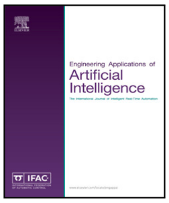
Engineering Applications of Artificial Intelligence 118 (2023) 105638

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



A stochastic dynamic programming approach for the machine replacement problem

Ali Forootani[a,*], Majid Ghaniee Zarch[b], Massimo Tipaldi[c], Raffaele Iervolino[d]

[a] *Hamilton Institute, Maynooth University, Co., Kildare W23 F2K8, Ireland*
[b] *Department of Electrical Engineering, Bu-Ali Sina University, Hamedan, Iran*
[c] *Department of Electrical and Information Engineering, Polytechnic University of Bari, Bari, 70126, Italy*
[d] *Department of Electrical Engineering and Information Technology, University of Naples, Napoli, 80125, Italy*

ARTICLE INFO

Keywords:
Machine replacement problem
Markov Decision Process
Dynamic Programming
Least-Squares Temporal Difference
Monte Carlo Simulations

ABSTRACT

This paper addresses both the modeling and the resolution of the replacement problem for a population of machines. The main objective is the computation of a minimum cost replacement policy, which, based on the status of each machine, determines whether one or more machines have to be replaced over a given finite time horizon.

The replacement problem of a set of machines can be regarded as a sequential decision-making problem under uncertainty. Thanks to this, we propose a novel formulation for such problems consisting of a composition of discrete-time multi-state Markov Decision Processes (MDPs), one for each specific machine. The underlying optimization problem is formulated as a stochastic Dynamic Programming (DP), and then solved by using the principles of the backward DP algorithm. Moreover, to deal with the curse of dimensionality due to the high-cardinality state–space of real-world/industrial applications, a new generalized multi-trajectory Least-Squares Temporal Difference (LSTD) based method is introduced. The resulting algorithm computes an approximate optimal cost function by: (i) running Monte Carlo simulations over different trajectories of a given length; (ii) embedding the policy improvement step within the recursive LSTD iterations; (iii) enforcing an off-policy mechanism to improve the LSTD exploration capabilities. A study on the convergence properties of the proposed approach is also provided. Several numerical examples are given to illustrate its effectiveness in terms of parametric sensitivity, computational burden, and performance of the computed policies compared with some heuristics defined in the literature.

1. Introduction

The machine replacement problem has been addressed since a long time and is a relevant topic in industrial engineering and management science (Jones et al., 1991; Gress et al., 2012; Pan and Thomas, 2010). In a typical industrial ecosystem, machines are under constant usage, and their key performance indicators can deteriorate over time (Wang, 2002; Jiang et al., 2022; Lin and Dou, 2015). As a result of this, such machines need replacement at a certain point in time, otherwise the efficiency of the overall industrial ecosystem can suffer a lot. Proper maintenance measures need to be put in place in order to make a trade-off between the replacement costs and the ones related to degraded operational conditions (e.g., production losses).

In this paper, we are interested in modeling and solving the replacement problem for a population of machines. A fixed number of machines is in operation at all times. The operating cost of each machine increases as time passes and the machine gets older. An older machine may have to be replaced by a new one when its operating costs become too high. There is a fixed purchase and installation cost associated with new machines. Our main objective is the computation of a minimum cost replacement policy, specifying the “keep” or the “replace” actions for all the machines at each time slot over a given finite time horizon.

Machine replacement problems can be regarded as sequential decision-making problems under uncertainty, and can be tackled using Markov Decision Processes (MDPs) and Dynamic Programming (DP) methods, see Gress et al. (2012), Pan and Thomas (2010) and Bertsekas (2017). We model the overall machine replacement problem as a composition of Markov Decision Processes (MDPs), one for each specific machine. In principle, the resulting stochastic DP problem can be solved exactly by using the backward DP algorithm for the finite time horizon case, while the Value Iteration (VI) algorithm or the Policy Iteration (PI) algorithm can be adopted for the infinite time

* Corresponding author.
E-mail addresses: ali.forootani@mu.ie, aliforootani@ieee.org (A. Forootani), m.ghaniee@basu.ac.ir (M.G. Zarch), massimo.tipaldi@poliba.it (M. Tipaldi), rafierv@unina.it (R. Iervolino).

https://doi.org/10.1016/j.engappai.2022.105638
Received 26 February 2022; Received in revised form 18 October 2022; Accepted 10 November 2022
Available online 28 November 2022

horizon case (Bertsekas, 2017, 2012). However, the usage of these exact techniques in real-world applications is limited by the so-called curse of dimensionality, since complex systems are involved, formulated through models featured by high-cardinality state spaces (Bertsekas, 2012; Li et al., 2012; Hu et al., 2022; Forootani et al., 2020a). For this reason, efforts have been devoted to finding suitable techniques able to address such issue in an approximate way (Forootani et al., 2022a; Bertsekas, 2019). This research field has evolved under different names (e.g., Approximate Dynamic Programming (ADP) and Reinforcement Learning (RL), see Bertsekas (2012, 2011a) and Garí et al. (2021)), and owes its results to the fruitful cross-fertilization among artificial intelligence, optimal control theory, and operations research (Bertsekas, 2011a; Forootani et al., 2019; Iervolino et al., 2021).

In this paper, we introduce a multi-trajectory Least-Squares Temporal Difference (LSTD) based method to solve machine replacement problems with large state spaces. This method falls within the category of cost function approximations (Bertsekas, 2011a). A cost function gives the expected cumulative cost when starting from a specific state, and then following a given policy (with the expectation performed over all the possible trajectories). An optimal policy is one minimizing the cost function for each state (Bertsekas, 2012). The main technique to cope with systems with large state spaces is to approximate such cost function via a more compact parametric representation in the subspace of selected features (which is also referred to as the approximation architecture Bertsekas, 2012), and then perform Monte Carlo simulations to collect samples and solve the underlying optimization problem (Bertsekas, 2011b; Geist and Pietquin, 2013).

The paper main contributions together with its organization can be summarized as follows:

- After providing some preliminaries about MDP, DP, and ADP in Section 2, we formulate the replacement problem of a set of machines in terms of a composition of MDPs in Section 3. As also shown in the following paragraph, to the best of our knowledge, this modeling has never been reported elsewhere.
- The generalized multi-trajectory LSTD algorithm is presented in Sections 4 and 6. This algorithm computes an approximate optimal cost function by running Monte Carlo simulations over different trajectories of a given length, by embedding the policy improvement step within the recursive LSTD iterations, and enforcing an off-policy mechanism to improve the LSTD exploration capabilities. A study on its convergence properties is also provided. Moreover, some practical implementation aspects are given in Section 7.
- In Section 8, both the exact DP algorithm and the generalized multi-trajectory LSTD algorithm are applied to solve machine replacement problems formulated as a set of MDPs. To this aim, a MATLAB-based application has been developed to formulate and solve such problems as well as to analyze and evaluate the corresponding computed policies. In particular, the parametric sensitivity of the proposed algorithm is shown, its computational burden measured, and the performance of the computed policies compared with some heuristics defined in the literature.
- Moreover, still in Section 8, we show and examine the results of the generalized multi-trajectory LSTD algorithm in its different configurations and for different sets of features to prove its flexibility. Section 9 concludes the paper.

Finally, it is worth highlighting that the proposed multi-trajectory LSTD algorithm can be applied to any (not necessarily industrial) application formulated as a sequential decision-making problem under uncertainty (and featured by a large state space). In this regard, we can mention energy management systems (Zhu et al., 2022), electric vehicle fleet operations (Lee and Boomsma, 2022), railway traffic management (Ghasempour and Heydecker, 2019), and capacity allocation problems in the service industry (Schütz and Kolisch, 2012).

1.1. A short literature review

To the best of our knowledge, most of the existing papers dealing with machine replacement problems (and more in general, with maintenance issues of deteriorating systems Wang, 2002) address single-machine systems (or systems with very few machines, each responsible for specific tasks). For instance, in Ouaret et al. (2018), the problem of simultaneous production planning and replacement control of a single manufacturing machine was addressed. Random phenomena, such as the quality deterioration and customers' demand, were formulated via a continuous-time dynamic model. In Ouaret et al. (2019), the same authors focused on the production and replacement problem of a hybrid manufacturing system composed of a manufacturing machine and a remanufacturing machine. An MDP formulation, based on the extension of the state space of the system (operational, repair and replacement), was conceived for the resulting control law to take into account the history of breakdowns and repairs. The problem of joint optimization of production and replacement policies was studied in terms of the evolution of finished product inventory, returned product inventory, and the history of machine breakdowns and repairs. The underlying machine dynamics were described by a continuous time stochastic process with a discrete state transition representation. In both the papers, the corresponding optimality conditions were formulated and solved by using a second-order approximation of Hamilton–Jacobi–Bellman (HJB) equations (Bertsekas, 2017). The robustness analysis of the computed control policy versus specific model parameters was also performed. In Dong et al. (2021), a periodic replacement policy and an inspection replacement policy for a single unit system (subject to both stochastic deterioration and external shocks) are compared to determine which one is more profitable in terms of the relative gain on the average maintenance cost rate.

A hierarchical decision making approach in production and repair/replacement planning of a single machine was presented in Nodem et al. (2009). The main goal of the paper was to determine the production rate and the repair/replacement policy minimizing the total expected cost when the machine deteriorates with age, and is subject to damage failures. The authors showed how to operate the machine as the machine aged or when a higher number of breakdowns occurred.

The integration of replacement and preventive/corrective maintenance for a single machine subject to failures is another relevant topic addressed in the literature, see Sharifi and Taghipour (2021), Leu and Ying (2020), Cassady and Kutanoglu (2005) and Nodem et al. (2011). In this regard, an interesting overview can be found in Nowakowski and Werbińka (2009), where three different classes of maintenance optimization models were presented, that is to say, the block replacement models, the group maintenance models, and the opportunistic maintenance models. Moreover, an example of a two-unit system maintenance process was provided in order to compare various maintenance policies.

In the recent work (Liu et al., 2021), a Conditional Based Maintenance (CBM) model for a two-unit system over a finite time horizon was considered. In particular, the maintenance problem was modeled via an MDP framework and the maintenance cost was optimized by employing the backward procedure of the exact DP algorithm. The influence of both economic dependence and degradation processes on the optimal policy was also assessed. However, the applicability of the idea was limited to small size problems. In Schouten et al. (2022), a single component model for maintenance optimization under time varying costs was presented. The life of the component was modeled as a discrete-time MDP and two policies were evaluated, i.e., age-based and block-based replacement. The authors derived a periodic-age replacement policy under mild conditions as the optimal solution. In another line of research shown in van Staden et al. (2022), the historical machine failures and maintenance records were used to derive future failure estimates and schedule preventive maintenance. The optimization problem was formulated via a finite time horizon MDP, and a data driven solution was computed to determine when to deviate from the planned preventive maintenance.

In Childress and Durango-Cohen (2005), the authors also considered to model the parallel machine replacement problem via MDPs. By taking into account classes of replacement cost functions, the optimal policy was computed analytically with predefined assumptions. Despite of having valuable results, this work differs from our framework since we seek to compute more general and practical solutions based on approximate cost functions and Monte Carlo simulations. In the more recent work (Li, 2020), the authors studied the replacement problem of economically interdependent machines with the three actions: keep, replace, and general repair. They proved that, under common cost assumptions, there was an optimal policy such that machines in the same state were either all generally repaired or none of them. In Seif et al. (2019), parallel machine replacement problems were formulated as a two-stage stochastic program with an uncertain planning horizon and applied to construction projects. Numerical analyses were conducted to obtain managerial implications.

The distinction between our work and the existing literature can be highlighted as follows. First of all, in our paper, the replacement problem of a set of machines is modeled as the composition of discrete-time multi-state degradation Markov processes. Then, a multi-trajectory LSTD based method is used to solve replacement machine problems with a large state space by approximating the original cost function via a parametric representation in the subspace of selected features. The initial state of each trajectory is selected by applying a probability distribution different from the frequencies of the associated MDP at hand in order to generate a richer mixture of state visits. In the literature, there exist some examples of ADP/RL methods used to solve machine replacement problems. However, they do not usually address the scalability issue of real-world machine replacement problems. For instance, in Huang et al. (2019), the basic look-up table Q-learning algorithm was adopted to solve machine preventive replacement problems in serial production lines. In that paper, the authors themselves suggested the adoption of cost function approximations to deal with systems featured by a larger state space. Further similar examples can be found in Yousefi et al. (2020) and Zhang et al. (2021).

2. Preliminaries on MDPs, DP and ADP

This section provides an essential background on the mathematical models and tools we will exploit for the machine replacement problem formulation and the relative proposed solution. The basic structure of an MDP can be defined as follows Bertsekas (2012) and Forootani et al. (2020a)

- $X = \{x^1, \dots, x^\Omega\}$ is the finite set of states, where $x^v \in X$ and $x^w \in X$ denote two generic elements of this set and $|X| = \Omega$ its cardinality. The state variable at time slot $k \in \mathcal{T}$, $\mathcal{T} = \{0, 1, \dots, T\}$, is denoted by $x(k)$, with $x(k) \in X$.
- $U = \{u_1, \dots, u_\sigma\}$ is the finite set of actions (or decisions), where $u \in U$ denotes an element of this set and $|U| = \sigma$. We also define $\mu(x^v, k) : X \times \mathcal{T} \to U$ as the time-varying control function mapping that maps the state x^v into action u at the time slot k. $U(x^v)$ denotes the set of admissible actions at state x^v.
- The state transition probability function is defined as $\mathcal{P}_{x^v x^w}(u) := \left[P(x(k+1) = x^w | x(k) = x^v, u)\right]$, $P : X \times U \times X \to [0, 1]$. It represents the probability that an action $u \in U$ performed in the state $x^v \in X$ at the time slot k leads the system to the state $x^w \in X$ at time slot $k+1$. We denote with $\mathcal{P} \in \mathbb{R}^{\Omega \times \sigma \times \Omega}$ the state transition probability matrix with elements $\mathcal{P}_{x^v x^w}(u)$.
- $g(x^v, u) : X \times U \to [0, +\infty)$ is the cost per stage function.

By using MDP based frameworks, it is possible to formulate and solve stochastic sequential decision problems. At the core of such frameworks, there is the resolution of a stochastic optimization problem (Bertsekas, 2012).

Let $\pi = \{\mu(0), \dots, \mu(T-1)\}$ denotes the policy, that is to say, the sequence of control vector functions $\mu(k)$ applied on the whole state space X over the finite time horizon T. More specifically, $\mu(k)$ is a vector function with components $\mu(x(k), k), \forall x(k) \in X, \forall k \in \mathcal{T}$.

To evaluate the performance of a given policy π over the finite time horizon T, we need to introduce the following (cumulative) cost function

$$J_\pi\big(x(0)\big) = E\left\{ g_T\big(x(T)\big) + \sum_{k=0}^{T-1} \alpha^k g\big(x(k), \mu(x(k), k)\big) \right\}, \tag{1}$$

where $E\{\cdot\}$ is the expectation operator, $0 < \alpha < 1$ is the discount factor, $g_T(x(T))$ is a given bounded terminal cost, and $x(T)$ is the state at the terminal time T. Solving sequential decision making problems over a finite time horizon means finding an optimal policy $\pi^* = \{\mu^*(0), \dots, \mu^*(T-1)\}$ minimizing the cost function (1) from any possible initial state $x(0)$

$$J^*\big(x(0)\big) = \min_\pi J_\pi\big(x(0)\big), \tag{2}$$

where $J^*(\cdot)$ is called optimal cost function.

The expressions (1) and (2) can be easily extended to the infinite time horizon case by computing their limit value for $T \to \infty$ and setting g_T to zero (Bertsekas, 2012). In such a case, the optimization problem (2) only addresses stationary policies, i.e., $\pi = \{\mu, \mu, \dots\}$, where μ is the stationary control vector function with components $\mu(x(k)) : X \to U, \forall x(k) \in X$. As a result, the stationary optimal policy $\pi^* = \{\mu^*, \mu^*, \dots\}$ can be computed (Bertsekas, 2012). Later in the paper, stationary policies are also denoted with μ.

By defining a generic vector cost function $J : X \to \mathbb{R}^\Omega$ as a vector with components $J(x^v)$, we can introduce the optimal Bellman operator $\mathcal{F}^* : \mathbb{R}^\Omega \mapsto \mathbb{R}^\Omega$ whose components are

$$(\mathcal{F}^* J)(x^v) := \min_{u \in U}\Big(g(x^v, u) + \alpha \sum_{x^w \in X} \mathcal{P}_{x^v x^w}(u) J(x^w)\Big). \tag{3}$$

The Bellman operator $\mathcal{F}^*$ can be viewed as a mapping that transforms the vector cost function J on X into the vector function $\mathcal{F}^* J$ on X. The optimal cost function J^* for the infinite time horizon case is the fixed point of Bellman equation $J^* = \mathcal{F}^* J^*$ (Bertsekas, 2012).

When we refer to a specific stationary policy $\pi = \{\mu, \mu, \dots, \}$ (either optimal or not), we can introduce the Bellman operator $\mathcal{F} : \mathbb{R}^\Omega \mapsto \mathbb{R}^\Omega$ with components

$$(\mathcal{F} J)(x^v) := g(x^v, \mu(x^v)) + \alpha \sum_{x^w \in X} \mathcal{P}_{x^v x^w}(\mu(x^v)) J(x^w). \tag{4}$$

As shown in Forootani et al. (2022a), the two operators can be expressed in a more compact vector form. In particular, (4) becomes

$$\mathcal{F} J = g + \alpha \mathcal{P} J, \tag{5}$$

where g is the cost per stage vector with components $g(x^v, \mu(x^v))$ and, with a slight abuse of notation, the explicit dependency of $\mathcal{P}$ and g from the specific stationary policy has been removed. Likewise, we can define the transition probability matrix and the cost per stage vector associated to the optimal stationary policy π^* with $\mathcal{P}^*$ and g^*, respectively. Thus, the optimal Bellman operator (3) can be expressed in a more compact vector format as follows (Forootani et al., 2022a)

$$\mathcal{F}^* J = g^* + \alpha \mathcal{P}^* J. \tag{6}$$

We will denote by $\mathcal{F}^l$ the composition of the mapping $\mathcal{F}$ with itself l times, that is for all l we write

$$\Big(\mathcal{F}^l J\Big)(x^v) = \Big(\mathcal{F}\big(\mathcal{F}^{l-1} J\big)\Big)(x^v), \quad x^v \in X. \tag{7}$$

When $l \to \infty$, the associated cost satisfies the fixed point mapping (Bertsekas, 2012),

$$J_{\mathcal{F}}(x^v) = \lim_{l \to \infty}\Big(\mathcal{F}^l J\Big)(x^v). \tag{8}$$

2.1. On solving stochastic DP problems

In case of computationally tractable problems, we can use the exact Bellman DP algorithm to solve exactly the optimization problem (2) over the finite time horizon (Bertsekas, 2017), while the Value Iteration or the Policy Iteration algorithms can be used to solve it exactly over the infinite time horizon (Bertsekas, 2012).

On the other hand, in case of large scale stochastic optimization problems, we can employ ADP methods, which basically address the so-called curse of dimensionality via architecture approximations and simulations (Bertsekas, 2011a). In this paper, we approximate any cost function $J(x^v)$ by means of a parametric architecture of the form $\tilde{J}(x^v, r)$ (or $\tilde{J} : X \times \mathbb{R}^\gamma \to \mathbb{R}^\Omega$ in its vector function representation), where $r \in \mathbb{R}^\gamma$ is a parameter vector with the component r_i and has to be computed by training the selected architecture ($\gamma \ll \Omega$). More specifically, a linear feature-based parametric architecture is chosen and $\tilde{J}(x^v, r)$ is defined as the inner product $\phi(x^v)'r$, where $\phi(x^v) = [\phi_1(x^v), \dots, \phi_\gamma(x^v)]'$ is referred to as the feature vector with γ given features as components (in this paper, the symbol $'$ denotes the transpose operator).

As a result, the vector cost function J can be approximated by a vector in the feature subspace $\Delta = \{\Phi r | r \in \mathbb{R}^\gamma\}$, where $\Phi \in \mathbb{R}^{\Omega\times\gamma}$ is called feature matrix with each row $\phi(x^v)'$. In other words, we consider the approximation of the cost function J in the form of $J \approx \Phi r$, where the feature matrix Φ is defined by the designer by exploiting the knowledge of the system at hand and $r \in \mathbb{R}^\gamma$ is the parameter vector that has to be computed via Monte Carlo simulations approaches, e.g., the LSTD method (Bertsekas, 2011b). This leads us to address the following weighted least-squares minimization problem (Bertsekas, 2011a, 2012)

$$\tilde{r} = \arg\min_{r\in\mathbb{R}^\gamma} \|J - \Phi r\|^2_\epsilon, \tag{9}$$

where $\epsilon \in \mathbb{R}^\Omega_+$ with $\|\epsilon\|_1 = 1$ is a weighting vector. As shown later in the paper, we conveniently choose as weighting vector the steady state probability vector associated with the stochastic dynamics of the system at hand (Bertsekas, 2019). In this paper, the parameter vector used to approximate the optimal cost function J^* and the cost function J of a specific stationary policy π are denoted with $\tilde{r}^*$ and $\tilde{r}$, respectively (note that the explicit dependency on the policy π has been removed). Moreover, the following two assumptions are made throughout this paper.

Assumption 1. For each admissible stationary policy π (and, hence, for the optimal policy π^*), the underlying Markov chain is irreducible (i.e., it has a single recurrent class and no transient states Bertsekas, 2012). The related stochastic matrix $\mathcal{P}$ (and, hence, $\mathcal{P}^*$) has unique steady state probability vectors ϵ with components $\epsilon_{x^v} > 0$ ($\epsilon^* \in \mathbb{R}^\Omega_+$ with components $\epsilon^*_{x^v} > 0$).

Assumption 2. The matrix Φ has rank γ.

Thanks to them, as shown in Bertsekas (2011a), one can sample according to such steady probability distributions in order to compute via Monte Carlo simulations the parameter vector r of linear cost function approximations. All these aspects are exploited in the paper.

3. The machine replacement problem and its stochastic DP formulation

The machine replacement problem definition for a single machine can be found in Bertsekas (2017). More in general, let us consider the problem of operating m different machines over the finite time interval $\mathcal{T}$ in an efficient way. Each machine M_i (with $i \in \{1, \dots, m\}$) can be in any of Ω operating states,[1] denoted by $x_i^{v_i}$ with $v_i \in \{1, \dots, \Omega\}$. It is assumed that the conditions of the machine in the state $x_i^{v_i}$, with $v_i \in \{1, \dots, \Omega-1\}$, are better than those ones associated with the state $x_i^{v_i+1}$, and the state x_i^1 corresponds to a machine M_i in its fully operational conditions. Moreover, by denoting with $c_i(x_i^{v_i})$ the operating cost per time slot (or stage) when machine M_i is in state $x_i^{v_i}$, we have the following

$$c_i(x_i^1) \le c_i(x_i^2) \le \dots \le c_i(x_i^\Omega), \tag{10}$$

which is known as *non-decreasing cost function*.

At the beginning of each time slot $k \in \mathcal{T}$, we know the state $x_i^{v_i}$ of each machine M_i and one of the following two actions can be performed:

- Let the machine M_i operate one more time slot in the state it currently is, i.e., the action u_i is set to 0.
- Replace the machine M_i with $u_i = 1$ at a cost R_i and restart from the perfect state x_i^1.

During any time slot of operation $k \in \mathcal{T}$, in case $u_i = 0$, the state of any machine can become worse or it may stay unchanged. Hence, the state transition probabilities $\mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0)$ satisfy the conditions $\mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0) = 0$ (if $v_i > w_i$) and $\left(\mathcal{P}_{x_i^{v_i+1} x_i^{w_i}}(0) - \mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0)\right) \ge 0$, known as increasing *failure rate*. Note that non-decreasing cost function and increasing failure rate are common assumptions in the literature (Childress and Durango-Cohen, 2005).

In case we decide to replace ($u_i = 1$), we assume that the time required for the machine replacement is negligible with respect to the time slot needed to assess the operational conditions of the machines. As a result, the addressed machine M_i moves to the state x_i^1 with probability equal to 1 and stays in that state x_i^1 for at least the current time slot. In the subsequent ones, it may deteriorate according to the state transition probabilities $\mathcal{P}_{x_i^{v_i} x_i^{w_i}}$. At each time slot, it is assumed that more than one machine can be replaced simultaneously.

The state transition probability graph for one single machine can be found in Bertsekas (2017). The main objective of the machine replacement problem is to assess the level of deterioration of each machine at which it is worth paying the cost of the machine replacement, say R_i, thereby obtaining the benefit of smaller future operating costs. As a result, the replacement decision incurs a cost of $R_i + c(x_i^1)$, while the decision of not replacing the machine implies an operating cost equal to $c_i(x_i^{v_i})$.

By exploiting the stochastic DP formulation used by the authors to formulate resource allocation problems (see Forootani et al. (2020a, 2019, 2021a) and Forootani et al. (2021b)), we can model each machine M_i as an MDP by defining the tuple $M_i = \langle X_i, U_i, \mathcal{P}_i, g_i \rangle$ as follows

- X_i is the state space, $X_i = \{x_i^1, x_i^2, \dots, x_i^\Omega\}$. The state variable at time k is denoted with $x_i(k)$, where $x_i(k) \in X_i$. Moreover, we denote with $x_i^{v_i}$ and $x_i^{w_i}$ two generic states of the machine M_i.
- $U_i = \{1, 0\}$ is the finite set of actions (also called control inputs or decisions). Its element is denoted by u_i, with $u_i = 1$ representing the replace action and $u_i = 0$ the action of continuing to operate the machine M_i. The control function at time k is denoted as $\mu_i(x_i(k), k)$.
- $\mathcal{P}_i$ is the state transition probability matrix with elements

$$\mathcal{P}_{x_i^{v_i} x_i^{w_i}}\left(u_i\right) := P\left[x_i(k+1) = x_i^{w_i} | x_i(k) = x_i^{v_i}, u_i\right]. \tag{11}$$

- $g_i(x_i^{v_i}, u_i) : X_i \times U_i \to [0, \infty)$ is the cost per stage function

$$g_i(x_i^{v_i}, u_i) = c_i(x_i^{v_i})(1 - u_i) + u_i\left(R_i + c_i(x_i^1)\right). \tag{12}$$

Note that, for the sake of the simplicity, we use the same notation for the machine and its related MDP representation.

The composition of m MDPs associated with each machine M_i gives rise to the formulation of the overall machine replacement problem M. In particular, such problem can be modeled as an MDP, defined by the tuple $M = \langle X, U, \mathcal{P}, g \rangle$ where

[1] For the sake of simplicity and without loss of generality, it is assumed the same number of possible states for all the machines.

- X is the entire state space, $X = \{x^h = (x_1^{h_1}, \dots, x_m^{h_m})' \in \bigtimes_{i=1}^m X_i\}$. To simplify the complexity in the notation and with a slight abuse of notation, we denote with $x(k)$ the state variable at time k. Moreover, $x^v = (x_1^{v_1}, x_2^{v_2}, \dots, x_m^{v_m})'$ and $x^w = (x_1^{w_1}, x_2^{w_2}, \dots, x_m^{w_m})'$ are two generic states of the process.
- $u = (u_1, \dots, u_m)' \in U$ is the overall control input, with $U = \bigtimes_{i=1}^m U_i$. In line with the notation introduced in the previous section, the control function at time k is denoted as $\mu(x(k), k) = (\mu_1(x_1(k), k), \dots, \mu_m(x_m(k), k))'$.
- $\mathcal{P}$ is the state transition probability matrix with elements

$$\mathcal{P}_{x^v x^w}(u) := \prod_{i=1}^m \mathcal{P}_{x_i^{v_i} x_i^{w_i}}(u_i). \tag{13}$$

- $g(x^v, u) : X \times U \to [0, \infty)$ is the cost per stage function

$$g(x^v, u) = \sum_{i=1}^m g_i(x_i^{v_i}, u_i). \tag{14}$$

By using (1) and (14), the cost function of a given policy π for the overall machine replacement problem M over the finite time horizon T can be formulated as follows

$$J_\pi(x(0)) = E\left[g_T(x(T)) + \sum_{k=0}^{T-1} \alpha^k \sum_{i=1}^m g_i(x_i(k), \mu_i(x_i(k), k))\right]. \tag{15}$$

3.1. Remarks on the proposed machine replacement problem formulation

In the following, we present some useful properties and remarks on the introduced MDP formulation used to model the machine replacement problem.

Property 1. *If we denote by $N_i(x_i^{v_i}, u_i)$ the number of state transitions for a given state $x_i^{v_i}$ when the control input u_i is applied to a specific machine M_i, it is easy to verify that*

$$N_i(x_i^{v_i}, u_i) = \begin{cases} \Omega - v_i + 1, & \text{if } u_i = 0, \\ 1, & \text{if } u_i = 1. \end{cases} \tag{16}$$

More in general, for the overall set of machines, the total number of state transitions $N(x^v, u)$, when applying an admissible control input u to any given state x^v, is

$$N(x^v, u) = \prod_{i=1}^m N_i(x_i^{v_i}, u_i). \tag{17}$$

Lemma 1. *The transition probability matrix corresponding to a single machine M_i is upper triangular when the control input "keep" is applied.*

Proof. The proof simply follows from the definition of the problem since $\mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0) = 0, \quad \forall w_i < v_i$. □

Lemma 2. *Consider the MDP formulation M_i for a specific machine. Assume that, $\forall v_i \in \{1, \dots, \Omega - 1\}$, it is*

(i) $0 \le \mathcal{P}_{x_i^{v_i} x_i^{v_i}}(0) < 1$;
(ii) $\mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0) \neq 0$, *for at least one* $w_i > v_i$.

If we always apply the control input "keep", the state x_i^Ω is an attractive equilibrium state.

Proof. Let us define with $p_{x_i}(k)$ the vector whose elements $p_{x_i^{v_i}}(k)$ correspond to the probability that the state of the machine at time k is $x_i^{v_i}$. Moreover, for the sake of simplicity, let us define $\mathcal{P}_{x_i^{v_i} x_i^{w_i}} := \mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0)$. The probabilities of the underlying Markov chain entering specific states at time $k+1$ are related to the ones of the previous time step k by the expression $p_{x_i}(k+1)' = p_{x_i}(k)' \mathcal{P}_i(0)$ (Luenberger, 1979). By applying Lemma 1, such relationship can be written as follows for the machine M_i

$$p_{x_i^1}(k+1) = \mathcal{P}_{x_i^1 x_i^1} p_{x_i^1}(k), \tag{18}$$

$$p_{x_i^2}(k+1) = \mathcal{P}_{x_i^1 x_i^2} p_{x_i^1}(k) + \mathcal{P}_{x_i^2 x_i^2} p_{x_i^2}(k), \tag{19}$$

$$p_{x_i^3}(k+1) = \mathcal{P}_{x_i^1 x_i^3} p_{x_i^1}(k) + \mathcal{P}_{x_i^2 x_i^3} p_{x_i^2}(k) + \mathcal{P}_{x_i^3 x_i^3} p_{x_i^3}(k), \tag{20}$$

$$\vdots \tag{21}$$

$$p_{x_i^\Omega}(k+1) = \mathcal{P}_{x_i^1 x_i^\Omega} p_{x_i^1}(k) + \mathcal{P}_{x_i^2 x_i^\Omega} p_{x_i^2}(k) + \cdots + \mathcal{P}_{x_i^\Omega x_i^\Omega} p_{x_i^\Omega}(k). \tag{22}$$

Being the sum of all the row elements of any transition probability matrix equal to 1 and from Lemma 1, we have $\mathcal{P}_{x_i^\Omega x_i^\Omega} = 1$. The equilibrium point $\bar{p}_{x_i}$ of the above system of linear iterative equations can be computed by letting $k \to \infty$. In particular, by denoting with $\bar{p}_{x_i^{v_i}}$ the generic element of $\bar{p}_{x_i}$, we have the following

$$\bar{p}_{x_i^1} = \mathcal{P}_{x_i^1 x_i^1} \bar{p}_{x_i^1}, \tag{23}$$

$$\bar{p}_{x_i^2} = \mathcal{P}_{x_i^1 x_i^2} \bar{p}_{x_i^1} + \mathcal{P}_{x_i^2 x_i^2} \bar{p}_{x_i^2}, \tag{24}$$

$$\bar{p}_{x_i^3} = \mathcal{P}_{x_i^1 x_i^3} \bar{p}_{x_i^1} + \mathcal{P}_{x_i^2 x_i^3} \bar{p}_{x_i^2} + \mathcal{P}_{x_i^3 x_i^3} \bar{p}_{x_i^3}, \tag{25}$$

$$\vdots \tag{26}$$

$$\bar{p}_{x_i^\Omega} = \mathcal{P}_{x_i^1 x_i^\Omega} \bar{p}_{x_i^1} + \mathcal{P}_{x_i^2 x_i^\Omega} \bar{p}_{x_i^2} + \cdots + \mathcal{P}_{x_i^\Omega x_i^\Omega} \bar{p}_{x_i^\Omega}. \tag{27}$$

From the assumption (i) and by considering that $\mathcal{P}_{x_i^\Omega x_i^\Omega} = 1$, it easy to verify that x_i^Ω is the (unique) solution of the above system of equations. Thanks to the assumption (ii) (i.e., for each state $x_i^{v_i}$, there is always the possibility of entering at least one state $x_i^{w_i}$ with $w_i > v_i$) and Lemma 1 (i.e., it is not possible to enter a state $x_i^{v_i}$ once entering a state $x_i^{w_i}$ with $w_i > v_i$), the equilibrium point x_i^Ω is also attractive. □

Corollary 1. *We define with $\mathcal{P}_i(u_i)$ the transition probability matrix associated with the machine M_i when the control input u_i is applied on its entire state space. Similarly, $\mathcal{P}(u)$ is the transition probability matrix of the overall machine replacement problem M when the control input u is applied. Then, the following relation holds*

$$\mathcal{P}(u) = \mathcal{P}_1(u_1) \otimes \mathcal{P}_2(u_2) \cdots \otimes \mathcal{P}_m(u_m), \tag{28}$$

where $\otimes$ denotes the Kronecker product.[2]

Corollary 2. *From Lemma 2 and Corollary 1, the vector $(x_1^\Omega, x_2^\Omega, \dots, x_m^\Omega)'$ is the equilibrium point of the overall machine replacement problem M if the policy "keep" is always applied to all the machines M_i.*

Remark 1. The Markov chain corresponding to the policy used in Corollary 2 (i.e., $u_i = 0$ in all the states, for all the time steps and all the machines) is not irreducible (indeed, each machine has $\Omega - 1$ transient state), and thus such policy does not fulfill Assumption 1. In the remaining parts of the paper, we always consider stationary policies fulfilling Assumption 1. In particular, as for the machine replacement problem M, it is sufficient to consider stationary policies with the associated Markov chains satisfying at least the following two properties for each machine M_i:

(i) in the state x_i^Ω, we always apply the replacement action $u_i = 1$;
(ii) for each state $x_i^{v_i}$ there exists at least one $w_i > v_i$ such that $\mathcal{P}_{x_i^{v_i} x_i^{w_i}}(0) \neq 0$.

Fig. 1 shows a schematic diagram of the replacement problem for a single machine with $\Omega = 3$ states. To have a better understanding of the machine replacement problem formulation, we present the following two illustrative examples.

[2] The joint transition probabilities are given by the product of the single transition probabilities being the machine state evolutions independent.

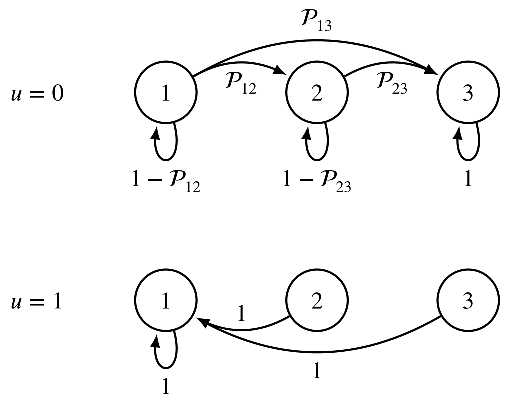


Fig. 1. Schematic diagram of machine replacement problem for a single machine with 3 states.

Example 3.1. Consider a machine replacement problem for a single machine with $\Omega = 5$ states and with the following state transition probabilities

$$\mathcal{P}_{x^v x^w}\big(u\big) = \frac{1}{N(x^v, u)}. \tag{29}$$

More specifically, the state transition probabilities with $u = 0$ are

$$\mathcal{P}_{11}\big(u=0\big) = \mathcal{P}_{12}\big(u=0\big) = \mathcal{P}_{13}\big(u=0\big) = \mathcal{P}_{14}\big(u=0\big) = \mathcal{P}_{15}\big(u=0\big) = \frac{1}{5},$$

$$\mathcal{P}_{22}\big(u=0\big) = \mathcal{P}_{23}\big(u=0\big) = \mathcal{P}_{24}\big(u=0\big) = \mathcal{P}_{25}\big(u=0\big) = \frac{1}{4},$$

$$\mathcal{P}_{33}\big(u=0\big) = \mathcal{P}_{34}\big(u=0\big) = \mathcal{P}_{35}\big(u=0\big) = \frac{1}{3},$$

$$\mathcal{P}_{44}\big(u=0\big) = \mathcal{P}_{45}\big(u=0\big) = \frac{1}{2},$$

while, with $u = 1$, we have

$$\mathcal{P}_{11}\big(u=1\big) = \mathcal{P}_{21}\big(u=1\big) = \mathcal{P}_{31}\big(u=1\big) = \mathcal{P}_{41}\big(u=1\big) = \mathcal{P}_{51}\big(u=1\big) = 1.$$

We applied the exact DP algorithm with the two different replacement costs $R = 4$ and $R = 5$ to evaluate their impact on the computed control action for each state. The results are presented in Figs. 2 and 3. It is straightforward to see that, if we decrease the replacement costs, the obtained DP policy tends to use $u = 1$ more often. Such results also comply with the conclusion reported in Childress and Durango-Cohen (2005), i.e., in case of increasing failure rates and non-decreasing replacement costs, it is optimal to replace the machine if it enters any state worse than a particular state. In this example, for the case of replacement cost $R = 4$, it is optimal to replace the machine in any state worse than $x^v = 2$, while, for the case of replacement cost $R = 5$, it is optimal to replace the machine in any state worse than $x^v = 3$.

Example 3.2. Now suppose to apply the exact DP algorithm for the case of two machines each having $\Omega_i = 5$, $i = 1, 2$ states with different transition probabilities and different replacement costs. We assume that the first machine M_1 has the same state transition probabilities as defined in (29), while the second one M_2 has the following state transition probabilities

$$\mathcal{P}_{x_2^{v_2} x_2^{w_2}}\big(u_2 = 0\big) = \begin{cases} \frac{N_2(x_2^{v_2},0) - v_2}{N_2(x_2^{v_2},0)}, & \text{if } x_2^{v_2} = x_2^{w_2}, \\ \frac{1 - \frac{N_2(x_2^{v_2},0) - v_2}{N_2(x_2^{v_2},0)}}{N_2(x_2^{v_2},0) - 1}, & \text{if } x_2^{v_2} \neq x_2^{w_2}, \end{cases} \tag{30}$$

while the state transition probabilities are equal to 1 in case the replacement action is carried out in any state. The results of this experiment are shown in Figs. 4 and 5. Being the cardinality of the state space $\Omega = 25$, it is not convenient to show the state-control pairs for the entire time horizon as we did in Example 3.1. Instead, we performed the exact DP algorithm and saved its result as a lookup table. Then, we ran a Monte Carlo simulation to observe the visiting states, while employing the optimal control action extracted from the computed lookup table. Fig. 4 shows the visited states and the applied control actions for M_1, while Fig. 5 presents the same information for the second machine. It is worth highlighting that M_2 is replaced less frequently than M_1 thanks to the lower state transition probabilities. Indeed, from (30), it is easy to verify that if we apply the control action "not-replacement($u_2 = 0$)", it is more likely that M_2 stays in its current state rather than being deteriorated.

4. The generalized multi-trajectory LSTD algorithm

In this Section, we introduce the generalized multi-trajectory LSTD algorithm. In short, this algorithm computes an approximate optimal cost function by running Monte Carlo simulations over different trajectories of a given length and enforcing both off-policy or on-policy mechanism to improve the LSTD exploration capabilities.

The proposed algorithm is a generalized ADP method that belongs to the family of LSTD based enhanced exploration approaches. In particular, the proposed algorithm integrates on-policy single trajectory LSTD (Bertsekas, 2017), off-policy LSTD (Forootani et al., 2022a; Sutton et al., 2016; Forootani et al., 2022b), and multi-trajectory greedy LSTD (Forootani et al., 2020b) approaches. A study on its sufficient convergence properties is also provided.

To start with, let us fix a specific stationary policy μ. Note that we consider the fixed policy regardless of the finite or infinite time horizon.

Later in this Section, we include the policy improvements in our proposed approach within Monte Carlo trajectories.

The family of LSTD based algorithms makes use of Monte Carlo simulations to take samples for cost function of a fixed policy for the MDP at hand (see Bertsekas (2017, 2011a) for more details). Based on the sampling algorithm these methods are generally divided into on-policy and off-policy approaches (Sutton et al., 2016). The main challenge in Monte Carlo simulation approach is to enhance the exploration of the state space of the MDP, hence reaching better approximation. In this regard we use the multi-trajectory Monte Carlo simulation method. By doing so we are capable to visit more states and capture the nature of the MDP at hand.

In particular, we generate Q simulated trajectories, the states of a trajectory are generated according to the transition probabilities $\mathcal{P}_{x^v x^w}\big(\mu(x^v)\big)$, where μ is the stationary policy under evaluation, the transition cost is discounted by an additional factor α with each transition, and following each transition to a state x^w, the trajectory is terminated when we reach to the maximum length L.

Once a trajectory is terminated, an initial state for the next trajectory is chosen according to a fixed probability distribution $\epsilon(0) = \big(\epsilon_{x^1}(0), \ldots, \epsilon_{x^v}(0), \ldots, \epsilon_{x^\Omega}(0)\big)$, where

$$\epsilon_{x^v}(0) = Pr\big(x^v(0)\big), \tag{31}$$

and $Pr(\cdot)$ denotes any fixed probability distribution over the state space X, and $x^v(0)$ the initial state.

Let the jth trajectory have the form $\big(x^{v,j}(0), x^{v,j}(1), \ldots, x^{v,j}(L)\big)$, $j = 1, \ldots, Q$, where $x^{v,j}(0)$ is the initial state, and $x^{v,j}(L)$ is the state at which the trajectory is completed. For each state $x^{v,j}(l)$, $l = 0, \ldots, L-1$, of the jth trajectory, the accumulative simulated cost by setting the terminal cost function to its approximation $\phi\big(x^{v,j}(L)\big)' r_s$ is given by

$$\lambda_{j,l}(r_s) = \sum_{q=l}^{L-1} \alpha^{q-l} g\big(x^{v,j}(l), \mu(x^{v,j}(l))\big) + \alpha^{L-l} \phi\big(x^{v,j}(L)\big)' r_s, \tag{32}$$

where Φr_s is the representation of cost function J at stage s under any stationary policy μ and we assume it is given.

In particular, once the cost function $\lambda_{j,l}(r_s)$ is computed for all states $x^{v,j}(l)$ of the jth trajectory and for all trajectories $j = 1, \ldots, Q$, the vector r_{s+1} is obtained from r_s by a least-squares fit of these values

$$r_{s+1} = \arg\min_{r \in \mathbb{R}^\gamma} \sum_{j=1}^{Q} \sum_{l=0}^{L-1} \Big(\phi\big(x^v(l)\big)' r - \lambda_{j,l}(r_s)\Big)^2. \tag{33}$$

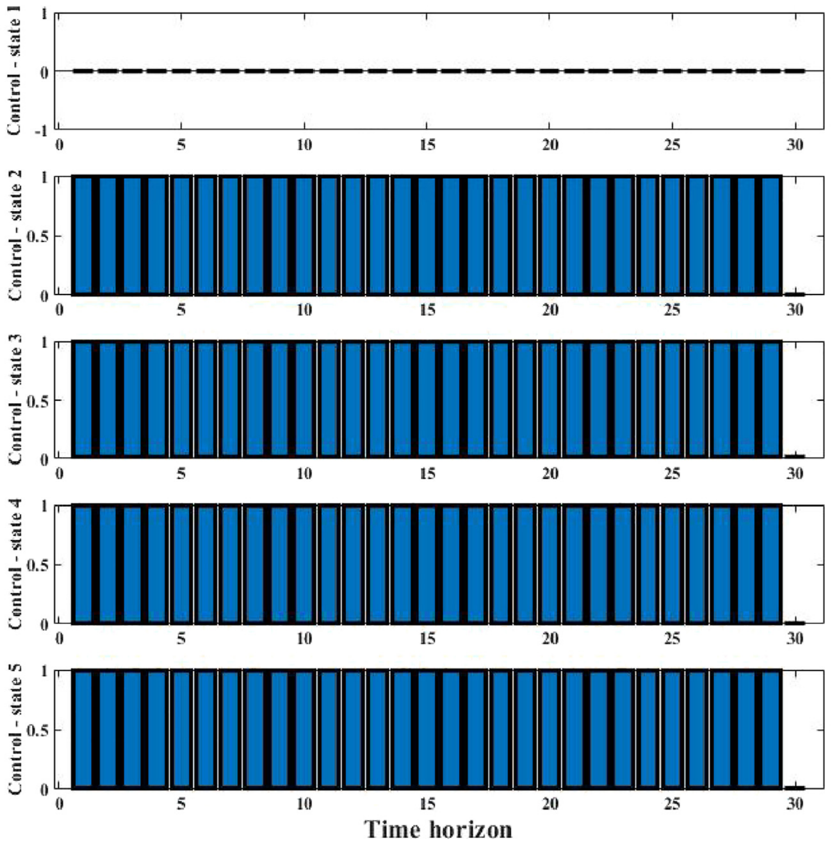


Fig. 2. State-Control pairs for the case of single machine, with time horizon $\mathcal{T} = 30$ and replacement cost $R = 4$.

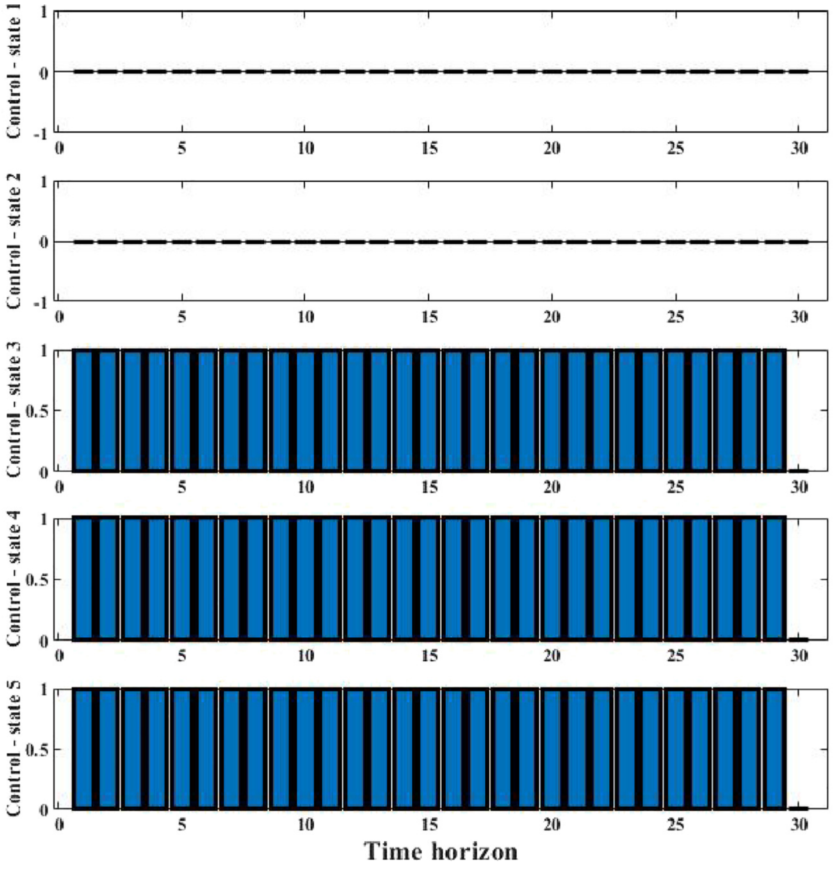


Fig. 3. State-Control pairs for the case of single machine, with time horizon $\mathcal{T} = 30$ and replacement cost $R = 5$.

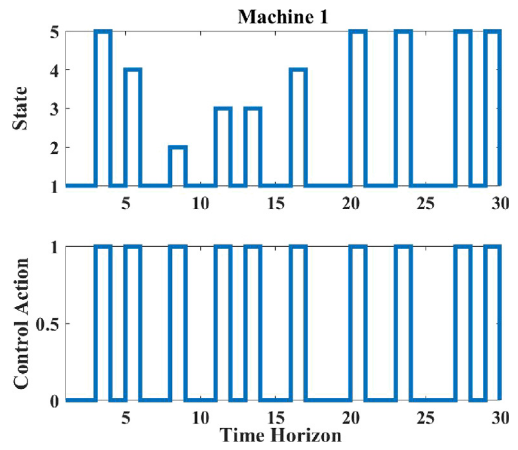


Fig. 4. State-Control pairs for M_1, with time horizon $\mathcal{T} = 30$ and replacement cost $R_1 = 4$ in one experiment.

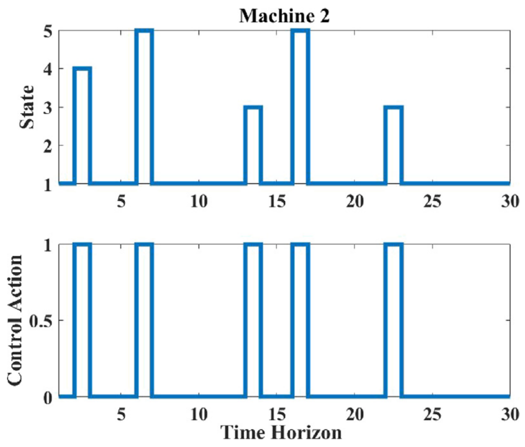


Fig. 5. State-Control pairs for M_2, with time horizon $\mathcal{T} = 30$ and replacement cost $R_2 = 6$ in one experiment.

It is worth highlighting that we can view (33) as a policy evaluation step in the policy iteration algorithm of the DP where r_{s+1} corresponds to the new policy that is being evaluated and r_s corresponds to the old policy. We can write the solution of least-squares problem (33) explicitly as

$$r_{s+1} = \left(\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\phi\big(x^v(l)\big)\phi\big(x^v(l)\big)'\right)^{-1}\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\phi\big(x^v(l)\big)\lambda_{j,l}(r_s). \tag{34}$$

In the following, we provide more details about multi-trajectory LSTD method.

To gain the insight regarding (33), let us utilize (5) and view $\mathcal{F}^{l+1}J$ as the vector of total discounted costs over a horizon of $(l+1)$ stages with the terminal cost function being J, and write one iteration of the Bellman's operator in the vector form

$$\mathcal{F}J = g + \alpha\mathcal{P}J. \tag{35}$$

Indeed over a horizon with $(l+1)$ stages and the terminal cost function J, in vector form we have

$$\mathcal{F}^{l+1}J = \alpha^{l+1}\mathcal{P}^{l+1}J + \sum_{q=0}^{l}\alpha^q\mathcal{P}^q g, \qquad l = 0,\dots,L-1. \tag{36}$$

As a result equivalently, we can write

$$\Big(\mathcal{F}^{l+1}J\Big)(x^v) = E\left\{\alpha^{l+1}J\big(x^v(l+1)\big) + \sum_{q=0}^{l}\alpha^q g\big(x^v,\mu(x^v(q))\big)\right\},$$
$$l = 0,\dots,L-1. \tag{37}$$

Hence $\Big(\mathcal{F}^{l+1}J\Big)(x^v)$ can be seen as the expected value of $(l+1)$-stages cost of the policy under evaluation starting at state x^v. Now consider to replace the terminal cost function J with its approximate value, i.e. $J \approx \Big(\Phi r_s\Big)(x^v)$, then we have

$$\Big(\mathcal{F}^{l+1}(\Phi r_s)\Big)(x^v) = E\left\{\alpha^{l+1}\phi(x^v(l+1))'r_s + \sum_{q=0}^{l}\alpha^q g\big(x^v,\mu(x^v(q))\big)\right\},$$
$$l = 0,\dots,L-1. \tag{38}$$

Comparing (38) with the right hand side of (32) follows that cost sample $\lambda_{j,l}(r_s)$ produced by the simulation earlier can be used to estimate $\Big(\mathcal{F}^{l+1}(\Phi r_s)\Big)(x^v)$ for all x^v by Monte Carlo averaging. The final estimation formula is

$$\Big(\mathcal{F}^{L}(\Phi r_s)\Big)(x^v) \approx \lim_{Q\to\infty}\frac{1}{\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)} \cdot \sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)\lambda_{j,l}(r_s), \tag{39}$$

where for any event, we denote by $\delta(event)$ the indicator function of the event.

Suppose we denote by η_{x^v} the probability of being at the state x^v after exploring the state space with Q trajectories each having the length L, then by using the definition of the projection with respect to this probability distribution, we can write

$$r_{s+1} = \arg\min_{r\in\mathbb{R}^\gamma}\sum_{x^v\in X}\eta_{x^v}\left(\phi(x^v)'r - \Big(\mathcal{F}^L(\Phi r_s)\Big)(x^v)\right)^2. \tag{40}$$

From (39) and consistency of Monte Carlo simulation for policy evaluation (Bertsekas and Tsitsiklis, 1995), the simulation based solution of above relation is in the form of

$$r_{s+1} = \arg\min_{r\in\mathbb{R}^\gamma}\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\Big(\phi\big(x^v(l)\big)'r - \lambda_{j,l}(r_s)\Big)^2, \tag{41}$$

which is (33).

In order to solve the least-squares minimization problem for computing r_s, we note that by using (38) (which is the approximation of (37)) and replacing into (40) we have

$$r_{s+1} = \arg\min_{r\in\mathbb{R}^\gamma}\sum_{x^v\in X}\eta_{x^v} \times\left(\phi(x^v)'r - E\left\{\alpha^{l+1}\phi(x^v(l+1))'r_s + \sum_{q=0}^{l}\alpha^q g\big(x^v,\mu(x^v(q))\big)\right\}\right)^2, \tag{42}$$

and by taking gradient from above relation with respect to r, equivalently we have

$$r_{s+1} = \left(\sum_{x^v\in X}\eta_{x^v}\phi(x^v)\phi(x^v)'\right)^{-1}\left(\sum_{x^v\in X}\eta_{x^v}\phi(x^v)\Big(\mathcal{F}^L(\Phi r_s)\Big)(x^v)\right). \tag{43}$$

In the following we try to find an operative implementation formula for (43) by using simulations, therefore we replace r_{s+1} by its estimated one, computed via simulation, that, with some abuse of notation, we still denote with r_{s+1}.

Let $\tilde{\eta}_{x^v}$ be the empirical relative frequency of state x^v during the simulation, given by

$$\tilde{\eta}_x = \frac{1}{Q\times L}\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big). \tag{44}$$

Note that $\tilde{\eta}_{x^v}$ is the long-term occupancy probability of state x^v during the simulation process.

If we replace (44) into (43), then the simulation-based estimate (34) can be written as

$$
\begin{aligned}
r_{s+1} &= \left(\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\phi\big(x^{v,j}(l)\big)\phi\big(x^{v,j}(l)\big)'\right)^{-1}\cdot\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\phi\big(x^{v,j}(l)\big)\lambda_{j,l}(r_s)\\
&= \left(\sum_{x^v\in X}\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\phi(x^v)\phi(x^v)'\right)^{-1}\cdot\sum_{x^v\in X}\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)\lambda_{j,l}(r_s)\\
&= \left(\sum_{x^v\in X}\tilde{\eta}_{x^v}\phi(x^v)\phi(x^v)'\right)^{-1}\cdot\sum_{x^v\in X}\frac{1}{Q\times L}.\phi(x^v)\\
&\quad .\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)\lambda_{j,l}(r_s)\\
&= \left(\sum_{x^v\in X}\tilde{\eta}_{x^v}\phi(x^v)\phi(x^v)'\right)^{-1}\sum_{x^v\in X}\frac{\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)}{Q\times L}.\phi(x^v)\\
&\quad .\frac{1}{\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)}\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)\lambda_{j,l}(r_s).
\end{aligned}
\tag{45}
$$

If we define

$$
\Lambda_Q(x^v)=\frac{1}{\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)}.\sum_{j=1}^{Q}\sum_{l=0}^{L-1}\delta\big(x^{v,j}(l)=x^v\big)\lambda_{j,l}(r_s),\tag{46}
$$

then finally have

$$
r_{s+1}=\left(\sum_{x^v\in X}\tilde{\eta}_{x^v}\phi(x^v)\phi(x^v)'\right)^{-1}\sum_{x^v\in X}\tilde{\eta}_{x^v}\phi(x^v)\Lambda_Q(r_s).\tag{47}
$$

Since $\big(\mathcal{F}(\Phi r_s)\big)(x^v)=\lim_{Q\to\infty}\Lambda_Q(x^v)$ and $\eta_{x^v}=\lim_{Q\to\infty}\tilde{\eta}_{x^v}$ hence we see that the iteration (43) and the simulation-based implementation (47) asymptotically coincide. For a general discussion on the consistency of Monte Carlo simulation for policy evaluation step see section 5.2 in Bertsekas and Tsitsiklis (1995).

In the following we provide some remarks that will be discussed in the rest of the paper.

Remark 2. It is noticed that one can consider the length of each trajectory different from another. In the sense that if we consider Q trajectories then length of trajectory L_j may differ from L_{j+1}. In this case we have

$$
\tilde{\eta}_{x^v}=\frac{1}{L_1+L_2+\cdots+L_Q}\sum_{j=1}^{Q}\sum_{l=0}^{L_j-1}\delta\big(x^j(l)=x^v\big),\tag{48}
$$

accordingly we can modify the rest of formulas for the new framework.

Remark 3. If $Q=1$ then the solution of the least-squares minimization (33) will be single trajectory Monte Carlo simulation method and it is considered as an on policy LSTD approach (see Bertsekas (2017) for more details).

Remark 4. If $Q\gg 1$ and $L=1$ then the multi-trajectory simulations method is categorized as an off-policy LSTD or off-policy LSPE approach, where the projection is with respect to the weighted sup-norm corresponding to probability distribution of the initial state of each trajectory (Bertsekas, 2017) (see (31)).

Remark 5. If $Q\gg 1$ and $L\gg 1$ then the multi-trajectory simulations method is categorized as an on-policy LSTD approach since the number of visiting states within multiple trajectories are considerably greater than the number of initial states of the trajectories.

Remark 6. If the probability distribution of selecting initial state of the next trajectory i.e. $\epsilon(0)$, was corresponding to the steady state distribution of the policy under evaluation with the transition probability matrix $\mathcal{P}$, then the method is simplified to the single trajectory on-policy LSTD method.

Remark 7. The least-squares minimization of the form (33) can run into some practical issues: (i) collecting whole samples at once and computing the new parameter vector r_{s+1} from r_s is susceptible to simulation noise in optimistic policy iteration setting (see discussion in Thiery and Scherrer (2010)); (ii) for the MDPs with a large state space and a large action space including policy improvement step in policy iteration algorithm in the form of (33) is impractical since it requires the collection of many samples between policy updates (see section 6.3.5 in Bertsekas (2017)); (iii) in the case of large policy space and practical size MDPs it is difficult to determine the base policy to initialize the Monte Carlo simulation for collecting samples (Forootani et al., 2020b), and finally (iv) implementation of (33) requires summation of whole trajectory (one-shot) which makes it difficult for real time update of the parameter vector r_{s+1}.

In the next section, we will discuss the convergence property of the proposed generalized multi-trajectory LSTD approach in the extreme case when the condition of Remark 4 holds. Moreover a modified version of (33) will be presented which allows to integrate policy improvement step within the Monte Carlo simulations and address the susceptibility to simulation noise.

5. Discussion on the convergence of the generalized multi-trajectory LSTD approach

The probability distribution in (31) to select the initial state of each trajectory is important for the convergence of the proposed LSTD algorithm, especially when the length of each trajectory is short. In particular, the extreme case as mentioned in Remark 4 occurs when we set the length of each trajectory equal to 1, i.e. $L=1$. By doing so, the generalized multi-trajectory LSTD method is categorized as an off policy approach whereby sufficient condition is required to guarantee its convergence. Indeed if the length of each trajectory is long the recursive algorithm which computes estimated parameter vector r gradually forgets its initial state that started the Monte Carlo simulation. In the following, we consider the most critical case where the length of each trajectory is $L=1$ since the other cases are derived from this case, hence providing the sufficient condition for its convergence covers the rest.

Note that in this case $\mathcal{F}^L=\mathcal{F}$ and the initial probability distribution ϵ_{x^v} in (31) coincides with η_{x^v} defined in the previous section.

The policy that we consider within a trajectory is fixed, hence we suppress in our notation the dependency of the transition probability matrix $\mathcal{P}$ to control μ. Therefore we say $\mathcal{P}$ to represent the whole matrix.

Inspired by the work (Bertsekas and Yu, 2009), in this paper we formally assume that the initial state of each trajectory is chosen based on the probability distribution ϵ derived from the irreducible transition probability matrix $\tilde{\mathcal{P}}$ which is defined as follows (Bertsekas, 2017)

$$
\tilde{\mathcal{P}}=(I-\mathcal{B})\mathcal{P}+\mathcal{B}\mathcal{D},\tag{49}
$$

where I is the identity matrix, $\mathcal{B}$ is a diagonal matrix with diagonal components $\beta_i\in[0,1)$, and $\mathcal{D}$ is another transition probability matrix[3] with the elements $d_{x^vx^w}$. In this framework, at state x^v, the next state x^w is generated with probability $1-\beta_i$ according to transition probabilities $\mathcal{P}_{x^vx^w}$, and with probability β_i according to transition probabilities $d_{x^vx^w}$. It is worth highlighting that a computer program is required to generate state transitions based on $\tilde{\mathcal{P}}$. Moreover, the pair (x^v,x^w) with $d_{x^vx^w}$ needs not correspond to physically plausible transitions. With this choice the projection that we introduced in the LSTD method is based on the following relation in the vector form

$$
\Phi r=\bar{\Pi}\mathcal{F}(\Phi r),\tag{50}
$$

[3] In our case, $\tilde{\mathcal{P}}$ is irreducible if $\mathcal{P}$ is.

where

$$\bar{\Pi} = \Phi(\Phi'\bar{\Xi}\Phi)^{-1}\Phi'\bar{\Xi}, \tag{51}$$

and $\bar{\Xi} \in \mathbb{R}^{\Omega\times\Omega}$ is the diagonal matrix, having ϵ along its diagonal.

The following Lemma provides the sufficient condition to preserve the convergence of $\bar{\Pi}\mathcal{F}$.

Lemma 3 (Forootani et al., 2022b). *Assume that $\tilde{\mathcal{P}}$ is an irreducible state transition probability matrix and that ϵ is its unique steady-state probability vector with positive components. Then, $\mathcal{F}$ and $\bar{\Pi}\mathcal{F}$ are contraction mappings with respect to $\|\cdot\|_\epsilon$, and the associated modulus of contraction is at most equal to $\bar{\alpha}$, where*

$$\bar{\alpha} = \alpha/\sqrt{1-\beta}, \quad \textit{with} \quad \beta = \max_{i=1,\dots,n} \beta_i. \tag{52}$$

Note that the relation $\beta < 1-\alpha^2$ has to be fulfilled for $\bar{\Pi}\mathcal{F}$ to be a contraction mapping w.r.t. $0 < \bar{\alpha} < 1$.

Since $\bar{\Pi}\mathcal{F}$ is a contraction mapping w.r.t. ϵ and by exploiting the assumption that Φ has full rank γ, it is easy to prove the following lemma, which extends the Proposition 6.3.1 of Bertsekas (2017).

Lemma 4. *Let the assumptions of Lemma 3 hold, and let the matrix Φ be of full column rank γ. Then, we have*

$$\left\|J_{\mathcal{F}} - \Phi r\right\|_\epsilon \le 1/(\sqrt{1-\bar{\alpha}^2})\left\|J_{\mathcal{F}} - \bar{\Pi}J_{\mathcal{F}}\right\|_\epsilon, \tag{53}$$

where r is the unique solution of the projected Bellman equation (50) and $J_{\mathcal{F}}$ is the fixed point of the mapping $\mathcal{F}$.

Proof. The fact that Φ has full column rank γ guarantees that the unique fixed point of the operator $\bar{\Pi}\mathcal{F}$ can be represented by a unique parameter vector r. Moreover, we have

$$\begin{aligned}\left\|J_{\mathcal{F}} - \Phi r\right\|_\epsilon^2 &= \left\|J_{\mathcal{F}} - \bar{\Pi}J_{\mathcal{F}}\right\|_\epsilon^2 + \left\|\bar{\Pi}J_{\mathcal{F}} - \Phi r\right\|_\epsilon^2 \\ &= \left\|J_{\mathcal{F}} - \bar{\Pi}J_{\mathcal{F}}\right\|_\epsilon^2 + \left\|\bar{\Pi}\mathcal{F}J_{\mathcal{F}} - \bar{\Pi}\mathcal{F}(\Phi r)\right\|_\epsilon^2 \\ &\le \left\|J_{\mathcal{F}} - \bar{\Pi}J_{\mathcal{F}}\right\|_\epsilon^2 + \bar{\alpha}^2\left\|J_{\mathcal{F}} - \Phi r\right\|_\epsilon^2,\end{aligned} \tag{54}$$

where the first equality uses the Pythagorean theorem,[4] the second equality holds because $J_{\mathcal{F}}$ is the fixed point of $\mathcal{F}$ and Φr is the fixed point of $\bar{\Pi}\mathcal{F}$, and the last inequality uses the contraction property of $\bar{\Pi}\mathcal{F}$. □

Unfortunately the computation of $\bar{\Pi}$ requires the matrix inversion and multiplication with large size matrices, which is impractical for large scale MDP problems. To tackle this practical issue, in the following, we discuss the construction of simulation based approximations to the projected $\Phi r = \bar{\Pi}\mathcal{F}(\Phi r)$ and related convergence analysis. In this regard, let us consider the iterative estimation of the parameter vector r given by $\Phi r_{s+1} = \bar{\Pi}\mathcal{F}(\Phi r_s)$. By expressing the projection as a least-squares minimization, we see that r_{s+1} is given by

$$r_{s+1} = \arg\min_{r\in\mathbb{R}^\gamma} \|\Phi r - \mathcal{F}(\Phi r_s)\|_\epsilon^2, \tag{55}$$

or equivalently

$$r_{s+1} = \arg\min_{r\in\mathbb{R}^\gamma} \sum_{x^v\in X} \epsilon_{x^v}\left(\phi(x^v)'r - \left\{g(x^v,\mu(x^v)) + \alpha\sum_{x^w\in X}\mathcal{P}_{x^v x^w}\phi(x^w)'r_s\right\}\right)^2, \tag{56}$$

which is identical to (40) with $L = 1$. By setting the gradient of the minimized expression above to 0 we have

$$r_{s+1} = \left(\sum_{x^v\in X}\epsilon_{x^v}\phi(x^v)\phi(x^w)'\right)^{-1}$$

$$\times\left(\sum_{x^v\in X}\epsilon_{x^v}\phi(x^v)\left\{g(x^v,\mu(x^v)) + \alpha\sum_{x^w\in X}\mathcal{P}_{x^v x^w}\phi(x^w)'r_s\right\}\right). \tag{57}$$

In spite of considering a model-based method in this paper, the approach can be extended to model-free cases by assuming that a computer simulation can be employed to generate samples according to the probability distribution ϵ_{x^v} and the transition probability $\mathcal{P}_{x^v x^w}$ (Bertsekas, 2012, 2019).

In particular, such a simulator can generate the needed samples along the Monte Carlo trajectories, that is to say, the sequence of states $\left\{x^v(0), x^v(1), \dots\right\}$ and the sequence of state transitions $\left\{\left(x^v(0), x^w(0)\right), \left(x^v(1), x^w(1)\right), \dots\right\}$. In other words, we estimate the above recursive equation by using its sampled version

$$r_{s+1} = \left(\sum_{t=0}^{s}\phi\left(x^v(t)\right)\phi\left(x^v(t)\right)'\right)^{-1}$$

$$\times\left(\sum_{t=0}^{s}\phi\left(x^v(t)\right)\left(g\left(x^v(t),\mu(x^v(t))\right) + \alpha\phi\left(x^w(t)\right)'r_t\right)\right), \tag{58}$$

which is the same as (45) for $L = 1$. The probabilistic mechanism in (58) is subject to the following conditions:

1. The sequence $\left\{x^v(0), x^v(1), \dots\right\}$ is generated based on the distribution ϵ associated with $\tilde{\mathcal{P}}$ in (49), which defines projection norm $\|\cdot\|_\epsilon$, in the sense that with probability 1,

$$\lim_{s\to\infty}\sum_{t=0}^{s}\delta\left(x^v(t) = x^v\right)/(s+1) = \epsilon_{x^v}, \quad \forall x^v \in X, \tag{59}$$

   where $\delta(\cdot)$ denotes the indicator function.
2. The sequence $\left\{\left(x^v(0), x^w(0)\right), \left(x^v(1), x^w(1)\right), \dots\right\}$ is generated according to stochastic matrix $\mathcal{P}$ with state transition probabilities $\mathcal{P}_{x^v x^w}$, that is

$$\lim_{s\to\infty}\frac{\sum_{t=0}^{s}\delta\left(x^v(t) = x^v, x^w(t) = x^w\right)}{\sum_{t=0}^{s}\delta\left(x^v(t) = x^v\right)} = \mathcal{P}_{x^v x^w}, \ \forall x^v, x^w \in X. \tag{60}$$

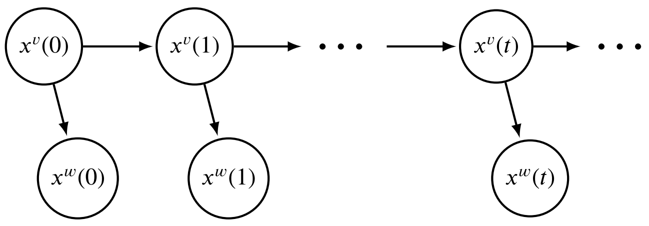


Fig. 6. The basic simulation scheme.

Fig. 6 shows the basic simulation methodology which consists of generating states $\{x^v(0), x^v(1), \dots\}$ according to the distribution ϵ, and a sequence of transitions $\left\{\left(x^v(0), x^w(0)\right), \left(x^v(1), x^w(1)\right)\right\}$.

For the convergence analysis of the proposed algorithmic implementation, it is useful to express the least-squares minimization (55) in the following form

$$r_{s+1} = \arg\min_{r\in\mathbb{R}^\gamma}\|\Phi r - (g + \alpha\mathcal{P}\Phi r_s)\|_\epsilon^2. \tag{61}$$

By setting to 0 the gradient with respect to r of the above quadratic expression, we obtain the orthogonality condition

$$\Phi'\bar{\Xi}\left(\Phi r_{s+1} - (g - \alpha\mathcal{P}\Phi r_s)\right) = 0, \tag{62}$$

which yields

$$r_{s+1} = r_s - G(Cr_s - y), \tag{63}$$

where

$$C = \Phi'\bar{\Xi}(I - \alpha\mathcal{P})\Phi, \quad y = \Phi'\bar{\Xi}g, \quad G = \left(\Phi'\bar{\Xi}\Phi\right)^{-1}. \tag{64}$$

[4] It is noticed that from the Pythagorean theorem we have $\left\|J_{\mathcal{F}} - \Phi r\right\|_\epsilon^2 = \left\|\bar{\Pi}(J_{\mathcal{F}} - \Phi r)\right\|_\epsilon^2 + \left\|(I - \bar{\Pi})(J_{\mathcal{F}} - \Phi r)\right\|_\epsilon^2$..

We approximate the matrix C and vector y by

$$C_s = 1/(s+1)\sum_{t=0}^{s} \phi\big(x^v(t)\big)\Big(\phi\big(x^v(t)\big) - \alpha\phi\big(x^w(t)\big)\Big)', \tag{65}$$

and

$$y_s = 1/(s+1)\sum_{t=0}^{s} \phi\big(x^v(t)\big)g\big(x^v(t), \mu(x^v(t))\big), \tag{66}$$

where we estimate the probabilities ϵ_{x^v} and $\mathcal{P}_{x^v x^w}$ by their empirical frequencies in the Monte Carlo simulations as follows

$$C_s = \sum_{x^v\in X} \frac{\sum_{t=0}^{s}\delta\big(x^v(t)=x^v\big)}{s+1}\phi\big(x^v\big) \times \left(\phi\big(x^v\big) - \alpha\sum_{x^w\in X}\frac{\sum_{t=0}^{s}\delta\big(x^v(t)=x^v, x^w(t)=x^w\big)}{\sum_{t=0}^{s}\delta\big(x^v(t)=x^v\big)}\phi\big(x^w\big)\right)', \tag{67}$$

and finally

$$C_s = \sum_{x^v\in X} \hat{\epsilon}_{t,x^v}\phi(x^v)\left(\phi(x^v) - \alpha\sum_{x^w\in X}\hat{\mathcal{P}}_{t,x^v x^w}\phi(x^w)\right). \tag{68}$$

Similarly we can write

$$y_s = \sum_{x^v\in X} \hat{\epsilon}_{t,x^v}\phi(x^v)g(x^v, \mu(x^v)). \tag{69}$$

Since the empirical frequencies $\hat{\epsilon}_{t,x^v}$ and $\hat{\mathcal{P}}_{t,x^v x^w}$ asymptotically converge to the probabilities ϵ_{x^v} and $\mathcal{P}_{x^v x^w}$ respectively, we have with probability 1, $C_s \to C$ and $y_s \to y$ (Bertsekas, 2011b).

Considering the Assumptions 1 and 2, the following results guarantee the convergence of the algorithm described above.

Lemma 5 (*Forootani et al., 2022a*)**.** *Matrix C is positive definite.*

Lemma 6 (*Forootani et al., 2022a*)**.** *$G^{-1} = (\Phi'\bar{\Xi}\Phi)^{-1}$ exists and it is symmetric positive definite.*

Theorem 7 (*Forootani et al., 2022a*)**.** *If matrix C is positive definite (but not symmetric in general) then the following holds: (i) Eigenvalues of C have positive real parts, (ii) $\det(G^{-1}C) < 1$, (iii) $G^{-1}C$ is positive definite, (iv) $I - G^{-1}C$ has the eigenvalues strictly within unit circle.*

Corollary 3. *Since $I - G^{-1}C$ has the eigenvalues strictly within unit circle as proven in Theorem 7, then the recursive iteration (63) or equivalently (58) is convergent.*

Note that the multi-trajectory LSTD approach proposed above is scale free. Indeed let us define the set

$$\mathcal{A} = \{\Phi r | r \in \mathbb{R}^\gamma\}. \tag{70}$$

For any invertible $m\times m$ matrix Θ such that $\Phi = \Psi\Theta$ then we have

$$\mathcal{A} = \{\Psi f | f \in \mathbb{R}^\gamma\}, \tag{71}$$

hence $\mathcal{A}$ can be represented as the span of a different set of basis functions, and any vector $\Phi r \in \mathcal{A}$ can be written as Θf, where the weight vector f is equal to Θr. In addition, each row $\phi(x)'$, i.e. the feature vector of state x in the representation based on Φ, is equal to $\psi(x)'\Theta$, i.e. the linearly transformed feature vector of x in the representation based on Ψ. Consider to denote by $C_{s,\Phi}$ and $y_{s,\Phi}$ the matrix and the vector corresponding to feature matrix Φ (see (65) and (66)), and denote by $C_{s,\Psi}$ and $y_{s,\Psi}$ the matrix and the vector corresponding to feature matrix Ψ. It is straightforward to show that $C_{s,\Phi} = \Theta' C_{s,\Psi}\Theta$ and $y_{s,\Phi} = \Theta' y_{s,\Psi}$. The following lemma proves that the LSTD method is scale free.

Lemma 8. *If $\Phi = \Psi\Theta$, then we have $\Phi r_s = \Psi f_s$ for all s.*

Proof. It is easy to verify that

$$\Phi r_s = \Phi\big(C_{s,\Phi}\big)^{-1}y_{s,\Phi} = \Psi\Theta\big(\Theta' C_{s,\Psi}\Theta\big)^{-1}\Theta' y_{s,\Psi} = \Psi\big(C_{s,\Psi}\big)^{-1}y_{s,\Psi} = \Psi f_s. \tag{72}$$

which proves that LSTD is scale free for all s. □

6. Solution to near singularity and policy improvement issues

In this section, we will discuss near singularity issues of matrix C, and consider the policy improvement step within the Monte Carlo simulation trajectories. Moreover, we provide more details about feature matrix selection.

6.1. Solution to near singularity of matrix C

In Lemma 5, we stated that the matrix C is invertible and positive definite. However, this property may not hold for C_s until a sufficient number of samples in the Monte Carlo simulation are acquired for its calculation. Near-singularity of C_s can be due either to the columns of Φ being nearly linearly dependent or to the matrix $\bar{\Xi}(I - \alpha\mathcal{P})$ being nearly singular (see (64)). To resolve this issue, a regularization term is introduced. More specifically, in each iteration along the Monte Carlo trajectory, we compute r_s by solving the following least-squares problem

$$\min_r\Big\{\big(y_s - C_s r\big)'\Sigma^{-1}\big(y_s - C_s r\big) + \sigma \parallel r - r_s \parallel^2\Big\}. \tag{73}$$

By setting the gradient of above function to 0, we have

$$r_{s+1} = \big(C_s'\Sigma^{-1}C_s' + \sigma I\big)^{-1}\big(C_s'\Sigma^{-1}y_s + \sigma r_s\big), \qquad s = 1,\dots,Q, \tag{74}$$

where the quadratic term $\sigma\|r - r_s\|^2$ is known as a regularization term (here, $\|\cdot\|$ denotes the L_2-norm), and has the effect of biasing the estimate r_{s+1} towards the previous parameter vector estimation r_s. We consider the heuristic guess $\bar{r}$ for the parameter vector r_0. It is based on some intuition about the problem at hand. Moreover, the matrix Σ and the coefficient σ are respectively positive definite and positive (Hoffman et al., 2011). To see more discussion on the selection of matrix Σ, we refer the reader to Bertsekas (2011b), and for an empirical study, to Forootani et al. (2022a, 2020b).

The convergence of the iteration (74) follows from $C_s \to C$, $y_s \to y$ (see Bertsekas (2011b)), and the following Lemma, the proof of which can be found in Forootani et al. (2022a).

Lemma 9 (*Forootani et al., 2022a*)**.** *The recursive iteration (74) is convergent.*

6.2. Integrating the policy improvement

We can modify (57) in order to integrate the policy improvement step within Monte Carlo simulations by using the results in Forootani et al. (2022a, 2020b) as follows:

$$r_{s+1}7 = \arg\min_{r\in\mathbb{R}^{\gamma}} \sum_{x^v\in X} \epsilon_{x^v} \times \left(\phi(x^v)'r - \min_{\mu} \left\{ g\left(x^v, \mu(x^v)\right) - \alpha \sum_{x^w\in X} \mathcal{P}_{x^v x^w}(\mu(x^v))\phi\left(x^w\right)' r_s \right\} \right)^2, \tag{75}$$

by setting to 0 the gradient of above equation

$$r_{s+1} = \left(\sum_{x^v\in X} \epsilon_{x^v} \phi(x^v)\phi(x^v)' \right)^{-1} \times \left(\sum_{x^v\in X} \epsilon_{x^v} \phi(x^v) \min_{\mu} \left\{ g\left(x^v, \mu(x^v)\right) + \alpha \sum_{x^w\in X} \mathcal{P}_{x^v x^w}(u)\left(\phi\left(x^w\right)' r_s\right) \right\} \right), \tag{76}$$

this recursive iteration can be approximately written as

$$r_{s+1} = \left(\sum_{t=0}^{s} \phi(x^v(t))\phi(x^v(t))' \right)^{-1} \times \left(\sum_{t=0}^{s} \phi(x^v(t)) \left\{ \left(g(x^v, \mu(x^v(t)))\right) + \alpha\left(\phi\left(x_{\mu}^{w}(t)\right)' r_s\right) \right\} \right), \tag{77}$$

where $x_{\mu}^{w}(t)$ is selected based on the following control law

$$\mu^*(x^v(s)) := \arg\min_{\mu} \left(g\left(x^v(s), \mu(x^v(s))\right) + \alpha\phi\left(x_{\mu}^{w}(s)\right)' r_s \right), \quad \forall\ s, \tag{78}$$

in the other words, at the start of the iteration s, we have the current parameter vector r_s, we are at the state $x^v(s)$, and we have chosen a control $\mu^*(x^v(s-1))$.

6.3. Discussion on feature selection

In many other ADP algorithms based on linear cost-to-go approximation architectures (such as Least-Squares Policy Iteration (Thiery and Scherrer, 2010), Least-Squares Policy Evaluation (Bertsekas, 2017), LSTD (Sutton et al., 2016), etc.), the feature vectors enter into the algorithm as outer products of the form $\phi(x^v)\phi(x^w)'$ (see Eq. (58)). This approach works well when the number of features is small, but if it is desirable to use a large number of features, difficulties are encountered since computing the outer product becomes computationally expensive.

One possible solution is to exploit the graph structure of the state space of the MDPs to select the feature matrix. In particular, e.g. for a fixed policy, the state transition dynamics of the MDP are described by a Markov chain, where each state x^v is represented as a node in the graph and is connected to states that are reachable from x^v in one step with positive probability. In this regard one can define a feature matrix based on the graph which measures the number of steps necessary to move from one state to another. To clarify, consider the following feature matrix

$$\vartheta(x^v) = \phi(x^v) - \alpha \sum_{x^w\in X} \mathcal{P}_{x^v x^w}\phi(x^w), \tag{79}$$

where $\vartheta(x^v)$ represents a new feature mapping that accounts for the structure of the MDP dynamics. More specifically, it demonstrates a combination of the features at state x^v and all states x^w that can be reached in one step from x^v. The following Lemma states an important property of the new feature mapping and proves that the set of vectors $\{\vartheta(x^v)|x^v \in X\}$ are linearly independent.

Lemma 10. *Assume the feature matrix Φ is full column rank. Then the vectors $\{\vartheta(x^v)|x^v \in X\}$, where $\vartheta(x^v) = \phi(x^v) - \alpha\sum_{x^w\in X}\mathcal{P}_{x^v x^w}\phi(x^w)$, are also linearly independent.*

Proof. Consider the real vector space Υ spanned by the vectors $\{\phi(x^v)|x^v \in X\}$. It is evident that $\vartheta(x^v)$ is a linear combination of vectors in Υ, so a linear operator B that maps $\phi(x^v)$ to $\vartheta(x^v)$ can be defined

$$\vartheta(x^v) = B\phi(x^v). \tag{80}$$

Since

$$\vartheta(x^v) = \phi(x^v) - \alpha \sum_{x^w\in X} \mathcal{P}_{x^v x^w}\phi(x^w), \tag{81}$$

the matrix of B is $(I - \alpha\mathcal{P})$, where I is the identity matrix and $\mathcal{P}$ is the probability transition matrix for the fixed policy. Since $\mathcal{P}$ is a stochastic matrix, its largest eigenvalue is 1 and all other eigenvalues have absolute value less than 1; thus all the eigenvalues of $\alpha\mathcal{P}$ have absolute value less than or equal to $\alpha < 1$. Since all eigenvalues of I are equal to 1, $(I - \alpha\mathcal{P})$ is full rank and $dim\left(ker(B)\right) = 0$. Hence,

$$dim\left(\{\vartheta(x^v)|x^v \in X\}\right) = dim\left(\{\phi(x^v)|x^v \in X\}\right), \tag{82}$$

so the vectors $\{\vartheta(x^v)|x^v \in X\}$ are linearly independent. $\square$

By exploiting the graph structure of an MDP as the future work, we plan to develop a new family of algorithms similar in spirit to Bellman residual methods (Antos et al., 2008). Note that Bellman residual approaches attempt to minimize the error incurred in solving Bellman's equation at a set of sample states. However, in our future work by exploiting kernel-based regression techniques with non-degenerate kernel functions as the underlying cost function ADP architecture, our algorithms are able to construct cost function for which the Bellman residuals are explicitly forced to zero at the sample stats. The following remark summarizes our main goal for employing kernel-based regression in the future work.

Remark 8. By using kernel-based regression techniques, our algorithms will compute only inner products of the feature vectors, which can be computed efficiently using kernels even when the effective number of features is very large or even infinite. Thus, our future approach can allow the use of very high-dimensional feature vectors in a computationally tractable way.

7. Practical algorithm for generalized multi-trajectory LSTD

In the previous sections, we introduced multi-trajectories and its various forms in the LSTD-ADP framework. We also discussed the solution to near singularity of matrix C_s at the beginning of the Monte Carlo simulation. Finally, we considered policy improvement within Monte Carlo simulation. In this section, we integrate all the previous steps in one framework to provide a general simulation algorithm as follows for the generalized multi-trajectory LSTD approach.

1. Choose the number of trajectories Q and their length L, set $j = 1$ and $l = 0$,
2. Initialize the vector $r_0 = \bar{r}$, the initial state $x^{v,1}(0)$, the matrix $C_0 = 0$, the vector $y_0 = 0$, and index $s = 0$.
3. While $j \le Q, j \in \{1, \ldots, Q\}$:
    - While $l \le L$, $l \in \{0, 1, \ldots, L\}$:
    - Set $s \leftarrow s + 1$
    - For each $u \in U(x^{v,j}(l))$ do:
        - From the current state $x^{v,j}(l)$, generate a candidate next state $x_u^{v,j}(l+1)$ by Monte Carlo simulation and by applying the admissible control u based on the transition probability matrix $\mathcal{P}(u)$
        - Compute the corresponding feature vector $\phi\left(x_u^j(l)\right)$ and the reward of the current state $g\left(x^{v,j}(l), (u)\right)$

– Calculate the matrix $C_s(u)$

$$C_s(u) = (1-\frac{1}{s+1})C_{s-1}+\frac{1}{s+1}\phi\big(x^{v,j}(l)\big)\Big(\phi\big(x^{v,j}(l)\big)-\alpha\phi\big(x_u^{v,j}(l+1)\big)\Big)' \tag{83}$$

– Calculate the vector y_s

$$y_s(u) = (1-\frac{1}{s+1})y_{s-1}+\frac{1}{s+1}\phi\big(x^{v,j}(l)\big)g\big(x^{v,j}(l),u\big) \tag{84}$$

– Compute the candidate parameter $r_s(u)$ as follows

$$r_s(u) = \big(C_s'(u)\Sigma^{-1}C_s(u)+\sigma I\big)^{-1}.\big(C_s'(u)\Sigma^{-1}y_s(u)+\sigma r_{s-1}\big) \tag{85}$$

- Choose the pair $(r_s(\bar{u}),\ x_{\bar{u}}^{v,j}(l+1))$ and the related greedy control $\bar{u}$ by

$$\underset{r_s(u),x_u^{v,j}(l+1)}{\arg}\quad\min\Bigg(g\big(x^{v,j}(l),(u)\big)+\alpha\phi\ \big(\ x_u^{v,j}(l+1)\ \big)\ 'r_s(u)\Bigg) \tag{86}$$

- Set $C_s \leftarrow C_s(\bar{u}),\ x^{v,j}(l+1) \leftarrow x_{\bar{u}}^{v,j}(l+1),\ r_s \leftarrow r_s(\bar{u}),\ l \leftarrow l+1$
- Set $j \leftarrow j+1$,
- Generate the new initial state $x^{v,j}(0)$ from $x^{v,j}(l-1)$ based on $\tilde{\mathcal{P}}$ in (49).

We can note that the sample computational complexity in the proposed multi-trajectory LSTD algorithm mainly depends on matrix inversion operations. In particular, we need to perform a $\gamma\times\gamma$ matrix inversion (γ is the number of features) in order to compute the candidate parameter vector $r_s(u)$ for each admissible action u in the current state $x^{v,j}(l)$, see (85). Such matrix inversion operation has a computational cost of $O(\gamma^3)$, which can be reduced to $O(\gamma^2)$ by using the Sherman–Morrison formula (Dann et al., 2014). The algorithmic complexity along with its convergence rate determines the computational resources needed for evaluating the estimated parameter vector $\tilde{r}^*$ from the parameter vector values r_s, the latter recursively calculated through the proposed algorithm iterations.

It is known that, for linear function approximation, the convergence of LSTD-based methods is guaranteed if the states are sampled according to the underlying system dynamics (Bertsekas, 2011b). As discussed in Section 5 and numerically verified in the next one, the asymptotic convergence rate of the LSTD based methods is not influenced by the choice of LSTD input parameters (such as Σ and σ, provided that the matrix Σ is symmetric positive definite Bertsekas, 2011b; Yu and Bertsekas, 2009; Nedić and Bertsekas, 2003), but it is influenced by the state sampling mechanism itself (Bertsekas, 2011b). This means that the convergence rate of the classical single-trajectory LSTD algorithm (where the cost of a specific policy is evaluated, see Bertsekas (2012)) is expected to be different from the one of the proposed multi-trajectory LSTD algorithm (where the policy under evaluation is renewed at each iteration, see (86)): the two state sampling mechanisms, even though based on the underlying system dynamics, are different. Thus, the convergence rate of the multi-trajectory LSTD algorithm is expected to be slower than the one of the single trajectory LSTD. On the other hand, besides the feature matrix, two critical parameters are the number of trajectories Q and their length L. As also shown in the next Section, they can influence both the quality of the value function approximation and the convergence rate of the algorithm.

8. Numerical simulation

In this section, we provide some examples to illustrate the effectiveness of the proposed approach for machine replacement problems. A MATLAB-based application has been developed to construct the system state space, to define the state transition probability matrix concerning the machine replacement, to solve the optimization problem via both the exact DP and the proposed generalized multi-trajectory LSTD algorithm (in short, ADP algorithm in this Section), as well as to perform Monte Carlo simulations for evaluating the computed policies. To do this, such application takes as input parameters and all the data necessary to set up the MDP formulation and the related stochastic DP framework (e.g., the operating cost per stage, the replacement cost, and the finite time horizon).

For all the simulation examples, we considered the following operating cost per stage

$$c_i(x_i) = 0.1 + 0.9e^{0.3x_i}, \tag{87}$$

while the transition probabilities are defined as

$$\mathcal{P}_{x_i^{v_i}x_i^{w_i}}\big(u_i\big) = \frac{1}{N_i(x_i^{v_i},u_i)}. \tag{88}$$

It is quite simple to verify that (88) satisfies the condition $\big(\mathcal{P}_{x_i^{v_i+1}x_i^{w_i}}(0)-\mathcal{P}_{x_i^{v_i}x_i^{w_i}}(0)\big) \geq 0$, which implies that there is a higher probability of ending at a worse state $x_i^{w_i}$ if we start at the less operational state $x_i^{v_i+1}$.

In this paper, for each state $x^v = (x_1^{v_1}, x_2^{v_2}, \dots, x_m^{v_m})$, we define the following components of the feature vector (note that, in our case, $\gamma = m$)

$$\phi_i(x_i^{v_i}) = \frac{1}{1+e^{-x_i^{v_i}+3}}, \quad i = 1,2,\dots,m, \tag{89}$$

where $x_i^{v_i}$ is the state of each machine.

This paper does not address the construction of the feature functions, which is indeed an important research area, see Bertsekas and Yu (2009), Busoniu et al. (2011) and Barker and Ras (2019). A limited number of well-crafted feature functions can capture the dominant non-linearities of cost functions for complex systems, and thus their linear combination can work well as an approximation architecture, see Bertsekas (2017) and Busoniu et al. (2011).

The cost function approximation $\tilde{J}^* = \Phi\tilde{r}^*$ of the machine replacement problem is used over the finite time horizon T. More specifically, we can calculate the (stationary) approximate optimal policy $\tilde{\mu}^*(\cdot)$ by replacing $\tilde{J}^*$ as the terminal cost function in the Bellman optimality operator

$$\tilde{\mu}^*(x^v) = \arg\min_{u\in U}\Bigg[g(x^v,u)+\alpha\sum_{x^w\in X}\mathcal{P}_{x^vx^w}\big(u\big)\phi'(x^w)\tilde{r}^*\Bigg]. \tag{90}$$

It is also worth highlighting that all the policies computed in this section (i.e., both via the exact DP and the proposed LSTD algorithm) as well as the heuristics/non-optimal policies have been tested and compared each other via Monte Carlo simulations with initial conditions randomly generated for each experiment. All the algorithms used for policy computations and their subsequent testing have been run by using a laptop equipped with a 2 GHz Quad-Core Intel Core i5 processor and 16 GB RAM.

Example 8.1. Consider the machine replacement problem M with $m = 3$ machines and $\Omega = 6$. The total number of states is 6^3. The operating cost per stage of each machine is shown in Fig. 7, which satisfies the non-decreasing property of $c(x_i)$, see (10). We performed the exact DP algorithm and saved its results. Then, by considering a set of 100 experiments with $T = 30$, we compared the expected cumulative costs resulted from the exact DP algorithm with the following four non-optimal policies/heuristics:

- Non-optimal policy (i): choosing a machine based on a round robin policy, i.e., sequentially replacing machines regardless of their operating status.
- Non-optimal policy (ii): choosing one or two machines randomly to replace among them.
- Non-optimal policy (iii): sequentially replacing machines choosing the one with the worst operating status.

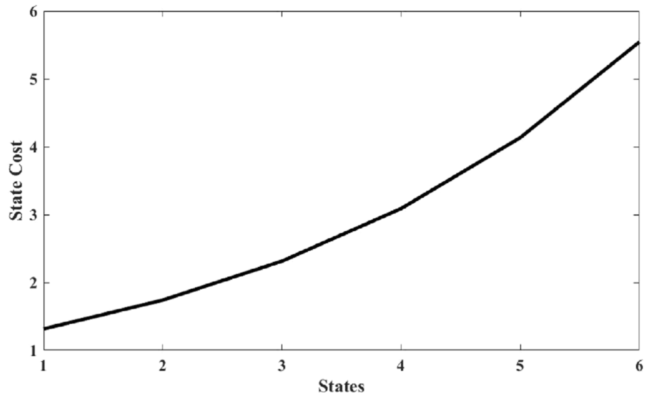


Fig. 7. Non-decreasing behavior of the cost-per stage (87).

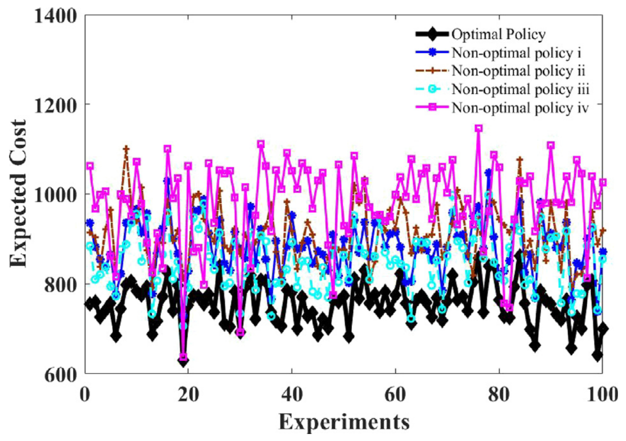


Fig. 8. Comparison of the exact DP with the non-optimal policies in Example 8.1.

- Non-optimal policy (iv): a machine is replaced only if all machines in worse states are replaced (worse cluster replacement rule, see Childress and Durango-Cohen (2005)).

The results of the comparison between the exact DP and such non-optimal polices is shown in Fig. 8. It is evident that the exact DP policy outperforms the other non-optimal policies and achieves minimum expected total cost. Fig. 9 shows the number of replaced machines when applying the exact DP policy together with the ones corresponding to the above non-optimal policies/heuristics. It can be noticed that the exact DP policy tends to replace fewer machines than all the non-optimal policies, with the exception of policies (i) and (iii) which replace a fixed number of machines in each experiment. Fig. 10 displays the total costs over 100 experiments of the exact DP policies computed with two different replacement costs $R_1 = 7$ and $R_2 = 9$. This way, one can assess the impact of the replacement costs on the resulting total costs, which are higher in case of machine replacement problems with higher replacement costs.

Finally, Figs. 11 and 12 compare the results of our proposed LSTD-based approach with the exact DP by computing, respectively, the total cost and the total number of replacements over 100 experiments. As expected, the exact DP provides better performance results, also achieved with fewer replacement operations. The computational time required to determine the exact DP policy was 989 s, while it took 413 s to compute the parameter vector $\tilde{r}^*$ in case of the proposed LSTD-based algorithm (both Q and L were set to 10).

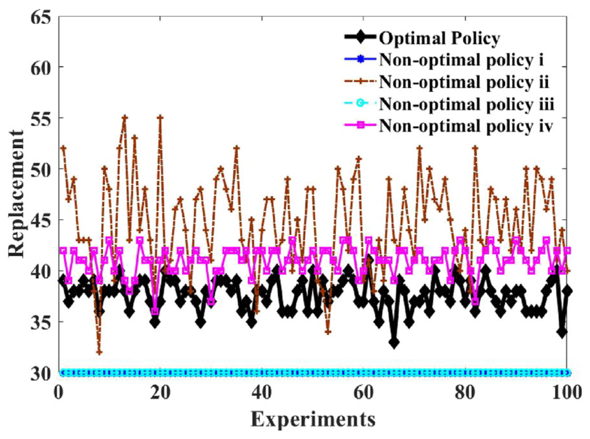


Fig. 9. Number of replacements with the exact DP approach and the non-optimal policies in Example 8.1.

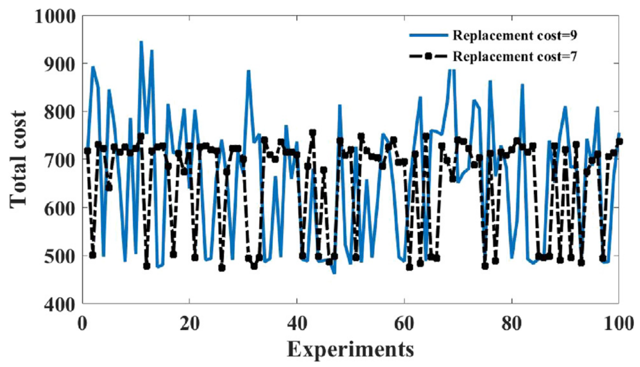


Fig. 10. Comparison of the exact DP with two different replacement costs in Example 8.1.

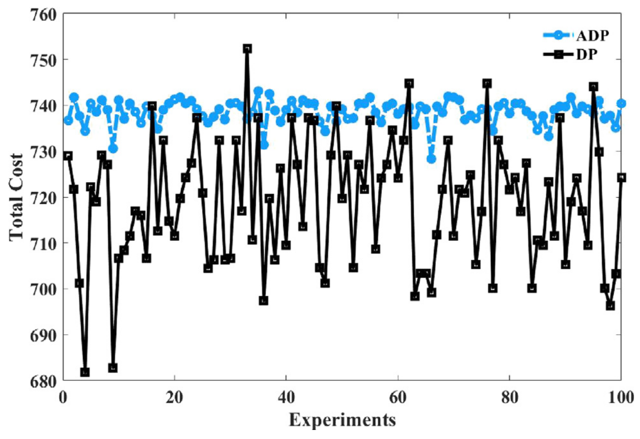


Fig. 11. Comparison of the exact DP with ADP in Example 8.1.

Example 8.2. Consider the machine replacement with $m = 6$ machines and $\Omega = 10$ operating states. Being $|X| = 10^6$, the problem suffers from the curse of dimensionality in the state space. Thus, we applied our proposed ADP method for the following different scenarios:

1. We executed the proposed algorithm 10 times, with $Q = 1$ and $L = 10\,000$ in each run. Moreover, we applied the same values for the parameters Σ, ζ, and $\bar{r}$. and performed the Monte Carlo simulations for a set of experiments to compute the estimated parameter vector $\tilde{r}^*$ from the parameter vector values r_s recursively calculated over the ADP algorithm iterations. In this

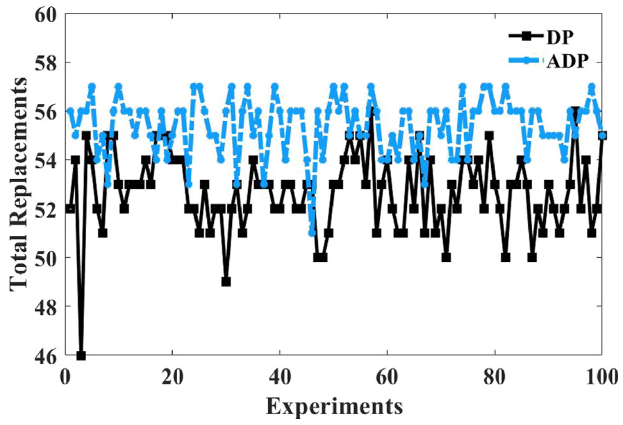


Fig. 12. Number of replacements with the exact DP and the ADP in Example 8.1.

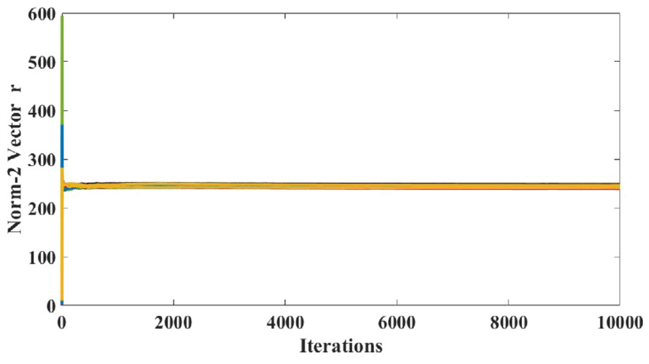


Fig. 13. Computed Norm-2 of the vector r_s as in (1) in Example 8.2.

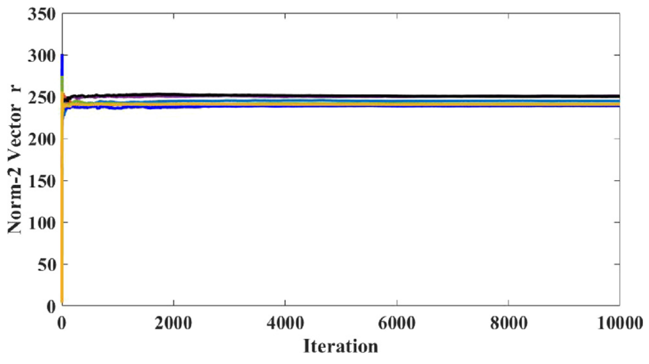


Fig. 14. Computed Norm-2 of the vector r_s as in (2) in Example 8.2.

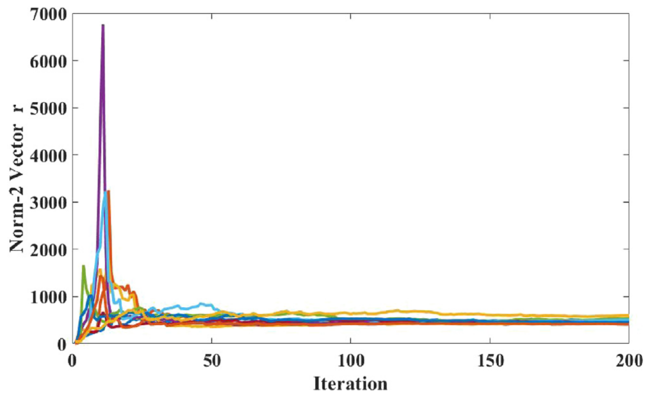


Fig. 15. Computed Norm-2 of the vector r_s as in (3) in Example 8.2.

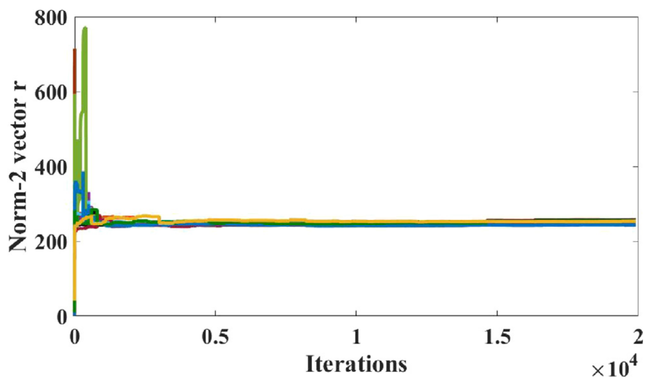


Fig. 16. Computed Norm-2 of the vector r_s as in (4) in Example 8.2.

regard, norm-2 of the parameter vectors r_s is plotted in Fig. 13. As we can see, all the curves converge to very close Norm-2 values. By taking the average of these results, we obtained $\tilde{r}^* = [131.6388,\ 127.3602,\ 131.8741,\ 46.7978,\ 46.8144,\ 48.1488]'$. The computational time required to determine the parameter vector $\tilde{r}^*$ for this scenario was 112 478 s.

2. In the second scenario, we used the same setup as (1), with the exception of running the Monte Carlo simulations with different values of Σ, ζ and $\bar{r}$. The outcome of this computation for a set of 10 algorithm executions is shown Fig. 14. The averaged value of the parameter vector is $\tilde{r}^* = [127.4561,\ 124.7104,\ 139.655,\ 46.9587,\ 46.9683,\ 49.8320]'$. As expected, the resulting parameter vector is quite close to the one computed in the previous scenario. The computational time required to determine the parameter vector $\tilde{r}^*$ was almost the same as the previous scenario since a similar set-up was used.
3. In the third scenario, we considered $Q = 200$ and $L = 1$. We performed 10 algorithm executions with different values of Σ, ζ and $\bar{r}$. The outcome of this scenario is shown in Fig. 15. The averaged value of the parameter vector is $\tilde{r}^* = [238.35,\ 234.2,\ 180.7,\ 211.6,\ 177.02,\ 170.02]'$. As expected, the computed parameter vector is different from the ones computed in the two previous scenarios. The computational time required to determine the parameter vector $\tilde{r}^*$ for this scenario was 1329 s.
4. In the fourth scenario, we considered $Q = 200$ and $L = 100$. We performed 10 executions of the proposed algorithm with different values of Σ, ζ and $\bar{r}$. The outcome of this scenario is shown in Fig. 16. The averaged value of the parameter vector is $\tilde{r}^* = [135.47,\ 132.86,\ 128.13,\ 48.07,\ 48.05,\ 48.61]'$. Again, due to a different choice of the algorithm parameters, we obtained a different parameter value. The computational time required to determine the parameter vector $\tilde{r}^*$ for this scenario was 50 499 s.

The main goal of Example 8.2 is to verify the convergence of the proposed multi-trajectory LSTD algorithm (when applied to machine replacement problems) and assess its parametric sensitivity. In particular, we considered different number of trajectories Q with different lengths L. The obtained parameter vectors $\tilde{r}^*$ are quite similar for the cases 1, 2, and 4, where trajectory lengths are sufficiently large to allow the algorithm to explore the state space over a given trajectory. As for case 3, the trajectory length is set to 1, and this results in a different value of the parameter vector $\tilde{r}^*$ since the algorithm performs only one step along a given trajectory. In conclusion, Q and L influence both the quality of the computed value function approximation and the convergence rate of the proposed multi-trajectories LSTD algorithm

In the next example, we modify the feature matrix in order to perform the generalized multi-trajectory LSTD by applying the result of Lemma 10 on the Machine Replacement problem.

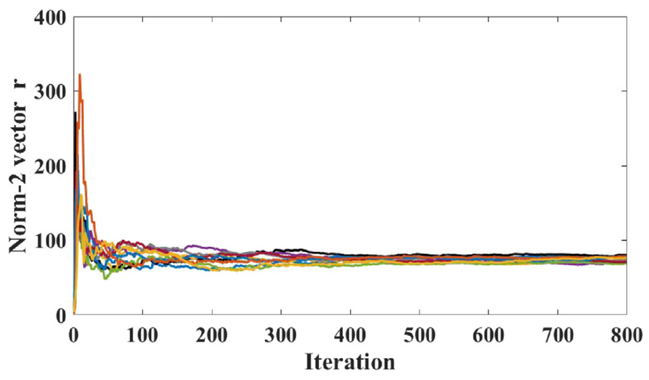


Fig. 17. Computed Norm-2 of the vector r_s for the feature matrix corresponding Lemma 10 in Example 8.3.

Example 8.3. Consider the machine replacement problem of Example 8.2 with the feature matrix resulted from Lemma 10.

The implementation of the our ADP algorithm with the feature vectors in the form of $\delta(x) = B\phi(x)$ has slower convergence speed compared to the previous feature vectors $\phi(x)$. This comes from the fact that the feature vector $\delta(x)$ requires the generation of the transition probabilities $\mathcal{P}_{x^v x^w}(u)$, which have to be computed along the Monte Carlo simulation trajectories. In particular, we also considered the policy improvement step as explained in Section 6.2, which makes the simulation speed slower.

In Fig. 17, norm-2 of the resulting parameter vector r is shown. In particular, we considered the case of $Q = 800$ trajectories with length $L = 1$, and run the proposed ADP algorithm 10 times. As we can see, all the curves converge to very close Norm-2 values. By taking the average of these results, we obtained $\tilde{r}^* = [37.58,\ 37.26,\ 36.02,\ 25.53,\ 24.05,\ 12.93]'$. The computational time for this example was 5124 s.

As expected, the computed parameter vector based on Lemma 10 has different values compared to the parameter vector of Example 8.2.

Example 8.4. Consider the machine replacement problem with $m = 5$ machines and $\Omega = 10$ operating states. The total number of states is 10^5. We performed our proposed ADP approach for the case of $L = 1$ and $Q \gg 1$ to compute the associated parameter vector r. We made use of the feature matrix of the form (89). The computed parameter vector with this setting is $\tilde{r}^* = [178.5,\ 179.6,\ 180.2,\ 181.2,\ 184.8]'$. We considered a set of 100 experiments each having a length of $T = 50$, and by using (90), we compared the result of our ADP approach with the non-optimal policies defined in Example 8.1. The outcome of this comparison is shown in Figs. 18 and 19. As shown in Fig. 18, the ADP approach has lower cumulative costs compared to the other non-optimal policies over the 100 experiments, and replaces fewer machines to reach such lower total cost with the exception of the non-optimal policies (i) and (iii), see Fig. 19. The computational time for this example was 4051 seconds. Finally, it was not possible to solve the machine replacement problem of this example via the exact DP approach in a reasonable time.

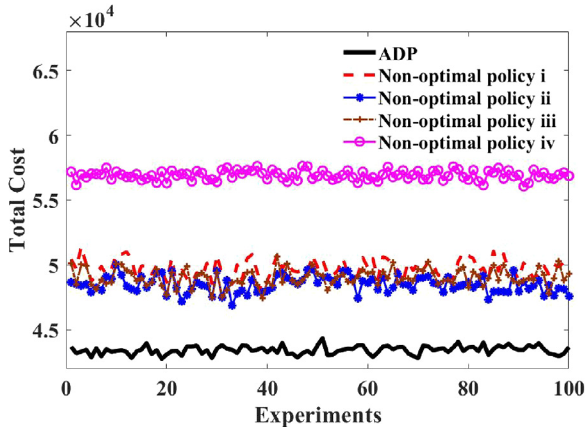


Fig. 18. Comparison of the total cost computed with the ADP approach with the non-optimal policies in Example 8.4.

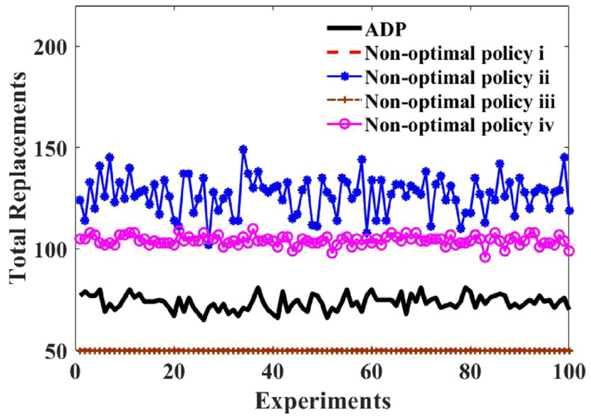


Fig. 19. Number of replacements with the ADP approach and the non-optimal policies in Example 8.4.

9. Conclusion

This paper has presented an approach to formulate the replacement problem of a set of machines as a composition of Markov Decision Processes (MDPs), one for each specific machine. The underlying stochastic optimization problem has been defined and solved over a finite time horizon by exploiting the principles of the backward Dynamic Programming (DP) algorithm. This way, a minimum operational cost policy can be determined, specifying the "keep" or the "replace" actions for all the machines, at each time slot, over the given finite time horizon.

We have also introduced the generalized multi-trajectory Least-Squares Temporal Difference (LSTD) algorithm to solve machine replacement problems featured by a large state space. This algorithm computes an approximate optimal cost function by running Monte Carlo simulations over different trajectories of a given length and enforcing an off-policy mechanism to improve the LSTD exploration capabilities. A study on its convergence properties has also been provided. Several numerical examples have been reported to illustrate the effectiveness of the proposed modeling and solution approaches.

As for future work, we plan to model and solve machine replacement problems featured by a set of operational constraints, also expressing a higher level of interdependence among the machines. The analytical formulation of the problem can take the form of a composition of Constrained Markov Decision Processes (CMDPs) with more complex cost per stage function definitions. Inspired by the available literature, preventive and corrective maintenance can also be integrated into the MDP formulation. In this paper, the states are assumed to be fully observed. Finally, the assumption of fully observable states can be removed to deal with the interesting case of machine replacement problems with hidden states.

CRediT authorship contribution statement

Ali Forootani: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Majid Ghaniee Zarch:** Software, Validation, Writing – original draft, Writing review & editing. **Massimo Tipaldi:** Methodology, Validation, Writing

– original draft, Writing – review & editing, Supervision. **Raffaele Iervolino:** Methodology, Validation, Writing – original draft, Writing – review & editing, Supervision.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

References

Antos, A., Szepesvári, C., Munos, R., 2008. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Mach. Learn. 71 (1), 89–129.

Barker, E., Ras, C.J., 2019. Unsupervised basis function adaptation for reinforcement learning. J. Mach. Learn. Res. 20, 1–73.

Bertsekas, D.P., 2011a. Approximate policy iteration: A survey and some new methods. J. Control Theory Appl. 9 (3), 310–335.

Bertsekas, D.P., 2011b. Temporal difference methods for general projected equations. IEEE Trans. Automat. Control 56 (9), 2128–2139.

Bertsekas, D.P., 2012. Dynamic Programming and Optimal Control, Volume II, fourth ed. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D., 2017. Dynamic Programming and Optimal Control, Volume I, fourth ed. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D.P., 2019. Feature-based aggregation and deep reinforcement learning: A survey and some new implementations. IEEE/CAA J. Autom. Sin. 6 (1), 1–31.

Bertsekas, D.P., Tsitsiklis, J.N., 1995. Neuro-dynamic programming: an overview. In: Proceedings of 1995 34th IEEE Conference on Decision and Control, Vol. 1. IEEE, pp. 560–564.

Bertsekas, D.P., Yu, H., 2009. Projected equation methods for approximate solution of large linear systems. J. Comput. Appl. Math. 227 (1), 27–50.

Busoniu, L., Ernst, D., De Schutter, B., Babuska, R., 2011. Cross-entropy optimization of control policies with adaptive basis functions. IEEE Trans. Syst. Man Cybern. B 41 (1), 196–209.

Cassady, C.R., Kutanoglu, E., 2005. Integrating preventive maintenance planning and production scheduling for a single machine. IEEE Trans. Reliab. 54 (2), 304–309.

Childress, S., Durango-Cohen, P., 2005. On parallel machine replacement problems with general replacement cost functions and stochastic deterioration. Nav. Res. Logist. 52 (5), 409–419.

Dann, C., Neumann, G., Peters, J., et al., 2014. Policy evaluation with temporal differences: A survey and comparison. J. Mach. Learn. Res. 15, 809–883.

Dong, W., Liu, S., Cao, Y., Javed, S.A., 2021. Scheduling optimal replacement policies for a stochastically deteriorating system subject to two types of shocks. ISA Trans. 112, 292–301.

Forootani, A., Iervolino, R., Tipaldi, M., 2019. Applying unweighted least-squares based techniques to stochastic dynamic programming: Theory and application. IET Control Theory Appl. 13 (15), 2387–2398.

Forootani, A., Iervolino, R., Tipaldi, M., Dey, S., 2022a. Transmission scheduling for multi-process multi-sensor remote estimation via approximate dynamic programming. Automatica 136, 110061.

Forootani, A., Iervolino, R., Tipaldi, M., Neilson, J., 2020a. Approximate dynamic programming for stochastic resource allocation problems. IEEE/CAA J. Autom. Sin. 7 (4), 975–990.

Forootani, A., Liuzza, D., Tipaldi, M., Glielmo, L., 2021a. Allocating resources via price management systems: a dynamic programming-based approach. Internat. J. Control 94 (8), 2123–2143.

Forootani, A., Tipaldi, M., Ghaniee Zarch, M., Liuzza, D., Glielmo, L., 2020b. A least-squares temporal difference based method for solving resource allocation problems. IFAC J. Syst. Control 13, 100106.

Forootani, A., Tipaldi, M., Ghaniee Zarch, M., Liuzza, D., Glielmo, L., 2021b. Modelling and solving resource allocation problems via a dynamic programming approach. Internat. J. Control 94 (6), 1544–1555.

Forootani, A., Tipaldi, M., Iervolino, R., Dey, S., 2022b. Enhanced exploration least-squares methods for optimal stopping problems. IEEE Control Syst. Lett. 6, 271–276.

Garí, Y., Monge, D.A., Pacini, E., Mateos, C., Garino, C.G., 2021. Reinforcement learning-based application autoscaling in the cloud: A survey. Eng. Appl. Artif. Intell. 102, 104288.

Geist, M., Pietquin, O., 2013. Algorithmic survey of parametric value function approximation. IEEE Trans. Neural Netw. Learn. Syst. 24 (6), 845–867.

Ghasempour, T., Heydecker, B., 2019. Adaptive railway traffic control using approximate dynamic programming. Transp. Res. Procedia 38, 201–221.

Gress, E.S.H., Monta, O., Armenta, J.R.C., Reyes, A.O.O., et al., 2012. A reward functional to solve the replacement problem. Intell. Control Autom. 3 (4), 413–418. http://dx.doi.org/10.4236/ica.2012.34045.

Hoffman, M.W., Lazaric, A., Ghavamzadeh, M., Munos, R., 2011. Regularized least squares temporal difference learning with nested l_2 and l_1 penalization. In: European Workshop on Reinforcement Learning. Springer, pp. 102–114.

Hu, J., Wang, Y., Pang, Y., Liu, Y., 2022. Optimal maintenance scheduling under uncertainties using linear programming-enhanced reinforcement learning. Eng. Appl. Artif. Intell. 109, 104655.

Huang, J., Chang, Q., Chakraborty, N., 2019. Machine preventive replacement policy for serial production lines based on reinforcement learning. In: 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). IEEE, pp. 523–528.

Iervolino, R., Tipaldi, M., Forootani, A., 2021. A Lyapunov-based version of the value iteration algorithm formulated as a discrete-time switched affine system. Internat. J. Control 1–16.

Jiang, K., Li, Z., Zhou, Y., Sarkodie-Gyan, T., Li, W., 2022. Transient waveform matching based on ascending multi-wavelets for diagnostics and prognostics of bearing deterioration. ISA Trans. 120, 330–341.

Jones, P.C., Zydiak, J.L., Hopp, W.J., 1991. Parallel machine replacement. Nav. Res. Logist. 38 (3), 351–365.

Lee, S., Boomsma, T.K., 2022. An approximate dynamic programming algorithm for short-term electric vehicle fleet operation under uncertainty. Appl. Energy 325, 119793.

Leu, S.-S., Ying, T.-M., 2020. Replacement and maintenance decision analysis for hydraulic machinery facilities at reservoirs under imperfect maintenance. Energies 13 (10), 2507.

Li, Y., 2020. Optimal parallel machine replacement policy under general repair. In: IIE Annual Conference. Proceedings. Institute of Industrial and Systems Engineers (IISE), pp. 382–387.

Li, F.-D., Wu, M., He, Y., Chen, X., 2012. Optimal control in microgrid using multi-agent reinforcement learning. ISA Trans. 51 (6), 743–751.

Lin, J., Dou, C., 2015. The diagnostic line: A novel criterion for condition monitoring of rotating machinery. ISA Trans. 59, 232–242.

Liu, B., Pandey, M.D., Wang, X., Zhao, X., 2021. A finite-horizon condition-based maintenance policy for a two-unit system with dependent degradation processes. European J. Oper. Res. 295 (2), 705–717.

Luenberger, D.G., 1979. Introduction to Dynamic Systems; Theory, Models, and Applications. Wiley.

Nedić, A., Bertsekas, D.P., 2003. Least squares policy evaluation algorithms with linear function approximation. Discrete Event Dyn. Syst. 13 (1), 79–110.

Nodem, F.D., Gharbi, A., Kenné, J.-P., 2011. Preventive maintenance and replacement policies for deteriorating production systems subject to imperfect repairs. Int. J. Prod. Res. 49 (12), 3543–3563.

Nodem, F.D., Kenné, J.-P., Gharbi, A., 2009. Hierarchical decision making in production and repair/replacement planning with imperfect repairs under uncertainties. European J. Oper. Res. 198 (1), 173–189.

Nowakowski, T., Werbińka, S., 2009. On problems of multicomponent system maintenance modelling. Int. J. Autom. Comput. 6 (4), 364–378.

Ouaret, S., Kenné, J.-P., Gharbi, A., 2018. Joint production and replacement planning for an unreliable manufacturing system subject to random demand and quality. IFAC-PapersOnLine 51 (11), 951–956.

Ouaret, S., Kenné, J.-P., Gharbi, A., 2019. Production and replacement planning of a deteriorating remanufacturing system in a closed-loop configuration. J. Manuf. Syst. 53, 234–248.

Pan, Y., Thomas, M.U., 2010. Repair and replacement decisions for warranted products under Markov deterioration. IEEE Trans. Reliab. 59 (2), 368–373.

Schouten, T.N., Dekker, R., Hekimoğlu, M., Eruguz, A.S., 2022. Maintenance optimization for a single wind turbine component under time-varying costs. European J. Oper. Res. 300 (3), 979–991.

Schütz, H.J., Kolisch, R., 2012. Approximate dynamic programming for capacity allocation in the service industry. European J. Oper. Res. 218 (1), 239–250.

Seif, J., Shields, B.A., Yu, A.J., 2019. Parallel machine replacement under horizon uncertainty. Eng. Econ. 64 (1), 1–23.

Sharifi, M., Taghipour, S., 2021. Optimal production and maintenance scheduling for a degrading multi-failure modes single-machine production environment. Appl. Soft Comput. 106, 107312.

van Staden, H.E., Deprez, L., Boute, R.N., 2022. A dynamic “predict, then optimize” preventive maintenance approach using operational intervention data. European J. Oper. Res. 302 (3), 1079–1096.

Sutton, R.S., Mahmood, A.R., White, M., 2016. An emphatic approach to the problem of off-policy temporal-difference learning. J. Mach. Learn. Res. 17 (1), 2603–2631.

Thiery, C., Scherrer, B., 2010. Least-squares λ policy iteration: Bias-variance trade-off in control problems. In: International Conference on Machine Learning.

Wang, H., 2002. A survey of maintenance policies of deteriorating systems. European J. Oper. Res. 139 (3), 469–489.

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

Yu, H., Bertsekas, D.P., 2009. Convergence results for some temporal difference methods based on least squares. IEEE Trans. Automat. Control 54 (7), 1515–1531.

Zhang, P., Zhu, X., Xie, M., 2021. A model-based reinforcement learning approach for maintenance optimization of degrading systems in a large state space. Comput. Ind. Eng. 161, 107622.

Zhu, J., Chen, J., Zhuo, Y., Mo, X., Guo, Y., Liu, L., Liu, M., 2022. Stochastic energy management of active distribution network based on improved approximate dynamic programming. IEEE Trans. Smart Grid 13 (1), 406–416.