otherwise, every component is replaced at T_i. In the published paper (our reference), these replacement intervals are obtained by minimizing D_i, for each i separately. D_i is the downtime per unit time for each component i. T_i is obtained by solving the following optimization problem:

$$\text{argmin}_{T_i} D_i = \frac{tp_i \cdot (1 - F(T_i) + tf_i \cdot F(T_i)}{(T_i + tp_i) \cdot (1 - F(T_i) + \left[(tf_i + E[t|t \leq T_i])\right] \cdot F(T_i)} \quad for\ every\ i \qquad (1)$$

where tp_i is the time to replace preventively the component i, tf_i is the time to replace the component i at failure, $F(T_i)$ is the probability of failure of component i at time T_i and $E[t|t \leq T_i]$ is the expected time to failure given that it occurs before T_i (Abdel Haleem and Yacout 1998). It is to be noted that the solution of the optimization problem that is given by Eq. (1) leads to a local optimal times to failures of the system, since the optimal time to replace each component is obtained separately and without taking into consideration a system optimization approach. Equation (1) is used in the reference model only. In our proposed model, the optimal replacement times T_i are obtained through the MCRL technique.
Strategy III: It is based on strategy II, to which it is added a scheduled overhaul. In other words, as strategy II, every component is replaced at failure and at replacement intervals T_i for each component i but also, the whole system is replaced at a known fixed time. The time to overhaul the whole system is ts $< \sum tp_i$. The similarity between these two strategies permits the analysis of the results of the MCRL model, as is shown in the next sections.
Strategy IV: This is a group based strategy carrying a preventive replacement opportunities for other components. When a component fails, components that are in the neighbourhood, are also replaced together, or when a component has reached or passed an age control limit.

Reinforcement learning model

Markov decision process

Markov decision process is a formalism to describe a stochastic dynamic system, which has five main components:

1. The discrete state space of the system denoted as $\mathcal{S}$. We denote the state at time t as $s_t \in \mathcal{S}$.
2. A set of actions that depends on a given state denoted as$\mathcal{A}(s)$. We denote the action at time t as $a_t \in \mathcal{A}(s_t)$
3. The transition probabilities of the system denoted as $p(s_{t+1}|s_t, a_t)$, which gives the probability of being in s_{t+1} given that the system was in s_t and an agent performs an action a_t.
4. The reward function (for maximization problem) denoted as $R(s_t, a_t)$, which is the reward of performing action a_t at state s_t

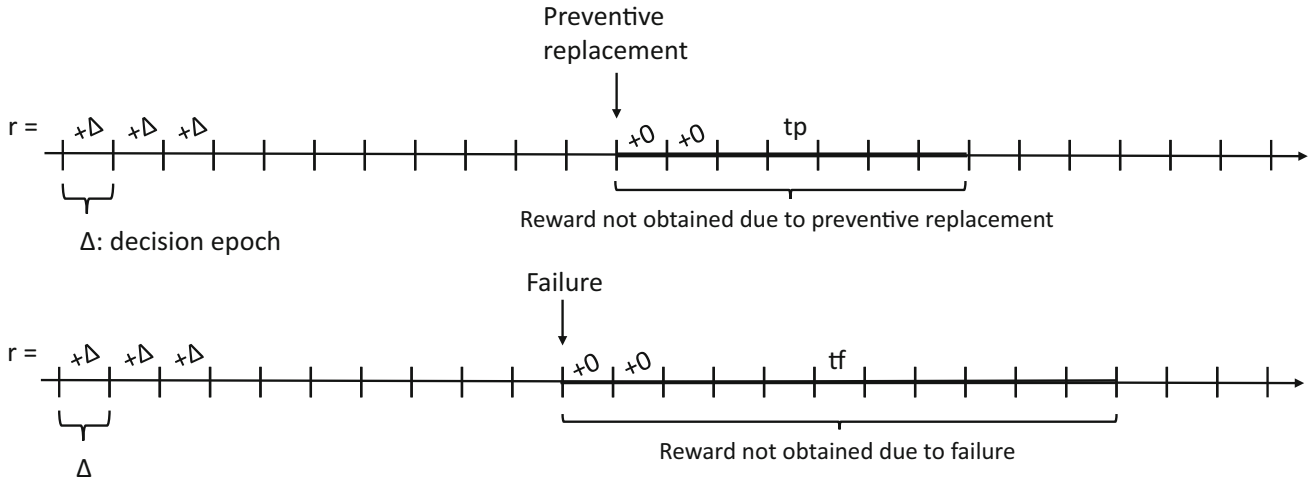


Fig. 1 Intermediate reward formulation by using the uptime and downtime trade-off

5. A discount factor $\gamma \in (0,1)$. If the problem is defined in a finite horizon, then the discount factor can be equal to one.

A deterministic policy function π, which defines the behavior of an agent is defined as $a_t = \pi(s_t)$. An objective function is defined as $F_\pi = \mathrm{E}_\pi\left[\sum_{t=0}^{T} \gamma^t \cdot R(s_t, \pi(a_t))\right]$ when it follows a policy π. The objective of this modeling method is to find the optimal policy π, that maximizes F_π. In other words, we search for $F^* = \max_\pi F_\pi$. The action-value function or Q-function is defined as the value of taking action a_t in state s_t under a policy π denoted as $\mathcal{Q}^\pi(s_t, a_t) = \mathrm{E}_\pi\left[\sum_{t=0}^{T} \gamma^t \cdot R(s_t, \pi(a_t))|s_t, a_t\right]$. The optimal action-value function $\mathcal{Q}^*$ is expressed as $\mathcal{Q}^*(s_t, a_t) = \max_\pi \mathcal{Q}^\pi(s_t, a_t), \forall s_t \in \mathcal{S} \,\&\, \forall a_t \in \mathcal{A}$ (Sutton and Andrew 1998). Hence, the optimal policy can be derived directly from $\mathcal{Q}^*$. In other words, $a_t^* = \pi^*(s_t) = \operatorname{argmax}_a Q^*(s_t, a)$ (Sutton and Andrew 1998; Powell 2007).

Markov decision process formulation

We give the MDP formulation for each strategy. Since the reinforcement problem is model-free, which means that there is no knowledge about the state-transition probabilities, we formulate only the state, action and reward function component.

MDP formulation of strategy II Let G_j denote the age of component j. Let w_j be the status of a component j. Let $w_j = 1$ denotes a failure, whereas $w_j = 0$ denotes a normal status. Let N be the number of components in the system. Then, the state of the system at time t is the vector defined as follow:

$$s_t = (G_1, \ldots, G_N, w_1, \ldots, w_N). \tag{2}$$

Let a_j denotes if there is a PM action or not on a component j. let T_j be a threshold on the age of a component j that permits the determination of action a_j. We assume that $a_j = 1$ corresponds to PM action, whereas $a_j = 0$ corresponds to "do-nothing" action. Then, a_j is defined by the following rule:

$$a_j = \begin{cases} 1, & if\, G_j \geq T_j \\ 0, & if\, G_j < T_j \end{cases}. \tag{3}$$

Thus, the action on the system at time t is the vector

$$a_t = (a_1, \ldots, a_N). \tag{4}$$

For the choice of the reward function, it is formulated by using a trade-off between uptime (productive time of a physical asset) and the downtime (non-productive time of a physical asset). More precisely, when there is a preventive replacement at time t for component j, the system goes down for tp_j time, which is equivalent to a loss equal to the time to replace preventively. This downtime is applied as a penalty for the agent when choosing the preventive action. Similarly, when there is a failure at time t for component j, the system goes down for tf_j time, which is equivalent to a loss equal to the time to replace due to failure. This downtime is applied as a penalty for the agent when choosing the "do-nothing" action but failure occurs at that interval (See Fig. 1).

However, there is a scaling problem as tp_j and tf_j are much smaller than Δ, that is the time interval between two decisions epoch which is chosen such that there aren't two different components that fail during that time interval. There are two ways to choose Δ: we can either diminish the time

interval between two epochs Δ, or we can scale tp_j such that it has at least the same time period than Δ. If we choose on the first option, the state space becomes larger which increases the computational time of the MCRL algorithm. For that reason, the second option which is the scaling method is chosen. Hence, the downtime are scaled by maintaining the ratio $\frac{tp_j}{tf_j}$, which represents the trade-off between two actions, the preventive replacement and the "do-nothing" until failure. The scale factor $\alpha_j = \frac{\Delta}{tp_j}$ is chosen, and tp_j is scaled such that it is equal to Δ. We denote the ceiling function as $\lceil x \rceil = \min\{n \in \mathrm{N} | n \geq x\}$. Hence, we have the following reward function

$$R(S_t, a_t) = \begin{cases} -\alpha_j \cdot tp_j, & if\ a_j = 1 \quad (5) \\ -\alpha_j \cdot \Delta \cdot \left\lceil \frac{tf_j}{\Delta} \right\rceil, & if\ w_j = 1\ \&\ a_j = 0 \quad (6). \\ \Delta, & otherwise \quad (7) \end{cases}$$

The condition in Eq. (5) represents the preventive replacement action of component j which is taken before a failure occurs. The condition in Eq. (6) means that the failure of component j occurs before the preventive replacement. Equation (7) represents the uptime as a reward since no event caused the downtime to occur.

MDP formulation of strategy III Let G_j denote the age of component j. Let w_j be the status of a component j. Let $w_j = 1$ denotes the failure, whereas $w_j = 0$ denotes the normal status. Let $\mathrm{O} = 1$ denotes if there is a scheduled overhaul, whereas $\mathrm{O} = 0$ denotes no scheduled overhaul. Let N be the number of components in the system. Then, the state of the system at time t is the vector defined as follow:

$$s_t = (G_1, \ldots, G_N, O, w_1, \ldots, w_N). \quad (8)$$

The action on the system are defined as

$$a_t = (a_1, \ldots, a_N). \quad (9)$$

The same idea as in strategy II is applied to formulate the reward function, which is

$$R(S_t, a_t) = \begin{cases} -\alpha_j \cdot tp_j, & if\ a_j = 1\ \&\ O = 0 \quad (10) \\ -\alpha_j \cdot \Delta \cdot \left\lceil \frac{tf_j}{\Delta} \right\rceil, & if\ w_j = 1\ \&\ a_j = 0\ \&\ O = 0 \quad (11). \\ -\alpha_j \cdot \Delta \cdot \left\lceil \frac{\beta \cdot \sum_{i=1}^{N} tp_i}{\Delta} \right\rceil, & if\ O = 1 \quad (12) \\ \Delta & otherwise \quad (13) \end{cases}$$

The condition in Eq. (10) represents that the preventive replacement action of component j before failure and without scheduled overhaul. The condition in Eq. (11) means that the failure of component j occurs before the preventive replacement time and the scheduled overhaul. The condition in Eq. (12) represents the scheduled overhaul. The given parameter $\beta \epsilon (0, 1)$ comes from the second assumption in the "Problem Description" section, which states that the time to replace the whole system is less than the sum of times to replace each component separately. The α_j parameter is defined in the same manner as in strategy II. Finally, Eq. (13) represents the uptime as a reward since no event that caused downtime occurred.

MDP formulation of strategy IV The notation to model the group based strategy in this section is inspired from (Jia 2010). Let G_j denote the age of component j. Let w_j be the status of a component j. Let $w_j = 1$ denotes the failure, whereas $w_j = 0$ denotes the normal status. Let N be the number of components in the system. Let K be the number of groups in the system. Let N_i be the number of components in group i. Let ϕ_i be the set of components' indexes in group i. From the published paper (Abdel Haleem and Yacout 1998), the following groups are formed: $(\phi_1, \phi_2, \phi_3, \phi_4, \phi_5) = (\{1, 3\}, \{3, 8\}, \{3, 5\}, \{7, 6\}, \{4, 2\})$.

Then, the state of the system at time t is the vector defined as follow:

$$s_t = (G_1, \ldots, G_N, w_1, \ldots, w_N). \quad (14)$$

The action on the system are defined, identically to strategy II, as

$$a_t = (a_1, \ldots, a_N). \quad (15)$$

The formulation of the reward function is defined as follow

$$R(S_t, a_t) = \begin{cases} -\alpha_j \cdot \beta \cdot \sum_{l \in \phi_j} tp_l, & if\ a_k = 1\ \&\ k \in \phi_j \quad (16) \\ -\alpha_j \cdot \Delta \cdot \left\lceil \frac{\beta \cdot \sum_{l \in \phi_j} tf_l}{\Delta}, \right\rceil & if\ w_k = 1\ \&\ a_k = 0\ \&\ k \in \phi_j \quad (17), \\ \Delta, & otherwise \quad (18) \end{cases}$$

The condition in Eq. (16) represents the preventive replacement action taken for component k, which belongs to the group j, before a failure occurs. The condition in Eq. (17) means that a failure occurred to component k, which belongs to the group j, before the preventive replacement time. The given parameter $\beta \epsilon (0, 1)$ is defined in the same manner as before. The α_j parameter is defined as $\alpha_j = \frac{\Delta}{\beta \Delta \sum_{l \in \phi_j} tp_l}$. Finally, Eq. (18) represents the uptime as a reward, since no event that caused downtime occurred.

Monte Carlo reinforcement learning In the model-free problem setting, Monte Carlo method is based on averaging sample returns from a simulated experience. It directly learns

from episodes generated by a policy derived from an estimated Q-function. It can be only applied to episodic MDPs, which means that all simulated episodes must terminate. Since the objective is to find the optimal deterministic policy for the preventive maintenance strategies without a complete knowledge of the environment, a model-free control algorithm is applied. This algorithm optimizes the action-value function of an unknown MDP. The parameters of the MDP are its states, its actions, the time-to-failure found from actual data, or modeled by any distribution function, for example the Weibull probability distribution of each component, and finally the reward function, which depend on the action chosen by the agent given the current state. The on-policy first-visit Monte Carlo control algorithm for ε-soft policy is applied. This is a well-known algorithm in the model-free control problem in the Reinforcement Learning field. The algorithm starts with a stochastic policy and converges to a deterministic one as the algorithm generates new episodes from the current policy. A stochastic policy π is defined as a mapping from each state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ to the probability $\pi(a_t|s_t)$ of taking action a_t in state s_t. (Sutton and Andrew 1998).

The algorithm's main process has two steps that are related to each other. On one hand, it evaluates a generated policy, and on the other hand, it improves the current policy by taking greedy action with an exploration measure, which consists of taking another action, different than the greedy one, with a low probability. The probability of choosing another action decreases hyperbolically. This process is described as follow: First, from the estimated action-value function Q, we derive the corresponding policy, which permits to generate a new episode. During the episode generation, which is different for strategy II, III and IV, an immediate reward r is chosen form the reward function that we defined in the MDP formulation section. Then, the generated policy is evaluated. At the first time-step t where the state s_t and action a_t are visited, a counter is incremented such that $N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$, total return $S(s_t, a_t) \leftarrow S(s_t, a_t) + G_t$, and the state-action value function is evaluated by $Q(s_t, a_t) \leftarrow S(s_t, a_t)/N(s_t, a_t)$.

The policy is then improved by taking ε-greedy action such that

$$\pi(a_t|s_t) = \begin{Bmatrix} 1 - \varepsilon_t + \frac{\varepsilon_t}{|\mathcal{A}(s)|} & if a = a^* \\ \frac{\varepsilon_t}{|\mathcal{A}(s_t)|} & if a \neq a^* \end{Bmatrix}, \tag{19}$$

where $a^* = \text{argmax}_{a \epsilon \mathcal{A}(s_t)} Q(s_t, a)$ (Sutton and Andrew 1998).

Our decision making problem involves a fundamental tradeoff: exploitation, which consists of making the best decision given current information, whereas exploration, which consists of gathering more information about the system under study. Hence, we have to gather enough information about the system under study in order to make the best decisions. A greedy algorithm chooses action with the highest value of the Q-function given in the current state: $a_t = \underset{a}{\text{argmax}}\, Q^*(s_t, a)$. In this context, the algorithm does only the exploitation part. However, by adding the notion of stochastic policy, which implies that instead of only staying with the best action forever, we can explore—gather more information—and change the action as we learn more about the system. This is especially essential at the beginning of the estimation of the Q-function. This is what the ε-greedy algorithm does. The algorithm is greedy at the limit with infinite exploration (GLIE), which is proven to converge to the optimal policy if the Robbins–Monro conditions (Szepesvari 2010) for the exploration is met. We choose $\varepsilon_t = \frac{1}{t}$, which guarantees optimal convergence as $\sum_{t=0}^{\infty} \varepsilon_t = \infty$ and $\sum_{t=0}^{\infty} \varepsilon_t^2 < \infty$ are satisfied (Tsitsiklis 2003).

Algorithm 1 shows the complete pseudo code of the model-free control algorithm. It is adapted and modified from the pseudocode in (Sutton and Andrew 1998). The algorithm is implemented by using MATLAB.

It is assumed that each component failure probability is independent, the algorithm is used to learn and find the optimal replacement time T_i^{RL} for each component seperately. In order words, the algorithm searches for the optimal Q-function for each component, and T_i^{RL} that corresponds to the age where the value of the action 'replace preventively' is higher than the value of the action 'do-nothing'.

Algorithm 1. Pseudo-code of on-policy first-visit Monte Carlo control algorithm

Input: $\mathcal{S}$, $\mathcal{A}$, time-to-failure Weibull probability distribution of each component and an intermediate reward r

Repeat for all the 8 components

Repeat for all s $\in \boldsymbol{\mathcal{S}_i}$ **and a** $\in \boldsymbol{\mathcal{A}}$ **(initialization)**

$Q_i(s,a) \leftarrow arbitrary\ matrix\ of\ dimension\ \mathcal{S} \times \mathcal{A}$

$Returns(s,a) \leftarrow empty\ matrix\ of\ dimension\ \mathcal{S} \times \mathcal{A}$

$N(s,a) \leftarrow empty\ matrix\ of\ dimension\ \mathcal{S} \times \mathcal{A}$

$\varepsilon \leftarrow 1\ (GLIE)$

$\pi(a|s) \leftarrow arbitrary\ matrix\ of\ dimension\ \mathcal{S} \times \mathcal{A}$ with $\sum_{a \in \mathcal{A}} \pi(a|s) = 1$

Repeat Forever

Generate an episode using π

For each pair (s,a) appearing in the episode

G ← Return following the first occurence of (s, a)

$Returns(s,a) \leftarrow Returns(s,a) + G$

$N(s,a) \leftarrow N(s,a) + 1$

$Q_i(s,a) \leftarrow Returns(s,a)/N(s,a)$

End for

For each s in the episode

For all a $\in \boldsymbol{\mathcal{A}}$

$$\pi(a|s) = \begin{Bmatrix} \frac{\varepsilon}{m} + 1 - \varepsilon & if\ a^* = \underset{a \in \mathcal{A}}{argmax}\ Q(s,a) \\ \frac{\varepsilon}{m} & otherwise \end{Bmatrix}$$

End for

End for

$\varepsilon = \frac{1}{k}$, where k is the k^{th} generated episode

End repeat

End repeat

End repeat

Output: Optimal policy π^* and Optimal state-action function Q_i^*

Model evaluation description

Based on the gathered times to failure of the different components, the time-to-failure probability distribution for each component is modeled by a Weibull cumulative density function.

$P\ (t; \lambda_i, k_i)$ is the Weibull probability density function of component i, with scale and shape parameters λ_i and k_i, respectively. Table 1 shows the mean time to failure and the Weibull's distribution's parameters for each component. Table 2 shows the values of the replacement times at failure, tf_i, the values of the planned replacement time tp_i, and the reference replacement times T_i that are obtained from Eq. 1, and are used in the reference model.

To evaluate the performance of the reference model and the MCRL model, we apply a discrete event simulation where the time interval between every two decision epochs is 5 h.

Table 1 The parameters of Weibull density function for each component

	Component							
	Tire	Transmission	Wheel	Coupling	Motor	Brake	Steering wheel	Shifting gears
Mean	2361.80	991.70	708.50	1399.90	342.60	3917.50	817.70	2037.00
Lambda (scale)	2365.08	996.88	713.55	1406.84	343.76	3933.12	828.19	2040.95
K (shape)	414.16	109.25	79.81	115.21	169.81	143.60	43.83	296.48

Table 2 Data for simulation

	Component							
	Tire	Transmission	Wheel	Coupling	Motor	Brake	Steering wheel	Shifting gears
tf_i	2	6.5	2.5	6	5	3.5	3	3.5
tp_i	0.4	5.42	0.625	0.857	1.25	0.7	0.429	0.875
T_i	1440	1830	At failure	2160	248	2250	306	1400

This value is chosen because the probability that two components will fail during this time interval is approximately zero. The simulation for all the strategies have three counters, where the downtime counter keeps track of the downtime of the system, the failed component counter keeps track of the number of failed components, and the preventive counter that keeps track of the number of preventive replacement for each component. The simulation is run for 100,000 h.

Simulation of strategy I The input parameters are tf_i as in Table 2, the time interval between two inspections (5 h), the Weibull's distribution parameters as in Table 1, or the actual data of the times to failures, and the simulation running time (100,000 h).

At each time period, a random time-to-failure from a Weibull probability density function (λ_i, k_i) for each component i, is generated. For each component, if the generated time-to-failure is larger than the current component's age, then there is no failure and we continue to the next time period, whereas if the generated time-to-failure is smaller than the current component's age, then there is a failure, the simulation reinitializes to 0 the age of the component that failed, tf_i is added in the downtime counter, and 1 is added to the component counter of the failed component.

The model's outputs are the total downtime in 100,000 h, the number of failures of each component, and the number of preventive replacements.

Simulation of strategy II The inputs are the same as strategy I. At each decision epoch, a random time-to-failure is generated from a Weibull probability density function (λ_i, k_i) for each component. For each component, the following condition statement is used: the generated time-to-failure is larger than the current component's age (True), and the current time is smaller than the optimal replacement time (True). By evaluating this condition statement, four cases are considered:

Case 1: when the condition statement is true and true, then the simulation continues to the next time period.
Case 2: when the condition statement is false and true, then there is a failure, and tf_i is added in the downtime counter, 1 is added to the component counter of the component that failed, and the age of the component that failed is reinitialize to 0.
Case 3: when the condition statement is true and false, then the equipment is replaced preventively, tp_i is added in the downtime counter, and the age of the component that was replaced preventively, is reinitialize to.
Case 4: when the condition statement is false and false, the generated time-to-failure and the optimal replacement time are compared: if the optimal replacement time is smaller than the generated time-to-failure, then tp_i is added in the downtime counter. The age of the component that was replaced preventively is reinitialized to 0. Otherwise, tf_i is added in the downtime counter, 1 to the component counter of the component that failed, and the age of the component that failed is reinitialize.

The outputs are the total downtime and the number of failures of each components.

Simulation of strategy III The inputs are the same as for strategy I and II. At each decision epoch, a random time-to-failure from a Weibull probability density function (λ_i, k_i) for each component i, is generated. For each component, the following condition statement is used: If the generated time-to-failure is larger than the current component's age (True), and the current time is smaller than the optimal replacement time (True), and there is not a scheduled overhaul (True). By evaluating this condition statement, we will have four cases:

Case 1: when the condition statement is true and true and true, then we continue to the next time period.
Case 2: when the condition statement is false and true and true, then there is a failure, tf_i is added to the downtime counter, 1 is added to the component counter of the component that failed, and the age of the component that failed is reinitialize to 0.
Case 3: when the condition statement is true and false and true, then tp_i is added to the downtime counter. The age of the component that was replaced preventively is reinitialized to 0.
Case 4: when the condition statement is true and true and false, then the overhaul schedule takes place all the components are replaced, and ts is added in the downtime counter. The age of all the components is reinitialized to 0.
Case 5: when the condition statement is false and false and true, a comparison between the generated time-to-failure and the optimal replacement time is executed: if the optimal replacement time is smaller than the generated time-to-failure, then tp_i is added in the downtime counter. The age of the component that was replaced preventively is reinitialized to 0. Otherwise, tf_i is added in the downtime counter, 1 is added to the component counter of the component that failed, and the age of the component that failed is reinitialized to 0.
Case 6: when the condition statement is true and false and false, a comparison between the optimal replacement time and the scheduled replacement time is executed: if the optimal replacement time is smaller than the scheduled overhaul time, tp_i is added in the downtime counter. The age of the component that was replaced preventively is reinitialized to 0. Otherwise, ts is added in the

downtime counter. The age of all the components is reinitialized to 0.

Case 7: when the condition statement is false and true and false, a comparison between the generated time-to-failure and the scheduled overhaul time is executed: if the generated time-to-failure is smaller than the scheduled overhaul time, then tf_i is added in the downtime counter, 1 is added to the component counter of the component that failed, and the age of the component that failed is reinitialized to 0. Otherwise, we consider that the scheduled overhaul time takes place, all the components are replaced, and ts is added in the downtime counter. The age of all the components is reinitialized to 0.

Case 8: when the condition statement is false and true and false, the smallest value among the generated time-to-failure, the scheduled overhaul time and the optimal replacement time are identified. The same process of the components' replacement and the update of the counters are executed as in case 7.

The outputs of the simulation are the total downtime and the number of failed components for each one.

Simulation of strategy IV The input parameters are tf_i, tp_i, T_i as in Table 2, the time interval between two decision epochs (5 h), the Weibull's distribution parameters as in Table 1, the simulation running time (100,000 h), and the formed groups which are {(1,3), (3,8), (3,5), (7,6), and (4,2)}.

The process is similar to policy II. Except that when there is failure or preventive replacement, all the components belonging to the same group of the failed or preventively replaced component, are also replaced, In the case of failure, $\alpha \cdot \sum_i tf_i, where\ 0 < \alpha < 1$ is calculated, whereas in the case of preventive replacement, $\alpha \cdot \sum_i tp_i$, where $0 < \alpha < 1$ is calculated.

The simulation outputs are the total downtime and the number of each failed component.

Analysis of the results

Comparison of the performance of each strategy is performed between the reference and the MRCL simulations. First, we run the simulations by using the T_i given in Table 2 for strategy II, III and IV. The results correspond to the reference simulation. Then, the simulation is run with the same settings, but the optimal replacement times are obtained through the MCRL.

In Fig. 2, a comparison between the results of strategy II is presented. The histogram represents the total downtime. The solid-line graph represents the number of failed components,

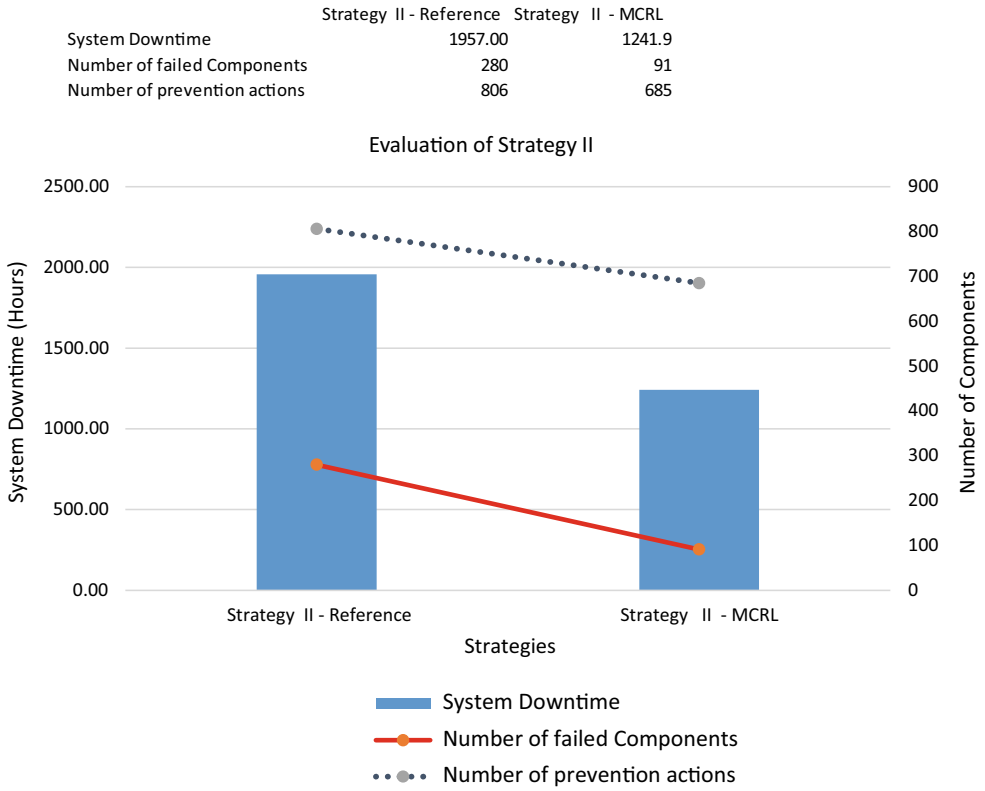


Fig. 2 Comparison between the reference and the MCRL results for strategy II

Component Name	Strategy II - Reference	Strategy II - MCRL
Tire	1440.00	2350
Transmission	1830.00	985
Wheel Rim	2000.00	690
Coupling	2160.00	1360
Motor	248.00	335
Brake	2250.00	3830
Steering Wheel	306.00	760
Shifting Gears	1400.00	1995

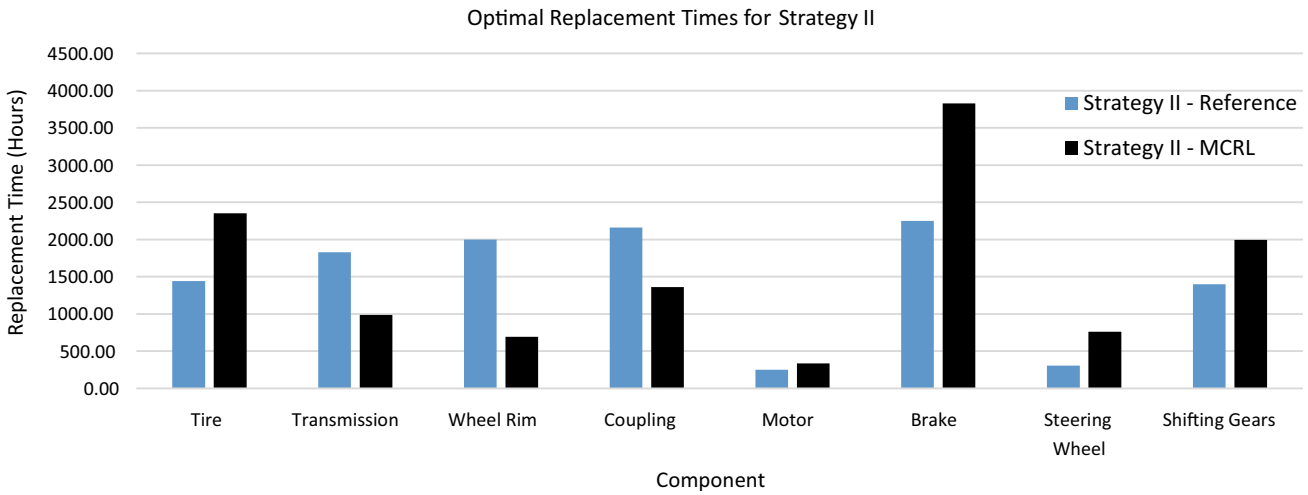


Fig. 3 Optimal replacement times for strategy II

whereas the dashed-line graph represents the number of components that was replaced preventively. It can be seen that the reference simulation has a total downtime of 1957 h with 280 failed components and 806 components replaced due to preventive action, whereas the MCRL simulation yields 1241.9 h with 91 failed components and 685 components replaced due to preventive action. The MCRL outperforms the reference simulation by 36.54 % of improvement in terms of total downtime, 189 less failures, and 121 less components replaced. Moreover, the MCRL simulation, not only provides lower total downtime, but also lower cost since both of the number of failed components and preventively replaced components are lower. In order words, the MCRL simulation has less non-productive hours, and lower replacement cost. In the MCRL simulation, although the agent makes decision in an infinite horizon, 1000 episodes were sufficient in order to reach the final results. Contrary to the reference model, the virtual agent finds the optimal policy by trial and error in the sense that it is not based on an expected value while in the Monte Carlo simulation reference model, the optimization is done locally for every component separately since the horizon is finite and based on an expected value which does not take into account the variance of the times to failure.

Figure 3 represents a histogram where the light colored bar is the reference simulation for each component, and the dark colored bar is the MCRL. First, it is observed that there is practically no difference between the optimal replacement times for the motor component. However, for the other seven components there is a clear difference in the results. For the transmission, wheel rim and coupling components, the reference replacement times are much larger than the MCRL replacement times. They exceed largely their component's corresponding mean time between failures (MTBF). Their probability of failure at that time is equal to one, which means that these three components are replaced at failure in the reference simulation, whereas in the MCRL, the failure probability at the replacement time are 0.552, 0.02 and 0.0015 for transmission, wheel rim and coupling, respectively. The failure probability at the replacement time decreases as the ratio tp/tf decreases, which means that as the preventive replacement becomes more advantageous, the agent chooses to replace preventively earlier than later. This pattern is also found for the other components too. This result is expected since the agent chooses the actions that are directly related to the trade-off between the penalty of replacing preventively and the penalty of replacing due to failure.

Secondly, the results of the strategy III are compared. The agent learns and finds the optimal way to react by generating 1000 episodes. Table 3 shows the total downtime for the reference simulation and the MCRL method for different value

Table 3 Comparison of system downtime obtained when using the reference simulation and the MCRL method for strategy III with different overhaul times

Schedule overhaul	System downtime	
	Strategy III—reference	Strategy III—MCRL
500	2002.1	1928.7
1000	1946	2309.5
1500	2167.7	1592.8
2000	2006.2	1566
2500	1870.5	1416.1
3000	2142.6	1519.3
3500	1934.7	1417.2
4000	1945.1	1426
4500	2005.4	1337.9
5000	1993.2	1482.5

Table 4 Optimal replacement times for the eight components obtained by the MC reference and the MCRL in strategy III

Component name	Strategy III—reference	Strategy III—MCRL
Tire	1440	2340
Transmission	1830	990
Wheel rim	At failure	685
Coupling	2160	1375
Motor	248	335
Brake	2250	3805
Steering wheel	306	725
Shifting gears	1400	2010

of scheduled overhaul with $\alpha = 0.9$. The reference simulation achieves the lowest downtime as 1870.5 at a scheduled overhaul of 2500 h, with 216 failed components, and 964 components replaced due to preventive action, whereas the MCRL achieves the lowest downtime of 1234.3 at a scheduled overhaul at 4500 h, with 34 failed components, and 826 components replaced due to preventive action. In this problem too, the MCRL method outperforms the reference simulation with a 34 % of improvement in terms of total downtime. Moreover it is observed that the MCRL solution produces better performance both in terms of downtime and cost. Table 4 shows the optimal replacement times for the reference simulation and the MCRL algorithm for strategy III. The solution are practically the same as strategy II because, as described earlier, strategy III is strategy II with a scheduled overhaul. This shows that the MCRL algorithm performs as expected.

Thirdly, the results in Fig. 4 belong to strategy IV. The agent learns and finds the optimal way to behave by generating 1000 episodes. The reference simulation has a total downtime of 2168.2 h, with 181 failed components, and 1532 components replaced due to preventive action, whereas the MCRL approach yields 1700.2 h of downtime, with 85 failed components, and 1430 components replaced due to preventive action. The MCRL outperforms the reference simulation by 21.58 % of improvement in terms of total downtime, with 96 less failures, and 102 less components replaced. The MCRL simulation produces better performance both in terms of downtime and cost.

Finally, the optimal replacement times of strategy IV are given in Fig. 5. The replacement times are much lower in the MCRL than in the reference simulation. This is due to the structure of the group. For example, for group 1, which is composed of tire and wheel rim, the MTBF of a tire is 2361.8 whereas the wheel rim's MTBF is 708.5. Since, the MTBF of the wheel rim is much smaller than the tire, the replacement policy depends largely on the failure or preventive action done to wheel rim. Hence, the optimal replacement time of the tire will follow the wheel rim's optimal replacement time. This logic follows for the other groups as well. Hence, the replacement strategy of the components of one group follows the component that has the lowest MTBF.

In Figs. 6 and 7, the results of different strategies with the reference and the MCRL simulations are compared, respectively in both figures. The histogram represents the total downtime for all the strategies. The solid line-graph represents the number of failed components, whereas the dashed-line graph represents the number of components that are replaced preventively. In Fig. 6, strategy II, III and IV outperform strategy I, the corrective maintenance,

	Strategy IV - Reference	Strategy IV - MCRL
System Downtime	2168.20	1700.20
Number of failed Components	181	85
Number of prevention actions	1532	1430

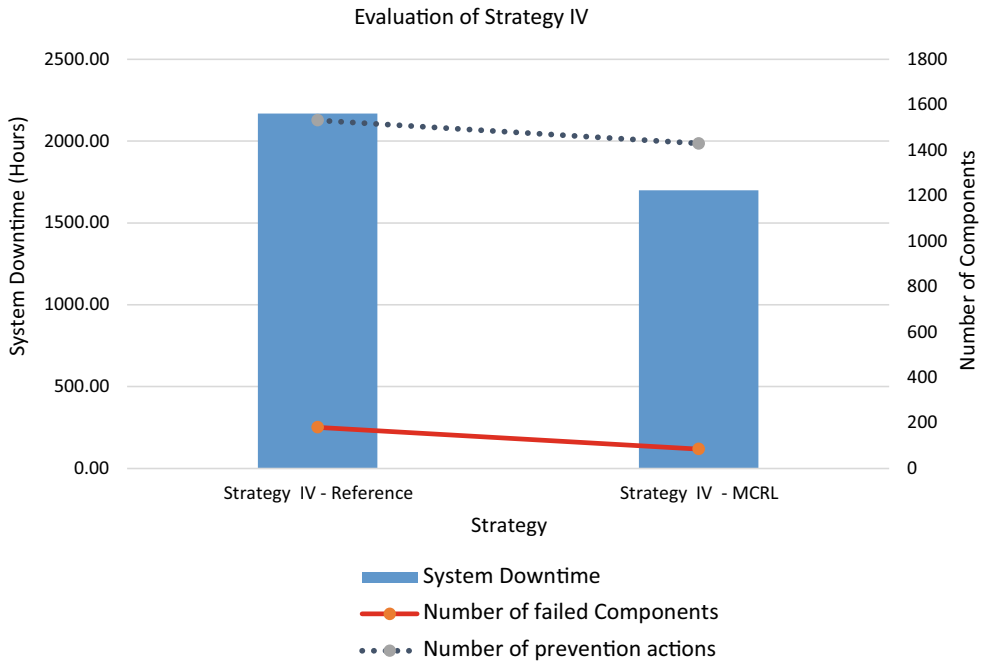


Fig. 4 Evaluation of strategy IV

Component Name	Strategy IV - Reference	Strategy IV - MCRL
Tire	1440.00	700
Transmission	1830.00	990
Wheel Rim	2000.00	315
Coupling	2160.00	985
Motor	248.00	310
Brake	2250.00	770
Steering Wheel	306.00	720
Shifting Gears	1400.00	690

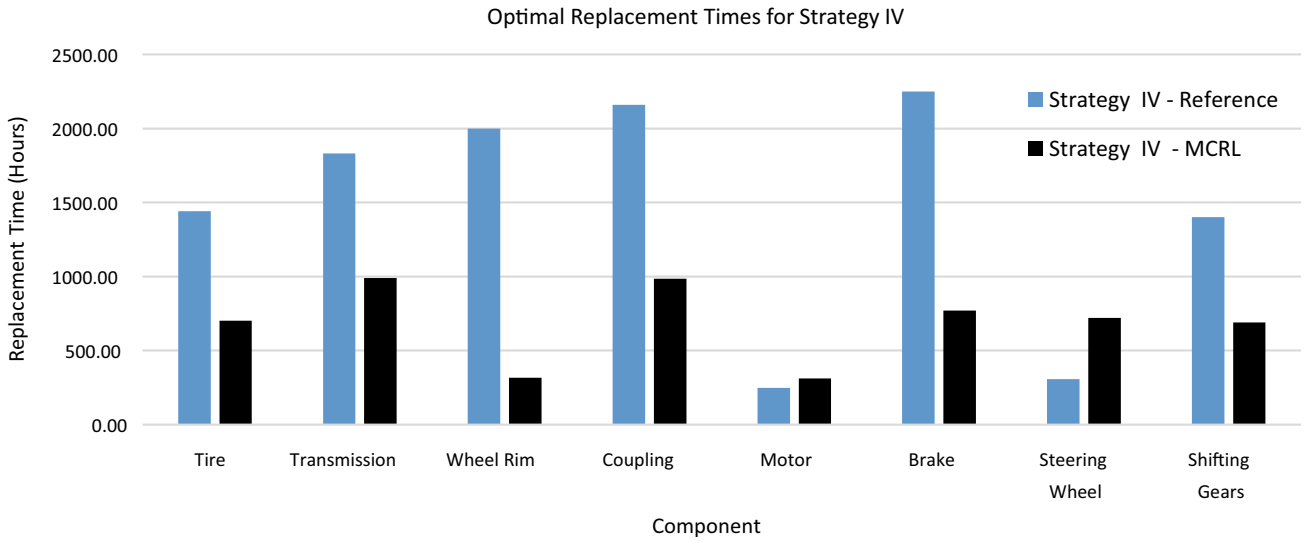
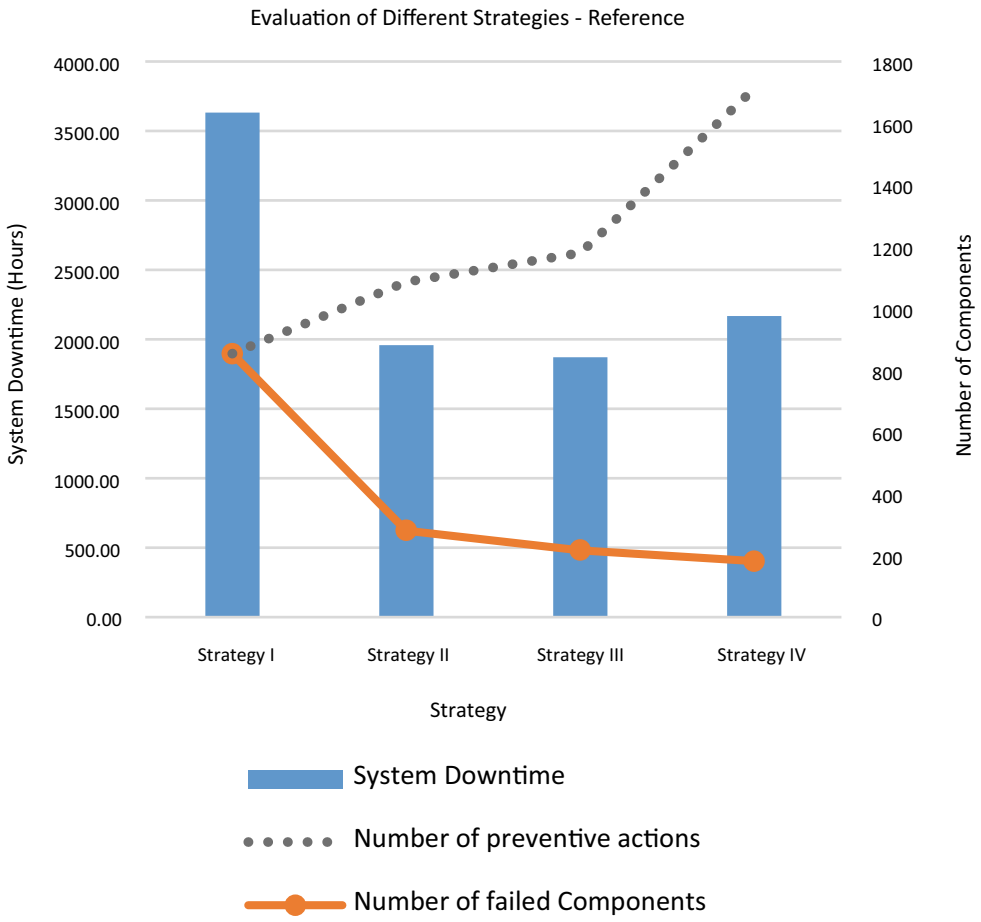


Fig. 5 Optimal replacement times for strategy IV

Fig. 6 Evaluation of different strategies—reference

Evaluation of Different Policies - Reference

	Strategy I	Strategy II	Strategy III	Strategy IV
System Downtime	3634.00	1957.00	1870.5	2168.20
Number of failed Components	853	280	216	181
Number of preventive actions	0	806	964	1532



in terms of total downtime. This shows that preventive maintenance's performance is higher than the corrective maintenance. Strategy III has the lowest total downtime among the four strategies with 1870.5 h of downtime. For the MCRL approach, strategy II has the lowest total downtime. The MCRL approach dominates in all dimensions, in downtime and in cost, since it has lower downtime, for lower number of failed components, and lower number of preventive replacements.

Conclusion

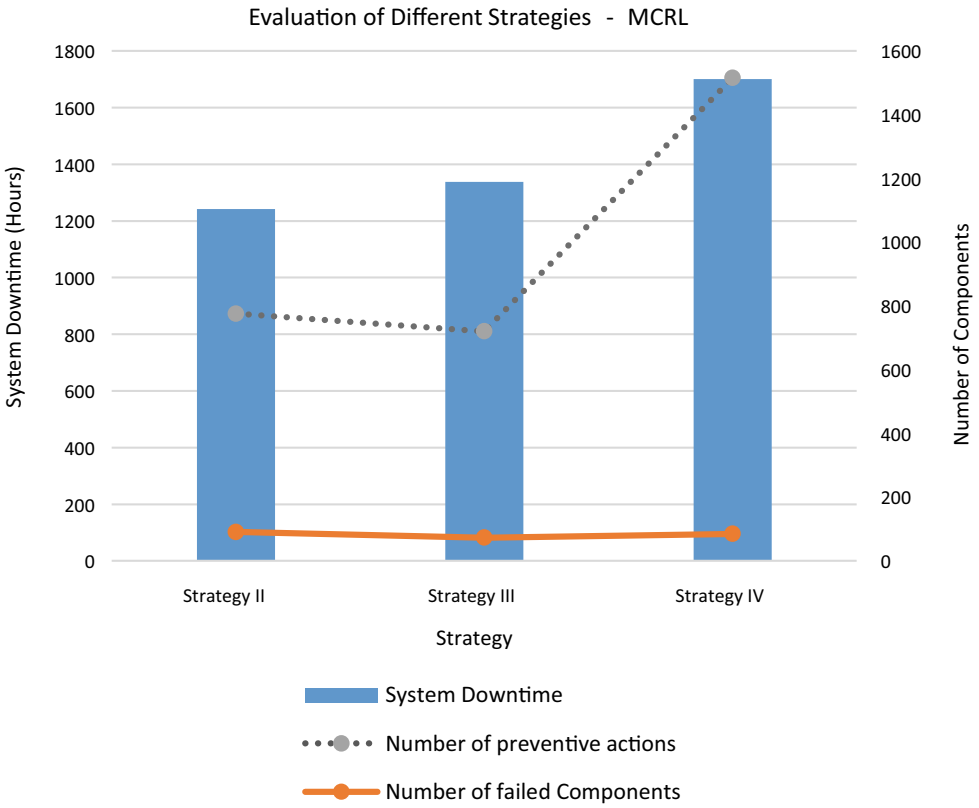
This paper presents an application of Monte Carlo reinforcement learning to find optimal replacement times for an equipment composed of multiple non identical components which have different time-to-failures. It was found that the MCRL method outperforms the reference model in terms of total downtime, and also in term of cost. The reason is that the analytical formula of the reference method has an open form; hence it is solved by using the iterative method of the traditional Monte Carlo simulation where its solution is local due to this approximation. Whereas, the MCRL model directly the problem and obtain an optimal solution for all the three strategies. Thus, the MCRL is proven to converge to the optimal solution. It was also found that, indeed, the planned maintenance method is better than the corrective one.

The advantage of the MCRL method is:

1. A mathematical formula is not needed to find the optimal replacement time for strategies II, III and IV. In other words, the traditional reference simulation solves formula (1) in order to find the replacement time for each component by minimizing the downtime only for that component, which is only locally optimal.
2. The problem is modeled by using a MDP framework, but it is solved without the knowledge of the transition probabilities and also without a reward function.

Fig. 7 Evaluation of different strategies—MCRL

Evaluation of Different Policies - MCRL			
	Strategy II	Strategy III	Strategy IV
System Downtime	1241.9	1337.9	1700.20
Number of failed Components	91	73	85
Number of preventive actions	685	647	1430



3. The problem is solved by using a MCRL algorithm, which converges to the optimal solution.

Areas of further research are: (i) to implement the reinforcement learning for real-time learning and control for this type of problem since it can be well implemented for autonomous control. (ii) Since it was assumed that the failures are independent among components, a single agent was needed. By relaxing this assumption, one can solve the current problem by applying a multi-agent reinforcement learning algorithm where failures of the components are dependent. (iii) The use of the information coming from condition monitoring to update the information needed by the agent in order to make a decision about the action to take. (iv) To apply condition based maintenance (CBM) for maintenance problem of equipment composed of multi-non-identical components. (v) Extending the current work in order to include the notion of resilience. This notion refers to the ability of the equipment to recover its functions after partial damage, thus leading to successes from failures (Zhang and Van Lutterveld 2011).

References

Abdel Haleem, B., & Yacout, S. (1998). Simulation of components replacement policies for a fleet of military trucks. *Quality Engineering*, *11*(2), 303–308.

Das, T. K., & Sarkar, S. (1999). Optimal preventive maintenance in a production inventory system. *IIE Transactions*, *31*(6), 537–551.

Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., et al. (2012). The grand challenge of computer Go: Monte Carlo tree search and extensions. *Communications of the ACM*, *55*(3), 106–113.

Gosavi, A. (2004). Reinforcement learning for long-run average cost. *European Journal of Operational Research*, *155*(3), 654–674.

Jardine, A. K., & Tsang, A. H. (2013). *Maintenance, replacement, and reliability: Theory and applications*. Boca Raton: CRC Press.

Jia, Q.-S. (2010). A structural property of optimal policies for multi-component maintenance problems. *IEEE Transactions on Automation Science and Engineering*, *7*(3), 677–680.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality* (Vol. 703). New York: Wiley.

Steven, B. (2001). J. D. Campbell, A. K. Jardine, & W. M. Dekker (Eds.), *Maintenance excellence, optimizing equipment life-cycle decisions*, pp. 43–44.

Sutton, R. S., & Andrew, G. B. (1998). *Reinforcement learning: An introduction* (Vol. 1, No. 1). Cambridge: MIT press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *4*(1), 1–103.

Tsitsiklis, J. N. (2003). On the convergence of optimistic policy iteration. *The Journal of Machine Learning Research*, *3*, 59–72.

Tuncel, E., Zeid, A., & Kamarthi, S. (2014). Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *Journal of Intelligent Manufacturing*, *25*(4), 647–659.

Wang, X., Wang, H., & Qi, C. (2014). Multi-agent reinforcement learning based maintenance policy for a resource constrained flow line system. *Journal of Intelligent Manufacturing*, *27*(2), 325–333.

Wang, J. W., Wang, H., Ip, W. H., Furuta, K., & Zhang, W. J. (2013). Predatory search strategy based on swarm intelligence for continuous optimization problems. *Mathematical Problems in Engineering*. 11 pp. doi:10.1155/2013/749256

Zhang, W. J., & Van Luttervelt, C. A. (2011). Toward a resilient manufacturing system. *CIRP Annals-Manufacturing Technology*, *60*(1), 469–472.