Approximate Dynamic Programming for Selective Maintenance in Series-Parallel Systems

Khatereh Ahadi and Kelly M. Sullivan

***Abstract*—The problem of allocating limited resources to maintain components of a multi-component system, known as selective maintenance, is naturally formulated as a high-dimensional Markov Decision Process (MDP). Unfortunately, these problems are difficult to solve exactly for realistically sized systems. With this motivation, we contribute an Approximate Dynamic Programming (ADP) algorithm for solving the selective maintenance problem for a series-parallel system with binary-state components. To the best of our knowledge, this paper describes the first application of ADP to maintaining multi-component systems. Our ADP is compared, using a numerical example from the literature, against exact solutions to the corresponding MDP. We then summarize the results of a more comprehensive set of experiments that demonstrate the ADP's favorable performance on larger instances in comparison to both the exact (but computationally intensive) MDP approach and the heuristic (but computationally faster) one-step-lookahead approach. Finally, we demonstrate that the ADP is capable of solving an extension of the basic selective maintenance problem in which maintenance resources are permitted to be shared across stages.**

***Index Terms*—Maintenance optimization, Selective maintenance, Approximate dynamic programming**

ACRONYMS

ADP	Approximate Dynamic Programming
MDP	Markov Decision Process
MWO	Multiple Weighted Objectives

NOTATION

m	Number of subsystems
n_i	Number of components in subsystem $i \in \{1,\dots,m\}$
T	Number of missions
r_i	Reliability of components in subsystem $i \in \{1,\dots,m\}$
S_i^t	Number of failed components in subsystem $i \in \{1,\dots,m\}$ at the end of mission $t \in \{0,\dots,T\}$, where S_i^0 is the number of initially failed components in subsystem i
a_i^t	Number of components in subsystem $i \in \{1,\dots,m\}$ to be repaired in break $t \in \{0,\dots,T-1\}$
$S_i^{x,t}$	Number of failed components in subsystem $i \in \{1,\dots,m\}$ immediately after maintenance is completed in break $t \in \{0,\dots,T-1\}$, $S_i^{x,t} = S_i^t - a_i^t$
ρ_{il}	Amount of resource $l \in \{1,\dots,p\}$ required to maintain components in subsystem $i \in \{1,\dots,m\}$
β_l	Amount of resource $l \in \{1,\dots,p\}$ available during each break
$\mathcal{S}$	State space given as $\mathcal{S} = \{0,1,\dots,n_1\} \times \{0,1,\dots,n_2\} \times \cdots \times \{0,1,\dots,n_m\}$
$V^t(s^t)$	Value function of state $s^t \in \mathcal{S}$, $t \in \{0,\dots,T\}$, defined as the maximum expected number of successful missions remaining among missions $t+1,\dots,T$ if the system is in state s^t after mission t
$V^t(s^{x,t})$	Value function of post-decision state $s^{x,t} \in \mathcal{S}$, defined as the maximum expected number of successful missions remaining among missions $t+2,\dots,T$ if (after completing maintenance action a^t in break $t \in \{0,\dots,T-1\}$) the system is in state $s^{x,t}$
$\bar{V}^t(s^{x,t}\|\eta^t)$	Approximate value function of post-decision state $s^{x,t} \in \mathcal{S},\ t \in \{0,\dots,T-1\}$
η^t	Parameter vector of the approximate value function $\bar{V}^t(\cdot\|\eta^t)$ associated with mission $t \in \{0,\dots,T-1\}$

Khatereh Ahadi is employed at the Naveen Jindal School of Management, University of Texas at Dallas, Richardson, TX, 75080, USA; email: kahadi@utdallas.edu

Kelly M. Sullivan is employed in the Department of Industrial Engineering, University of Arkansas, Fayetteville, AR, 72701, USA; email: ksulliv@uark.edu

I. Introduction

Maintenance is necessary for reliable operation of industrial and military systems, but it is also very costly. Due to resource limitations, it may be cost-prohibitive or impractical to fully maintain a system at every opportunity. This paper presents an approximate dynamic programming approach for optimizing *selective* maintenance decisions over time. We consider a series-parallel system subjected to a sequence of missions in which we have the opportunity to replace a subset of the failed components between each pair of successive missions. Our base model is identically the model of Maillart et al. [1]; however, we are the first to address the issue of solving large-scale instances of this problem.

The topic of selective maintenance has garnered recent attention in the literature. Rice et al. [2] contributed the first work in the area of selective maintenance, formulating a mathematical model to maximize the reliability of a series-parallel system given limited maintenance time. They considered repair time as a limited resource and developed a heuristic to identify which subset of failed components to repair to maximize reliability. Chen et al. [3] considered selective maintenance of multi-state series-parallel systems, developing a mathematical model that accounts for system reliability and attempts to minimize maintenance cost. Cassady et al. [4] compared optimal selective maintenance decisions for a series-parallel system under three different objective functions: maximizing reliability, minimizing cost, and minimizing repair time. Cassady et al. [5] considered practical extensions of the reliability-maximization model, including Weibull-distributed component lifetimes, multiple maintenance actions, minimal repair, and corrective or preventive replacement. To enhance the efficiency of the enumerative solution procedure used by Cassady et al. [5], Rajagopalan and Cassady [6] proposed a series of improvements that reduce the solution space.

Researchers have incorporated a variety of elements to make these types of models more realistic. For instance, Dao et al. [7] and Xu et al. [8] proposed selective maintenance models for multi-state series parallel system under economic dependence that arises when there are savings to be realized by jointly maintaining components. Dao and Zuo [9] incorporated dependence and component degradation within a multi-component maintenance problem. Chen et al. [10] proposed a mathematical model for selective maintenance of aging components with uncertainty maintenance duration. Khatab et al. [11], [12] extended the basic selective maintenance problem to allow for stochastic duration of missions and breaks.

In all selective maintenance problems for series-parallel systems reviewed thus far, decisions are made for only one future mission. Maillart et al. [1] formulated a selective maintenance model that considers a finite or infinite horizon of future missions. This model is formulated and solved as a Markov decision process (MDP) and the resulting optimal infinite-horizon policy is compared to both the optimal single-mission and two-mission policies. This study considers only relatively small systems having no more than three subsystems. Our research extends the work of Maillart et al. [1] by enabling maintenance planning for series-parallel systems with a greater number of subsystems and/or components. To overcome the difficulty of solving MDPs, an approximate dynamic programming (ADP) approach is proposed. The idea behind ADP is to develop useful approximations of a high-dimensional MDP that enable time and/or memory savings as compared to solving the MDP exactly. We refer the reader to [13] and [14] for an introduction to ADP.

Relative to the ADP literature, our primary contribution is the selective maintenance application. ADP has applications to numerous other sequential and stochastic decision problems in the literature, including: managing energy storage in the electric grid [15], [16], [17], [18]; scheduling links in a wireless communications network [19]; positioning emergency service vehicles [20], [21]; controlling the flight of an aircraft [22]; routing vehicles [23]; assigning truck drivers to loads [24]; relocating empty containers in a distribution network [25]; and batching and dispatching products for outbound transportation [26]. To the best of our knowledge, ADP has not yet been applied to optimize maintenance of multi-component systems.

Our contributions are as follows: We (i) provide an ADP algorithm that exploits the structure of the selective maintenance problem. We perform computational experiments to (ii) verify that the ADP identifies solutions that are near-optimal for small instances that can be solved to optimality as MDPs, and (iii) demonstrate that the ADP identifies solutions for larger and/or more complex instances that are superior to those attainable via less sophisticated (e.g., myopic) approaches. In order to solve larger instances using ADP in a reasonable time, we (iv) incorporate a heuristic method called Multiple Weighted Objectives (MWO), due to Coit and Konak [27], inside the ADP. Leveraging the ADP solution methodology, we also (v) extend the selective maintenance model of Maillart et al. [1] to the expanded-state-space case where resources are shared across missions and (vi) present insights upon solving numerical examples of this

extended model.

The remainder of this paper is organized as follows: Section II summarizes the selective maintenance MDP model of Maillart et al. [1]. Section III develops the ADP algorithm, "ADP-Concave," and Section IV compares its performance with (exact) MDP solutions for a numerical example from the literature. In Section V, the MWO heuristic is incorporated within ADP-Concave to enable solving larger instances in a reasonable time. Section VI extends ADP-Concave to allow resources to shift between missions. Section VII summarizes computational experiments investigating solution properties, sensitivity analysis, and algorithm performance for larger instances, and Section VIII concludes.

II. Model Definition and Assumptions

In this section, we present the selective maintenance MDP model for series-parallel systems. With the exception of minor notational differences, this model is identical to the MDP model of Maillart et al. [1].

Consider a series-parallel system comprised of m independent subsystems connected in series such that each subsystem i has independent components connected in parallel. Let n_i denote the number of components in subsystem $i = 1, \ldots, m$, and define $N_i = \{[i, j] : j = 1, \ldots, n_i\}$ as the set of components in subsystem i. Let $N = \cup_{i=1}^{m} N_i$ denote the set of components in the system. We consider a sequence of T missions denoted $t = 1, \ldots, T$, with breaks between missions. For $t = 1, \ldots, T-1$, "break t" refers to the break that occurs immediately after mission t, and "break 0" refers to a break before the first mission. Failures happen only during the missions and maintenance is performed during breaks. Fig. 1 illustrates the timing of missions and breaks implied by the above assumptions.

Let the state vector $S^t = [S_1^t, S_2^t, \ldots, S_m^t] \in \mathcal{S}$, $t = 0, \ldots, T$, denote the number of failed components in each subsystem after mission t where S^0 denotes the initial state of the system. In what follows, we will use the capitalized notation S_i^t to refer to an uncertain future state and the lower-case s_i^t to refer to a specific realization of S_i^t.

There are limited resources available to repair the failed components during each break. In state s^t, $t = 0, \ldots, T-1$, define the set of maintenance actions $A(s^t)$ as the set of a_i^t satisfying

$$\sum_{i=1}^{m} \rho_{il} a_i^t \le \beta_l, \ \forall l = 1, \ldots, p, \tag{1}$$

$$0 \le a_i^t \le s_i^t, \ \text{integer}, \ \forall i = 1, \ldots, m, \tag{2}$$

where a_i^t is the number of components to be repaired in subsystem i. Thus, the reliability of the series-parallel system for mission $t+1$ is given by

$$R(s^t, a^t) = \prod_{i=1}^{m} [1 - (1 - r_i)^{n_i - s_i^t + a_i^t}], \ t = 0, \ldots, T-1. \tag{3}$$

Let Z_i^{t+1} denote the number of components failing during mission $t+1$. Thus, the number of failed components at the end of mission $t+1$ is

$$S_i^{t+1} = s_i^t - a_i^t + Z_i^{t+1}, \tag{4}$$

for all $t = 0, \ldots, T-1$, where

$$Z_i^{t+1} \sim \text{binomial}(n_i - s_i^t + a_i^t, 1 - r_i). \tag{5}$$

Let $b(y, n, p) = \binom{n}{y} p^y (1-p)^{n-1}$ denote the binomial probability mass function. Then, for $i = 1, \ldots, m$ and $t = 0, \ldots, T-1$, the transition probabilities are

$$\begin{aligned} Pr(S_i^{t+1} = s_i^{t+1}) &= Pr(Z_i^{t+1} = s_i^{t+1} - s_i^t + a_i^t) \\ &= b(s_i^{t+1} - s_i^t + a_i^t, n_i - s_i^t + a_i^t, 1 - r_i). \end{aligned} \tag{6}$$

Under action $a^t \in A(s^t)$, the system transitions from state $s^t \in \mathcal{S}$ to state $s^{t+1} \in \{0, 1, \ldots, n_1 - s_1^t + a_1^t\} \times \{0, 1, \ldots, n_2 - s_2^t + a_2^t\} \times \cdots \times \{0, 1, \ldots, n_m - s_m^t + a_m^t\}$ with probability $p(s^{t+1} | s^t, a^t)$ defined as

$$\begin{aligned} p(s^{t+1} | s^t, a^t) &= \prod_{i=1}^{m} Pr(S_i^{t+1} = s_i^{t+1}) \\ &= \prod_{i=1}^{m} b(s_i^{t+1} - s_i^t + a_i^t, n_i - s_i^t + a_i^t, 1 - r_i). \end{aligned} \tag{7}$$

We define the value function $V^t(s^t)$ as the maximum expected number of successful missions remaining among missions $t+1, \ldots, T$ if the system is in state $s^t \in \mathcal{S}$ after mission t. For $t = 0, \ldots, T$ and $s^t \in \mathcal{S}$, the value function can be expressed as

$$\begin{aligned} V^t(s^t) &= \max_{a^t \in A(s^t)} \Big\{ R(s^t, a^t) \\ &\qquad + E\{V^{t+1}(S^{t+1}) | S^t = s^t\} \Big\} \\ &= \max_{a^t \in A(s^t)} \Big\{ R(s^t, a^t) \\ &\qquad + \sum_{s^{t+1} \in \mathcal{S}} V^{t+1}(s^{t+1}) p(s^{t+1} | s^t, a^t) \Big\}. \end{aligned} \tag{8}$$

We define $V^T(s^T) = 0$ for all $s^T \in \mathcal{S}$ such that (8) remains defined for $t = T$.

III. Maintenance Decision Making Using ADP

By increasing the number of components and/or subsystems, the state space, action space, and outcome space

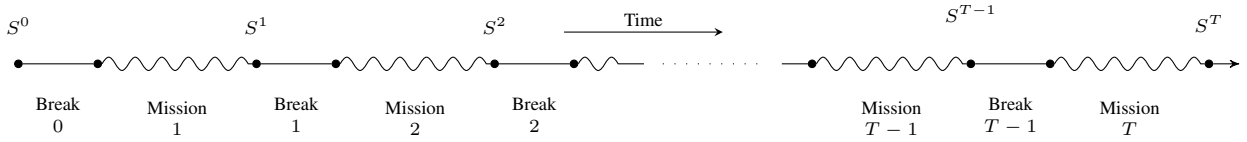

Fig. 1: Timeline of events in the selective maintenance problem

grow quickly, thus making the MDP difficult to solve. ADP aims to overcome these challenges by employing tractable approximations of $V^t(\cdot)$ that avoid looping over every possible state of the value function and integrating over the range of all random variables.

A. Optimality Recursion for Post-Decision States

Reformulating Equation (8) around the so-called *post-decision state* variable is a well-known strategy (see [14]) for reducing the difficulty associated with approximating the expectation in (8). Let the post-decision state $S_i^{x,t}$ denote the number of failed components in subsystem i immediately after maintenance is completed in break $t = 0, \ldots, T-1$, i.e.,

$$S_i^{x,t} = S_i^t - a_i^t. \tag{9}$$

Let $V^t(S^{x,t})$ denote the maximum expected number of successful missions remaining among missions $t+2, \ldots, T$ if (after completing maintenance action a^t in break t), the system is in state $S^{x,t} = S^t - a^t$. Conditioning on the outcome of mission $t+1$, we have

$$V^t(s^{x,t}) = \sum_{s^{t+1} \in \mathcal{S}} V^{t+1}(s^{t+1}) p(s^{t+1}|s^t, a^t), \quad t = 0, \ldots, T-1, \tag{10}$$

or equivalently

$$V^t(s^{x,t}) = E\{V^{t+1}(S^{t+1})|S^{x,t} = s^{x,t}\}, \quad t = 0, \ldots, T-1. \tag{11}$$

Therefore, substituting (10) into (8), we have

$$V^t(s^t) = \max_{a^t \in A(s^t)} \left\{R(s^t, a^t) + V^t(s^{x,t})\right\}, \quad t = 0, \ldots, T-1. \tag{12}$$

Substituting (12) for period $t+1$ into (11) for $t = 0, \ldots, T-2$, the optimality equations are formulated around the post-decision state variables as

$$V^t(s^{x,t}) = E\Big\{ \max_{a^{t+1} \in A(S^{t+1})} \left\{R(S^{t+1}, a^{t+1}) + V^{t+1}(S^{x,t+1})\right\} \Big| S^{x,t} = s^{x,t} \Big\}, \ t = 0, \ldots, T-1, \tag{13}$$

where $V^{T-1}(s^{x,T-1}) = 0, \ \forall s^{x,T-1} \in \mathcal{S}$ and $V^0(s^0) = \max_{a^0 \in A(s^0)} \left\{R(s^0, a^0) + V^0(S^{x,0})\right\}$ can be used to compute the value function for the initial state.

Using post-decision state variables, the expectation in (13) is now outside of the maximization. This is advantageous, because evaluating (or approximating) (13) for state $s^{x,t}$ under known (or approximated) $V^{t+1}(\cdot)$ requires only solving a deterministic optimization problem for each possible realization (or sampled subset of realizations) of S^{t+1}. By comparison, evaluating (8) for a single state requires solving a complex stochastic program.

B. Value Function Approximation

The state space is exponential for this problem, meaning it is impractical to store the value function $V^t(s^{x,t})$ for each discrete state $s^{x,t}$. Thus, there is a need for defining reasonable approximating strategies to compute value function for each state. There are various methods to approximate the value function in ADP, which typically entail building predictive models—in a manner that is computationally less burdensome than building a full lookup table—that can be used to approximate the value function associated with a given state. Commonly, the value function is approximated by defining a set of *basis functions* $\phi_f(s), \ f \in F$, over a set of important features F, such that $\phi_f(s)$ will represent the contribution of feature f to the approximated value function in state $s \in \mathcal{S}$. We define features corresponding to each subsystem (i.e., such that $F = \{1, \ldots, m\}$) and therefore denote the basis functions as $\phi_i(s), \ i = 1, \ldots, m$.

Let $\phi_i(s_i^{x,t})$ denote a basis function that will approximate the value function reduction from the unreliability of subsystem i. We assume ϕ_i has the form

$$\phi_i(s_i^{x,t}) = (\eta_i^t)^{n_i - s_i^{x,t}}, \tag{14}$$

where $0 \leq \eta_i^t \leq 1$ is a parameter. This basis function satisfies

$$\phi_i(1) - \phi_i(0) > \phi \cdots > \phi_i(n_i) - \phi_i(n_i - 1), \tag{15}$$

which is desirable because subsystem unreliability is also concave and increasing in the number of failed

components. Because the basis functions have been constructed to satisfy (15), we will refer to the resulting ADP as "ADP-Concave."

To further capture the correspondence between system reliability and the system's value function, we assume functions $\phi_i(\cdot)$ are combined to approximate the value function as

$$\bar{V}^t(s^{x,t}|\eta^t) = \eta_0^t \prod_{i=1}^{m} \left(1 - (\eta_i^t)^{n_i - s_i^{x,t}}\right), \quad t = 0, \ldots, T-1, \tag{16}$$

where parameter $0 \leq \eta_0^t \leq T - t$ represents an upper bound on the number of remaining successful missions.

Our choice of Equation (16) is somewhat nontraditional in the sense that ADPs commonly approximate the value function as a weighted sum of basis functions, and not as a product. The product form of Equation (16) is dually motivated, as summarized below.

Motivation 1: This form seeks to capture the structure of serially connected subsystems. The approximate value function $\bar{V}^t(s^{x,t}|\eta^t)$ is intended to provide an estimate of the true value function $V^t(s^{x,t})$, which equals the maximum expected number of successful missions remaining among missions $t+2, \ldots, T$. Therefore, for $i \in \{1, \ldots, m\}$, the i-th term in the product of Equation (16) might loosely be interpreted as "an estimate of the maximum probability that subsystem i survives a given future mission $t' \in \{t+2, \ldots, T\}$, given the current post-decision state $s_i^{x,t}$." Thus, this form enables (i) separately estimating the impact of each subsystem's failed components after break t on system failures among missions $t' \in \{t+2, \ldots, T\}$ and (ii) using knowledge of the system's structure to aggregate these estimates into an approximation of the value function.

Motivation 2: As demonstrated in Section V, this form yields optimization subproblems of the ADP that are conveniently solved by established methodologies for the classical redundancy allocation problem.

Parameters η^t are updated through a sequence of iterations $n = 1, \ldots, N$. Let $\eta^{t,n}$ denote the $(m+1)$-vector of parameter estimates associated with period $t = 0, \ldots, T-1$ and iteration $n = 1, \ldots, N$. In each iteration $n = 1, \ldots, N$ a sample path represented by $\{s^t\}_{t=0}^{T}$ and $\{s^{x,t}\}_{t=0}^{T-1}$ is generated by solving for each $t = 0, \ldots, T-1$ the approximate version of Equation (12), i.e.,

$$\tilde{V}^{t,n}(s^t) = \max_{a^t \in A(s^t)} \left\{R(s^t, a^t) + \bar{V}^t(s^{x,t}|\eta^{t,n-1})\right\}. \tag{17}$$

Given parameter estimates $\eta^{t,n-1},\ t \in \{1, \ldots, T-1\}$, we generate the updated parameter estimates $\eta^{t-1,n}$ by seeking to minimize

$$g^{t-1,n}(\eta^{t-1}) = \mathbb{E}\frac{1}{2}\left(\bar{V}^{t-1}(S^{x,t-1}|\eta^{t-1}) - \tilde{V}^{t,n}(S^t)\right)^2, \tag{18}$$

subject to $0 \leq \eta_0^t \leq T - t$ and $0 \leq \eta_i^t \leq 1,\ \forall i = 1, \ldots, m$, where the expectation is taken over $S^{x,t-1}$ and S^t. In view of (18) and the observed states $s^{x,t-1}$ and s^t from this iteration, we apply a stochastic gradient descent update to generate $\eta^{t-1,n},\ \forall t = 1, \ldots, T-1$, as

$$\eta^{t-1,n} = \eta^{t-1,n-1} - \alpha_n\left(\bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1,n-1}) - \tilde{V}^{t,n}(s^t)\right) \times \nabla_{\eta^{t-1}} \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1,n-1}), \tag{19}$$

where $0 \leq \alpha_n \leq 1$ is the step size at iteration n. Element 0 of $\nabla_{\eta^{t-1}} \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1})$ is

$$\frac{\partial \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1})}{\partial \eta_0^{t-1}} = \prod_{i=1}^{m} \left(1 - (\eta_i^{t-1})^{n_i - s_i^{x,t-1}}\right), \tag{20}$$

and element $i = 1, \ldots, m$ of $\nabla_{\eta^{t-1}} \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1})$ is

$$\frac{\partial \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1})}{\partial \eta_i^{t-1}} = -\left(n_i - s_i^{x,t-1}\right)(\eta_i^{t-1})^{n_i - s_i^{x,t-1} - 1}(\eta_0^{t-1}) \times \prod_{i' \in \{1,\ldots,m\} \setminus i} \left(1 - (\eta_{i'}^{t-1})^{n_{i'} - s_{i'}^{x,t-1}}\right). \tag{21}$$

The stochastic gradient descent algorithm is a well-known machine learning method to update regression parameters (see, e.g., [28]) incrementally as training observations (in this case, $s^{x,t-1}$ and s^t) are added. In our problem, there is a feasible region defined for $\eta^{t,n-1}$ based on two constraints: $0 \leq \eta_0^t \leq T - t$ and $0 \leq \eta_i^t \leq 1,\ \forall i = 1, \ldots, m$. If updating the parameters based on (19) violates one of the constraints, we do not update the parameter in this iteration.

Given an initial state s^0, ADP-Concave determines the maintenance actions during break 0. A formal statement of ADP-Concave follows.

ADP-CONCAVE

Step 0: Initialization

(0a) Set $n = 1$ (N is the maximum number of iterations)

(0b) Initialize $\eta_0^{t,0} = 0$ and $\eta_i^{t,0} = 1 - r_i,\ \forall t = 0, \ldots, T-1, \forall i = 1, \ldots, m$

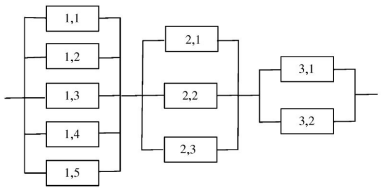

Fig. 2: Small system with $n = [5, 3, 2]$

Step 1: For $t = 0, \ldots, T-1$,
(1a) Set $\tilde{V}^{t,n}(s^t)$ as

$$\tilde{V}^{t,n}(s^t) = \max_{a^t \in A(s^t)} \left\{ R(s^t, a^t) + \bar{V}^t(s^{x,t} | \eta^{t,n-1}) \right\} \tag{22}$$

and let a^{*t} denote the obtained optimal action.
(1b) Set $s^{t+1} = s^t - a^{*t} + Z^{t+1}$, where Z_i^{t+1}, $i = 1, \ldots, m$ is generated from $Z_i^{t+1} \sim \text{binomial}(n_i - s_i^t + a^{*t}, 1 - r_i)$.

Step 2: For $t = 1, \ldots, T-1$, generate $\eta^{t-1,n}$ from $\eta^{t-1,n-1}$ by applying Equation (19).

Step 3: If $n \leq N$, increment n and go to Step 1; Otherwise, return the value functions $\bar{V}^{t,N}$ for $t = 0, \ldots, T-1$.

As is common in ADP, the idea behind ADP-Concave is to focus computational effort—by stepping forward through time instead of solving the MDP by traditional backward dynamic programming—on states that are relatively likely to be visited, given the initial state s^0. The resulting value function approximations may therefore differ depending on the initial state. Thus, if decisions are desired for multiple possible initial states, we suggest re-running ADP-Concave for each initial state.

IV. Numerical Example

In this section, we represent the results of ADP algorithm for a small example. In order to validate the developed ADP algorithm, we choose the example illustrated in [1], which is solved optimally using finite-horizon MDP approach for different initial states. This example is a small system with three subsystems and 10 components, as depicted in Fig. 2. The reliabilities for subsystems are $r_1 = 0.7962$, $r_2 = 0.8623$, and $r_3 = 0.9558$. There are limited resources for maintenance at each time period $t = 0, 1, \ldots, T-1$ as follows:

$$1.93a_1^t + 3.82a_2^t + 1.52a_3^t \leq 11.7$$
$$2.85a_1^t + 3.28a_2^t + 3.47a_3^t \leq 20.1$$
$$2.84a_1^t + 1.06a_2^t + 3.76a_3^t \leq 20.2$$

The ADP-Concave method is employed to solve the problem with $N = 500$ and $\alpha = 0.1$ for $T \in \{1, 2\}$.

The optimal action and value function based on the finite-horizon MDP $(a_M^*(s), V_M(s))$ and ADP-Concave $(a_{AC}^*(s), V_{AC}(s))$ are represented for $T \in \{1, 2, 5, 10\}$ in Table I. The value reported for $V_{AC}(s)$ is the sample average of 500 sample paths based upon using the approximated value function to determine maintenance actions during each break. Based on Table I, the selected actions based on the ADP method for $T = 1$ are the same as the finite-horizon MDP for all of the initial states. This is the expected result because both the MDP and ADP capture the first mission's reliability explicitly when $T = 1$. For $T \in \{2, 5, 10\}$, the best actions selected using ADP-Concave are the same as the finite-horizon MDP as well. For most of the initial states, both the MDP and ADP maintain all of the failed components in the third subsystem because it has comparatively fewer components and requires more resources to repair.

V. Enhanced ADP with MWO Heuristic

Although the previous section demonstrates that the ADP performs well on a small instance, there are remaining challenges that must be overcome in order to solve larger problem instances with the ADP. Notably, the statement of ADP-Concave given in Section III assumes availability of an oracle for solving Equation (22). The method (i.e., totally enumerating $A(s^t)$) utilized in the previous section may be too time consuming for larger instances, especially when considering that Equation (22) must be solved repeatedly throughout the ADP. This section enhances ADP-Concave by applying a specialized method to solve Equation (22).

The maximization problem inside Equation (22) is an extended version of the redundancy allocation problem (see, e.g., [29], [30], [31]). Because this problem is known to be NP-hard [32], we develop a heuristic method to solve Equation (22). This method is based on the MWO heuristic proposed by Coit and Konak [27], and it involves the transformation of a single-objective problem into one with multiple objectives. When using MWO to solve Equation (22), we refer to the resulting ADP-Concave implementation as "ADP-MWO."

We now summarize the MWO heuristic used in ADP-MWO. Our implementation of MWO follows closely to Coit and Konak [27]; therefore, we have opted for a brief description here, referring the reader to Coit and Konak [27] for a detailed description. For $t = 0, \ldots, T-$

TABLE I: Selected actions and value functions for instancein Section IV

	$T=1$				$T=2$				$T=5$				$T=10$			
Initial state	MDP		ADP-Concave		MDP		ADP-Concave		MDP		ADP-Concave		MDP		ADP-Concave	
s	$a_M^{*1}(s)$	$V_M^1(s)$	$a_{AC}^{*1}(s)$	$V_{AC}^1(s)$	$a_M^{*2}(s)$	$V_M^2(s)$	$a_{AC}^{*2}(s)$	$V_{AC}^2(s)$	$a_M^{*5}(s)$	$V_M^5(s)$	$a_{AC}^{*5}(s)$	$V_{AC}^5(s)$	$a_M^{*10}(s)$	$V_M^{10}(s)$	$a_{AC}^{*10}(s)$	$V_{AC}^{10}(s)$
[0,3,1]	[0,2,1]	0.979	[0,2,1]	0.979	[0,2,1]	1.974	[0,2,1]	1.973	[0,2,1]	4.974	[0,2,1]	4.961	[0,2,1]	9.969	[0,2,1]	9.896
[0,3,2]	[0,2,2]	0.979	[0,2,2]	0.979	[0,2,2]	1.974	[0,2,2]	1.974	[0,2,2]	4.975	[0,2,2]	4.947	[0,2,2]	9.973	[0,2,2]	9.897
[1,2,2]	[0,2,2]	0.994	[0,2,2]	0.994	[0,2,2]	1.989	[0,2,2]	1.989	[0,2,2]	4.980	[0,2,2]	4.977	[0,2,2]	9.976	[0,2,2]	9.922
[1,3,0]	[0,3,0]	0.994	[0,3,0]	0.994	[0,3,0]	1.989	[0,3,0]	1.989	[0,3,0]	4.980	[0,3,0]	4.944	[0,3,0]	9.980	[0,3,0]	9.889
[1,3,1]	[1,2,1]	0.979	[1,2,1]	0.979	[1,2,1]	1.974	[1,2,1]	1.974	[1,2,1]	4.978	[1,2,1]	4.954	[1,2,1]	9.975	[1,2,1]	9.940
[1,3,2]	[0,2,2]	0.978	[0,2,2]	0.978	[0,2,2]	1.972	[0,2,2]	1.972	[0,2,2]	4.969	[0,2,2]	4.919	[0,2,2]	9.968	[0,2,2]	9.875
[2,2,1]	[1,2,1]	0.994	[1,2,1]	0.994	[1,2,1]	1.989	[1,2,1]	1.989	[1,2,1]	4.943	[1,2,1]	4.943	[1,2,1]	9.939	[1,2,1]	9.882
[2,2,2]	[0,2,2]	0.987	[0,2,2]	0.987	[0,2,2]	1.982	[0,2,2]	1.982	[0,2,2]	4.973	[0,2,2]	4.961	[0,2,2]	9.972	[0,2,2]	9.890
[2,3,0]	[0,3,0]	0.987	[0,3,0]	0.987	[0,3,0]	1.982	[0,3,0]	1.982	[0,3,0]	4.983	[0,3,0]	4.951	[0,3,0]	9.976	[0,3,0]	9.877
[2,3,1]	[1,2,1]	0.978	[1,2,1]	0.978	[1,2,1]	1.972	[1,2,1]	1.972	[1,2,1]	4.972	[1,2,1]	4.938	[1,2,1]	9.969	[1,2,1]	9.943
[2,3,2]	[0,2,2]	0.971	[0,2,2]	0.971	[0,2,2]	1.966	[0,2,2]	1.966	[0,2,2]	4.971	[0,2,2]	4.948	[0,2,2]	9.964	[0,2,2]	9.884
[3,1,2]	[2,1,2]	0.994	[2,1,2]	0.994	[2,1,2]	1.989	[2,1,2]	1.989	[2,1,2]	4.982	[2,1,2]	4.940	[2,1,2]	9.973	[2,1,2]	9.930
[3,2,0]	[2,2,0]	0.994	[2,2,0]	0.994	[2,2,0]	1.989	[2,2,0]	1.988	[2,2,0]	4.988	[2,2,0]	4.967	[2,2,0]	9.987	[2,2,0]	9.974
[3,2,1]	[1,2,1]	0.987	[1,2,1]	0.987	[1,2,1]	1.982	[1,2,1]	1.982	[1,2,1]	4.987	[1,2,1]	4.939	[1,2,1]	9.986	[1,2,1]	9.983
[3,2,2]	[2,1,2]	0.978	[2,1,2]	0.978	[2,1,2]	1.972	[2,1,2]	1.972	[2,1,2]	4.977	[2,1,2]	4.952	[2,1,2]	9.974	[2,1,2]	9.879
[3,3,0]	[2,2,0]	0.978	[2,2,0]	0.978	[2,2,0]	1.972	[2,2,0]	1.972	[2,2,0]	4.976	[2,2,0]	4.926	[2,2,0]	9.968	[2,2,0]	9.890
[3,3,1]	[1,2,1]	0.971	[1,2,1]	0.971	[1,2,1]	1.965	[1,2,1]	1.965	[1,2,1]	4.973	[1,2,1]	4.963	[1,2,1]	9.971	[1,2,1]	9.964
[3,3,2]	[1,2,1]	0.939	[1,2,1]	0.939	[0,2,2]	1.933	[0,2,2]	1.933	[0,2,2]	4.950	[0,2,2]	4.905	[0,2,2]	9.945	[0,2,2]	9.846
[4,1,1]	[3,1,1]	0.994	[3,1,1]	0.994	[3,1,1]	1.989	[3,1,1]	1.989	[3,1,1]	4.984	[3,1,1]	4.951	[3,1,1]	9.979	[3,1,1]	9.932
[4,1,2]	[2,1,2]	0.987	[2,1,2]	0.987	[2,1,2]	1.982	[2,1,2]	1.982	[2,1,2]	4.979	[2,1,2]	4.949	[2,1,2]	9.978	[2,1,2]	9.919
[4,2,0]	[2,2,0]	0.987	[2,2,0]	0.987	[2,2,0]	1.982	[2,2,0]	1.982	[2,2,0]	4.953	[2,2,0]	4.936	[2,2,0]	9.949	[2,2,0]	9.922
[4,2,1]	[3,1,1]	0.978	[3,1,1]	0.978	[3,1,1]	1.972	[3,1,1]	1.972	[3,1,1]	4.967	[3,1,1]	4.918	[3,1,1]	9.962	[3,1,1]	9.956
[4,2,2]	[2,1,2]	0.971	[2,1,2]	0.971	[2,1,2]	1.965	[2,1,2]	1.965	[2,1,2]	4.975	[2,1,2]	4.925	[2,1,2]	9.969	[2,1,2]	9.926
[4,3,0]	[2,2,0]	0.971	[2,2,0]	0.971	[2,2,0]	1.966	[2,2,0]	1.966	[2,2,0]	4.970	[2,2,0]	4.955	[2,2,0]	9.965	[2,2,0]	9.890
[4,3,1]	[2,2,0]	0.939	[2,2,0]	0.939	[1,2,1]	1.933	[1,2,1]	1.930	[1,2,1]	4.944	[1,2,1]	4.905	[1,2,1]	9.938	[1,2,1]	9.869
[4,3,2]	[1,2,1]	0.908	[1,2,1]	0.899	[1,2,1]	1.890	[1,2,1]	1.882	[1,2,1]	4.930	[1,2,1]	4.899	[1,2,1]	9.929	[1,2,1]	9.844
[5,0,2]	[4,0,2]	0.994	[4,0,2]	0.994	[4,0,2]	1.989	[4,0,2]	1.989	[4,0,2]	4.978	[4,0,2]	4.951	[4,0,2]	9.977	[4,0,2]	9.888
[5,1,0]	[4,1,0]	0.994	[4,1,0]	0.994	[4,1,0]	1.989	[4,1,0]	1.989	[4,1,0]	4.981	[4,1,0]	4.943	[4,1,0]	9.971	[4,1,0]	9.969
[5,1,1]	[3,1,1]	0.987	[3,1,1]	0.987	[3,1,1]	1.982	[3,1,1]	1.982	[3,1,1]	4.976	[3,1,1]	4.954	[3,1,1]	9.975	[3,1,1]	9.917
[5,1,2]	[4,0,2]	0.978	[4,0,2]	0.978	[4,0,2]	1.972	[4,0,2]	1.972	[4,0,2]	4.966	[4,0,2]	4.950	[4,0,2]	9.960	[4,0,2]	9.917
[5,2,0]	[4,1,0]	0.978	[4,1,0]	0.978	[4,1,0]	1.972	[4,1,0]	1.972	[4,1,0]	4.960	[4,1,0]	4.926	[4,1,0]	9.954	[4,1,0]	9.871
[5,2,1]	[3,1,1]	0.971	[3,1,1]	0.971	[3,1,1]	1.966	[3,1,1]	1.966	[3,1,1]	4.974	[3,1,1]	4.944	[3,1,1]	9.970	[3,1,1]	9.925
[5,2,2]	[3,1,1]	0.939	[3,1,1]	0.939	[2,1,2]	1.932	[2,1,2]	1.932	[2,1,2]	4.922	[2,1,2]	4.892	[2,1,2]	9.916	[2,1,2]	9.845
[5,3,0]	[2,2,0]	0.939	[2,2,0]	0.939	[2,2,0]	1.933	[2,2,0]	1.933	[2,2,0]	4.924	[2,2,0]	4.909	[2,2,0]	9.919	[2,2,0]	9.840
[5,3,1]	[2,2,0]	0.908	[2,2,0]	0.899	[2,2,0]	1.891	[2,2,0]	1.890	[2,2,0]	4.884	[2,2,0]	4.867	[2,2,0]	9.876	[2,2,0]	9.845
[5,3,2]	[3,1,1]	0.825	[3,1,1]	0.825	[2,1,2]	1.817	[2,1,2]	1.817	[2,1,2]	4.802	[2,1,2]	4.789	[2,1,2]	9.793	[2,1,2]	9.725

1, the maximization problem in (22) is

$$\max \left\{ \prod_{i=1}^{m} [1-(1-r_i)^{n_i - s_i^t + a_i^t}] + \eta_0^{t,n-1} \prod_{i=1}^{m} [1-(\eta_i^{t,n-1})^{n_i - s_i^t + a_i^t}] \right\}, \quad (23)$$

s.t. Constraints (1)–(2).

The MWO heuristic attempts to identify high-quality solutions to Model (23) by employing a surrogate model in which each of the m terms in the first product and $(m+1)$ terms in the second product of Objective (23) are maximized separately. The surrogate model is the $(2m+1)$-objective problem

$$\max \Big\{ 1-(1-r_1)^{n_1 - s_1^t + a_1^t}, 1-(1-r_2)^{n_2 - s_2^t + a_2^t}, \ldots, 1-(1-r_m)^{n_m - s_m^t + a_m^t}, \eta_0^{t,n-1}, 1-(\eta_1^{t,n-1})^{n_1 - s_1^t + a_1^t}), 1-(\eta_2^{t,n-1})^{n_2 - s_2^t + a_2^t}, \ldots, 1-(\eta_m^{t,n-1})^{n_m - s_m^t + a_m^t} \Big\}, \quad (24)$$

s.t. Constraints (1)–(2).

The surrogate model, Model (24), conveniently allows for the nonlinear objectives to be reformulated into linear objectives. Removing the constant-valued objective $\eta_0^{t,n-1}$ and negating the remaining objectives yields that that Model (24) is equivalent to the $2m$-objective model

$$\min \Big\{ (1-r_1)^{n_1 - s_1^t + a_1^t}, (1-r_2)^{n_2 - s_2^t + a_2^t}, \ldots, (1-r_m)^{n_m - s_m^t + a_m^t}, (\eta_1^{t,n-1})^{n_1 - s_1^t + a_1^t}, (\eta_2^{t,n-1})^{n_2 - s_2^t + a_2^t}, \ldots, (\eta_m^{t,n-1})^{n_m - s_m^t + a_m^t} \Big\}. \quad (25)$$

s.t. Constraints (1)–(2).

After defining $\lambda_i = -\ln(1-r_i)$ and $\gamma_i^{t,n-1} = -\ln(\eta_i^{t,n-1})$, applying a natural logarithm to each of the $2m$ objectives in Model (25) yields the equivalent, linear, $2m$-objective model

$$\max \Big\{ (n_1 - s_1^t + a_1^t)\lambda_1, (n_2 - s_2^t + a_2^t)\lambda_2, \ldots, (n_m - s_m^t + a_m^t)\lambda_m, (n_1 - s_1^t + a_1^t)\gamma_1^{t,n-1}, (n_2 - s_2^t + a_2^t)\gamma_2^{t,n-1}, \ldots, (n_m - s_m^t + a_m^t)\gamma_m^{t,n-1} \Big\}, \quad (26)$$

s.t. Constraints (1)–(2).

Following Coit and Konak [27], we derive solutions to Model (26) by solving a single-objective model in which the objectives of Model (26) are combined using weights $\omega_i,\ i = 1, \ldots, m$. Similar to Coit and Konak [27], we set $\omega_i = 1/(2m),\ \forall i = 1, \ldots, m$, placing equal weight on the $2m$ objectives from Objective (26). This assignment of weights is sensible because the system is structurally symmetric with regard to the set of subsystems (i.e., failure of any subsystem causes the system to fail). These weights yield the integer linear program

$$\max \sum_{i=1}^{m} \omega_i \left(n_i - s_i^t + a_i^t\right) \left(\lambda_i + \gamma_i^{t,n-1}\right), \quad (27)$$

s.t. Constraints (1)–(2).

Solving Model (27) yields a feasible solution to Model (23); however, the "weighted objectives" form of Model (27) tends to lead to solutions in which one or more of the subsystems $i \in \{1, \ldots, m\}$ yields a value of $1-(1-r_i)^{n_i - s_i^t + a_i^t}$ or $1-(\eta_i^{t,n-1})^{n_i - s_i^t + a_i^t}$ that is "too small"—that is, such a solution would have prohibitively reduced the value of Objective (23) even though it happens to be optimal for Model (27). In order to address this issue, we add the constraints

$$(n_i - s_i^t + a_i^t)\lambda_i > -\ln(1-R_i),\ \forall i = 1, \ldots, m, \quad (28)$$

$$(n_i - s_i^t + a_i^t)\gamma_i^{t,n-1} > -\ln(1-R_i'),\ \forall i = 1, \ldots, m, \quad (29)$$

where R_i and R_i' respectively denote unacceptable values of subsystem i's reliability and value function contribution that must be exceeded in future iterations of solving Model (27). These are written as "greater than" constraints, but in practice, they are changed to a "greater than or equal to" by adding a small ϵ term to the right hand-side. (We use $\epsilon = 0.00001$ for all of the instances reported in this paper.) We solve Model (27)–(29) iteratively in which Constraints (28) and (29) are tightened (by adding a pre-specified Δ-value) in order to improve the smallest reliability and value function contribution in the next iteration. Model (27)–(29) can be solved using available integer programming algorithms and software. The algorithm for a given Δ-value is described below.

MWO HEURISTIC

Step 0: Set $q \leftarrow 1$. Set $R_i \leftarrow 0$, $R_i' \leftarrow 0$, and $\omega_i \leftarrow 1/(2m)$ for $i = 1, \ldots, m$.

Step 1: Solve Model (27)–(29) using CPLEX. If the model is infeasible, go to Step 3. Otherwise, retain the obtained optimal action a^{tq} and optimal value $\tilde{V}_q^{t,n}$ for iteration q, and go to Step 2.

Step 2: For iteration q, let $k \in \arg\min_i (R_i(a_i^{tq}))$

and $j \in \arg\min_i(1-\eta^{n_i-s_i^t+a_i^{tq}})$, where $R_i(a_i^{tq}) = (1-(1-r_i)^{n_i-s_i^t+a_i^{tq}})$. Update $R_k \leftarrow R_k(a_k^q)+\Delta$, $R'_j \leftarrow (1-\eta^{n_j-s_j^t+a_j^{tq}})+\Delta$, and $q \leftarrow q+1$. Go to Step 1.

Step 3: Return the best objective value $\tilde{V}^{*t,n} = \max_q \tilde{V}_q^{t,n}$ and solution $a^* = a^{tv}$, where $v \in \arg\max_q \tilde{V}_q^{t,n}$.

VI. Maintenance with Shared Resources

We now demonstrate that ADP-Concave can be extended to the case where resources are shared across maintenance breaks. For simplicity, we assume in this section that there is only one resource (hereafter referred to as *budget*) with β_1 units available. In order to model this case, we add the state variable $B^t,\ t=0,\dots,T-1$, to represent the remaining budget at the end of mission t, where $B^0 \equiv \beta_1$ represents the total available budget. Thus, $A(s^t)$ is changed to $A(s^t,B^t),\ t=0,\dots,T-1$, by replacing Constraint (1) with

$$\sum_{i=1}^{m} \rho_{i1} a_i^t \le B^t. \tag{30}$$

State transitions occur as before with the exception that we now add for $t=0,\dots,T-1$ the transition function

$$B^{t+1} = B^t - \sum_{i=1}^{m} \rho_{i1} a_i^t, \tag{31}$$

in order to update the remaining budget. We modify the ADP-Concave algorithm by adjusting Equation (16) as

$$\bar{V}^t(s^{x,t}, B^t|\eta_0^t,\eta_i^t,\theta^t) = \eta_0^t(1-\theta^{t\frac{B^{t+1}}{\beta_1}})\prod_{i=1}^{m}(1-\eta_i^{t(n_i-s_i^{x,t})}), \tag{32}$$

where $0 \le \theta^t \le 1$ is a new parameter. Analogous to the discussion after Equation (16), the term $(1-\theta^{t\,B^{t+1}/\beta_1})$ might be interpreted loosely as "an estimate of the maximum probability that shortage of resources does not cause failure in a given future mission $t' \in \{t+2,\dots,T\}$." By incorporating θ^t, the modified value function approximation in Equation (32) now has the ability to capture the impact of remaining budget on the maximum number of remaining successful missions—because $(1-\theta^{t\,B^{t+1}/\beta_1})$ is increasing in B^{t+1}, Equation (32) enables penalizing over-aggressive use of the budget in early periods.

The parameters in Equation (32) are updated based on stochastic gradient descent method represented in (19). Element 0 of $\nabla_{\eta^{t-1}}\bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1},\theta^t)$ is

$$\frac{\partial \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1},\theta^t)}{\partial \eta_0^{t-1}} = (1-\theta^{t\frac{B^{t+1}}{\beta_1}})\prod_{i=1}^{m}\left(1-(\eta_i^{t-1})^{n_i-s_i^{x,t-1}}\right), \tag{33}$$

element $i=1,\dots,m$ of $\nabla_{\eta^{t-1}}\bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1},\theta^t)$ is

$$\frac{\partial \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1},\theta^t)}{\partial \eta_i^{t-1}} = -(1-\theta^{t\frac{B^{t+1}}{\beta_1}}) \times \left(n_i - s_i^{x,t-1}\right)(\eta_i^{t-1})^{n_i-s_i^{x,t-1}-1}(\eta_0^{t-1}) \times \prod_{i'\in\{1,\dots,m\}\setminus i}\left(1-(\eta_{i'}^{t-1})^{n_{i'}-s_{i'}^{x,t-1}}\right), \tag{34}$$

and θ^t is updated based on

$$\frac{\partial \bar{V}^{t-1}(s^{x,t-1}|\eta^{t-1},\theta^t)}{\partial \theta} = -\frac{B^{t+1}}{\beta}\theta^{t(\frac{B^{t+1}}{\beta_1}-1)}\eta_0^{t-1} \times \prod_{i=1}^{m}\left(1-(\eta_i^{t-1})^{n_i-s_i^{x,t-1}}\right). \tag{35}$$

To implement MWO inside the ADP, we define $\mu^{t,n-1} = -\ln(\theta^{t,n-1})$ and rewrite the objectives (26) as

$$\max\Big\{(n_1-s_1^t+a_1^t)\lambda_1,(n_2-s_2^t+a_2^t)\lambda_2,\dots,(n_m-s_m^t+a_m^t)\lambda_m, \frac{B^{t+1}}{\beta_1}\mu^{t,n-1}, (n_1-s_1^t+a_1^t)\gamma_1^{t,n-1},(n_2-s_2^t+a_2^t)\gamma_2^{t,n-1},\dots,(n_m-s_m^t+a_m^t)\gamma_m^{t,n-1}\Big\}. \tag{36}$$

Let R'' denote an unacceptable value of $(1-\theta^{t\,B^{t+1}/\beta_1})$ that is initialized to 0 and updated at each iteration. We then solve the MWO algorithm in Section V in which we replace Model (27) with

$$\max\ \frac{1}{2m+1}\left(\sum_{i=1}^{m}\left(n_i-s_i^t+a_i^t\right)\left(\lambda_i+\gamma_i^{t,n-1}\right)+\frac{B^{t+1}}{\beta_1}\mu^{t,n-1}\right), \tag{37}$$

$$\text{s.t. } a^t \in A(s^t,B^t), \tag{38}$$

$$B^{t+1}\mu^{t,n-1} > -\beta_1 \ln(1-R''), \tag{39}$$

Constraints (28), (29), and (31).

Following Coit and Konak [27] and the discussion in Section V, the above model results after applying equal weights to the $(2m+1)$ objectives in Equation (36).

VII. Computational Results

In this section, we test the performance of the ADP algorithms and analyze their solutions. Using a set of instances that vary in size, Section VII-A compares the ADP algorithm performance against the MDP and myopic algorithms and analyzes ADP solutions in detail. Section VII-B performs a further investigation of the ADP's performance on a broader set of instances. Section VII-C analyzes the effect of varying input parameters on the value function and maintenance decisions. Section VII-D applies and analyzes the shared-budget model on the instances from Section VII-A and compares solutions to the shared-budget and original models.

A. ADP Performance and Solution Analysis

In this section, we compare the performance of ADP-Concave against the (exact) backward dynamic programming algorithm for solving the MDP (see [14]). We also compare against a myopic approach in which we sequentially solve the one-step-lookahead redundancy allocation problem to determine the maintenance actions in each state—that is, the myopic approach solves $\max_{a^t \in A(s^t)} R(s^t, a^t)$ (ignoring the expected number of successful missions after mission $t+1$) to determine the maintenance actions in state $s^t \in \mathcal{S}$. Both ADP-Concave and the myopic approach are amenable to solving subproblems using the MWO heuristic of Section V, so we have implemented two versions of each of these approaches: one that includes MWO, one that solves by enumeration. We utilize $N = 500$ and step size $\alpha_n = 0.1,\ \forall n = 1, \ldots, N$ in both of the ADP implementations and $\Delta = 0.0001$ in both of the MWO implementations. In what follows, we will refer to the five solution approaches as "MDP" (the backward dynamic program), "Myopic" (the myopic approach in which subproblems are solved by enumeration), "ADP-Concave" (the version of ADP-Concave in which subproblems are solved by enumeration), "Myopic-MWO" (the myopic approach in which subproblems are solved by MWO), and "ADP-MWO" (the version of ADP-Concave in which subproblems are solved by MWO. Each algorithm is implemented in JAVA using the University of Arkansas High Performance Computing Center and the time limit is set for three days $(259,200$ s).

We compare the algorithms across eight instances with $m \in \{3, 4, 5, 6, 7, 8, 9, 10\}$ and parameters given in Table II. The instances are solved for $T \in \{1, \ldots, 10\}$ and one available resource (hereafter referred to as "budget") with $\beta_1 = 5m$ units available for maintenance in each break and 2 failed components in each subsystem as the initial state. Tables III, IV, and V respectively display (i) each algorithm's expected number of successful missions, (ii) the computation time required by each algorithm, and (iii) the selected maintenance actions in break 0 for the instances with $T = 10$. In Table III, the value functions for the MDP are exact; however, for the ADP and myopic approaches, we report a sample average over 500 simulated sample paths. In Table IV, the computation time for both ADP algorithms is the time required to train the model parameters over 500 sample paths. However, no training is required for the myopic approaches, and we therefore (for consistency with the ADP results) report the time required to find the best maintenance policy over a single generated sample path. Consequentially, the time required to evaluate the performance of the myopic and ADP algorithms (by generating 500 additional sample paths) is not included in the time reported in Table IV.

TABLE II: Numerical instance parameters

Subsystem, i	n_i	r_i	ρ_{i1}
1	5	0.8	5
2	5	0.8	5
3	5	0.7	5
4	5	0.9	5
5	5	0.9	5
6	5	0.8	5
7	5	0.8	5
8	5	0.7	5
9	5	0.9	5
10	5	0.9	5

For small systems (e.g., $m \leq 5$), the MDP is preferable to the other algorithms in the sense that it yields an optimal solution whose value function exceeds that of all the other algorithms; however, as indicated in Tables III–IV, the MDP runs out of memory for $T \geq 8$ when there are $m = 6$ subsystems and for $T \geq 4$ when there are $m = 10$ subsystems. By contrast, the remaining algorithms solve within a few minutes for all of the instances. On the instances that MDP can solve, the reported ADP-Concave and ADP-MWO value functions tend to be only slightly less than the MDP value function. Both ADP approaches and both myopic approaches are capable of solving larger instances in a reasonable time, but with ADP-Concave and ADP-MWO requiring more computation time than Myopic and Myopic-MWO; however, there are significant differences in the quality

TABLE III: Value function (expected number of successful missions) for instances in Section VII-A, where ">" denotes that the algorithm did not terminate within $259,200$s

m	Algorithm	$T=1$	$T=2$	$T=3$	$T=4$	$T=5$	$T=6$	$T=7$	$T=8$	$T=9$	$T=10$
	MDP	0.9949	1.9885	2.9827	3.9769	4.9711	5.9654	6.9596	7.9539	8.9481	9.9423
	Myopic	0.9949	1.9769	2.9339	3.8732	4.8279	5.7818	6.7421	7.7017	8.6612	9.6188
3	ADP-Concave	0.9949	1.9826	2.9733	3.9666	4.9478	5.9348	6.9299	7.9139	8.8989	9.8722
	Myopic-MWO	0.9899	1.9692	2.9249	3.8637	4.7859	5.7033	6.6445	7.5769	8.5028	9.4409
	ADP-MWO	0.9912	1.9811	2.9669	3.9469	4.9228	5.9097	6.8916	7.8738	8.8528	9.8212
	MDP	0.9934	1.9858	2.9792	3.9727	4.9662	5.9598	6.9533	7.9468	8.9403	9.9338
	Myopic	0.9934	1.9759	2.9229	3.8571	4.7962	5.6866	6.5992	7.5168	8.4518	9.3605
4	ADP-Concave	0.9934	1.9828	2.9681	3.9528	4.9374	5.9112	6.8989	7.8678	8.8227	9.8124
	Myopic-MWO	0.9872	1.9639	2.8953	3.7894	4.6986	5.6038	6.5221	7.4518	8.3809	9.2998
	ADP-MWO	0.9899	1.9748	2.9622	3.9417	4.9239	5.9008	6.8828	7.8514	8.8115	9.7997
	MDP	0.9937	1.9867	2.9805	3.9742	4.9679	5.9617	6.9554	7.9491	8.9428	9.9366
	Myopic	0.9937	1.9769	2.9494	3.8589	4.7899	5.6902	6.6312	7.5778	8.5212	9.4578
5	ADP-Concave	0.9937	1.9829	2.9695	3.9419	4.9196	5.8797	6.8522	7.8188	8.8022	9.7787
	Myopic-MWO	0.9819	1.9487	2.8797	3.7756	4.6772	5.5988	6.5299	7.4698	8.4034	9.3379
	ADP-MWO	0.9891	1.9771	2.9653	3.9327	4.8902	5.8582	6.8149	7.7926	8.7701	9.7479
	MDP	0.9899	1.9779	2.9821	3.9781	4.9756	5.9712	6.9702	$>$	$>$	$>$
	Myopic	0.9899	1.9642	2.8792	3.8153	4.7588	5.6712	6.6087	7.5623	8.5569	9.5411
6	ADP-Concave	0.9899	1.9712	2.9524	3.9298	4.9046	5.8522	6.8297	7.8272	8.8136	9.8018
	Myopic-MWO	0.9689	1.8992	2.8059	3.7276	4.6537	5.5588	6.5026	7.4999	8.4767	9.4628
	ADP-MWO	0.9888	1.9689	2.9512	3.9202	4.8886	5.8423	6.8101	7.7889	8.7668	9.7342
	MDP	0.9884	1.9765	2.9789	3.9755	4.9721	5.9711	$>$	$>$	$>$	$>$
	Myopic	0.9884	1.9626	2.8655	3.7836	4.7178	5.6455	6.6135	7.5511	8.5398	9.5274
7	ADP-Concave	0.9884	1.9708	2.9511	3.9178	4.8923	5.8489	6.8276	7.8142	8.8012	9.7889
	Myopic-MWO	0.9675	1.8839	2.7665	3.6855	4.6093	5.5349	6.5003	7.4867	8.4622	9.4589
	ADP-MWO	0.9882	1.9628	2.9444	3.9095	4.8802	5.8388	6.8124	7.7799	8.7578	9.7245
	MDP	0.9887	1.9889	2.9776	3.9742	4.9689	$>$	$>$	$>$	$>$	$>$
	Myopic	0.9887	1.9515	2.8539	3.7511	4.6851	5.6401	6.6388	7.5354	8.5112	9.5188
8	ADP-Concave	0.9887	1.9799	2.9397	3.9067	4.8612	5.8588	6.8135	7.8098	8.7865	9.7612
	Myopic-MWO	0.9662	1.8764	2.7466	3.6533	4.5616	5.5288	6.4994	7.4728	8.4411	9.4389
	ADP-MWO	0.9876	1.9616	2.9225	3.8788	4.8443	5.8222	6.8109	7.7675	8.7312	9.7188
	MDP	0.9871	1.9824	2.9722	3.9714	$>$	$>$	$>$	$>$	$>$	$>$
	Myopic	0.9871	1.9698	2.8452	3.7424	4.6717	5.6288	6.6211	7.5887	8.5513	9.5189
9	ADP-Concave	0.9871	1.9765	2.9274	3.8855	4.8599	5.8434	6.8298	7.7968	8.7598	9.7467
	Myopic-MWO	0.9654	1.8727	2.7323	3.6353	4.5487	5.5026	6.4875	7.4612	8.4267	9.4154
	ADP-MWO	0.9856	1.9735	2.9178	3.8616	4.8299	5.8034	6.8078	7.7599	8.7225	9.7079
	MDP	0.9865	1.9812	2.9785	$>$	$>$	$>$	$>$	$>$	$>$	$>$
	Myopic	0.9865	1.9621	2.8298	3.7369	4.6488	5.6005	6.6189	7.5728	8.5226	9.5065
10	ADP-Concave	0.9865	1.9722	2.9145	3.8612	4.8387	5.8221	6.8187	7.7865	8.7377	9.7325
	Myopic-MWO	0.9622	1.8653	2.6813	3.6133	4.5122	5.4998	6.4767	7.4436	8.4101	9.4021
	ADP-MWO	0.9843	1.9685	2.9188	3.8478	4.8112	5.7824	6.7788	7.7364	8.7101	9.6821

TABLE IV: Computation time (s) for instances in Section VII-A, where "$>$" denotes that the algorithm did not terminate within $259,200$s

m	Algorithm	$T=1$	$T=2$	$T=3$	$T=4$	$T=5$	$T=6$	$T=7$	$T=8$	$T=9$	$T=10$
	MDP	<1	<1	<1	1	1	1	1	2	2	2
	Myopic	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
3	ADP-Concave	<1	<1	<1	2	3	4	5	6	7	7
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	<1	<1	<1	1	1	2	2	3	4	4
	MDP	<1	2	46	95	144	199	240	292	340	401
	Myopic	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
4	ADP-Concave	<1	5	9	11	16	18	25	34	42	54
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	<1	2	4	6	10	12	15	21	26	34
	MDP	<1	103	8267	17009	25641	34441	43309	51402	60791	69275
	Myopic	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
5	ADP-Concave	<1	12	19	25	38	45	56	77	95	136
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	1	4	7	10	24	27	35	48	66	89
	MDP	25	5369	8992	16467	22560	60880	125788	$>$	$>$	$>$
	Myopic	<1	<1	<1	<1	<1	2	4	8	15	21
6	ADP-Concave	28	62	112	159	191	224	264	297	324	356
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	2	8	15	24	35	56	79	95	136	188
	MDP	60	8877	15225	30665	77445	145660	$>$	$>$	$>$	$>$
	Myopic	<1	<1	<1	<1	<1	<1	2	4	5	8
7	ADP-Concave	65	115	185	215	266	287	311	356	378	412
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	5	16	30	55	78	110	142	178	212	245
	MDP	109	12556	25665	66055	156223	$>$	$>$	$>$	$>$	$>$
	Myopic	<1	<1	<1	<1	<1	1	3	5	8	11
8	ADP-Concave	105	187	236	289	365	412	466	499	557	621
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	12	38	65	88	112	152	176	198	236	287
	MDP	256	26650	65990	167880	$>$	$>$	$>$	$>$	$>$	$>$
	Myopic	<1	<1	<1	<1	<1	3	6	10	12	16
9	ADP-Concave	244	297	322	345	388	421	465	499	534	565
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	42	90	125	142	167	216	244	267	297	343
	MDP	368	50340	159440	$>$	$>$	$>$	$>$	$>$	$>$	$>$
	Myopic	<1	<1	2	4	8	12	17	21	27	35
10	ADP-Concave	368	426	489	532	579	612	645	675	699	736
	Myopic-MWO	<1	<1	<1	<1	<1	<1	<1	<1	<1	<1
	ADP-MWO	62	122	136	185	211	244	245	274	322	364

TABLE V: Maintenance actions in break 0 for instances in Section VII-A with $T = 10$, where ">" denotes that the algorithm did not terminate within $259,500$s

	Reliability, r_i	0.8	0.8	0.7	0.9	0.9	0.8	0.8	0.7	0.9	0.9
m	Algorithm	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$	$i=9$	$i=10$
3	MDP	0	1	2	-	-	-	-	-	-	-
	Myopic	0	1	2	-	-	-	-	-	-	-
	ADP-Concave	0	1	2	-	-	-	-	-	-	-
	Myopic-MWO	0	1	2	-	-	-	-	-	-	-
	ADP-MWO	0	1	2	-	-	-	-	-	-	-
4	MDP	1	1	2	0	-	-	-	-	-	-
	Myopic	1	1	2	0	-	-	-	-	-	-
	ADP-Concave	1	1	2	0	-	-	-	-	-	-
	Myopic-MWO	1	1	2	0	-	-	-	-	-	-
	ADP-MWO	1	1	2	0	-	-	-	-	-	-
5	MDP	1	1	2	1	0	-	-	-	-	-
	Myopic	1	2	2	0	0	-	-	-	-	-
	ADP-Concave	1	1	2	1	0	-	-	-	-	-
	Myopic-MWO	1	2	2	0	0	-	-	-	-	-
	ADP-MWO	1	1	2	1	0	-	-	-	-	-
6	MDP	$>$	$>$	$>$	$>$	$>$	$>$	-	-	-	-
	Myopic	1	2	2	0	0	1	-	-	-	-
	ADP-Concave	1	1	2	1	0	1	-	-	-	-
	Myopic-MWO	1	2	2	0	0	1	-	-	-	-
	ADP-MWO	1	1	2	1	0	1	-	-	-	-
7	MDP	$>$	$>$	$>$	$>$	$>$	$>$	$>$	-	-	-
	Myopic	1	2	2	0	0	1	1	-	-	-
	ADP-Concave	1	1	2	1	0	1	1	-	-	-
	Myopic-MWO	1	2	2	0	0	1	1	-	-	-
	ADP-MWO	1	1	2	1	0	1	1	-	-	-
8	MDP	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$	-	-
	Myopic	1	1	2	0	0	1	1	2	-	-
	ADP-Concave	1	1	2	0	0	1	1	2	-	-
	Myopic-MWO	1	1	2	0	0	1	1	2	-	-
	ADP-MWO	1	1	2	0	0	1	1	2	-	-
9	MDP	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$	-
	Myopic	1	2	2	0	0	1	1	2	0	-
	ADP-Concave	1	1	2	1	0	1	1	2	0	-
	Myopic-MWO	1	2	2	0	0	1	1	2	0	-
	ADP-MWO	1	1	2	1	0	1	1	2	0	-
10	MDP	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$
	Myopic	1	2	2	0	0	1	2	2	0	0
	ADP-Concave	1	1	2	1	0	1	1	2	1	0
	Myopic-MWO	1	2	2	0	0	1	2	2	0	0
	ADP-MWO	1	1	2	1	0	1	1	2	1	0

of these solutions, with the ADP approaches consistently outperforming the myopic approaches. Fig. 3(a), 3(b), 3(c), and 3(d) display the optimality gap (defined as 100% times the ratio of the difference between an algorithm's attained value and the MDP's attained value, divided by the MDP's value) for the instances with $m = 3$, $m = 4$, $m = 5$, and $m = 6$.

For each value of m and each solution approach, the optimality gap is increasing in the number of missions. Further, ADP-Concave and ADP-MWO exhibit significantly smaller optimality gaps in comparison to Myopic and Myopic-MWO.

From Table V, all of the algorithms tend to maintain all failed components in subsystems 3 and 8, which have the smallest reliability. Whereas all failed components in subsystems 2 and 7 are maintained in Myopic and Myopic-MWO, these resources are used in MDP, ADP-Concave, and ADP-MWO to maintain one component in subsystems 2, 4, 7, and 9. By increasing the number of missions, there is a greater probability that the highly reliable components in subsystems 4 and 9 will fail during the planning horizon. Because both Myopic and Myopic-MWO maximize the reliability of the next mission, neither of these algorithms captures the effect of a prolonged planning horizon on the need to maintain components of subsystems 4 and 9. By incorporating information about the expected impact on future missions, both ADP-Concave and ADP-MWO are thus able to improve upon the myopic algorithms.

B. ADP Performance for Expanded Instance Set

In order to further test the performance of ADP, we vary the parameters in Table II. Table VI defines three settings for the component reliabilities r_i, $i \in \{1, \ldots, m\}$: original (R_o), high (R_h) and low (R_l). Table VII defines four settings (I, II, III, and IV) for the number of components n_i in each subsystem $i \in \{1, \ldots, m\}$, with I corresponding to equal number of components in each subsystem and II–IV becoming progressively less balanced (i.e., with some subsystems containing many more components than other subsystems). Table VIII defines four settings (I, II, III, and IV) for the maintenance resource requirements ρ_{i1} in each subsystem $i \in \{1, \ldots, m\}$, with setting I representing balanced resource requirements (i.e., maintenance actions in each subsystem have the same cost) and II–IV progressing toward maintenance actions costing much more in some subsystems than in others. All algorithms from Section VII-A are compared for $m = 10$ and $T \in \{3, 10\}$, and we report the results for all combinations of the three "component reliability" settings with

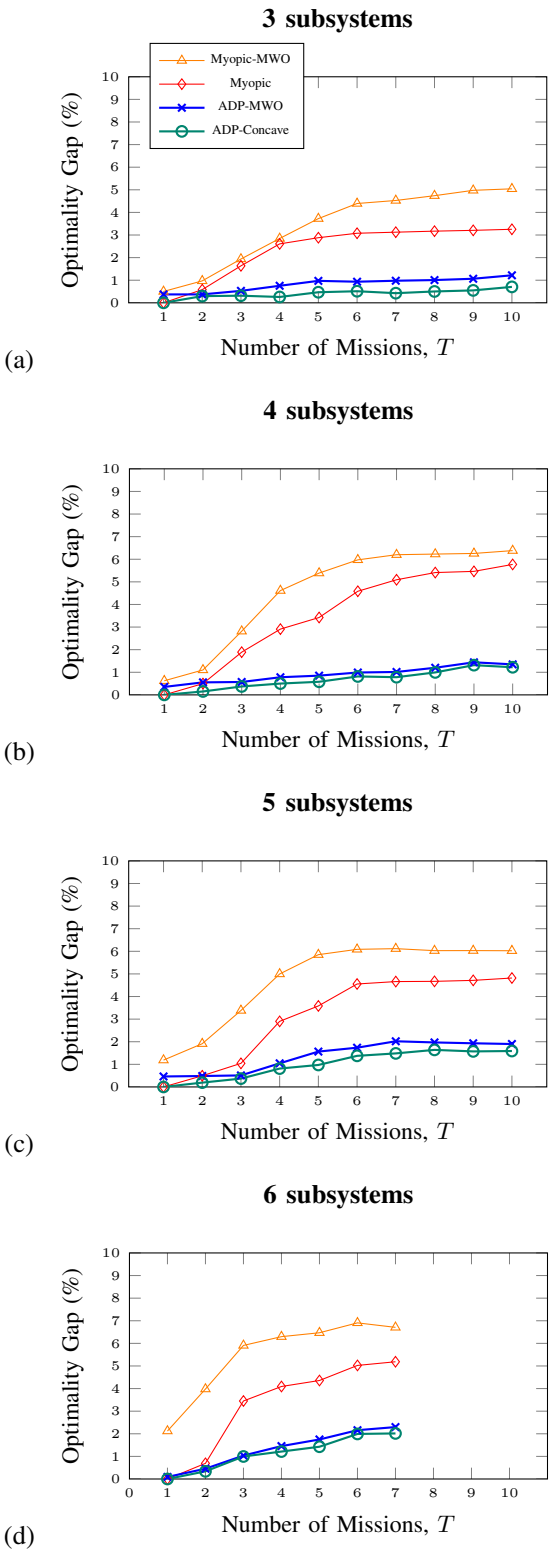

Fig. 3: Optimality gap versus number of missions for (a) 3 subsystems, (b) 4 subsystems, (c) 5 subsystems, and (d) 6 subsystems.

TABLE VI: "Component reliability" settings R_o, R_l, and R_h for instances from Section VII-B

	Component reliability, r_i		
Subsystem, i	R_o	R_l	R_h
1	0.8	0.75	0.85
2	0.8	0.75	0.85
3	0.7	0.65	0.75
4	0.9	0.85	0.95
5	0.9	0.85	0.95
6	0.8	0.75	0.85
7	0.8	0.75	0.85
8	0.7	0.65	0.75
9	0.9	0.85	0.95
10	0.9	0.85	0.95

TABLE VII: "Number of components" settings I, II, III, and IV for instances from Section VII-B

	Number of components, n_i			
Subsystem, i	I	II	III	IV
1	5	4	3	2
2	5	4	3	2
3	5	4	3	2
4	5	4	3	2
5	5	4	3	2
6	5	6	7	8
7	5	6	7	8
8	5	6	7	8
9	5	6	7	8
10	5	6	7	8

TABLE VIII: "Maintenance resource requirement" settings I, II, III, and IV for from Section VII-B

	Resource requirement, ρ_{i1}			
Subsystem, i	I	II	III	IV
1	5	4	3	2
2	5	4	3	2
3	5	4	3	2
4	5	4	3	2
5	5	4	3	2
6	5	6	7	8
7	5	6	7	8
8	5	6	7	8
9	5	6	7	8
10	5	6	7	8

either (in Table IX) the four "number of components" settings or (in Table X) the four "maintenance resource requirement" settings.

Tables IX and X indicate that the reported value functions for ADP-Concave and ADP-MWO are slightly less than the MDP for $T = 3$. The MDP runs out of memory for $T = 10$, but the remaining algorithms are capable of solving the large instances. Computationally, the remaining algorithms perform similarly to the results in Section VII-A: For all instances, ADP-Concave and ADP-MWO significantly outperform both Myopic and Myopic-MWO in terms of solution quality, both for $T = 3$ and $T = 10$. Furthermore, on the $T = 3$ instances, the optimality gaps (as depicted in Fig. 4 and 5) with respect to the exact MDP value are small for both ADP-Concave and ADP-MWO. The value functions in Tables IX decrease upon changing from balanced (setting I) to imbalanced (settings II, III, and IV) allocations of component redundancies; however, the value functions in Table X increase upon changing the maintenance resource requirement from balanced (setting I) to imbalanced (settings II, III, and IV) across subsystems.

In both Sections VII-A and VII-B, ADP-Concave slightly outperforms ADP-MWO in terms of solution quality for all of the instances, but at the expense of a significant increase in computation time. Thus, depending on the size of the "maintenance action" sets $A(s^t),\ s^t \in \mathcal{S}$, either ADP-Concave or ADP-MWO could be justified as a solution approach. Throughout the remainder of this paper, we consider a number of instances in which the maintenance action sets are larger due to either (in Section VII-C) increased numbers of components and maintenance budgets, or (in Section VII-D) allowing for resource-sharing across the time horizon. We have therefore opted to use ADP-MWO throughout the remainder of our computational results.

C. Sensitivity Analysis

In this section, we investigate the effect of changing input parameters on the value function and computation time. We vary the total budget level, number of components in each subsystem, and initial state for the 10-subsystem instance from Table II with $T = 5$, and solve using ADP-MWO. The results are presented in Tables XII and XIII. The value function (in Table XII) and computation time (in Table XIII) increase by increasing the number of components in each subsystem. Also, by increasing the total budget, the value function first increases and then levels off when there are enough resources to maintain all failed components in each subsystem during each mission. There is an increasing and then decreasing trend in computation time because little room for optimization exists when the budget is small or large. For example, when initial state is $[1, 1, \ldots, 1]$, all

TABLE IX: Value function (expected number of successful missions) for instance combinations defined in in Tables VI and VII

		n_i (I)		n_i (II)		n_i (III)		n_i (IV)	
r_i	Algorithm	$T=3$	$T=10$	$T=3$	$T=10$	$T=3$	$T=10$	$T=3$	$T=10$
	MDP	2.9246	-	2.8756	-	2.8129	-	2.5896	-
	Myopic	2.7788	9.4666	2.7582	9.4213	2.6995	9.3824	2.5112	9.2666
R_l	ADP-Concave	2.8788	9.7011	2.8518	9.6523	2.7987	9.5722	2.5799	9.5125
	Myopic-MWO	2.6544	9.3823	2.6325	9.3323	2.6112	9.2765	2.4501	9.0761
	ADP-MWO	2.8665	9.6688	2.8302	9.6339	2.7756	9.5323	2.5588	9.4898
	MDP	2.9785	-	2.9124	-	2.8434	-	2.6422	-
	Myopic	2.8298	9.5065	2.7886	9.5598	2.7234	9.5147	2.5334	9.4112
R_o	ADP-Concave	2.9145	9.7325	2.8687	9.6777	2.8001	9.6012	2.6102	9.5523
	Myopic-MWO	2.6813	9.4021	2.6415	9.4727	2.6098	9.4389	2.4289	9.3367
	ADP-MWO	2.9188	9.6821	2.8565	9.6515	2.7889	9.5812	2.5988	9.5289
	MDP	2.9889	-	2.9556	-	2.9024	-	2.8233	-
	Myopic	2.8545	9.6577	2.8231	9.5883	2.7786	9.5441	2.7272	9.4623
R_M	ADP-Concave	2.9434	9.7555	2.9088	9.7033	2.8612	9.6333	2.8022	9.5828
	Myopic-MWO	2.7128	9.6145	2.6896	9.5221	2.6478	9.4935	2.5923	9.4221
	ADP-MWO	2.9217	9.7092	2.8876	9.6777	2.8482	9.6016	2.7926	9.5445

TABLE X: Value function (expected number of successful missions) for instance combinations defined in in Tables VI and VIII

		ρ_{i1} (I)		ρ_{i1} (II)		ρ_{i1} (III)		ρ_{i1} (IV)	
r_i	Algorithm	$T=3$	$T=10$	$T=3$	$T=10$	$T=3$	$T=10$	$T=3$	$T=10$
	MDP	2.9246	-	2.9395	-	2.9537	-	2.9767	-
	Myopic	2.7788	9.4666	2.7999	9.4826	2.8329	9.5115	2.8633	9.5338
R_l	ADP-Concave	2.8788	9.7011	2.9145	9.7336	2.9396	9.7566	2.9656	9.7839
	Myopic-MWO	2.6544	9.3823	2.6888	9.4144	2.7118	9.4432	2.7339	9.4898
	ADP-MWO	2.8665	9.6688	2.8996	9.6856	2.9229	9.7147	2.9553	9.7446
	MDP	2.9785	-	2.9823	-	2.9888	-	2.9912	-
	Myopic	2.8298	9.5065	2.8555	9.5333	2.8832	9.5655	2.9146	9.5988
R_o	ADP-Concave	2.9145	9.7325	2.9345	9.7819	2.9637	9.8115	2.9746	9.8624
	Myopic-MWO	2.6813	9.4021	2.7033	9.4334	2.7256	9.4882	2.7638	9.5119
	ADP-MWO	2.9188	9.6821	2.9212	9.7635	2.9433	9.7889	2.9667	9.8223
	MDP	2.9889	-	2.9898	-	2.9912	-	2.9967	-
	Myopic	2.8545	9.6577	2.8676	9.6788	2.8777	9.7088	2.8897	9.7335
R_M	ADP-Concave	2.9434	9.7555	2.9635	9.7967	2.9726	9.8212	2.9812	9.8515
	Myopic-MWO	2.7128	9.6145	2.7889	9.6314	2.8019	9.6555	2.8225	9.6727
	ADP-MWO	2.9217	9.7092	2.9413	9.7399	2.9588	9.7833	2.9727	9.7999

of the failed components can be maintained with $B \geq 50$ in the first mission and the problem gets easier to solve by increasing the total budget level. Increasing the initial number of failed components in each subsystem reduces the value function and increases the computation time.

The best maintenance action for the first mission depends only on the total available budget for maintenance and does not change by varying the number of components in each subsystem or initial state. By increasing the total budget level, the number of maintained components in each subsystem changes according to the reliability of subsystems. For example, Table XI presents the ADP-MWO solutions under varying budget values for the instance with 5 components at each subsystem and initial state $[5, 5, \ldots, 5]$. The number of components maintained in each subsystem increases as the budget increases, and the least reliable components (in this case, subsystems 3 and 8) are more frequently maintained as compared to higher-reliability components.

D. The Shared-Budget Model

In this section, we report the results from applying the shared-budget model of Section VI to the instances of Table III. The total available budget for this model is the

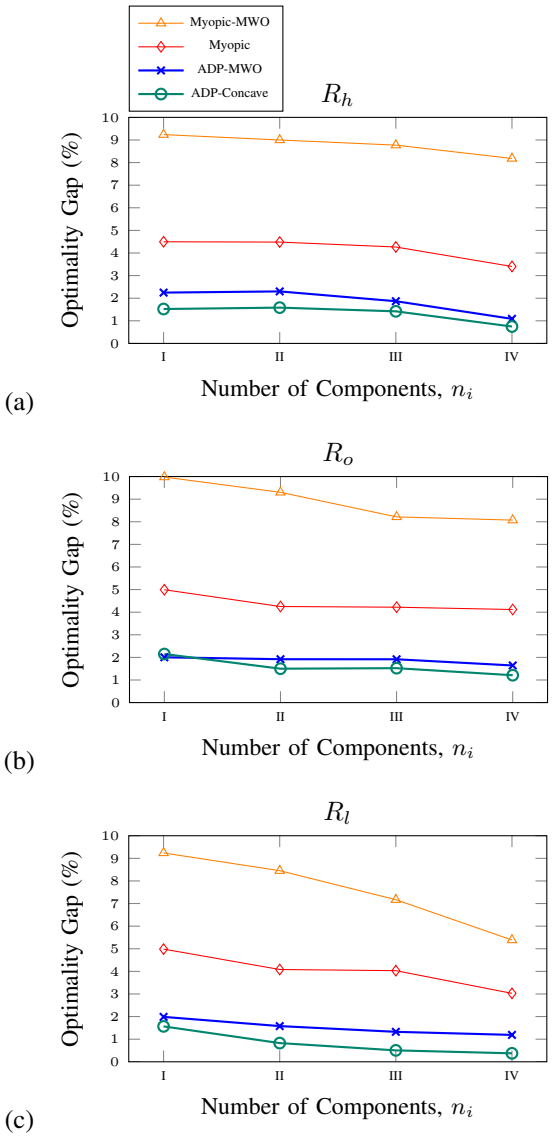

Fig. 4: Optimality gap versus number of components for (a) R_h, (b) R_o, and (c) R_l.

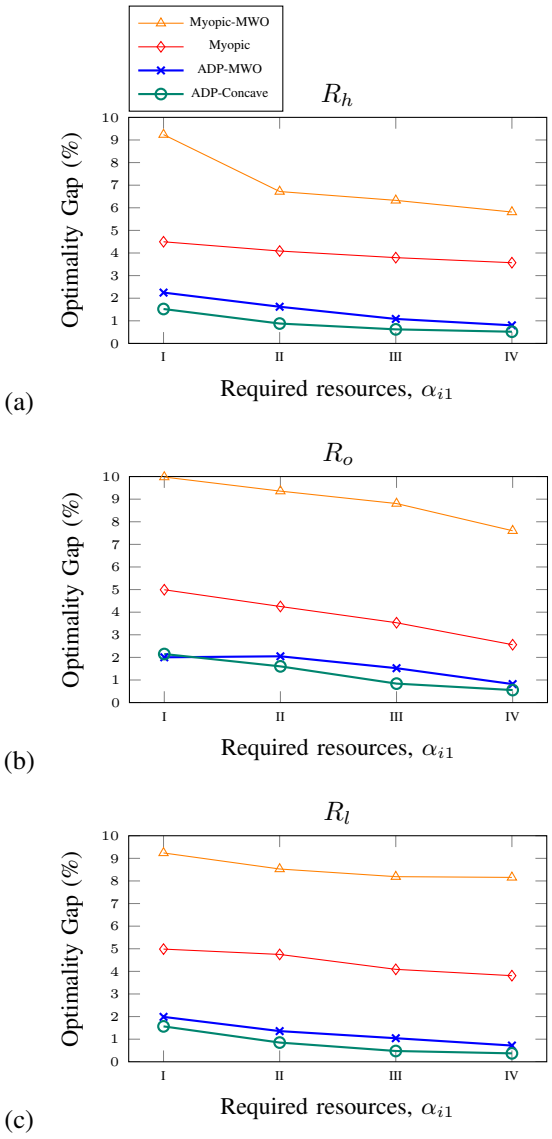

Fig. 5: Optimality gap versus number of required resource for (a) R_h, (b) R_o, and (c) R_l.

summation of budget available for maintenance at each mission for the original model, i.e., $B^0 = T\beta_1 = 5Tm$. The value function and computation time are represented in Tables XIV and XV for both MDP and ADP-MWO for $T \geq 2$ (because the the shared-budget model is identical to the original model for $T = 1$). Because the missions are related together in this model, there is no clear extension of the myopic approach. Thus, we have not implemented Myopic and Myopic-MWO for these instances. Based on the results, the MDP runs out of memory when $T \geq 8$ and there are $m = 4$ subsystems. Not surprisingly, this problem turns out to be more difficult to solve than previous problem because the state space is larger. However, ADP-MWO works well in comparison to the exact solution obtained based on MDP for smaller instances and it can solve larger instances in a reasonable time.

By sharing the total budget across the missions, there is on average a 2.14% increase in the value function in comparison with the previous problem. The maximum possible improvement relative to upper bound T, is on average 2.87%. Thus, by allowing the optimization

TABLE XI: First-period maintenance decision for instances in Section VII-C

	Num. components maintained			
Subsystem, i	$B = 50$	$B = 80$	$B = 100$	$B = 120$
1	1	2	3	3
2	1	2	2	3
3	3	4	5	5
4	0	0	0	1
5	0	0	0	0
6	1	2	3	3
7	1	2	2	3
8	3	4	5	5
9	0	0	0	1
10	0	0	0	0

model to determine when to expend resources, we obtain a significant improvement to the system's reliability.

Table XVI presents the maintenance actions of the first mission for both the original and shared-budget models based on ADP-MWO for a 10-subsystem instance, initial state $[2, 2, \ldots, 2]$, total budget $B^0 = 50T$, and $T \in \{1, \ldots, 10\}$. Under the the original model (with budget separated by period), all failed components in subsystems 3 and 8 are maintained they have the smallest reliability. Also, all failed components in subsystems 2 and 7 are maintained for $T \leq 5$, but for $T \geq 6$ these resources are reallocated to maintain one component in subsystems 2, 4, 7 and 9. Evidently, the longer planning horizon forces the model to place more importance on the increased possibility that components in the (highly reliable) subsystems 4 and 9 will fail during the planning horizon. As T increases, changes in the maintenance action for each subsystem are formatted in bold.

For the shared-budget model, the maintenance actions for $T \in \{1, 2\}$ are identical to the original. By increasing the number of missions (increasing the total available budget), the shared-budget model tends to maintain more failed components at the beginning of planning horizon.

VIII. Conclusion

In this paper, we develop an approximate dynamic program for the multi-mission selective maintenance problem. We implement a heuristic approach inside the ADP algorithm to solve the larger instances in a reasonable time. Using numerical experimentation, we evaluate the performance of the ADP algorithm in comparison with exact solution of the MDP. The results demonstrate that the developed algorithm performs well for small instances and solves larger instances in a reasonable time. The ADP's increased capabilities open the door to allowing the model to determine how the maintenance budget should be allocated across the planning horizon. The results for the shared-budget model suggest that incorporating this feature may lead to significant improvements in the system's reliability.

Future work may seek to further improve solution methods for this class of problems. Additionally, the development of ADP methodology for selective maintenance problems may open the door to other modeling extensions, including (i) selective preventive maintenance actions on a system with increasing failure rate components and (ii) selective maintenance on more generally structured (i.e., not series-parallel) systems.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1751191. The authors are thankful for the suggestions of three anonymous referees.

References

[1] L. M. Maillart, C. R. Cassady, C. Rainwater, and K. Schneider, "Selective maintenance decision-making over extended planning horizons," *IEEE Transactions on Reliability*, vol. 58, no. 3, pp. 462–469, 2009.

[2] W. Rice, C. Cassady, and J. Nachlas, "Optimal maintenance plans under limited maintenance time," in *Proceedings of the Seventh Industrial Engineering Research Conference*, May 1998.

[3] C. Chen, M. Q.-H. Meng, and M. J. Zuo, "Selective maintenance optimization for multi-state systems," in *1999 IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 3. IEEE, 1999, pp. 1477–1482.

[4] C. R. Cassady, E. A. Pohl, and W. Paul Murdock, "Selective maintenance modeling for industrial systems," *Journal of Quality in Maintenance Engineering*, vol. 7, no. 2, pp. 104–117, 2001.

[5] C. R. Cassady, W. P. Murdock, and E. A. Pohl, "Selective maintenance for support equipment involving multiple maintenance actions," *European Journal of Operational Research*, vol. 129, no. 2, pp. 252–258, 2001.

[6] R. Rajagopalan and C. R. Cassady, "An improved selective maintenance solution approach," *Journal of Quality in Maintenance Engineering*, vol. 12, no. 2, pp. 172–185, 2006.

[7] C. D. Dao, M. J. Zuo, and M. Pandey, "Selective maintenance for multi-state series–parallel systems under economic dependence," *Reliability Engineering & System Safety*, vol. 121, pp. 240–249, 2014.

[8] Q. Xu, L. Guo, and N. Wang, "Selective maintenance model and its solution algorithm for multi-state series-parallel system under economic dependence," in *2016 Chinese Control and Decision Conference (CCDC)*. IEEE, 2016, pp. 4781–4788.

[9] C. D. Dao and M. J. Zuo, "Selective maintenance for multistate series systems with s-dependent components," *IEEE Transactions on Reliability*, vol. 65, no. 2, pp. 525–539, 2016.

[10] L. Chen, Z. Shu, Y. Li, and X. Li, "Selective maintenance model considering time uncertainty," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, no. 16, pp. 2723–2727, 2012.

[11] A. Khatab, E.-H. Aghezzaf, I. Djelloul, and Z. Sari, "Selective maintenance for series-parallel systems when durations of missions and planned breaks are stochastic," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 1222–1227, 2016.

TABLE XII: Value function (expected number of successful missions) for instances in Section VII-C

Initial State	# of Components	$B = 50$	$B = 80$	$B = 100$	$B = 120$
$[1, 1, \ldots, 1]$	5	4.8932	4.9125	4.9351	4.9540
	10	4.8981	4.9292	4.9421	4.9594
	20	4.9113	4.9323	4.9511	4.9623
	30	4.9249	4.9418	4.9641	4.9747
$[2, 2, \ldots, 2]$	5	4.7451	4.8913	4.9108	4.9333
	10	4.8259	4.9132	4.9327	4.9481
	20	4.8531	4.9248	4.9457	4.9512
	30	4.8982	4.9310	4.9533	4.9692
$[3, 3, \ldots, 3]$	5	4.5862	4.8721	4.8955	4.9257
	10	4.6222	4.8834	4.9211	4.9308
	20	4.6457	4.8988	4.9367	4.9389
	30	4.6750	4.9061	4.9452	4.9574
$[4, 4, \ldots, 4]$	5	4.4611	4.6892	4.7462	4.8357
	10	4.4777	4.7143	4.7539	4.8636
	20	4.4979	4.7964	4.8212	4.8714
	30	4.5252	4.8123	4.8378	4.8976
$[5, 5, \ldots, 5]$	5	4.2123	4.5032	4.5618	4.6643
	10	4.2887	4.5859	4.6446	4.7562
	20	4.3388	4.6334	4.6692	4.7824
	30	4.4287	4.6458	4.7572	4.8661

TABLE XIII: Computation time (s) for instances in Section VII-C

Initial State	# of Components	$B = 50$	$B = 80$	$B = 100$	$B = 120$
$[1, 1, \ldots, 1]$	5	262	225	201	184
	10	589	521	487	425
	20	1085	988	912	886
	30	2669	2214	2036	1955
$[2, 2, \ldots, 2]$	5	288	388	310	266
	10	744	845	612	575
	20	1177	1233	1120	966
	30	2812	2966	2588	2289
$[3, 3, \ldots, 3]$	5	312	416	588	675
	10	655	989	1199	1295
	20	1222	1578	1655	1767
	30	2945	3175	3289	3424
$[4, 4, \ldots, 4]$	5	389	485	612	721
	10	766	1146	1293	1365
	20	1346	1715	1804	1916
	30	3156	3283	3377	3515
$[5, 5, \ldots, 5]$	5	467	545	688	811
	10	826	1322	1512	1495
	20	1480	1824	1927	2038
	30	3328	3478	3593	3664

TABLE XIV: Value function under shared budget (expected number of successful missions)

m	Algorithm	$T=2$	$T=3$	$T=4$	$T=5$	$T=6$	$T=7$	$T=8$	$T=9$	$T=10$
3	MDP	1.9897	2.9895	3.9808	4.9794	5.9769	6.9751	7.9726	8.9701	9.9674
	ADP-MWO	1.9839	2.9794	3.9766	4.9626	5.9522	6.9417	7.9221	8.9112	9.8999
4	MDP	1.9891	2.9866	3.9861	4.9838	5.9811	6.9766	-	-	-
	ADP-MWO	1.9878	2.9762	3.9636	4.9406	5.9318	6.9265	7.9181	8.8998	9.8865
5	MDP	1.9896	2.9889	3.9878	4.9815	5.9789	-	-	-	-
	ADP-MWO	1.9785	2.9739	3.9499	4.9289	5.9178	6.8924	7.8842	8.8688	9.8579
6	MDP	1.9809	2.9876	3.9882	4.9801	-	-	-	-	-
	ADP-MWO	1.9692	2.9675	3.9339	4.9135	5.8912	6.8777	7.8689	8.8511	9.8491
7	MDP	1.9799	2.9866	3.9845	-	-	-	-	-	-
	ADP-MWO	1.9686	2.9538	3.9248	4.9038	5.8813	6.8645	7.8524	8.8391	9.8314
8	MDP	1.9898	2.9812	-	-	-	-	-	-	-
	ADP-MWO	1.9679	2.9442	3.9111	4.8985	5.8731	6.8523	7.8401	8.8212	9.8101
9	MDP	1.9868	-	-	-	-	-	-	-	-
	ADP-MWO	1.9723	2.9388	3.9096	4.8902	5.8688	6.8416	7.8299	8.8178	9.8012
10	MDP	1.9845	-	-	-	-	-	-	-	-
	ADP-MWO	1.9686	2.9292	3.8925	4.8868	5.8515	6.8311	7.8119	8.8098	9.7988

TABLE XV: Computation time (s) under shared budget, where "$>$" denotes that the algorithm did not terminate within $259,200$s

m	Algorithm	$T=2$	$T=3$	$T=4$	$T=5$	$T=6$	$T=7$	$T=8$	$T=9$	$T=10$
3	MDP	<1	4	13	37	85	133	210	303	452
	ADP-MWO	<1	3	8	9	16	28	45	68	89
4	MDP	25	581	3539	11846	27266	$>$	$>$	$>$	$>$
	ADP-MWO	<1	8	11	17	36	66	98	154	189
5	MDP	715	10523	26889	65555	125777	$>$	$>$	$>$	$>$
	ADP-MWO	18	24	45	69	98	118	146	188	223
6	MDP	6676	28889	85545	168995	$>$	$>$	$>$	$>$	$>$
	ADP-MWO	49	112	140	201	268	312	406	512	609
7	MDP	12556	58885	115886	$>$	$>$	$>$	$>$	$>$	$>$
	ADP-MWO	136	195	242	298	366	442	566	678	726
8	MDP	66776	124445	$>$	$>$	$>$	$>$	$>$	$>$	$>$
	ADP-MWO	165	198	302	399	476	565	662	774	898
9	MDP	85550	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$
	ADP-MWO	245	297	369	489	569	667	754	882	926
10	MDP	116662	$>$	$>$	$>$	$>$	$>$	$>$	$>$	$>$
	ADP-MWO	336	412	564	641	755	869	978	1025	1156

[12] A. Khatab, E. H. Aghezzaf, C. Diallo, and I. Djelloul, "Selective maintenance optimisation for series-parallel systems alternating missions and scheduled breaks with stochastic durations," *International Journal of Production Research*, vol. 55, no. 10, pp. 3008–3024, 2017.

[13] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA: Athena Scientific, 2005, vol. 1.

[14] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2011, vol. 842.

[15] W. Scott and W. B. Powell, "Approximate dynamic programming for energy storage with new results on instrumental variables and projected bellman errors," Princeton University, Tech. Rep., 2012.

[16] D. F. Salas and W. B. Powell, "Benchmarking a scalable approximate dynamic programming algorithm for stochastic control of grid-level energy storage," *INFORMS Journal on Computing*, vol. 30, no. 1, pp. 106–123, 2017.

[17] D. R. Jiang and W. B. Powell, "Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming," *INFORMS Journal on Computing*, vol. 27, no. 3, pp. 525–543, 2015.

[18] J. Durante, J. Nascimento, and W. B. Powell, "Backward approximate dynamic programming with hidden semi-markov stochastic models in energy storage optimization," *arXiv preprint arXiv:1710.03914*, 2017.

[19] K. Papadaki and V. Friderikos, "Approximate dynamic programming for link scheduling in wireless mesh networks," *Computers & Operations Research*, vol. 35, no. 12, pp. 3848–3859, 2008.

[20] M. S. Maxwell, M. Restrepo, S. G. Henderson, and H. Topaloglu,

TABLE XVI: Best maintenance actions for break 0 under original and shared-budget models and varying planning horizons T

Subsystem, i		1	2	3	4	5	6	7	8	9	10
Reliability, r_i		0.8	0.8	0.7	0.9	0.9	0.8	0.8	0.7	0.9	0.9
Original Model	$T=1$	1	2	2	0	0	1	2	2	0	0
	$T=2$	1	2	2	0	0	1	2	2	0	0
	$T=3$	1	2	2	0	0	1	2	2	0	0
	$T=4$	1	2	2	0	0	1	2	2	0	0
	$T=5$	1	2	2	0	0	1	2	2	0	0
	$T=6$	1	**1**	2	**1**	0	1	**1**	2	**1**	0
	$T=7$	1	1	2	1	0	1	1	2	1	0
	$T=8$	1	1	2	1	0	1	1	2	1	0
	$T=9$	1	1	2	1	0	1	1	2	1	0
	$T=10$	1	1	2	1	0	1	1	2	1	0
Shared-Budget Model	$T=1$	1	2	2	0	0	1	2	2	0	0
	$T=2$	1	2	2	0	0	1	2	2	0	0
	$T=3$	**2**	2	2	0	0	**2**	2	2	0	0
	$T=4$	2	2	2	0	0	2	2	2	0	0
	$T=5$	2	2	2	0	0	2	2	2	0	0
	$T=6$	2	2	2	0	**1**	2	2	2	0	**1**
	$T=7$	2	2	2	0	1	2	2	2	0	1
	$T=8$	2	2	2	**1**	1	2	2	2	**1**	1
	$T=9$	2	2	2	**2**	1	2	2	2	1	**2**
	$T=10$	2	2	2	2	**2**	2	2	2	**2**	2

"Approximate dynamic programming for ambulance redeployment," *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 266–281, 2010.

[21] V. Schmid, "Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming," *European Journal of Operational Research*, vol. 219, no. 3, pp. 611–621, 2012.

[22] S. Ferrari and R. F. Stengel, "Online adaptive critic flight control," *Journal of Guidance Control and Dynamics*, vol. 27, no. 5, pp. 777–786, 2004.

[23] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 196, no. 2, pp. 509–515, 2009.

[24] W. B. Powell, "A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers," *Transportation Science*, vol. 30, no. 3, pp. 195–219, 1996.

[25] S.-W. Lam, L.-H. Lee, and L.-C. Tang, "An approximate dynamic programming approach for the empty container allocation problem," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 4, pp. 265–277, 2007.

[26] K. P. Papadaki and W. B. Powell, "An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem," *Naval Research Logistics*, vol. 50, no. 7, pp. 742–769, 2003.

[27] D. W. Coit and A. Konak, "Multiple weighted objectives heuristic for the redundancy allocation problem," *IEEE Transactions on Reliability*, vol. 55, no. 3, pp. 551–558, 2006.

[28] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[29] D. W. Coit and A. E. Smith, "Optimization approaches to the redundancy allocation problem for series-parallel systems," in *Fourth Industrial Engineering Research Conference Proceedings*, 1995, pp. 342–349.

[30] ——, "Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach," *Computers & Operations Research*, vol. 23, no. 6, pp. 515–526, 1996.

[31] D. E. Fyffe, W. W. Hines, and N. K. Lee, "System reliability allocation and a computational algorithm," *IEEE Transactions on Reliability*, vol. 17, no. 2, pp. 64–69, 1968.

[32] M.-S. Chern, "On the computational complexity of reliability redundancy allocation in a series system," *Operations Research Letters*, vol. 11, no. 5, pp. 309–315, 1992.

Khatereh Ahadi is a faculty member in Supply Chain and Operations Management program in Naveen Jindal School of Management at the University of Texas at Dallas. She received the Ph.D. degree in Industrial Engineering from the University of Arkansas, Fayetteville, AR, and the B.S. and M.S. degrees in Industrial Engineering from Sharif University of Technology, Tehran, Iran. Her primary research interest is in large-scale optimization, with applications in network optimization and maintenance optimization. Dr. Ahadi is a member of INFORMS and IISE.

Kelly M. Sullivan is an Assistant Professor in the Industrial Engineering Department at the University of Arkansas, Fayetteville, AR. His research focuses on advancing computational methodology for designing, maintaining, and securing complex systems.

Dr. Sullivan received a National Science Foundation CAREER Award in 2018 and was awarded the 2014 Glover-Klingman Prize for the best paper published in *Networks*. He is a member of IISE and INFORMS.