Optimally Replacing Multiple Systems in a Shared Environment

David T. Abdul-Malak[1] and Jeffrey P. Kharoufeh[2]
Department of Industrial Engineering
University of Pittsburgh
1025 Benedum Hall
3700 O'Hara Street
Pittsburgh, PA 15261 USA

Accepted Version
April 5, 2017

Abstract

We consider the problem of optimally replacing multiple stochastically degrading systems using condition-based maintenance. Each system degrades continuously at a rate that is governed by the current state of the environment, and each fails once its own cumulative degradation threshold is reached. The objective is to minimize the sum of the expected total discounted setup, preventive replacement, reactive replacement, and downtime costs over an infinite horizon. For each environment state, we prove that the cost function is monotone nondecreasing in the cumulative degradation level. Additionally, under mild conditions, these monotonicity results are extended to the entire state space. In the case of a single system, we establish that monotone policies are optimal. The monotonicity results help facilitate a tractable, approximate model with state- and action-space transformations and a basis-function approximation of the action-value function. Our computational study demonstrates that high-quality, near-optimal policies are attainable and significantly outperform heuristic policies.

1 Introduction

The emergence of low-cost, advanced sensing technologies and real-time condition-monitoring systems have led to increased interest in advanced maintenance planning strategies. Condition-based maintenance (CBM) techniques utilize up-to-date condition information to make well-informed maintenance decisions to achieve important objectives (e.g., minimize maintenance costs, maximize revenue, or maximize system availability). For example, modern wind turbine systems use advanced sensors to measure particle contamination levels in lubricating fluids, shaft torque, electrical discharge, vibrations, acoustic emissions, torsional vibration, and many other signals of degradation [39]. CBM provides an opportunity to exploit degradation measurements, or signals of degradation, for system prognosis and intelligent maintenance decision making that increases system uptime while reducing maintenance and operational costs.

[1]Email: dta10@pitt.edu
[2]Corresponding author. Ph: (412) 624-9832; Email: jkharouf@pitt.edu

Many large-scale systems that degrade stochastically over time are difficult to analyze in the presence of dependencies, including stochastic, economic, and structural dependencies. Stochastic dependencies are prevalent when integrated components, or systems, do not degrade (or fail) independently. For example, within a single wind farm, wind turbines are exposed to common, local weather conditions and, therefore, operate in a shared ambient environment. This exposure to similar environmental conditions may lead to dependencies in the degradation sample paths of individual wind turbines. Economic dependencies can be viewed as any financial linkages between maintenance actions, e.g., exploitation of shared downtime or rented equipment. Finally, structural dependencies result when maintenance activities performed on one component, or subsystem, require maintenance activities to occur on another component or subsystem. For instance, these types of dependencies exist when a multi-component system is enclosed in a single machine, and disassembly is required to repair or replace failed components.

In this paper, we consider the problem of determining optimal replacement strategies for multiple, stochastically degrading systems that exhibit both stochastic and economic dependencies. Within this context, we refer to multiple machines with similar characteristics operating in close proximity to one another. The systems operate in a shared, exogenous environment that evolves randomly over time and modulates the rates of degradation of each of the systems. The systems, which are stochastically heterogenous, degrade monotonically until the cumulative level of degradation reaches a threshold, or the system is replaced. The costs of maintaining this system with replacements include a substantial fixed cost that is incurred when any maintenance action is taken. Replacements may occur preventively (before failure) or reactively (in response to a failure). Preventive replacements are less costly than reactive replacements. Finally, a downtime cost is incurred whenever the system is taken off-line to perform replacements. Our objective is obtain cost-minimizing replacement policies that account for preventive and reactive replacement decisions, as well as downtime and fixed setup costs. To this end, we formulate a continuous-time, infinite-horizon discounted Markov decision process (MDP) model, establish important properties of the cost function, reformulate the model using well-devised approximation techniques, and customize an approximate dynamic programming (ADP) algorithm to obtain high-quality policies.

For well over five decades, maintenance optimization models have emerged in the applied probability and operations research communities. Many extensive surveys highlight some of the most prevalent models for both single- and multi-component systems [1, 22, 24, 25, 30, 31, 32, 37, 38]. Most classical models consider single-component systems that operate in a static environment. Furthermore, the majority of these models employ failure-based (as opposed to condition-based) decision making. Recent emphasis on condition-based maintenance strategies has led to a stream of research on degradation-based reliability of single-component systems. These strategies seek to model degradation (or signals of degradation) as a stochastic process evolving in continuous time. Some representative models consider degradation as a Brownian motion process [8, 12, 40], while others assume that degradation is modulated by Markov or semi-Markov processes [15, 16, 17, 18].

Additionally, there exists a significant body of literature on non-condition-based policies for multi-component systems, and surveys of this literature can be found in [9, 11, 23]. Recently, Ko and Byon [19] use asymptotic theory to analytically derive the cost minimizing policy for a large-scale system with finite condition states and independent and identically degrading components.

Over the past two decades, a modest body of literature has emerged for multi-component or multi-system CBM. Marseguerra et al. [21] formulated a joint optimization model that seeks to maximize availability and net profit within a large-scale system with stochastic dependencies. Optimal thresholds, beyond which preventive maintenance should be performed, are computed by Monte Carlo simulation embedded within a genetic algorithm. Castanier et al. [7] considered a discrete-time, series system with shared setup cost for inspection or replacement. The model's decision variables are thresholds on inspection and preventive, corrective, and opportunistic maintenance. These thresholds are determined analytically for a two-component system that degrades in a static environment. Bouvard et al. [6] studied a large-scale system with shared setup cost within the context of commercial heavy vehicle maintenance. Rolling horizon procedures were developed that incorporate component information, and dynamic, analytical maintenance intervals were obtained. Tian and Liao [34] investigated policies similar to those in [7] for the general multi-component case with multiple identical and independent components. Dual threshold policies, for which once any component's failure risk exceeds the first threshold, all components with failure risk over the second threshold are replaced, were obtained numerically. Zhu et al. [41] considered a multi-component system with non-identical, independently degrading components and large shared setup costs for maintenance activities. They examined the case where the degradation paths are described by a random coefficient model and developed a nested enumeration algorithm to simultaneously obtain the optimal maintenance interval and optimal preventive maintenance control limits.

The model we present here differs from existing replacement models in that we consider large-scale systems with condition-based maintenance, stochastic dependency (through a shared modulating environment), economic dependency (through a shared setup cost), non-identical systems, degradation modulated by an exogenous continuous-time stochastic process, and continuous degradation sample paths. These model features present challenges for: (i) exact analysis of the the structure of the value function and optimal policy; and (ii) numerical computation of optimal policies for particular problem instances. Those points notwithstanding, our model is significant, as the gap between theory and practice of maintenance models is appreciable, and these realistic features represent a step towards bridging that gap [10, 30]. In the general case of multiple systems, we establish monotonicity of the value function in the cumulative degradation level for each environment state. Then, under mild conditions, it is shown that this monotonicity extends to the entire state space. Additionally, this framework allows for analysis of the special single-system case for which we show that optimal replacement policies are monotone on the entire state space. This result partially resolves the conjecture of Ulukus et al. [36] when reactive replacements do not occur immediately. Subsequently, we exploit the monotonicity results of the value function to devise

a tractable, approximate model with state- and action-space transformations and a customized, basis-function approximation of the action-value function. The novel state space transformation maps abstract degradation levels to probabilities that are easily interpreted. In addition to improved interpretability, the action-value function can be more easily approximated using this new state space. We believe this type of transformation is sufficiently general to find applicability in a wide range of maintenance optimization problems. Finally, we provide a detailed computational study to demonstrate the efficacy of the approximate model in producing near-optimal policies. Specifically, we obtain policies that are nearly indistinguishable from the optimal policy in small-scale instances, as well as policies that significantly outperform heuristics in large-scale instances. To our knowledge, these techniques and algorithms are novel within the maintenance optimization literature.

The remainder of the paper is organized as follows. In Section 2, we describe the environment process and its relationship to the degradation of the systems, and formulate a mathematical model of the sequential decision process. Section 3 discusses attributes of the value function and the optimal decision rule. In Section 4, we reformulate the problem using an approximate dynamic programming (ADP) model and demonstrate the usefulness of this reformulation through numerical examples in Section 5.

2 Degradation Model and Problem Formulation

Consider a collection of n ($n < \infty$) systems operating in a shared, exogenous environment. The systems are assumed to begin operation in an as-good-as-new condition. The degradation rate of each system is governed by the randomly evolving environment, which occupies one of finitely many states at any point in time. Over time, each system accumulates degradation until it reaches its own fixed, deterministic threshold, above which it is considered to be failed. For the ith system, we denote this failure threshold by ξ_i ($0 < \xi_i < \infty$), $i = 1, \ldots, n$. In what follows, all random variables are defined on a common, complete probability space $(\Omega, \mathscr{F}, \mathbb{P})$.

Let $Z(t)$ denote the state of the environment at time t and $\mathcal{Z} \equiv \{Z(t) : t \geq 0\}$ is the environment process defined on the finite state space $S = \{1, \ldots, \ell\}$. For this model, it is assumed that $\mathcal{Z}$ evolves as an S-valued, irreducible, continuous-time Markov chain (CTMC). The environment state $j \in S$ can be understood as an abstract classification of the exogenous factors that impact the degradation of each system. Specifically, whenever $Z(t) = j \in S$, the ith system degrades linearly at a rate r_i^j. Without loss of generality, it is assumed that, for each system, the degradation rates are positive, finite and monotone increasing in the environment state, i.e., $0 < r_i^1 < r_i^2 < \cdots < r_i^\ell < \infty$, for $i = 1, \ldots, n$. Denote by $X_i(t)$ the cumulative degradation of system i at time t given by

$$X_i(t) = X_i(0) + \int_0^t r_i^{Z(u)} du, \quad t \geq 0, \tag{1}$$

where $X_i(0)$ denotes the initial degradation level of system i. Assuming $X_i(0) = 0$, and noting that

for each i

$$\int_0^t r_i^{Z(u)} du < \infty,$$

$X_i(t)$ is well defined for each $t \geq 0$. Moreover, as noted in [36], the non-negativity of the degradation rates, $\{r_i^j\}$, ensures that the sample paths of the degradation process $\mathcal{X} \equiv \{X_i(t) : t \geq 0\}$ are piecewise linear and monotone increasing in t.

Now, we introduce a Markov decision process (MDP) model to formulate the problem of optimally replacing multiple systems in a shared environment. The objective is to minimize the sum of the expected total discounted setup, replacement, and downtime costs over an infinite time horizon. This model can be viewed as an extension of the MDP model presented in [36] for a single system, with a minor variation in the costs and system dynamics. The state of the process is an $(n+1)$-dimensional vector of the form $(\boldsymbol{x}, j)$ – a realization of the joint process $(\mathcal{X}, \mathcal{Z})$ in which $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the vector of the systems' cumulative degradation levels, and $j \in S$ denotes the current state of the environment. Without loss of generality, we can scale the degradation rates appropriately and assume that $\xi_i = \xi$ for all i; therefore, the state space of the MDP model is the set $\Gamma \equiv [0, \xi]^n \times S$. The set of feasible actions (or action space) is $\mathcal{A} = \{0, 1\}^n$ where, for $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathcal{A}$, $a_i = 0$ corresponds to taking no action on the ith system, and $a_i = 1$ corresponds to replacing system i. System replacements can be done preventively (before failure) at a fixed cost c_p, or reactively (after a failure) at a fixed cost c_r. It is reasonable to assume that $c_p < c_r$. In addition to the system replacement costs, a fixed cost c_0 is assessed if any maintenance is performed. In practice, this cost may account for equipment costs, crew wages, or travel expenses associated with maintenance. Finally, a cost c_d for the expected per period lost productivity is assessed for each failed system that is inoperable, and we assume that all costs are non-negative and bounded from above, i.e., $c_0, c_p, c_r, c_d \in [0, \infty)$.

Because the environment process $\mathcal{Z}$ evolves as a CTMC on the finite state space S, we employ the common strategy of uniformization (cf. Puterman [27]) and convert to a discrete-time Markov chain (DTMC). Denote by $\mathcal{Q} = [q_{jk}]$ the infinitesimal generator matrix of $\mathcal{Z}$, and let q_j be the total transition rate out of state j, i.e.,

$$q_j \equiv -q_{jj} = \sum_{k:k \neq j} q_{jk}, \quad j \in S.$$

Define a uniformization rate $q \geq \max\{q_j : j \in S\}$. By uniformizing the environment process, the system is inspected at exponentially-distributed intervals of time. We denote by T_m the length of the mth inter-inspection period, which is exponentially-distributed with rate q for each $m = 1, 2, \ldots$. Denote the (random) degradation level, (random) environment state, and action taken at the mth inspection time to be $\boldsymbol{X}_m = [X_m^i]$, Z_m, and $\boldsymbol{A}_m = [A_m^i]$, respectively. Similarly, denote the (random) state at the mth period to be $\boldsymbol{S}_m = (\boldsymbol{X}_m, Z_m)$. For $j, k \in S$, define the transition probabilities of the discretized environment process,

$$p_{jk} = \mathbb{P}(Z_{m+1} = k | Z_m = j),$$

the probability that the environment transitions from state j to k during one inter-inspection period given by

$$p_{jk} = \begin{cases} q_{jk}/q, & k \neq j, \\ 1 - q_j/q, & k = j, \end{cases}$$

and let $\boldsymbol{P} = [p_{jk}]$ be the transition probability matrix of the uniformized chain. Each transition epoch of the uniformized process is then treated as an inspection time. Therefore, the minimum inspection rate is dictated by the environment process, and the inspection times are linked to the environment process through the embedded discrete-time Markov chain $\{Z_n : n \geq 0\}$.

Next, let Y_i^j be the (random) one-step accumulated degradation of system $i \in \{1, 2, \ldots, n\}$ while the environment is in state $j \in S$. The exponential length of the inter-inspection period and the constant rate of degradation jointly imply that Y_i^j is exponentially distributed with rate parameter q/r_i^j. That is,

$$F_i^j(y) := \mathbb{P}\left(Y_i^j \leq y\right) = \mathbb{P}\left(r_i^j\, T_m \leq y\right) = 1 - \exp\left(-qy/r_i^j\right).$$

For notational convenience, define the indicator function

$$\mathbb{I}_i(\boldsymbol{x}) = \begin{cases} 1, & x_i \geq \xi, \\ 0, & x_i < \xi. \end{cases}$$

That is, $\mathbb{I}_i(\boldsymbol{x})$ indicates whether or not system i is failed in the degradation vector $\boldsymbol{x}$. Therefore, the expected one-step cost associated with state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ is

$$c(\boldsymbol{s}, \boldsymbol{a}) \equiv c_j(\boldsymbol{x}, \boldsymbol{a}) = \begin{cases} c_0 + \displaystyle\sum_{i=1}^{n} \left[a_i(1 - \mathbb{I}_i(\boldsymbol{x}))c_p + a_i \mathbb{I}_i(\boldsymbol{x}) c_r + (1 - a_i)\mathbb{I}_i(\boldsymbol{x}) c_d\right], & \boldsymbol{a} \neq \boldsymbol{0}, \\ c_d \displaystyle\sum_{i=1}^{n} \mathbb{I}_i(\boldsymbol{x}), & \boldsymbol{a} = \boldsymbol{0}. \end{cases} \tag{2}$$

For many applications, the downtime cost c_d may depend on the length of the inter-inspection period. Within our framework, this dependence is reflected by the fact that c_d is a function of the uniformization constant q; however, for notational convenience, we suppress this dependence on q, as it is reasonable to assume that $c_d(q)$ is monotone nonincreasing in q. We discuss the implications of this assumption at the end of Section 3. Inspections are assumed to be costless and instantaneous. If the replacement action is taken, the degradation of that system ceases, and the replacement occurs during the current period and ends just prior to the start of the next period. These assumptions ensure that failures do not occur once preventive replacement is decided upon and that the system begins the next period in as-good-as-new condition. All one-step costs are incurred at the beginning of the period and discounted at rate α $(0 < \alpha < 1)$, where

$$\alpha = \frac{q}{\theta + q}$$

for some continuous-time discount rate θ ($0 < \theta < \infty$). The objective is to minimize the expected total discounted setup, replacement and downtime costs over an infinite planning horizon. The optimal expected total discounted cost, starting in state $\boldsymbol{s} = (\boldsymbol{x}, j)$ and denoted by $V_j(\boldsymbol{x})$, is given as a solution to the Bellman optimality equations

$$V_j(\boldsymbol{x}) = \min_{\boldsymbol{a} \in \mathcal{A}} \left\{ c_j(\boldsymbol{x}, \boldsymbol{a}) + \alpha \sum_{k=1}^{\ell} \left(\int_0^{\infty} V_k\left(\boldsymbol{x}'\right) q e^{-qt} dt \right) p_{jk} \right\}, \quad j \in S, \tag{3}$$

where $\boldsymbol{x}' = [x_i']$ with

$$x_i' = \begin{cases} \min\{\xi,\, x_i + t\, r_i^j\}, & a_i = 0, \\ 0, & a_i = 1. \end{cases}$$

It should be noted that this formulation does not force reactive replacements immediately upon failure. This added flexibility allows the decision maker to pool replacements when appropriate, thereby sharing the cost (c_0) amongst multiple repairs.

In order to solve this problem by conventional MDP solution techniques, the state space must first be discretized so that numerical methods can be applied (e.g., value iteration or policy iteration [3, 14]). It is clear from this formulation that as n, the number of systems, grows large, the problem will suffer from the curse of dimensionality. This occurs because the state space is an n-dimensional hypercube, and the number of permissible actions is 2^n. For example, with 50 systems and one environment state, the number of feasible actions is approximately 10^{15}, and discretizing the state space into $1,000$ states per component yields 10^{100} states. For these reasons, we propose handling this problem by approximate dynamic programming (ADP) methods. In particular, we customize a state-action-reward-state-action (SARSA) algorithm with eligibility traces and basis function approximation [26, 28]. Before doing so, we first provide some useful structural results that help characterize an optimal policy.

3 Structural Results

In this section, we characterize the attributes of the cost function and optimal policy of the MDP model presented in Section 2. We first consider the general, multiple system case before presenting results for the special case of a single system.

3.1 Results for the General Case

The first result for n ($n > 1$) systems establishes the existence of a stationary optimal policy, as well as the convergence of the standard value iteration algorithm.

Lemma 1 *There exists an optimal, non-randomized stationary replacement policy, and the value iteration algorithm converges to the optimal value.*

Proof. First note that the state space, $\Gamma = [0, \xi]^n \times S$, is Borel-measurable and the action space, $\mathcal{A} = \{0, 1\}^n$, is finite. Additionally, the immediate costs are strictly positive and bounded, and the problem is discounted. Therefore, the result follows immediately from Corollary 9.17.1 of Bertsekas and Shreve [4]. ∎

For the state space Γ, define the binary relation $(\leq)$ as the standard component-wise inequality. That is, for any two vectors $\boldsymbol{s}, \boldsymbol{s}' \in \Gamma \subset \mathbb{R}^{n+1}$, $\boldsymbol{s} \leq \boldsymbol{s}'$ if and only if $s_i \leq s'_i$ for each $i = 1, \ldots, n+1$. It can be verified that $\leq$ is a partial order on Γ and $\mathcal{A}$; therefore, $(\Gamma, \leq)$ and $(\mathcal{A}, \leq)$ are partially-ordered sets (posets). Additionally, as formalized in Lemma 2, $(\Gamma, \leq)$ is a lattice (see Birkhoff [5] for a detailed discussion of lattices).

Lemma 2 *The partially ordered sets* $(\Gamma, \leq)$ *and* $(\mathcal{A}, \leq)$ *are lattices.*

Proof. For any pair of states $\boldsymbol{s}, \boldsymbol{s}' \in \Gamma$, we have that

$$\bar{\boldsymbol{s}} := (\max\{s_1, s'_1\}, \ldots, \max\{s_{n+1}, s'_{n+1}\}) \in \Gamma,$$

and is an upper bound for $\boldsymbol{s}$ and $\boldsymbol{s}'$. Letting $\bar{\boldsymbol{s}}' = (\bar{s}'_1, \ldots, \bar{s}'_{n+1})$ be an arbitrary upper bound on $\boldsymbol{s}, \boldsymbol{s}'$, it is seen that $\bar{s}'_i \geq s_i$ and $\bar{s}'_i \geq s'_i$. Hence, $\bar{s}'_i \geq \max\{s_i, s'_i\}$ for each $i = 1, \ldots, n+1$. Therefore, $\bar{\boldsymbol{s}}' \geq \bar{\boldsymbol{s}}$, which implies $\boldsymbol{s} \vee \boldsymbol{s}' = \bar{\boldsymbol{s}} \in \Gamma$. Similarly, by component-wise minimization, $\boldsymbol{s} \wedge \boldsymbol{s}' \in \Gamma$. The proof that $(\mathcal{A}, \leq)$ is a lattice proceeds in an identical fashion. ∎

The notion of submodularity plays an important role in deriving our main results; therefore, we formally define it next.

Definition 1 *(Submodularity). A function* $f : A \times B \to \mathbb{R}$ *is said to be submodular on* $A \times B$ *if*

$$f(a_2, b_2) - f(a_2, b_1) \leq f(a_1, b_2) - f(a_1, b_1) \tag{4}$$

for any $a_1, a_2 \in A$ *and* $b_1, b_2 \in B$ *such that* $a_1 \leq a_2$ *and* $b_1 \leq b_2$.

Submodularity is useful for characterizing structural properties in optimization problems. Here, we restate an important result due to Topkis [35] as Theorem 1.

Theorem 1 *(Topkis [35]). Let* $f : A \times B \to \mathbb{R}$ *be a submodular function on* $A \times B$*, and let* $(A, \leq)$ *and* $(B, \leq)$ *be lattices. Then* $g^*(b) = \max\{a' \in \operatorname{argmin}_{a \in A} f(a, b)\}$ *is nondecreasing in* b.

For the results that follow, define the Q-function,

$$Q(\boldsymbol{s}, \boldsymbol{a}) \equiv Q_j(\boldsymbol{x}, \boldsymbol{a}) := c(\boldsymbol{s}, \boldsymbol{a}) + \alpha \mathbb{E}\left(V(\boldsymbol{S}_1) | \boldsymbol{S}_0 = \boldsymbol{s}, \boldsymbol{A}_0 = \boldsymbol{a}\right). \tag{5}$$

By Theorem 1, if it can be shown that Q is submodular on $\Gamma \times \mathcal{A}$, then the optimal decision rule,

$$d^*(\boldsymbol{s}) := \max\left\{\boldsymbol{a}' \in \operatorname{argmin}_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}, \boldsymbol{a})\right\}, \tag{6}$$

is monotone in Γ. This result is formalized under some specific conditions in Theorem 4. However, before proceeding to the main results, Lemma 3 provides basic insights into the structure of the

expected one-step cost function $c(\boldsymbol{s}, \boldsymbol{a})$; it asserts that the one-step cost does not decrease as the cumulative degradation level increases. Additionally, under a particular cost structure, the one-step cost function is submodular.

Lemma 3 *The expected one-step cost function $c(\boldsymbol{s}, \boldsymbol{a})$ is*

(a) monotone nondecreasing in $\boldsymbol{s}$, and

(b) submodular on $\Gamma \times \mathcal{A}$, if $c_r - c_p \leq c_d$.

Proof. We note that if $\boldsymbol{a} \neq \boldsymbol{0}$, then $c_j(\boldsymbol{x}, \boldsymbol{a})$ can be expressed as

$$c_j(\boldsymbol{x}, \boldsymbol{a}) = c_0 + \sum_{i=1}^{n} [a_i c_p + \mathbb{I}_i(\boldsymbol{x}) a_i (c_r - c_p) + \mathbb{I}_i(\boldsymbol{x})(1 - a_i) c_d].$$

Because it is assumed that $c_r > c_p$, Lemma 3(a) follows immediately. For Lemma 3(b), we seek to show that for $\boldsymbol{a} < \boldsymbol{a}'$ and $\boldsymbol{x} < \boldsymbol{x}'$,

$$c_j(\boldsymbol{x}', \boldsymbol{a}') + c_j(\boldsymbol{x}, \boldsymbol{a}) \leq c_j(\boldsymbol{x}', \boldsymbol{a}) + c_j(\boldsymbol{x}, \boldsymbol{a}'). \tag{7}$$

We begin with the case where $\boldsymbol{a} = \boldsymbol{0} < \boldsymbol{a}'$. The left-hand side (l.h.s.) of (7) is given by

$$\begin{aligned} c_j(\boldsymbol{x}', \boldsymbol{a}') + c_j(\boldsymbol{x}, \boldsymbol{a}) &= c_0 + \sum_{i=1}^{n} \left[\boldsymbol{a}_i' c_p + \boldsymbol{a}_i' \mathbb{I}_i(\boldsymbol{x}')(c_r - c_p) + \mathbb{I}_i(\boldsymbol{x}')(1 - \boldsymbol{a}_i') c_d \right] + c_d \sum_{i=1}^{n} \mathbb{I}_i(\boldsymbol{x}) \\ &= c_0 + \sum_{i=1}^{n} \left[\boldsymbol{a}_i' c_p + \boldsymbol{a}_i' \mathbb{I}_i(\boldsymbol{x}')(c_r - c_p) + \left(\mathbb{I}_i(\boldsymbol{x}')(1 - \boldsymbol{a}_i') + \mathbb{I}_i(\boldsymbol{x}) \right) c_d \right]. \end{aligned} \tag{8}$$

Similarly, the right-hand side (r.h.s.) of (7) is given by

$$c_j(\boldsymbol{x}', \boldsymbol{a}) + c_j(\boldsymbol{x}, \boldsymbol{a}') = c_0 + \sum_{i=1}^{n} \left[\boldsymbol{a}_i' c_p + \boldsymbol{a}_i' \mathbb{I}_i(\boldsymbol{x})(c_r - c_p) + \left(\mathbb{I}_i(\boldsymbol{x})(1 - \boldsymbol{a}_i') + \mathbb{I}_i(\boldsymbol{x}') \right) c_d \right]. \tag{9}$$

After some algebraic manipulation, it can be shown that the quantity in (8) is no greater than that in (9) if, and only if,

$$c_d \sum_{i=1}^{n} \boldsymbol{a}_i' (\mathbb{I}_i(\boldsymbol{x}') - \mathbb{I}_i(\boldsymbol{x})) \geq (c_r - c_p) \sum_{i=1}^{n} \boldsymbol{a}_i' (\mathbb{I}_i(\boldsymbol{x}') - \mathbb{I}_i(\boldsymbol{x})).$$

By supposition, $c_r - c_p \leq c_d$; therefore, inequality (7) holds. Similarly, in case $\boldsymbol{0} < \boldsymbol{a} < \boldsymbol{a}'$, it can be shown that inequality (7) holds if, and only if,

$$c_d \sum_{i=1}^{n} \left[\mathbb{I}_i(\boldsymbol{x})(\boldsymbol{a}_i - \boldsymbol{a}_i') + \mathbb{I}_i(\boldsymbol{x}')(\boldsymbol{a}_i' - \boldsymbol{a}_i) \right] \geq (c_r - c_p) \sum_{i=1}^{n} \left[\mathbb{I}_i(\boldsymbol{x})(\boldsymbol{a}_i - \boldsymbol{a}_i') + \mathbb{I}_i(\boldsymbol{x}')(\boldsymbol{a}_i' - \boldsymbol{a}_i) \right].$$

For each $i = 1, \ldots, n$,

$$\mathbb{I}_i(\boldsymbol{x})(\boldsymbol{a}_i - \boldsymbol{a}_i') + \mathbb{I}_i(\boldsymbol{x}')(\boldsymbol{a}_i' - \boldsymbol{a}_i) = (\mathbb{I}_i(\boldsymbol{x}') - \mathbb{I}_i(\boldsymbol{x}))(\boldsymbol{a}_i' - \boldsymbol{a}_i) \geq 0.$$

Therefore, by the condition, $c_r - c_p \leq c_d$, inequality (7) holds. ∎

We pause here to note that the condition of Lemma 3(b), $c_r - c_p \leq c_d$, may not hold in practice. In particular, for a large uniformization rate q, it is unlikely that this inequality is valid. Fortunately, a useful property still emerges, namely that the value function is monotone nondecreasing in the cumulative degradation level, even if the condition is relaxed. We formalize this result in Theorem 2.

Theorem 2 *For each $j \in S$, the value function $V_j(\boldsymbol{x})$ is monotone nondecreasing in the degradation level $\boldsymbol{x} \in \mathcal{X}$.*

Proof. For $(\boldsymbol{x}, j) \in \Gamma$, denote the mth iterate of the value iteration algorithm by $v_j^m(\boldsymbol{x}) \equiv v^m(\boldsymbol{x}, j)$. We prove the theorem by induction on m. Take $v_j^0(\boldsymbol{x}) = 0$ for all $(\boldsymbol{x}, j) \in \Gamma$. Therefore,

$$\begin{aligned} v_j^1(\boldsymbol{x}) &= \min_{\boldsymbol{a} \in \mathcal{A}} \left\{ c_j(\boldsymbol{x}, \boldsymbol{a}) + \alpha \mathbb{E}\left(v^0(\boldsymbol{X}_1, Z_1) | \boldsymbol{X}_0 = \boldsymbol{x}, Z_0 = j, \boldsymbol{A}_0 = \boldsymbol{a} \right) \right\} \\ &= \min_{\boldsymbol{a} \in \mathcal{A}} \left\{ c_j(\boldsymbol{x}, \boldsymbol{a}) + 0 \right\} \\ &= c_d \sum_{i=1}^{n} \mathbb{I}_i(\boldsymbol{x}), \end{aligned}$$

which is monotone nondecreasing in $\boldsymbol{x}$. For the induction hypothesis, assume $v_j^m(\boldsymbol{x})$ is monotone nondecreasing in $\boldsymbol{x} \in \mathcal{X}$ for each $j \in S$. Let $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ such that $\boldsymbol{x}_1 \leq \boldsymbol{x}_2$, then for each $\boldsymbol{a} \in \mathcal{A}$

$$\begin{aligned} v_j^{m+1}(\boldsymbol{x}_1, \boldsymbol{a}) &= c_j(\boldsymbol{x}_1, \boldsymbol{a}) + \alpha \sum_{k=1}^{\ell} \left(\int_0^{\infty} v_k^m(\boldsymbol{x}_1') q e^{-qt} dt \right) p_{jk} \\ &\leq c_j(\boldsymbol{x}_2, \boldsymbol{a}) + \alpha \sum_{k=1}^{\ell} \left(\int_0^{\infty} v_k^m(\boldsymbol{x}_2') q e^{-qt} dt \right) p_{jk} \\ &= v_j^{m+1}(\boldsymbol{x}_2, \boldsymbol{a}), \end{aligned} \tag{10}$$

where the inequality holds due to Lemma 3(a) and by noting that, for a fixed $\boldsymbol{a}$, $\boldsymbol{x}_1' \leq \boldsymbol{x}_2'$. Minimizing both sides of (10) over $\boldsymbol{a} \in \mathcal{A}$ shows that $v_j^{m+1}(\boldsymbol{x}_1) \leq v_j^{m+1}(\boldsymbol{x}_2)$. Finally, Lemma 1 implies that $v_j^m(\boldsymbol{x}) \to V_j(\boldsymbol{x})$, as $m \to \infty$, and the proof is complete. ∎

Theorem 2 asserts that the expected cost-to-go increases as the cumulative degradation level of the system increases. Informally, this corresponds to the intuitive idea that starting in a 'bad' state is, in fact, more costly than starting in a 'good' state. Under some reasonable conditions we can prove stronger structural results.

For the remainder of this section, we impose mild conditions on the uniformized environment process and the degradation rate vectors $\boldsymbol{r}_i$. For completeness, we review the notion of the increasing failure rate (IFR) property of a transition probability matrix.

Definition 2 *Let $\boldsymbol{P} = [p_{ij}]$ be the transition probability matrix of a DTMC with state space $S = \{1, \ldots, \ell\}$. Then $\boldsymbol{P}$ is said to be IFR if*

$$\eta_m(i) := \sum_{j=m}^{\ell} p_{ij} \tag{11}$$

is nondecreasing in $i \in S$ for all $m = 1, \ldots, \ell$.

Now, let us define the tail distribution function

$$q(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) := \mathbb{P}\left(\boldsymbol{S}_{m+1} \geq \boldsymbol{s}'|\boldsymbol{S}_m = \boldsymbol{s}, \boldsymbol{A}_m = \boldsymbol{a}\right). \tag{12}$$

The quantity in (12) represents the probability that the MDP transitions to a state at least as large as $\boldsymbol{s}'$, given a starting state $\boldsymbol{s}$ and action $\boldsymbol{a}$. For notational convenience, also define the following sets:

$$\mathcal{I}(\boldsymbol{x}') := \{i \in \{1, \ldots, n\} : x'_i = 0\}, \tag{13}$$

$$\mathcal{J}(\boldsymbol{a}) := \{i \in \{1, \ldots, n\} : a_i = 1\}, \tag{14}$$

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') := \{i \in \{1, \ldots, n\} : x_i < x'_i\}. \tag{15}$$

These sets are used to describe the form of $q(\cdot|\cdot)$, namely,

$$q((\boldsymbol{x}', k')|(\boldsymbol{x}, k), \boldsymbol{a}) = \begin{cases} \exp\left[-q \cdot \max\limits_{v \in \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')} \left\{\dfrac{x'_v - x_v}{r_v^k}\right\}\right] \displaystyle\sum_{m=k'}^{\ell} p_{km}, & \mathcal{J} \subseteq \mathcal{I},\ \mathcal{K} \neq \emptyset, \\ \displaystyle\sum_{m=k'}^{\ell} p_{km}, & \mathcal{J} \subseteq \mathcal{I},\ \mathcal{K} = \emptyset, \\ 0, & \mathcal{J} \nsubseteq \mathcal{I}. \end{cases} \tag{16}$$

The tail distribution is useful for establishing monotonicity of the value function (3) and the corresponding optimal decisions. We begin by describing some useful properties of the tail distribution in Lemma 4.

Lemma 4 *If $\boldsymbol{P}$ is IFR, and the degradation rates r_i^j are monotone nondecreasing in $j \in S$ for each $i = 1, \ldots, n$, then the tail distribution function $q(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ is*

(a) monotone nondecreasing in $\boldsymbol{s} \in \Gamma$, and

(b) submodular on $\Gamma \times \mathcal{A}$.

Proof. Let $\tilde{\boldsymbol{s}} = (\tilde{\boldsymbol{x}}, \tilde{j}) \in \Gamma$, $\boldsymbol{a} \in A$, and $\boldsymbol{s} = (\boldsymbol{x}, j) \leq \boldsymbol{s}' = (\boldsymbol{x}', j') \in \Gamma$. In the case where $\mathcal{J} \nsubseteq \mathcal{I}$, we have $q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}) = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}) = 0$. Because $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ is decreasing in $\boldsymbol{x}$, there are three subcases for $\mathcal{J} \subseteq \mathcal{I}$: (i) $\mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \neq \emptyset$ and $\mathcal{K}(\boldsymbol{x}', \tilde{\boldsymbol{x}}) = \emptyset$, (ii) $\mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \neq \emptyset$ and $\mathcal{K}(\boldsymbol{x}', \tilde{\boldsymbol{x}}) \neq \emptyset$, and (iii) $\mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \emptyset$ and $\mathcal{K}(\boldsymbol{x}', \tilde{\boldsymbol{x}}) = \emptyset$. For subcase (i),

$$q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}) = \sum_{m=\tilde{k}}^{\ell} p_{j'm} \geq \exp\left(-q \cdot \max_{v \in \mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left\{\frac{\tilde{x}_v - x_v}{r_v^j}\right\}\right) \sum_{m=\tilde{k}}^{\ell} p_{jm} = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}).$$

For subcase (ii),

$$q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}) = \sum_{m=\tilde{k}}^{\ell} p_{j'm} \geq \sum_{m=\tilde{k}}^{\ell} p_{jm} = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}),$$

because $\boldsymbol{P}$ is IFR. Lastly, for subcase (iii), note by monotonicity of $\boldsymbol{r_i}$, for each $i \in \{1, 2, \ldots, n\}$,

$$\frac{\tilde{x}_v - x'_v}{r_v^{j'}} \leq \frac{\tilde{x}_v - x_v}{r_v^{j}}.$$

Additionally, by $\mathcal{K}(\boldsymbol{x}', \tilde{\boldsymbol{x}}) \subseteq \mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$,

$$\max_{v \in \mathcal{K}(\boldsymbol{x}', \tilde{\boldsymbol{x}})} \left\{ \frac{\tilde{x}_v - x'_v}{r_v^{j'}} \right\} \leq \max_{v \in \mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left\{ \frac{\tilde{x}_v - x_v}{r_v^{j}} \right\}.$$

Therefore, by the monotonicity of $\exp(\cdot)$ and the increasing failure rate of $\boldsymbol{P}$ we have

$$\begin{aligned} q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}) &= \exp\left(-q \cdot \max_{v \in \mathcal{K}(\boldsymbol{x}', \tilde{\boldsymbol{x}})} \left\{ \frac{\tilde{x}_v - x'_v}{r_v^{j'}} \right\} \right) \sum_{m=\tilde{k}}^{\ell} p_{j'm} \\ &\geq \exp\left(-q \cdot \max_{v \in \mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left\{ \frac{\tilde{x}_v - x_v}{r_v^{j}} \right\} \right) \sum_{m=\tilde{k}}^{\ell} p_{jm} = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}). \end{aligned}$$

Thus the tail function is monotone in $\boldsymbol{s}$. For Lemma 4(b), we seek to show

$$q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}') + q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}) \leq q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}') + q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}) \tag{17}$$

for $\boldsymbol{a} \leq \boldsymbol{a}' \in A$. Noting $\mathcal{J}(\boldsymbol{a}) \subseteq \mathcal{J}(\boldsymbol{a}')$, we have three cases to consider: (i) $\mathcal{J}(\boldsymbol{a}), \mathcal{J}(\boldsymbol{a}') \nsubseteq \mathcal{I}$, (ii) $\mathcal{J}(\boldsymbol{a}) \subseteq \mathcal{I}$, $\mathcal{J}(\boldsymbol{a}') \nsubseteq \mathcal{I}$, and (iii) $\mathcal{J}(\boldsymbol{a}), \mathcal{J}(\boldsymbol{a}') \subseteq \mathcal{I}$. For case (i), it is clear that both sides of (17) equal 0. For case (ii), $q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}') = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}') = 0$, reducing inequality (17) to

$$q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}) \leq q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}),$$

which holds by Lemma 4(a). For case (iii), we note $q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}) = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}, \boldsymbol{a}')$ and $q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}) = q(\tilde{\boldsymbol{s}}|\boldsymbol{s}', \boldsymbol{a}')$. Hence, inequality (17) holds in equality. Thus, the tail function is submodular. ∎

Next, we review some concepts from stochastic ordering that are needed for the remaining results. The first is that of upper orthant ordering of random vectors. For two n-dimensional, random vectors $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$, we say that $\boldsymbol{Y}_1$ is less than $\boldsymbol{Y}_2$ in the upper orthant order, denoted $\boldsymbol{Y}_1 \leq_{uo} \boldsymbol{Y}_2$, if for all $y \in \mathbb{R}^n$,

$$\mathbb{P}(\boldsymbol{Y}_1 \geq y) \leq \mathbb{P}(\boldsymbol{Y}_2 \geq y).$$

A set U is said to be an upper set if $u_2 \in U$ whenever $u_2 \geq u_1$ and $u_1 \in U$. We say that $\boldsymbol{Y}_1$ is less than $\boldsymbol{Y}_2$ in the usual stochastic order, denoted $\boldsymbol{Y}_1 \leq_{st} \boldsymbol{Y}_2$, if for all upper sets $U \subseteq \mathbb{R}^n$,

$$\mathbb{P}(\boldsymbol{Y}_1 \in U) \leq \mathbb{P}(\boldsymbol{Y}_2 \in U).$$

It is important to note that monotonicity of the tail function, established in Lemma 4(a), is equivalent to the condition

$$[\boldsymbol{S}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}, \boldsymbol{A}_m = \boldsymbol{a}] \leq_{uo} [\boldsymbol{S}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}', \boldsymbol{A}_m = \boldsymbol{a}], \quad \text{for any } \boldsymbol{s} \leq \boldsymbol{s}'.$$

If $n = 1$, the upper orthant and the usual stochastic order are equivalent. If $n > 1$, the usual stochastic order is stronger than the upper orthant order. These concepts are useful in that they allow us to extend part Lemma 4(a) to the usual stochastic order as seen in Lemma 5.

Lemma 5 *Under the conditions stated in Lemma 4, for any $a \in \mathcal{A}$ and $\boldsymbol{s} \leq \boldsymbol{s}' \in \Gamma$,*

$$[\boldsymbol{S}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}, \boldsymbol{A}_m = \boldsymbol{a}] \leq_{st} [\boldsymbol{S}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}', \boldsymbol{A}_m = \boldsymbol{a}].$$

Proof. Let $\boldsymbol{S}_{m+1} = (\boldsymbol{X}_{m+1}, \boldsymbol{Z}_{m+1})$. By independence of $\boldsymbol{X}_{m+1}$ and $\boldsymbol{Z}_{m+1}$ it suffices to show:

$$[\boldsymbol{X}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}, \boldsymbol{A}_m = \boldsymbol{a}] \leq_{st} [\boldsymbol{X}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}', \boldsymbol{A}_m = \boldsymbol{a}], \tag{18}$$

and

$$[\boldsymbol{Z}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}, \boldsymbol{A}_m = \boldsymbol{a}] \leq_{st} [\boldsymbol{Z}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}', \boldsymbol{A}_m = \boldsymbol{a}]. \tag{19}$$

Inequality (19) follows from the IFR assumption of $\boldsymbol{P}$. For $t \geq 0$, define the functions $\psi_1(t)$ and $\psi_2(t)$ by

$$\psi_1(t) = [(1 - a_1)(x_1 + r_1^j t), (1 - a_2)(x_2 + r_2^j t), \ldots, (1 - a_n)(x_n + r_n^j t)],$$

and

$$\psi_2(t) = [(1 - a_1)(x_1' + r_1^{j'} t), (1 - a_2)(x_2' + r_2^{j'} t), \ldots, (1 - a_n)(x_n' + r_n^{j'} t)].$$

By the monotonicity of the degradation rates r_i^j, and the fact that, $\boldsymbol{x} \leq \boldsymbol{x}'$, we see that $\psi_1(t) \leq \psi_2(t)$ for all $t \geq 0$. Letting $T \sim \text{Exp}(q)$, we have

$$\psi_1(T) \stackrel{\text{d}}{=} [\boldsymbol{X}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}, \boldsymbol{A}_m = \boldsymbol{a}] \quad \text{and} \quad \psi_2(T) \stackrel{\text{d}}{=} [\boldsymbol{X}_{m+1}|\boldsymbol{S}_m = \boldsymbol{s}', \boldsymbol{A}_m = \boldsymbol{a}],$$

where $\stackrel{\text{d}}{=}$ denotes equality in distribution. The result $\mathbb{P}(\psi_1(T) \leq \psi_2(T)) = 1$ implies inequality (18) holds, and the proof is complete. ∎

Next, in Proposition 1, we state without proof a well-known result about the comparability of expectations for random vectors under the usual multivariate stochastic order.

Proposition 1 *(Shaked and Shanthikumar [29]). Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two n-dimensional, random vectors such that $\boldsymbol{X} \leq_{st} \boldsymbol{Y}$. For any nondecreasing function ψ on $\mathbb{R}^n$*

$$\mathbb{E}[\psi(\boldsymbol{X})] \leq \mathbb{E}[\psi(\boldsymbol{Y})].$$

We are now prepared to state our main result concerning the value function. Namely, under appropriate conditions the value function $V(\boldsymbol{s})$ is monotone in $\boldsymbol{s} \in \Gamma$.

Theorem 3 *If $\boldsymbol{P}$ is IFR and the degradation rates r_i^j are monotone nondecreasing in $j \in S$ for each $i = 1, \ldots, n$, then the value function $V(\boldsymbol{s})$ is monotone nondecreasing in $\boldsymbol{s} \in \Gamma$.*

Proof. For $(\boldsymbol{x}, j) \in \Gamma$, denote the mth iterate of the value iteration algorithm by $v_j^m(\boldsymbol{x}) \equiv v^m(\boldsymbol{s})$. We prove the theorem by induction on m. Take $v_j^0(\boldsymbol{x}) = 0$ for all $(\boldsymbol{x}, j) \in \Gamma$. Therefore, as in the proof of Theorem 2, $v_j^1(\boldsymbol{x}) = c_d \sum_{i=1}^n \mathbb{I}_i(\boldsymbol{x})$, which is monotone nondecreasing in $\boldsymbol{s}$. For the induction hypothesis, assume $v^m(\boldsymbol{s})$ is monotone nondecreasing in $\boldsymbol{s} \in \Gamma$. By Lemma 5 and Proposition 1, for $\boldsymbol{s} \leq \boldsymbol{s}' \in \Gamma$,

$$\mathbb{E}\left(v^m(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}, \boldsymbol{A}_0 = \boldsymbol{a}\right) \leq \mathbb{E}\left(v^m(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}', \boldsymbol{A}_0 = \boldsymbol{a}\right).$$

By Lemma 3, $c(\boldsymbol{s}, \boldsymbol{a})$ is monotone in $\boldsymbol{s}$; hence,

$$c(\boldsymbol{s}, \boldsymbol{a}) + \alpha\mathbb{E}\left(v^m(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}, \boldsymbol{A}_0 = \boldsymbol{a}\right) \leq c(\boldsymbol{s}', \boldsymbol{a}) + \alpha\mathbb{E}\left(v^m(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}', \boldsymbol{A}_0 = \boldsymbol{a}\right). \tag{20}$$

By minimizing over $\boldsymbol{a} \in \mathcal{A}$ on both sides of inequality (20), we obtain $v^{m+1}(\boldsymbol{s}) \leq v^{m+1}(\boldsymbol{s}')$. Thus, by Lemma 1, the result is proved. ∎

Theorem 3 asserts that, if the uniformized environment process is IFR, and the degradation rates are monotone, then the values of the value function are ordered over the entire state space.

3.2 Special Case: A Single System

In general, the optimal policy is not necessarily monotone over the entire state space. Indeed, in Section 5, we provide two numerical examples illustrating that, even under the conditions of Lemma 4, optimal policies need not be monotone. In this subsection, we restrict our attention to the special case of a single system and establish the monotonicity of optimal policies for this case. Before proceeding to the main result, Lemma 6 is first needed.

Lemma 6 *Suppose there is only a single system (i.e., $n = 1$). If $\boldsymbol{P}$ is IFR, and the degradation rates r^j are monotone nondecreasing in $j \in S$, then $\mathbb{E}(V(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}, A_0 = a)$ is submodular on $\Gamma \times \mathcal{A}$.*

Proof. By Lemma 4(b), the tail distribution function $q(\cdot|\cdot)$ is submodular on $\Gamma \times \mathcal{A}$. Define $F(\boldsymbol{s}|\boldsymbol{s}_0, a) = \mathbb{P}(\boldsymbol{S}_1 \leq \boldsymbol{s}|\boldsymbol{S}_0 = \boldsymbol{s}_0, A_0 = a)$. Then for any $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \Gamma$ and $a_1, a_2 \in \mathcal{A}$ such that $\boldsymbol{s}_1 \leq \boldsymbol{s}_2$ and $a_1 \leq a_2$,

$$\int_{\boldsymbol{s} \geq \boldsymbol{s}'} dF(\boldsymbol{s}|\boldsymbol{s}_2, a_2) + \int_{\boldsymbol{s} \geq \boldsymbol{s}'} dF(\boldsymbol{s}|\boldsymbol{s}_1, a_1) \leq \int_{\boldsymbol{s} \geq \boldsymbol{s}'} dF(\boldsymbol{s}|\boldsymbol{s}_1, a_2) + \int_{\boldsymbol{s} \geq \boldsymbol{s}'} dF(\boldsymbol{s}|\boldsymbol{s}_2, a_1)$$

for each $\boldsymbol{s}' \in \Gamma$. Thus,

$$\int_{\boldsymbol{s} \geq \boldsymbol{s}'} \left[dF(\boldsymbol{s}|\boldsymbol{s}_2, a_2) + dF(\boldsymbol{s}|\boldsymbol{s}_1, a_1)\right] \leq \int_{\boldsymbol{s} \geq \boldsymbol{s}'} \left[dF(\boldsymbol{s}|\boldsymbol{s}_1, a_2) + dF(\boldsymbol{s}|\boldsymbol{s}_2, a_1)\right].$$

Let us define $dF_1(\boldsymbol{s}) = dF(\boldsymbol{s}|\boldsymbol{s}_2, a_2) + dF(\boldsymbol{s}|\boldsymbol{s}_1, a_1)$ and $dF_2(\boldsymbol{s}) = dF(\boldsymbol{s}|\boldsymbol{s}_1, a_2) + dF(\boldsymbol{s}|\boldsymbol{s}_2, a_1)$. Noting that $V(\boldsymbol{s})$ is monotone nondecreasing (by Theorem 3), and that Proposition 1 extends to finite measures, we obtain

$$\int_{\boldsymbol{s} \in \Gamma} V(\boldsymbol{s}) dF_1(\boldsymbol{s}) \leq \int_{\boldsymbol{s} \in \Gamma} V(\boldsymbol{s}) dF_2(\boldsymbol{s}). \tag{21}$$

By expanding the terms on both sides of inequality (21), the result is obtained. ∎

We are now prepared to state the main result for a single system. Theorem 4 asserts that, under suitable conditions, the optimal decision rule is monotone nondecreasing over the entire state space Γ.

Theorem 4 *For the case of $n = 1$, if $c_r - c_p \leq c_d$, $\boldsymbol{P}$ is IFR, and the degradation rates r^j are monotone nondecreasing in $j \in S$, the optimal decision rule $d^*(\boldsymbol{s})$ is monotone in $\boldsymbol{s} \in \Gamma$.*

Proof. By Lemma 3(b), the expected one-step cost function $c(\boldsymbol{s}, \boldsymbol{a})$ is submodular on $\Gamma \times \mathcal{A}$. Similarly, by Lemma 6, we have that $\mathbb{E}(V(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}, A_0 = a)$ is submodular. Thus,

$$Q(\boldsymbol{s}, a) = c(\boldsymbol{s}, a) + \alpha \mathbb{E}(V(\boldsymbol{S}_1)|\boldsymbol{S}_0 = \boldsymbol{s}, A_0 = a)$$

is a non-negative, linear combination of submodular functions and is, therefore, submodular. Hence, by Theorem 1, the optimal decision rule $d^*(\boldsymbol{s})$ is monotone in $\boldsymbol{s} \in \Gamma$. ∎

Theorem 4 ensures that, under suitable conditions, threshold policies are optimal, and these thresholds are monotone over the environment states. Ulukus et al. [36] examined a similar model and proved the optimality of threshold policies for a single-system (or component) setting and conjectured (without proof) that the thresholds are monotone over the environment states. Theorem 4 is significant in that it provides at least a partial resolution to this conjecture. Specifically, setting $c_0 = 0$ and $c_d \gg c_r$ in our framework yields a model that is identical to the one studied in [36], except that reactive replacements do not occur immediately following a failure; they occur at the start of the subsequent period. However, the assumption that c_d is very large induces an optimal policy that forces reactive replacements; hence, our model is not more restrictive than the one studied in [36].

The actions satisfying the optimality equation (3) for each state (corresponding to an optimal policy) cannot be directly computed by standard numerical methods (e.g. using the value or policy iteration algorithms [27]). Section 4 presents an approximation scheme to address the numerical problems associated with obtaining replacement policies.

4 An Approximate Formulation

In this section, we address dual manifestations of Bellman's curses of dimensionality, namely a continuous state space and a combinatorially large action space [2, 26]. To obtain high-quality policies, we customize and employ multiple reinforcement learning techniques: Q-function approximation, reduction to a subset of the action space, and an on-policy approximate dynamic programming (ADP) algorithm, namely the state action reward state action, or SARSA(γ), algorithm. For a complete description of the algorithm, the reader is referred to [28, 33].

Before discussing the details of our approximation scheme, we first introduce a transformation of the state description. In lieu of considering the current cumulative degradation level directly, we

consider the state to be a vector of probabilities on the hypercube $[0,1]^n$. In particular, define the state by $\tilde{\boldsymbol{x}} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$, where

$$\tilde{x}_i = \mathbb{P}\left(X^i_{m+1} \geq \xi | X^i_m = x_i, Z_m = j\right) = \exp\left(\frac{-q(\xi - x_i)}{r^j_i}\right), \tag{22}$$

Z_m is the current environment state, X^i_m is the current degradation level of system i, and X^i_{m+1} is the degradation level at the time of the next inspection. That is, for each system i, the corresponding system in the transformed state vector, denoted by $\tilde{x}_i$, is the probability that the system will fail before the next decision epoch. We pause here to remark on two aspects of this transformation: (i) no information is lost in the transformation (given fixed model parameters, the mapping is a bijection); and (ii) the new state implicitly contains information about the underlying degradation process. The first point is important, as we do not fundamentally change the problem. The second point is of practical significance for our function approximation. We utilize a linear basis to approximate the Q-function, and direct inclusion of this information improves the performance of simple basis functions. Intuitively, the degradation level itself is abstract and uninformative, whereas probabilities require no additional knowledge of the underlying degradation process to interpret; hence, this state space is more appropriate. For convenience, we denote the modified state as $\tilde{\boldsymbol{s}}$ and state space as $\tilde{\Gamma} = [0,1]^n \times S$.

The first issue we address is reduction of the action space $\mathcal{A}$. It is clear that for a large number of systems, n, the action space $\mathcal{A} = \{0,1\}^n$ is too large to perform any practical computational operations. For example, for $n = 20$ the cardinality of the action space is over 10^6 and for $n = 30$ the cardinality is over one billion. Unfortunately, it is very difficult to characterize the form of an optimal policy in the multiple system case, so instead we consider a reduced action space based on the transformed state vector $\tilde{\boldsymbol{x}}$ that may, or may not, contain the optimal actions. We define the new action space as

$$\tilde{\mathcal{A}} = \{0, 1, \ldots, n\},$$

where $a = 0$ corresponds to taking no action, and $a = k$ corresponds to replacing all systems i such that $\tilde{x}_i \geq \tilde{x}_k$, for $k = 1, 2 \ldots, n$. The action $a = k > 0$ means that any system whose probability of failing in the next period is at least as high as that of system k is replaced. For example, if $\tilde{\boldsymbol{x}} = (0.3, 0.7, 0.25, 0.6)$ and action $a = 4$ is selected, then all systems whose probability of failing within the next period is at least 0.6 are replaced. In this case, systems 2 and 4 are replaced. It should be noted that this action space can be interpreted as the most general set of threshold policies in the transformed state. That is, for every state $\tilde{\boldsymbol{s}} \in \tilde{\Gamma}$ the optimal action in $\tilde{\mathcal{A}}$ is equivalent to replacing all systems above some optimal threshold $u(\tilde{\boldsymbol{s}})$. Most importantly, this set helps facilitate computational tractability, as it is linear in the number of systems.

The next issue we address is the continuous nature of $\mathcal{X}$. In order to overcome this difficulty, we employ function approximation techniques. First, we recall the definition of the Q-function, or action-value function [26] from equation (5). For each $j \in S$, the function $Q_j(\cdot)$ maps state-action

pairs to expected total discounted costs, i.e.,

$$Q_j(\boldsymbol{x}, \boldsymbol{a}) = c(\boldsymbol{s}, \boldsymbol{a}) + \alpha \sum_{k=1}^{\ell} \left(\int_0^{\infty} V_k\left(\boldsymbol{x}'\right) q e^{-qt} dt \right) p_{kj}.$$

It is seen that, by the relationship between (3) and (5), optimal actions can be determined by minimizing the Q-function over the original action space $\mathcal{A}$. We seek to approximate the Q-function as a weighted (linear) sum of basis functions that take the transformed state variables as input [13]. In particular, we approximate $Q_j(\boldsymbol{x}, \boldsymbol{a})$ by $\tilde{Q}_j(\tilde{\boldsymbol{x}}, a)$ defined as

$$\tilde{Q}_j(\tilde{\boldsymbol{x}}, a) := \sum_{k=1}^{K} \lambda_k^j \phi_k^j(\tilde{\boldsymbol{x}}, a), \tag{23}$$

where λ_k^j are real-valued coefficients and $\phi_k^j(\cdot)$ are basis functions. By Theorem 2, it is known that the value function is monotone in the state space; hence, we choose to represent the Q-function as a linear combination of monotone functions and constant functions. In particular, we utilize constant functions along with functions of the form $\cos(f(\tilde{\boldsymbol{x}}, a))$, where $f : \tilde{\Gamma} \to [0, 1]$. This choice of basis functions is practical, as they are bounded and monotone on $\tilde{\Gamma}$. Additionally, the non-constant basis functions are sigmoidal, enabling flexible modeling of nonlinearity in the value function, and appropriate choices for f allow for dependencies between systems to be captured. As described in [20], these basis functions can be viewed as adapted first-order Fourier basis functions, and we choose the particular form

$$\begin{aligned} \tilde{Q}_j(\tilde{\boldsymbol{x}}, a) = \lambda_0^j + \lambda_1^j \mathbb{I}\left(a > 0\right) + \sum_{i=1}^{n} \lambda_{i,1}^j \mathbb{I}\left(\tilde{x}_i \geq \tilde{x}_a\right) + \sum_{i=1}^{n} \lambda_{i,1}^j \mathbb{I}\left(\tilde{x}_i = 1, a > 0\right) \\ + \sum_{i=1}^{n} \sum_{k=i}^{n} \lambda_{i,k,1}^j \phi_{i,k,1}^j\left(\tilde{x}, a\right) + \sum_{i=1}^{n} \sum_{k=i+1}^{n} \lambda_{i,k,2}^j \phi_{i,k,2}^j\left(\tilde{x}, a\right), \end{aligned} \tag{24}$$

where

$$\phi_{i,k,m}^j = \begin{cases} \cos\left(\frac{\pi}{2} \cdot \left(\mathbb{I}\left(\tilde{x}_i < \tilde{x}_a\right) \tilde{x}_i + \mathbb{I}\left(\tilde{x}_k < \tilde{x}_a\right) \tilde{x}_k\right)\right), & m = 1, \\ \cos\left(\frac{\pi}{2} \cdot \mathbb{I}\left(\min\left\{\tilde{x}_i, \tilde{x}_k\right\} < \tilde{x}_a\right) \cdot \left(\tilde{x}_i + \tilde{x}_k\right)\right), & m = 2. \end{cases} \tag{25}$$

The first term in (24) is a constant, the second term accounts for the impact of the setup cost, the third term accounts for preventive maintenance cost, the fourth for reactive maintenance, and the final two terms account for the impact of the degradation level (given the current action taken). It can be seen in (25) that the approximation $\tilde{Q}_j(\cdot)$ considers only pairwise dependencies between systems. This assumption can be relaxed, but the computation time of any iterative learning process would naturally be adversely impacted. For the model in (24), the coefficients λ_k^j can be obtained using an iterative, on-policy learning algorithm, such as the SARSA(γ) algorithm, as described in Sutton and Barto [33].

5 Numerical Examples

In this section, we illustrate the efficacy of the proposed approximation framework in obtaining high-quality solutions. Both small- and large-scale examples are considered. Small problem instances are illustrated because exact solutions can be obtained using standard MDP machinery; hence, approximate solutions can be compared to these exact solutions. Larger problem instances are used to assess how well our approximation scheme performs on problems that are otherwise intractable.

Because our aim is to compare distinct policies, we first simulate the evolution of the environment process over a large number of sample paths. The simulation run length is given by the number of decision epochs N (recall that the time between decision epochs is exponentially distributed with rate parameter q). Along each sample path, the total discounted cost is computed for each policy of interest, and these values are compared. It should be noted that, because the expected one-step costs are bounded, and the cost function is discounted, we can determine *a priori* the simulation run length needed to ensure that the total discounted cost is accurate to a fixed constant. In particular, to guarantee the finite approximation is within ε ($\varepsilon > 0$) of the true total discounted cost, the length of the sample paths N must satisfy

$$N \geq \frac{\ln((1-\alpha)\varepsilon/C)}{\ln(\alpha)} - 1,$$

where C is any valid upper bound on the expected one-step costs, and α is the one-step discount rate. For all numerical examples, N was chosen to correspond to $\varepsilon = 0.01$ and C was taken to be $c_0 + nc_d + nc_r$. All problem instances were coded within the MATLAB R2016a computing environment and executed on a personal computer with a 3.50 GHz processor and 8 GB of RAM.

5.1 Single-System Problems

First, we present single-system examples to demonstrate the accuracy of our approach in the simplest case. We modify the cost structure of Section 2 to fit the single-system case. Rather than having four cost parameters, we simply have the preventive and reactive replacement costs, c_p and c_r respectively, and we force replacement of the system whenever it is found to be failed.

In the first example, we fix the parameter values and solve the resulting problem repeatedly using the approximate formulation. We compute multiple learned solutions over a single problem to explore the impact of a stochastic learning algorithm on the consistency of solutions. For the first instance, the discount rate is $\alpha = 0.99$ and the cost vector is $\boldsymbol{c} = (c_p, c_r) = (3, 10)$. The failure threshold is set to $\xi = 1$, and the environment $\mathcal{Z}$ has state space $S = \{1, 2, 3, 4\}$ with infinitesimal generator matrix

$$\mathcal{Q} = \begin{pmatrix} -5 & 5 & 0 & 0 \\ 2.5 & -5 & 2.5 & 0 \\ 0 & 2.5 & -5 & 2.5 \\ 0 & 0 & 5 & -5 \end{pmatrix}.$$

The inspection rate is $q = 10$ and degradation rates are $\boldsymbol{r} = (r_1, r_2, r_3, r_4) = (2.5, 3, 3.5, 4)$.

The degradation interval $[0, \xi]$ was discretized into $10,000$ states, and the optimal policy was obtained numerically using the value iteration algorithm. For each fixed environment state $j \in S$, the optimal policy was determined to be a threshold policy. This is intuitive, as the problem can viewed as the general problem with $c_0 = 0$ and c_d chosen sufficiently large so as to force reactive replacements. Therefore, Theorem 4 ensures that a monotone threshold policy is optimal. In particular, the thresholds were found to be $\boldsymbol{u} = (u_1, u_2, u_3, u_4) = (0.524, 0.469, 0.430, 0.387)$, where for state $(x, j) \in [0, 1] \times S$ the optimal action $a_j^*(x)$ is given by

$$a_j^*(x) = \begin{cases} 0, & x \leq u_j, \\ 1, & x > u_j. \end{cases}$$

To adapt to the modified cost structure, we simplified the basis functions for our approximation as follows:

$$\phi_j^i(\tilde{x}, a) = \begin{cases} 1, & i = 1, \\ \mathbb{I}(a = 0)\cos(\pi\tilde{x}), & i = 2, \\ \mathbb{I}(a = 1,\, \tilde{x} < 1), & i = 3, \\ \mathbb{I}(\tilde{x} \geq 1), & i = 4, \end{cases}$$

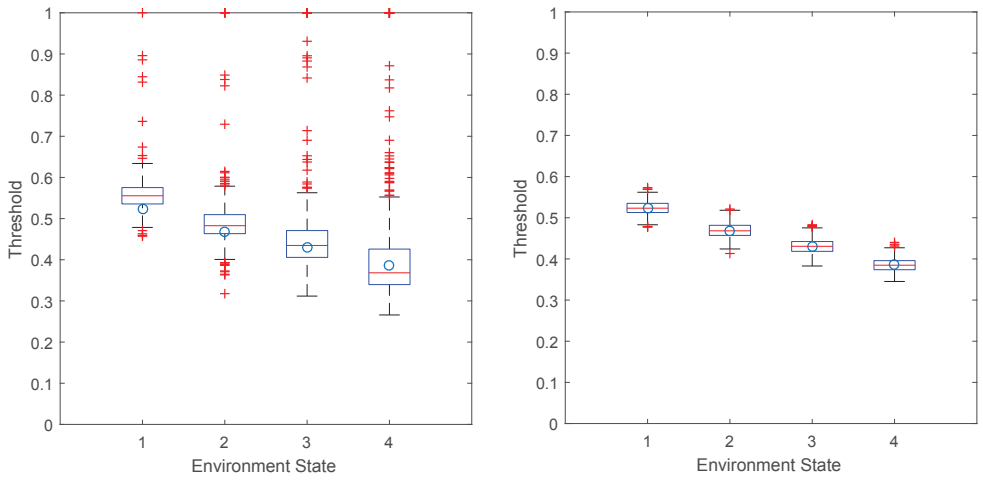
where $\tilde{x}$ is the transformed state as described by (22). Therefore, the expected total discounted cost-to-go, starting from state $\boldsymbol{s} = (x, j) \in \Gamma$ and taking action $a \in \mathcal{A}$, is approximated by

$$\tilde{Q}_j(\tilde{x}, a) = \sum_{i=1}^{4} \lambda_j^i \phi_j^i(\tilde{x}, a).$$

In order to test the quality of this approximation, 500 sample paths were simulated, each with $2,000,000$ decision epochs. On each of these sample paths, the approximation model was trained using a SARSA(γ) algorithm with $\gamma = 0.9$. Exploration is encouraged by using an ϵ-greedy action-selection approach, where the best myopic action is chosen with probability $1 - \epsilon$, and a random action is chosen with probability ϵ. For these examples, ϵ is initialized to 0.5 and linearly decreased in increments of 2.5×10^{-7} to 0.0001 by the final iteration.

Figure 1 shows box-and-whisker plots of the thresholds derived from the learned policies. In Figure 1(a), we see the thresholds from the policies if learning is terminated at 400,000 iterations. This plot indicates that the bias from parameter initialization is still significant in the learned policies. Generally, the thresholds are still too high, which matches intuition as the initial parameter values would induce a policy that never replaces components. By $2,000,000$ iterations, this upward bias is drastically diminished, and the variance has been greatly reduced. In fact, the mean absolute deviations from the true thresholds are under 0.015 for all environment states. The thresholds from value iteration and the mean learned thresholds are given in Table 1.

While these policies appear to be accurate, it is important to gauge the impact of small policy variations on the expected total discounted cost. To this end, we first simulate 5,000 sufficiently



(a) After 400,000 simulation iterations. (b) After 2,000,000 simulation iterations.

Figure 1: Boxplots of thresholds from learned policies (optimal thresholds given by circles).

Table 1: Comparison of exact and learned thresholds.

Technique	Environment State			
	$j = 1$	$j = 2$	$j = 3$	$j = 4$
Value iteration threshold	0.5238	0.4688	0.4301	0.3865
Mean learned threshold	0.5240	0.4701	0.4303	0.3850

long sample paths per policy (2,500,000 sample paths in total) as described at the beginning of this section. Each simulation run is initialized in state $(x, j) = (0, 1)$. For learned policy $m \in \{1, \ldots, 500\}$ and sample path $k \in \{1, \ldots, 5,000\}$, we then follow both the optimal policy and the learned policy over each of these sample paths to determine their respective total discounted costs, call them $\tilde{V}_k^m$ and $\hat{V}_k^m$, respectively. The sample-wise values are then averaged to yield approximate expected total discounted costs

$$\tilde{V}^m = \frac{1}{5000}\sum_{k=1}^{5000} \tilde{V}_k^m \quad \text{and} \quad \hat{V}^m = \frac{1}{5000}\sum_{k=1}^{5000} \hat{V}_k^m.$$

The results are summarized in the box-and-whisker plots of Figure 2. While the optimal policy does perform better on average (over the sample paths), the percent difference in the means is nominal at approximately

$$\frac{|\tilde{V} - \hat{V}|}{\tilde{V}} = 4.887 \times 10^{-4},$$

where $\tilde{V} = \sum_m \tilde{V}^m/500$ and $\hat{V} = \sum_m \hat{V}^m/500$. In fact, the approximate policy outperforms the optimal policy in 49.16% of the total sample paths. Therefore, the learned policies are not only superficially similar to the optimal policies, but almost identical in performance as well. Lastly, it should be noted that obtaining the exact solution took 80.5269 seconds (averaged over 100 function calls) and the average time for the approximate model to generate the simulations and train was

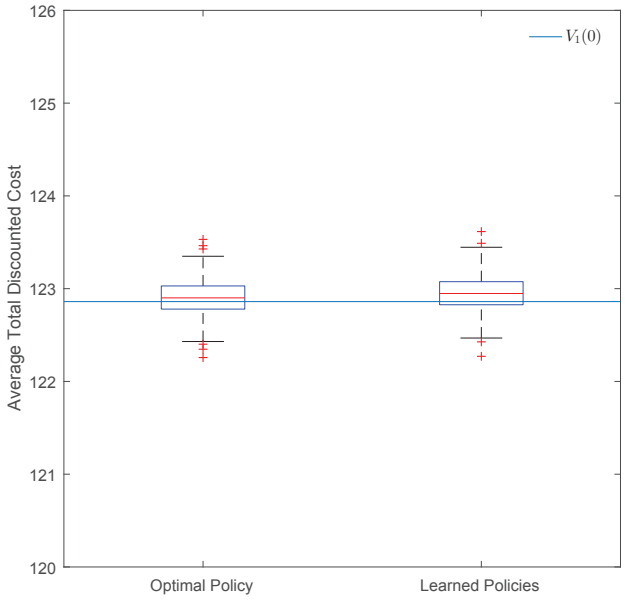


Figure 2: Boxplots comparing cost using the optimal policy versus the learned policies.

only 25.8465 (averaged over the 500 sample paths). Thus, even in the single-system case, this solution method is able to generate stable, accurate policies faster than value iteration.

In the next example, 200 single-system problems are randomly generated, and one solution to the approximate formulation is computed for each problem. Our aim is to vary the problem parameters over a wide range to assess the robustness of the approximate formulation. In what follows, $U(a, b)$ denotes a continuous uniform random variable on (a, b). Fixing $\alpha = 0.99$, and the number of environment states at $\ell = 4$, we randomly generated $M = 200$ problem instances. (These two parameters are fixed because they significantly affect the learning rate, and the training size will be fixed over the problem set.) For problem $m \in \{1, \ldots, M\}$, the cost vector is denoted $\boldsymbol{c}^m = (c_p^m, c_r^m)$, where $c_p^m \sim U(0, 5)$ and $c_r^m \sim c_p^m + U(0, 10)$. As before, the failure threshold is fixed at $\xi^m = 1$, corresponding to rates being normalized. The environment process $\mathcal{Z}^m$ has state space $S^m = \{1, 2, 3, 4\}$ with infinitesimal generator matrix $Q^m = [q_{ij}^m]$, where $q_{ij}^m \sim U(0, 10)$ for $j \neq i$ and $q_{ii}^m = -\sum_j q_{ij}^m$. The inspection rate is given as $q^m = 2 \cdot \max_i\{-q_{ii}^m\}$, and the degradation rates are $\boldsymbol{r}^m = (r_1^m, r_2^m, r_3^m, r_4^m)$, where $r_1^m \sim U(0, 4)$ and $r_j^m \sim r_{j-1}^m + U(0, 4)$ for $j = 2, 3, 4$.

For each problem, the degradation interval $[0, \xi]$ was discretized into $10,000$ states and the optimal policy was obtained numerically using the value iteration algorithm. In each case, a long sample path was simulated with 2,000,000 decision epochs, and the approximate model was trained using a SARSA(γ) algorithm with $\gamma = 0.9$. Exploration was again encouraged by using an ϵ-greedy action-selection approach, where ϵ was initialized to 0.5 and linearly decreased to 0.0001 by the final iteration. Next, to evaluate the quality of the solutions, 5,000 sufficiently long sample paths were simulated per problem using the initial state $(x, j) = (0, 1)$. For each problem m and sample path $k_m \in \{1, \ldots, 5,000\}$, the learned policy was followed over the path to determine the total discounted

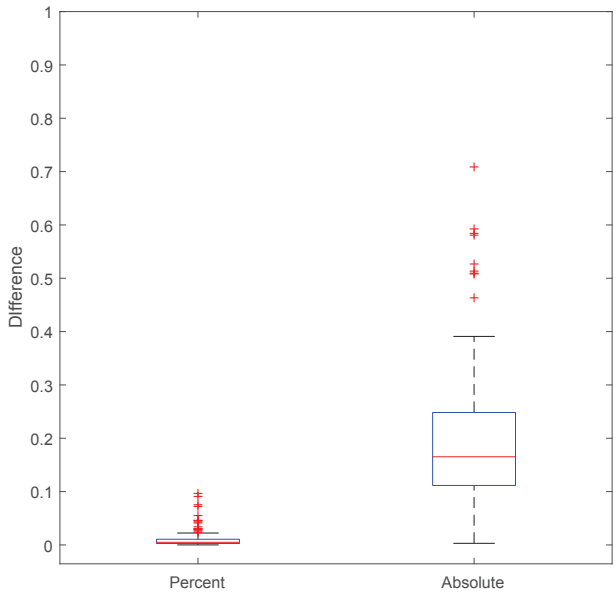


Figure 3: Boxplots of percent and absolute differences.

cost, denoted v_k^m. For problem m, we denote the value function, at state $(0,1) \in \Gamma$ by V^m, and assume that these are given by the final values from the value iteration algorithm. Similarly, for problem m, we denote the approximate value function, at state $(0,1) \in \Gamma$ by $\widehat{V}^m = \sum_k v_k^m/5000$. Lastly, we define $\delta^m = |V^m - \widehat{V}^m|$ and $\Delta^m = \delta^m/V^m$ to be, respectively, the absolute and percent differences between the actual and approximate value functions. From the data, we find that there is one significant outlier, where $\Delta^m = 0.38$. It is seen that at this data point, the actual value function is very small at $V^m = 1.0649$. We remove this outlier from the data set and present the data in Figure 3. For the percent differences, the sample mean of Δ^m is 0.0099 and the standard deviation is 0.0143. In fact, for approximately 97.5% of the cases $\Delta^m \leq 0.05$. For the absolute differences, we find that the sample mean of δ^m is 0.1908 and the standard deviation is 0.1170. In fact, for over 95% of the cases $\delta^m \leq 0.4$.

5.2 Multiple-System Problems

Next, we present multiple-system examples to illustrate solution quality and the tractability of our approach. We consider three problems with $n = 2$, $n = 3$ and $n = 50$, respectively. For cases $n = 2$ and $n = 3$, we obtain "exact" solutions by applying the value iteration algorithm on discretized versions of the problems. Our approximate solutions, obtained by using the methods developed in Section 4, are compared to these solutions. Additionally, we describe a set of heuristic policies and compare our solutions to those as well. When $n = 50$, exact solutions cannot be obtained, so our approximation is compared only to heuristic policies.

Example. For the first example, $n = 2$ systems, the discount rate is $\alpha = 0.99$ and the cost vector is $\boldsymbol{c} = (c_0, c_p, c_r, c_d) = (20, 3, 10, 8)$. The failure thresholds are $\xi_i = 1$, for all i, and the

environment $\mathcal{Z}$ has state space $S = \{1, 2, 3\}$ with infinitesimal generator matrix

$$\mathcal{Q} = \begin{pmatrix} -5 & 5 & 0 \\ 2.5 & -5 & 2.5 \\ 0 & 5 & -5 \end{pmatrix}.$$

The inspection rate is $q = 10$ and degradation rates were randomly generated as $\boldsymbol{r}_i = (r_i^1, r_i^2, r_i^3) = U(1, 2) \cdot (0.5, 1.0, 1.5)$, for $i = 1, 2$, where $U(1, 2)$ is a continuous uniform random variable on $(1, 2)$. The particular realization of $\boldsymbol{r}$ is

$$\boldsymbol{r} = \begin{pmatrix} 0.9567 & 1.9134 & 2.8701 \\ 0.5635 & 1.1270 & 1.6905 \end{pmatrix}.$$

To compute an exact solution, the degradation interval $[0, \xi]$ was uniformly discretized into 1,000 states for each system, yielding a discretized problem with 3,000,000 total states. The optimal policy was obtained numerically using the value iteration algorithm. For each $\boldsymbol{s}$ in the discretized set of states, let $v^m(\boldsymbol{s})$ denote the mth iterate of the value iteration algorithm. The algorithm terminates when the maximum norm of the difference between subsequent value function iterates is below 0.01, i.e.,

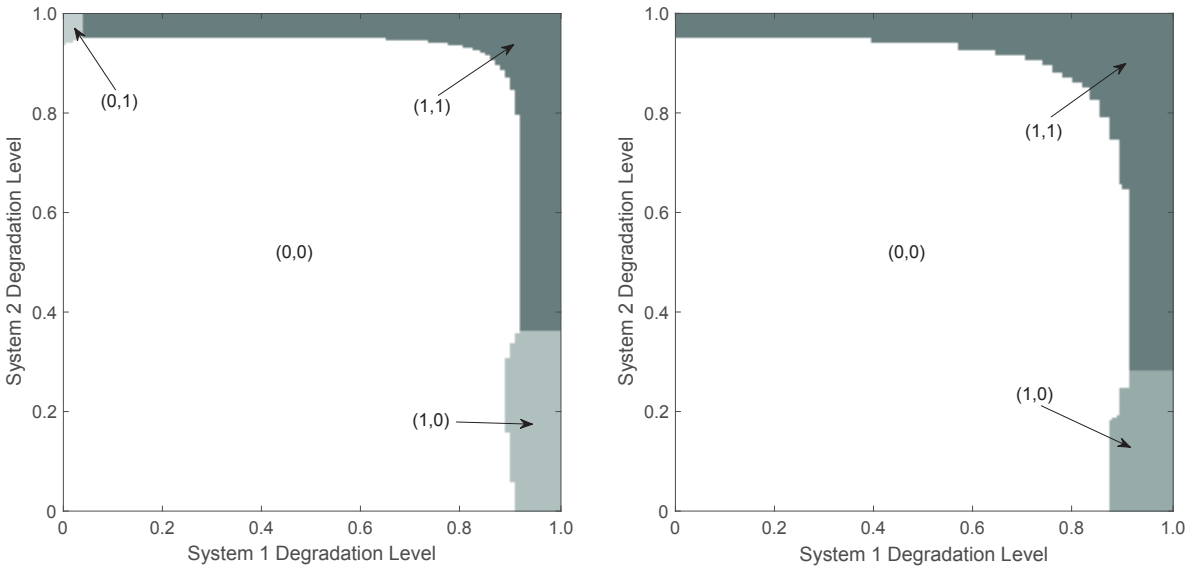
$$||v_{m+1} - v_m||_\infty = \max_{\boldsymbol{s}} \{|v_{m+1}(\boldsymbol{s}) - v_m(\boldsymbol{s})|\} \leq 0.01,$$

or after 604,800 seconds (7 days). The algorithm terminated after 401,029.97 seconds (approximately 4.6 days).

In order to train our approximate model, a long sample path was simulated with 4,000,000 decision epochs, and the approximate model was trained using a SARSA(γ) algorithm with $\gamma = 0.9$. Exploration was again encouraged by using an ϵ-greedy action-selection approach, where ϵ was initialized to 0.5 and linearly decreased to 0.0001 by the final iteration. The training took 369.54 seconds, three orders of magnitude faster than value iteration.

In Figure 4, the optimal and learned policies are depicted when the environment state is fixed at 1. We observe that the optimal policy does not have a monotone structure. While the absence of a monotone structure is counterintuitive, the overall structure is not. When one system is very heavily degraded, but the other system is nearly as-good-as-new, it is optimal to replace the heavily degraded system. However, due to economic dependencies, the region where replacing both systems is optimal dominates the replacement regions. This result should be expected, as the shared replacement cost is high and the difference between the preventive and reactive replacement costs is substantial. Additionally, it is optimal to wait longer to perform maintenance actions on system 2 as a consequence of the degradation rates. In particular, we note that $r_2^1 < r_1^1$. Contrasting the optimal and learned policies, we note that while similar, the learned policy is slightly more aggressive and replaces more frequently.

For a cost comparison, we also consider a set of threshold-based replacement policies. To understand these policies, we begin with the policy that ignores the dependency of the systems and



(a) Depiction of optimal policy.

(b) Depiction of learned policy.

Figure 4: Comparison between optimal and learned policy in environment state 1.

decouples the problem. For this policy, we consider each system separately and solve the single-system problem described in Section 5.1. We then use the computed thresholds to determine when to replace individual systems independently of the others. We refer to this as a 1-policy. We then generalize this policy to a k-policy, where we replace all systems above their independent threshold whenever there are at least k systems above their threshold. In the case where $n = 50$, two other similarly structured policies were considered, but failed to perform as well: (i) wait until at least k systems are failed and then replace all failed systems; and (ii) wait until at least k systems exceed their threshold and then replace all systems.

As in the single-system cases, we assess the quality of our solutions by following each policy over 5,000 sufficiently long sample paths simulated using the initial state $(x, j) = (\mathbf{0}, 1)$. The results of this comparison are summarized in the box-and-whisker plots of Figure 5.

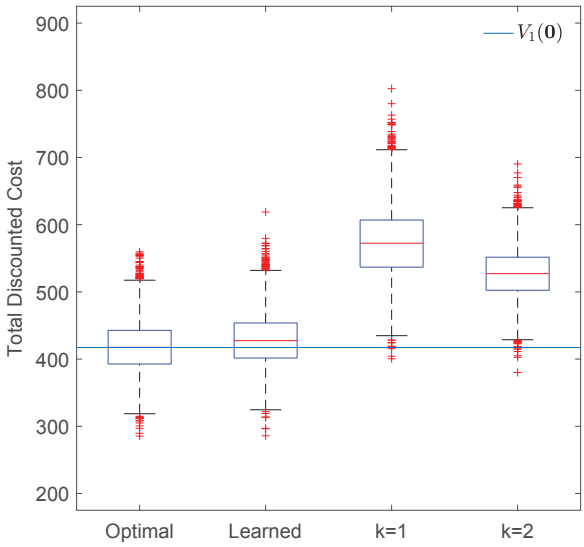


Figure 5: Boxplots comparing cost under different policies.

Beginning with the heuristic policies, we note that they are both significantly outperformed by the learned policy. On average, the learned policy outperforms the 1-policy by 32.79% and the 2-policy by 22.28%. Moreover, the learned policy outperforms the 1-policy on all sample paths and the 2-policy on all but three sample paths. This performance gap demonstrates the importance of considering structural and economic dependence within our model. Comparing the learned and optimal policies, we note the average total discounted cost for the learned policy was 431.32 compared to 418.74 for the optimal policy (approximately 3% difference). The learned policy even outperformed the optimal policy on 1,041 sample paths.

Example. For this example, $n = 3$ systems, the discount rate is $\alpha = 0.99$ and the cost vector is $\boldsymbol{c} = (c_0, c_p, c_r, c_d) = (20, 3, 10, 8)$. The failure thresholds are $\xi_i = 1$, for all i, and the environment $\mathcal{Z}$ has state space $S = \{1, 2\}$ with infinitesimal generator matrix

$$\mathcal{Q} = \begin{pmatrix} -5 & 5 \\ 5 & -5 \end{pmatrix}.$$

The inspection rate is $q = 10$ and degradation rates were randomly generated as $\boldsymbol{r}_i = (r_i^1, r_i^2) = U(1, 2) \cdot (0.5, 1.0)$. The particular realization of $\boldsymbol{r}$ is

$$\boldsymbol{r} = \begin{pmatrix} 0.9074 & 1.8147 \\ 0.9529 & 1.9058 \\ 0.5635 & 1.1270 \end{pmatrix}.$$

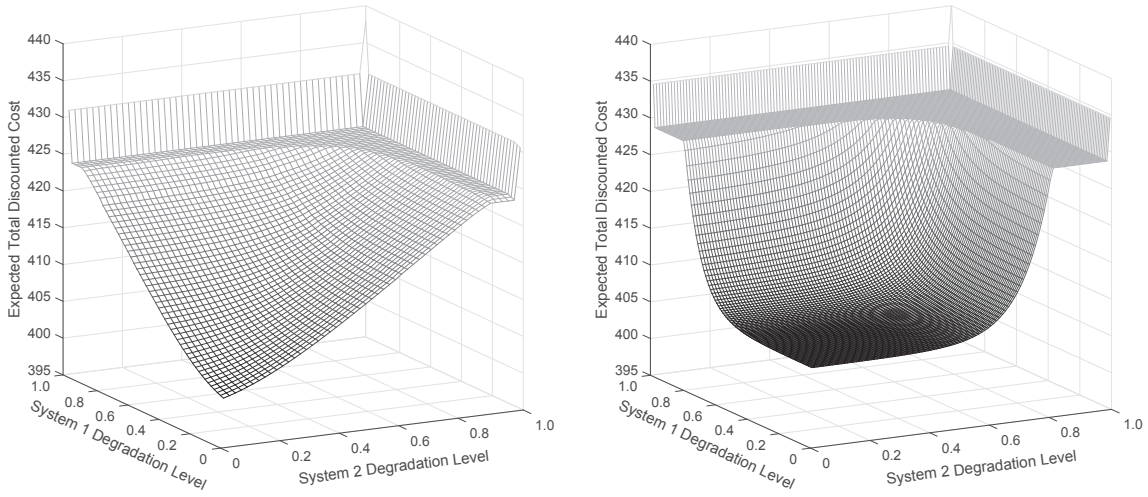
To compute an exact solution, the degradation interval $[0, \xi]$ was uniformly discretized into 125 states, for each system, yielding a discretized problem with 3,906,250 total states. The optimal policy was obtained numerically using the value iteration algorithm. For each $\boldsymbol{s}$ in the discretized set of states, let $v^m(\boldsymbol{s})$ denote the mth iterate of the value iteration algorithm. The algorithm

was set to terminate when the maximum norm of the difference between subsequent value function iterates fell below 0.01, i.e.,

$$||v_{m+1} - v_m||_\infty = \max_{\boldsymbol{s}} \{|v_{m+1}(\boldsymbol{s}) - v_m(\boldsymbol{s})|\} \leq 0.01,$$

or after 604,800 seconds (7 days). The algorithm terminated after reaching 604,800 seconds and the maximum norm of the difference between the final two value function iterates was 0.0253.

In order to train our approximate model, a long sample path was simulated with 4,000,000 decision epochs, and the approximate model was trained using a SARSA(γ) algorithm with $\gamma = 0.9$. Exploration was again encouraged by using an ϵ-greedy action-selection approach, where ϵ was initialized to 0.5 and linearly decreased to 0.0001 by the final iteration. The training took 482.48 seconds to complete.



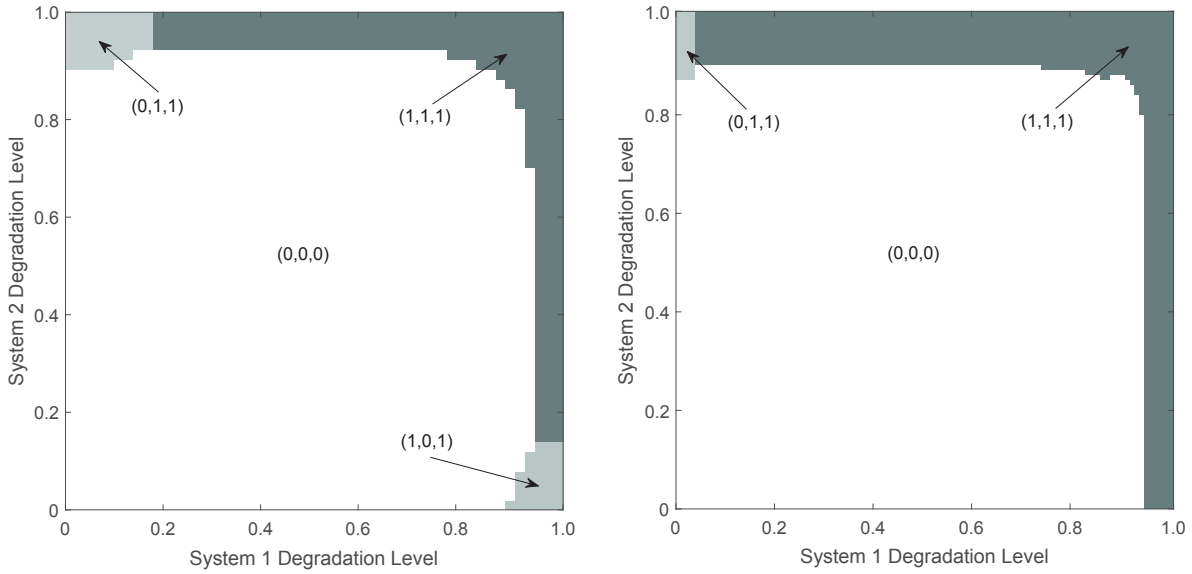
(a) Optimal value function.

(b) Learned value function.

Figure 6: Comparison between optimal and learned value function in environment state 2 (degradation level of system 3 is fixed at 0.40).

Figure 6 depicts the optimal and learned value functions when the environment is fixed at state 2 and the degradation level of system 3 is fixed at 0.40. At low degradation levels, in regions where it is optimal to wait to replace, the magnitude of the gradient of the value function approximation tends to be near zero. Otherwise, the shape and values of the function approximation tend to be very similar to the true optimal value function. These results are typical and were observed across all test cases.

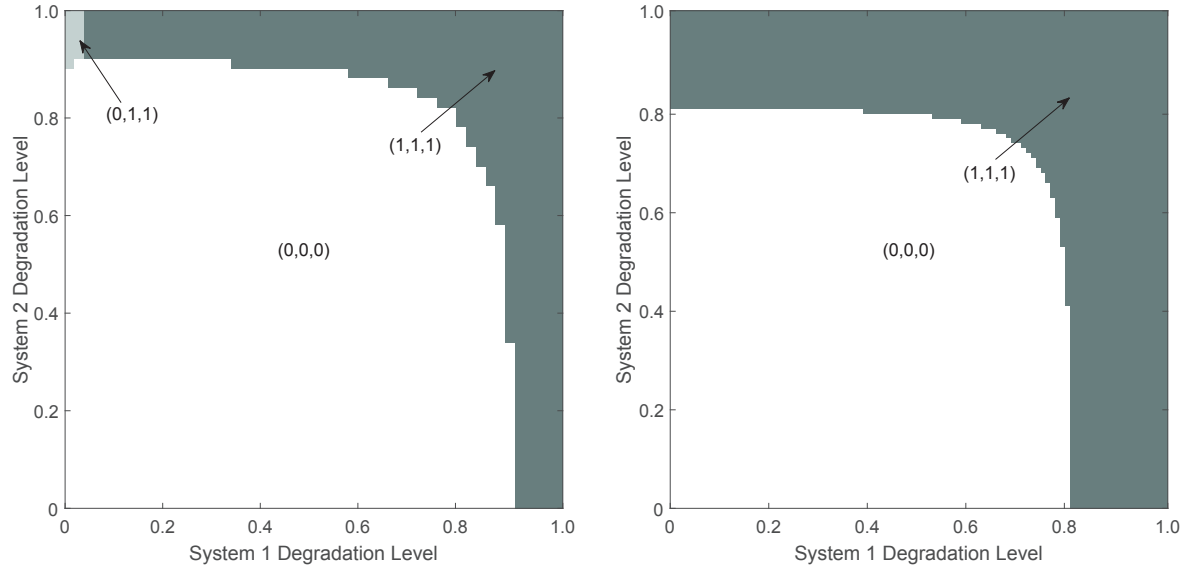
Figure 7 depicts the optimal and learned policies when the environment state is fixed at 1 and the degradation level of system 3 is fixed at 0.40. Again we observe that the optimal policy does not have a monotone structure. In contrast to the two-system case, we see that the policy is reasonably symmetric. This symmetry stems from the similarity between the degradation rates of the first two systems. In Figure 8, we compare the policies at environment state 2, noting that the regions



(a) Depiction of optimal policy.

(b) Depiction of learned policy.

Figure 7: Comparison between optimal and learned policy in environment state 1 (degradation level of system 3 is fixed at 0.40).



(a) Depiction of optimal policy.

(b) Depiction of learned policy.

Figure 8: Comparison between optimal and learned policy in environment state 2 (degradation level of system 3 is fixed at 0.40).

where replacing two systems have largely vanished, but the symmetry is still observed. In both environment states, over a majority of the degradation states, the learned policy calls for replacing at least as many systems as the optimal policy.

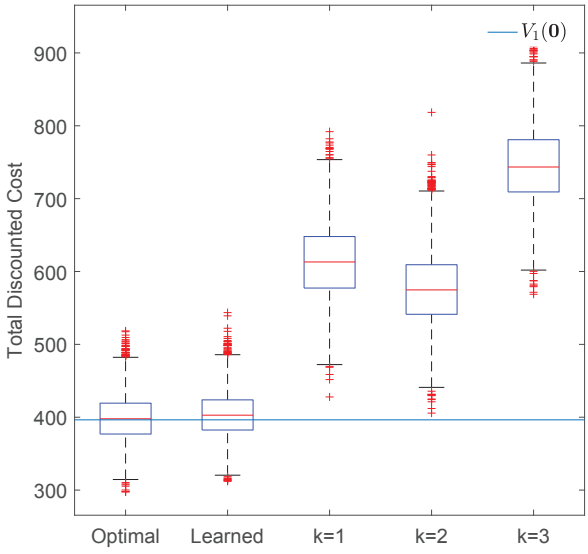


Figure 9: Boxplots comparing cost under different policies.

Again, we evaluate the quality of the solutions by following each policy over 5,000 sufficiently long sample paths simulated using the initial state $(x, j) = (\mathbf{0}, 1)$. The results of this comparison are summarized in the box-and-whisker plot in Figure 9.

The learned policy again outperforms the heuristic policies, in this case, on average it outperforms the 1-policy by 51.89%, the 2-policy by 42.66%, and the 3-policy by 84.61%. The learned policy outperforms all of the heuristic policies on every sample path. Comparing the learned and optimal policies, we note the average total discounted cost for the learned policy was 403.76 compared to 398.91 for the optimal policy (approximately 1.22% difference). The learned policy even outperformed the optimal policy on 1,854 sample paths (approximately 37%). Showing that for small $n > 1$ our approach is able to quickly determine near optimal policies.

Example. For this final example, we consider a larger problem for which finding an optimal solution by value iteration is computationally intractable. Here, there are $n = 50$ systems, the discount rate is $\alpha = 0.99$ and the cost vector is $\boldsymbol{c} = (c_0, c_p, c_r, c_d) = (100, 3, 10, 8)$. The failure thresholds are $\xi_i = 1$, for all i, and the environment $\mathcal{Z}$ has state space $S = \{1, 2, 3, 4\}$ with infinitesimal generator matrix

$$\mathcal{Q} = \begin{pmatrix} -5 & 5 & 0 & 0 \\ 2.5 & -5 & 2.5 & 0 \\ 0 & 2.5 & -5 & 2.5 \\ 0 & 0 & 5 & -5 \end{pmatrix}.$$

The inspection rate is $q = 10$ and degradation rates were randomly generated as $\boldsymbol{r}_i = (r_i^1, r_i^2, r_i^3, r_i^4) = U(1, 2) \cdot (2.5, 3, 3.5, 4)$, where $U(1, 2)$ is a continuous uniform random variable on $(1, 2)$.

For this problem, the learned policy was trained for $60,000$ seconds, allowing for approximately $12,000,000$ decision epochs. Once the training was complete, the performance of the learned policy

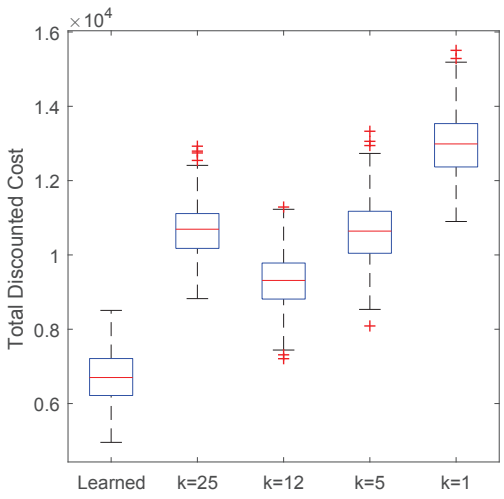


Figure 10: Boxplots comparing cost under different policies.

and the k-policies were compared over $5,000$ sample paths, beginning from state $(\boldsymbol{x}, j) = (\mathbf{0}, j)$. The results of this comparison are summarized in Figure 10.

Looking first at the k-policies, it appears that the expected total discounted cost is convex in k. This behavior occurs because, for large k, the downtime costs dominate the overall cost, yet for small k the setup costs are most significant. Even when the setup and downtime costs are most balanced ($k = 12$), the heuristic policy fails to outperform the learned policy. In fact, the learned policy outperforms all other policies sample-wise over all $5,000$ sample-paths. Specifically, the learned policy leads to an average cost savings of 38.05% when compared to the 12-policy and 92.61% when all dependency is ignored ($k = 1$).

Acknowledgements

The authors are grateful to the anonymous referees for constructive comments and suggestions. This research was sponsored, in part, by a grant from the U.S. National Science Foundation (CMMI-1266194).

References

[1] R. Barlow and F. Proschan. *Mathematical Theory of Reliability*. John Wiley & Sons, Inc., New York, NY, 1965.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[3] R. Bellman. A Markovian decision process. *Indiana University Mathematics Journal*, 6(4):679–684, 1957.

[4] D. Bertsekas and S. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, Belmont, MA, 1996.

[5] G. Birkhoff. *Lattice Theory, Third Edition*. American Mathematical Society, Providence, RI, 1967.

[6] K. Bouvard, S. Artus, C. Berenguer, and V. Cocquempot. Condition-based dynamic maintenance operataions planning and grouping: Application to commercial heavy vehicles. *Reliability Engineering and System Safety*, 96(6):601–610, 2011.

[7] B. Castanier, A. Grall, and C. Berenguer. A condition-based maintenance policy with non-periodic inspections for a two-unit series system. *Reliability Engineering and System Safety*, 87(1):109–120, 2005.

[8] N. Chen and K. Tsui. Condition monitoring and remaining useful life prediction using degradation signals: Revisited. *IIE Transactions*, 45(9):939–952, 2013.

[9] D. Cho and M. Parlar. A survey of maintenance models for multi-unit systems. *European Journal of Operational Research*, 51(1):1–23, 1991.

[10] R. Dekker. Applications of maintenance optimization models: A review and analysis. *Reliability Engineering and System Safety*, 51:229–240, 1996.

[11] R. Dekker, R. Wildeman, and F. van der Duyn Schouten. A review of multi-component maintenance models with economic dependence. *Mathematical Methods of Operations Research*, 45:411–435, 1997.

[12] N. Gebraeel, M. Lawley, R. Li, and J. Ryan. Residual-life distributions from component degradation signals: A Bayesian approach. *IIE Transactions*, 37(6):543–557, 2005.

[13] A. Geramifard, T. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. How. A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Foundations and Trends in Machine Learning*, 6(4):375–451, 2013.

[14] R. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA, 1960.

[15] J. Kharoufeh and S. Cox. Stochastic models for degradation-based reliability. *IIE Transactions*, 37(6):533–542, 2005.

[16] J. Kharoufeh, D. Finkelstein, and D. Mixon. Availability of periodically inspected systems with Markovian wear and shocks. *Journal of Applied Probability*, 43(2):303–317, 2006.

[17] J. Kharoufeh and D. Mixon. On a Markov-modulated shock and wear process. *Naval Research Logistics*, 56(6):563–576, 2009.

[18] J. Kharoufeh, C. Solo, and M. Ulukus. Semi-markov models for degradation-based reliability. *IIE Transactions*, 42(8):599–612, 2010.

[19] Y. Ko and E. Byon. Condition-based joint maintenance optimization for a large-scale system with homogeneous units. *IIE Transactions*, (DOI: 10.1080/0740817X.2016.1241457).

[20] G. Konidaris, S. Osentoski, and P. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, San Francisco, CA, 2011. The AAAI Press.

[21] M. Marseguerra, E. Zio, and L. Podofillini. Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation. *Reliability Engineering and System Safety*, 77(2):151–165, 2002.

[22] J. McCall. Maintenance policies for stochastically failing equipment: A survey. *Management Science*, 11(5):493–524, 1965.

[23] R. Nicolai and R. Dekker. Optimal maintenance of multi-component systems: A review. In *Complex System Maintenance Handbook*, pages 263–286, London, 2008. Springer London.

[24] H. Pham and H. Wang. Imperfect maintenance. *European Journal of Operational Research*, 94(3):425–438, 1996.

[25] W. Pierskalla and J. Voelker. A survey of maintenance models: The control and surveillance of deteriorating systems. *Naval Research Logistics Quarterly*, 23(3):353–388, 1976.

[26] W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, Hoboken, NJ, 2007.

[27] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, Hoboken, NJ, 2nd edition, 2005.

[28] G. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Cambridge University Engineering Department, 1994.

[29] M. Shaked and J. Shanthikumar. *Stochastic Orders*. Springer, New York, NY, 2007.

[30] A. Sharma, G. Yadava, and S. Deshmukh. A literature review and future perspectives on maintenance optimization. *Journal of Quality in Maintenance Engineering*, 17(1):5–25, 2011.

[31] Y. Sherif and M. Smith. Optimal maintenance models for systems subject to failure – A review. *Naval Research Logistics Quarterly*, 28(1):47–74, 1981.

[32] N. Singpurwalla. Survival in dynamic environments. *Statistical Science*, 10(1):86–103, 1995.

[33] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

[34] Z. Tian and H. Liao. Condition-based maintenance optimization for multi-component systems using proportional hazards model. *Reliability Engineering and System Safety*, 96(5):581–589, 2011.

[35] D. Topkis. *Supermodularity and Complementarity*. Princeton University Press, Princeton, NJ, 1998.

[36] M. Ulukus, J. Kharoufeh, and L. Maillart. Optimal replacement policies under environment-driven degradation. *Probability in the Engineering and Informational Sciences*, 26(3):405–424, 2012.

[37] C. Valdez-Flores and R. Feldman. A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistics*, 36(4):419–446, 1989.

[38] H. Wang. A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139(3):469–489, 2002.

[39] W. Yang, P. J. Tavner, C. J. Crabtree, Y. Feng, and Y. Qiu. Wind turbine condition monitoring: Technical and commercial challenges. *Wind Energy*, 17(5):673–693, 2012.

[40] Z. Ye, Y. Wang, K. Tsui, and M. Pecht. Degradation data analysis using Wiener processes with measurement errors. *IEEE Transactions on Reliability*, 62(4):772–780, 2013.

[41] Q. Zhu, H. Peng, and G. van Houtum. A condition-based maintenance policy for multi-component systems with a high maintenance setup cost. *OR Spectrum*, 37:1007–1035, 2015.