Motivated to apply sustainable supply chain principles to air-pollution control systems, this paper presents a dynamic inventory-management approach where substitution is possible to maintain these systems’ equipment. An air-pollution control system’s subsequent reliability depends on the replacement equipment selected. The corresponding problem is formulated as a stochastic dynamic program. Because the state and action space are prohibitively large, the approximate policy iteration algorithm is adapted to generate high-quality solutions. Therefore, this work replaces the value function with an affine combination of nonlinear basis functions and shows that a relaxation of the policy improvement step requires the solving of a mixed integer linear program. This approach helps in the designing of an algorithm that improves the quality of the approximation by solving a convex optimization problem. To assess the quality of resulting solutions, a lower bound is developed by considering a relaxation of the problem. In addition, two classes of heuristics are proposed based on a rolling horizon two-stage stochastic programming formulation of the problem and a standard base-stock ordering policy. The performance of proposed policies is tested on a variety of settings, and results show that the approximate dynamic programming (ADP) policies are near-optimal in the settings of interest and significantly outperform available benchmarks. The following analysis reveals that the proposed inventory replenishment policies resemble a base-stock policy with occasional deviations, and assignment and substitution decisions are determined by balancing the reliability with ordering, holding, and shortage costs.

Key words: Inventory management, substitution, reliability, approximate dynamic programming, sustainable supply chain

1. Introduction

Motivation. This research was prompted by the supply chain imperatives that the semiconductor manufacturing industry is facing related to the management of air-pollution control systems. With the nation’s emerging emphasis on sustainable manufacturing comes an increasing need for flexible and robust supply chains to provide the equipment required to maintain low levels of environmental impact (Wu and Pagell 2011). Manufacturing industries are confronted with significant pressure to decrease pollution, and they bear
the resulting costs imposed by environmental regulations (Brunnermeier and Cohen 2003). For instance, the U.S. Congress passed the Clean Air Act to regulate air emissions from stationary and mobile sources. This law authorizes the Environmental Protection Agency (EPA) to establish National Ambient Air Quality Standards that protect public health and welfare (EPA 2016).

In the motivating example for this research, a large semiconductor firm had a wafer fabrication facility in Europe that utilized two air-pollution control systems supplied by a U.S. manufacturer. One system reduced four chemical vapor deposition waste streams that include silane, a pyrophoric flammable hydride, and hydrogen fluoride, a poisonous corrosive, among other gases. The second air-pollution control system also lessened the emissions from four metal etch reactors. Those pollutants included chlorine, a toxic, and boron trichloride, a toxic corrosive. Since the air-pollution control system processed all eight reaction chambers, the disruption to production caused by closing the reactors would be severe and probably result in tens of thousands of dollars lost each day. However, given the nature of the waste and the total emissions flow of more than 200 liters per minute from each system, allowing the pollutants into the atmosphere was not a desirable option either because it could lead to fines totaling hundreds of thousands of dollars.

When an air-pollution control system requires replacement and its corresponding equipment is not available, discontinuing all production that relies on the failed air-pollution control device or continuing production without emission controls are expensive. Discontinuing production has benefits: the pollutants do not enter the environment, and the manufacturer avoids the high penalties imposed by environmental regulations. However, stopping production would also result in significant revenue losses because the semiconductor industry is mature and highly competitive. One key factor that adds more complexity to the inventory management of air-pollution control systems is the equipment replacement question. When the “native” equipment, made specifically for the individual system, is not available, it can sometimes be replaced with equipment made for, or native to, other air-pollution control systems. However, this substitution may increase the probability that an air-pollution control system will fail. In other words, the study presented here considers the reliability of an air-pollution control system as a function of what type of equipment is installed to maintain it.
This work presents a dynamic inventory management problem with substitution-dependent reliability, where a decision-maker, at the beginning of each period, may place orders for equipment, referred to hereafter as “service parts,” in the presence of lead time. The decision-maker then assigns the service parts to individual air-pollution control systems, hereafter referred to as “machines,” to minimize total cost.

**Background.** Three specific areas of research related to this work include the literature on inventory control of spare parts, supply chain of substitutable products, and the studies on approximate dynamic programming (ADP). Literature on spare parts inventory is quite rich. Spare parts in this context refers to parts that are used for maintenance of machines. Nahmias (1981) provided a review of early models on spare parts inventory control and corresponding solution techniques. For a more recent survey on spare parts inventory management see Driessen et al. (2015) and the references therein. Thonemann et al. (2002) studied a system approach to inventory control of spare parts where the fill rates for expensive items are lower as opposed to an item approach in which fill rates are constant across all items. An analytical model is presented and validated by a data set from a distribution center for computer spare parts. The results show the benefits that can be obtained by implementing a system approach to spare parts inventory control. Tiemessen et al. (2013) considered the dynamic allocation of spare parts from a network of warehouses to different customer classes to minimize the expected total delivery and penalty cost, where the base-stock levels in each warehouse are given. A one-step look-ahead policy is developed and its performance is compared with that of static allocation rules. Numerical results showed that savings of up to 8% is achievable using the dynamic policies. Howard et al. (2015) studied a two-echelon inventory system of spare parts where in case of a stock out in a local warehouse, an emergency shipment is requested from a central warehouse at additional cost. A model is presented that measures the benefits of using information on orders in the replenishment pipeline. However, current studies on spare parts inventory control do not consider the potential change in the stochastic evolution of the system due to replacement substitution. Therefore, existing methods in the literature do not apply in our setting.

Research concerning the supply chain of substitutable products has encompassed the development of optimization frameworks, as well as exact and/or heuristic solutions. Robinson (1990) considered a finite horizon, multi-outlet transshipment problem in which orders
are placed before demand is observed and transshipments are allowed after demand is realized. He showed that a base-stock ordering policy is optimal and designed a Monte Carlo heuristic for its computation. Bassok et al. (1999) formulated a single-period multiproduct inventory problem with full downward substitution and offered a framework of structural properties for optimal ordering policies. Rudi et al. (2001) considered two locations and examined the effects transshipment has on optimal ordering policies to each given location. Axsäter (2003) considered a single-period inventory system among parallel warehouses with compound Poisson demand, where transshipment between warehouses is possible. Shumsky and Zhang (2009) examined a multi-period allocation problem with upgrading: a retailer purchases the capacity of each product before the first period, and products are allocated to customers during each succeeding period. Nagarajan and Rajagopalan (2008) examined the structure of optimal inventory policies for retailers managing substitutable products. Their work showed that a partially decoupled policy is optimal under fairly general conditions.

The key difference that separates our work from the literature on inventory management with substitution is that in the literature, a firm allocates capacities to demands requested by customers where the demand is a sequence of independent and identically distributed (i.i.d.) random variables over time. However, in our settings the number of machines that will fail in a given period (which may be thought of as demand) depends on the service part installed (i.e., action taken). In particular, the work presented here examines a setting where any substitution configuration is possible, and more importantly, a machine’s reliability, in terms of the probability of failure, depends on the service part installed on it. Similarly, the presence of lead time, inventory replenishment, and non-i.i.d. sequence of multivariate random variables separates this work from the previous literature on revenue management with substitution; see Zhang (2011) and the references therein.

Reviewing the literature on approximate dynamic programming shows that a broad spectrum of operations management problems can be modeled as stochastic dynamic programs. However, solving the corresponding formulations is challenging due to the curse of dimensionality (Powell 2007). Many researchers have used ADP to come up with high-quality solutions for a variety of applications, e.g., allocating resources in service systems (Zhang and Adelman 2009, Jiang and Powell 2015); resource allocation in healthcare (Bertsimas
et al. 2013, Khademi et al. 2015); and supply chain management (Lai et al. 2010, Nadarajah et al. 2015). In applying ADP methodologies to inventory management, Van Roy et al. (1997) studied a two-echelon inventory problem in which products from a manufacturer are shipped to a warehouse and from the warehouse to the stores. Demand at each store is stochastic and stores and the warehouse are facing the problem of ordering and positioning their inventory to minimize total storage and transportation cost. Several ADP methods are used to provide solutions because the problem suffers from the curse of dimensionality. The results show that ADP solutions outperform standard order-up-to policies by up to 10%. Topaloglu and Kunnumkal (2006) considered a supply chain where a decision maker ships products from production plants to customers. Demand at each customer location is stochastic and the decision maker’s problem is to determine how much product to ship from each production plant to each customer location and how much inventory to keep at each plant to maximize profit. Two ADP methodologies are applied: one based on approximating the value function and linear programming representation of the optimality equation and the other based on Lagrangian relaxation and decomposing the problem. The results show that the ADP solutions significantly outperform available benchmarks. The work presented here proposes a novel application of ADP in inventory management of service parts for machines. Any substitution configuration is possible, and each substitution may result in a different stochastic system evolution in relation to machine reliability.

**Main contributions and results.** This work introduces a novel class of inventory control problems and considers general substitutions of service parts to machines, as well as inventory replenishment with lead time. A salient feature of this class of problems is that a machine’s reliability may depend on the service part installed. This class of problems is also different from inventory management problems with substitution discussed in the previous literature in that (i) this work considers the inventory replenishment decisions made in each period for every service part in the presence of lead time, and (ii) future demand, in terms of number of machines requiring service parts, depends on which service parts are assigned to which machines. The dynamic assignment of service parts to machines, with general substitution, during each period may be viewed as dynamic capacity allocation, where capacities and customers, respectively, resemble service parts and machines. However, in the dynamic capacity allocation setting, customer demand is a sequence of i.i.d. multivariate random variables over time. In the settings presented in this study, though,
the number of machines that require service depends on the service parts that have been installed on them, resulting in non-i.i.d. demand. Thus, a one-to-one analogy does not hold.

The problem is formulated as a stochastic dynamic program. Because the state and action space of this formulation grow exponentially, standard techniques fail to produce reliable solutions in the settings of interest. Therefore, ADP is used to find approximate solutions. Upper bounds are derived for the expected discounted cost minimization problem through an approximate policy iteration framework that substitutes the value function with an affine combination of nonlinear basis functions. Then, a relaxation of the resulting policy improvement step can be solved by a mixed integer linear program (MILP) which allows the set up of an iterative procedure that systematically improves the quality of the approximated value function by solving a convex optimization problem. In order to assess the quality of the solutions generated by the ADP approach, lower bounds on the value function are developed by constructing a relaxation of the problem amenable to exact solutions. In addition, several heuristic policies are developed based on the stochastic programming formulation of a two-stage problem that uses a rolling horizon approach to the multi-period problem, as well as base-stock inventory policies. The performance of the ADP policies is compared with heuristics across a broad range of problem settings via a Monte Carlo simulation. Furthermore, we relaxed several modeling assumptions made in Section 2 in a series of sensitivity analysis to test the robustness of the ADP policies.

Results indicate that ADP policies are near-optimal in a wide range of test instances, and their performance is significantly better than that of the heuristics. Analyzing the ADP policies shows that, in ordering service parts, ADP policies mirror a base-stock policy with occasional deviations (see Figure 3). In the settings with dense substitution feasibility graphs (see Section 2), the ordering policy resembles an aggregated base-stock policy, where the sum of all service parts in the inventory is equal to the base-stock inventory. However, the base-stock policy holds for each service part type in sparse substitution feasibility graphs. In assigning service parts to machines, the ADP policies consider the cost, as well as the reliability.

The rest of the paper is organized as follows: Section 2 states the problem setting, explains the modeling primitives and assumptions, and provides a stochastic dynamic programming formulation of the problem. Section 3 describes the ADP approach used to
derive high-quality policies and several heuristic policies, as well as the bounding system designed to assess the quality of the ADP policies. Section 4 develops several heuristic methods based on a rolling horizon two-stage stochastic programming formulation of the problem, as well as standard base-stock inventory policies. Section 5 describes the settings used to test the quality of the bounds and provides key insights for replenishment and replacement decisions in a variety of settings. Section 6 explores the robustness of the ADP policy and heuristics under milder assumptions. Section 7 concludes, and detailed proofs appear in the appendix.

2. Problem Formulation

This section provides an infinite-horizon Markov decision process (MDP) formulation of the problem. Let \( T = \{1, 2, \ldots \} \) be the set of decision epochs; \( \mathcal{M} = \{1, 2, \ldots, M\} \) denote the set of all machines, where \( M \) is the number of machines; and \( \mathcal{I} = \{1, 2, \ldots, I\} \) be the set of all service part types, where \( I \) is the number of service part types. As we study the inventory control of the key component of air-pollution control systems, we assume that each machine works with one service part. The feasibility of using a service part on a machine is described by a bipartite graph \( G = (\mathcal{M}, \mathcal{I}) \), where \( \mathcal{M} \) and \( \mathcal{I} \) are two disjoint sets of nodes, and an edge from service part \( i \) to machine \( m \) indicates that service part \( i \) is eligible for use on machine \( m \). Let \( e_{i,m} = 1 \) if an edge exists between service part \( i \) and machine \( m \), and 0, otherwise. We let the set of service parts that might be installed on machine \( m \) be \( F(m) \), and let the set of machines that can use service part \( i \) be \( \bar{F}(i) \), i.e., \( F(m) := \{i \in \mathcal{I} : e_{i,m} = 1\} \forall m \) and \( \bar{F}(i) := \{m \in \mathcal{M} : e_{i,m} = 1\} \forall i \). The reliability of a machine depends on the service part installed on it.

Let \( x_i(t) \) denote the number of service part \( i \) available at the beginning of period \( t \) and \( x(t) := (x_1(t), x_2(t), \ldots, x_I(t)) \). Also, let \( y_m(t) \) denote the service part installed on machine \( m \) at the beginning of period \( t \) and \( y(t) := (y_1(t), y_2(t), \ldots, y_M(t)) \). Note that \( x_i(t) \geq 0 \) for all \( i \), and for each \( m \), \( y_m(t) \in F(m) \cup \{0\} \), where 0 shows that machine \( m \) is down because its service part failed. The state of the system at period \( t \) is represented by \( s(t) := (x(t), y(t)) \in S \), where \( S \) is the set of all states.

At the beginning of each period, the decision-maker assigns service parts to the failed machines, based on availability of service parts, and then places an order for each part. We assume that lead times are one period: orders placed at the beginning of one period are
received at the beginning of the next period. If we assume that the lead time \( L \geq 2 \), the state of the system at the beginning of period \( t \) will contain all the information regarding orders placed in periods \( t - 1, t - 2, \ldots, t - \mathcal{L} + 1 \); see Zipkin (2008) and the references therein for a review of inventory control problems with lead time. Also, Goldberg et al. (2016) showed that as lead time grows in a lost-sales inventory system a significant amount of randomness results, such that a “smart algorithm” provides almost no benefit. Nonetheless, Section 6 tests the performance of the ADP policy in settings where the lead times may be greater than one and could vary across service parts. Figure 1 shows an overview of the problem during a given period.

![Figure 1 Schematic view of the decision process](image)

Let \( o_i(t) \) denote the number of service part \( i \) ordered at period \( t \) and \( o(t) := (o_1(t), o_2(t), \ldots, o_I(t)) \). We assume that if a machine is working, its service part will not be replaced. This assumption is aligned with the current practice of inventory management for air-pollution control systems’ service parts, as replacing a working service part may require discontinuing production that relies on that machine. Nonetheless, Section 6 measures the loss of optimality caused by this assumption via simulation of an ADP policy that may replace an item even though it is working on a machine. Let \( \bar{Y} := \{m \in M : y_m = 0\} \) denote the set of machines that require service part assignment. Define

\[
o^m_i(t) = \begin{cases} 
1 & \text{if service part } i \text{ is assigned to machine } m \text{ at time } t, \\
0 & \text{otherwise,}
\end{cases}
\]
and \( a(t) := (a^m_i(t) : \forall i, m) \). The action space at period \( t \) is given by

\[
A(s(t)) = \left\{(o(t), a(t)) : \sum_{m \in F(i) \cap \bar{Y}} a^m_i(t) \leq x_i(t), \forall i; \sum_{i \in F(m)} a^m_i(t) \leq 1, \forall m \in \bar{Y}\right\};
\]

\[
a^m_i(t) = 0, \forall i, m \in \mathcal{M} \setminus \bar{Y}, m \notin F(i); o_i(t) \geq 0, \forall i; a^m_i(t) \in \{0, 1\}, \forall i, m,
\]

where constraint \( \sum_{m \in F(i) \cap \bar{Y}} a^m_i(t) \leq x_i(t) \) guarantees that the number of service part \( i \) assigned must be less than or equal to the number of service part \( i \) available, and constraint \( \sum_{i \in F(m)} a^m_i(t) \leq 1 \) guarantees that at most one service part is assigned to machine \( m \). Let \( y^m_i(t) \) denote the state of machine \( m \) right after the service part assignment at period \( t \). Thus,

\[
y^m_i(t) = \begin{cases} 
\sum_{i \in F(m)} i a^m_i(t) & \text{if } y_m(t) = 0, \\
y_m(t) & \text{if } y_m(t) \neq 0.
\end{cases}
\]

We know that \( x_i(t+1) = x_i(t) + o_i(t) - \sum_{m \in F(i) \cap \bar{Y}} a^m_i(t) \) almost surely for all \( i \in \mathcal{I} \). Let \( p^m_i \) be the probability that machine \( m \) fails during a period when it is using service part \( i \). The transition probabilities are

\[
P\{y_{m(t+1)}|y_{m(t)}, a(t)\} = \begin{cases} 
1 & \text{if } y_m(t+1) = 0, y^m_i(t) = 0, \\
1 - p^m_i & \text{if } y_m(t+1) = i, y^m_i(t) = i, \\
p^m_i & \text{if } y_m(t+1) = 0, y^m_i(t) = i, \\
0 & \text{otherwise},
\end{cases}
\]

for all \( m \in \mathcal{M} \) and \( t \in T \). We assume that machines fail independently. Note that this assumption is not restrictive because the ADP framework in Section 3.1 can be extended to include settings where the failure probability of a machine may depend on the state of other machines. Therefore, one can compute \( P\{s(t+1)|s(t), a(t)\} \); the probability that the next state is \( s(t+1) \) when the current state is \( s(t) \) and action \( (o(t), a(t)) \) is taken.

We consider a decision-maker that minimizes the total expected discounted cost over a long horizon. Let \( c^o_i \) be the cost of ordering an item of service part \( i \), \( c^h_i \) be the holding cost of one item of service part \( i \), and \( c^s \) be the shortage cost if there is no available service part for a failed machine \( m \). Without loss of generality, we assume that costs such as environmental regulation costs are embedded in \( c^s \). Also, in order to avoid situations
where the optimal policy is trivial or nonsensical, we assume that \( c^m > \max_{i \in F(m)} (c^i, c^j) \) for all \( m \), i.e., the cost of a failed machine during a given period is greater than the ordering and holding cost of any service parts that might be installed on it. Let \( c(s, o, a) \) denote the expected immediate cost. Therefore,

\[
\begin{align*}
  c(s, o, a) &= \sum_{i \in I} c^i_o i + \sum_{i \in I} c^j_i (x_i - \sum_{m \in \bar{F}(i)} a^m) + \sum_{m \in \bar{Y}} c^m (1 - \sum_{i \in F(m)} a^m).
\end{align*}
\]

(1)

Let \( J_\pi(s) \) denote the total expected discounted cost when \( s(0) = s \) under policy \( \pi \in \mathcal{P} \), where \( \mathcal{P} \) denotes the set of all stationary non-anticipative policies. That is,

\[
J_\pi(s) = E\left\{ \sum_{t=0}^{\infty} \lambda^t c(s(t), \pi_t(s(t))) \mid s(0) = s \right\}, s \in S, \pi \in \mathcal{P},
\]

where \( \pi_t(s(t)) \) denotes the action selected by an admissible \( \pi \) in state \( s(t) \) at period \( t \), and \( 0 < \lambda < 1 \) is a discount factor. The decision-maker solves for

\[
v(s) = \inf_{\pi \in \Pi} \{ J_\pi(s) \},
\]

where \( \Pi \subseteq \mathcal{P} \) denotes the set of admissible policies under consideration and \( v(s) \) satisfies the Bellman optimality equation (Puterman 2005)

\[
v(s) = \min_{(o, a) \in A(s)} \{ c(s, o, a) + \lambda E(o, a) \{ v(s') \} \}, \forall s \in S,
\]

(2)

where the expectation is taken with respect to action \( (o, a) \), and \( s' \) denotes the state of the system at the beginning of the next period.

Because we do not consider the capacity limit for inventory, \( S \) is unbounded and solving formulation (2) in general is intractable. Another layer of difficulty in solving formulation (2) is the large size of the action space. The possible number of assigning service parts to machines grows exponentially, especially in dense substitution feasibility graphs. Moreover, with the current problem design, an unbounded number of service parts can be ordered. However, the next remark, by providing a natural upper bound, suggests that a finite number of service part orders over each period can be determined.

**Remark 1.** Let \( o^*_i \) be the optimal number of orders for service part \( i \). For a state \( s = (x, y) \), \( o^*_i \leq |\bar{F}(i)| - x_i \) for all \( i \).
Furthermore, if we impose an inventory capacity $K$, the size of the state space will be $|S| = (I + 1)^M \times \sum_{n=0}^{I} \binom{K-n-1}{n}$, which is exponential in model parameters. Nonetheless, it is straightforward to extend our solution framework to consider a capacity constraint by adding $\sum_{i \in I} (x_i + o_i - \sum_{m \in \mathcal{F}(i)} a_{im}) \leq K$ to the action space and all the solution techniques in Section 3 will hold.

We use Remark 1 in our solution framework in Section 3 to bound the number of orders placed for each service part. As discussed above, the size of state and action space is prohibitively large and standard techniques fail to solve formulation (2). The next section presents approximate solutions and provides bounds for the optimal solution.

3. Approximate Solutions and Performance Guarantee

This section details an approximate policy iteration approach that derives upper bounds on the optimal value function and uses a relaxation of the problem to derive lower bounds. In addition, the heuristics designed consider a single-period version of the problem and use a rolling horizon stochastic programming approach, as well as base-stock ordering policies.

3.1. Upper Bound

The standard policy iteration algorithm starts with an arbitrary policy $\pi^0$. Then, at iteration $n$, it evaluates $\pi^n$ by calculating $v^n(s)$ for all $s \in S$ by solving $v^n(s) = L_{\pi^n}v^n(s)$, where $L_{\pi^n}v^n(s) = c(s, \pi^n) + \lambda \mathbb{E}\{v^n(s')\}$. Next, it improves the policy by choosing the myopic policy relative to $v^n$, i.e., $\pi^{n+1}(s) \in \arg\min_{d \in \mathcal{D}^{MD}} \{c(s, d) + \lambda \mathbb{E}\{v^n(s')\}\}$, where $d \in \mathcal{D}^{MD}$ denotes an admissible decision rule in the set of stationary Markovian deterministic policies ($\mathcal{D}^{MD}$). This iterative procedure is continued until $\pi^{n+1} = \pi^n$. Because of the problem’s prohibitively large state and corresponding action space, the policy evaluation and improvement steps are intractable in this problem. In particular, each iteration of policy evaluation requires solving a $|S| \times |S|$ system of linear equations, where $|S| \sim \mathcal{O}(M^I I^M)$, and each iteration of policy improvement requires enumerating over all actions, which is in the order of $\mathcal{O}(M^I I^M)$. To overcome this issue, the value function is approximated by an affine combination of basis functions. Under this approximation, a relaxation of the policy improvement step for a state is equivalent to solving an MILP. Also, Monte Carlo simulation is used to estimate the expected values.

In order to develop an approximate policy iteration algorithm, we first approximate the value function by an affine combination of basis functions, i.e., $v(s) \approx \hat{v}(s) = \sum_j \eta_j \phi_j(s)$,
where $\phi_j(s)$ denotes the $j$th basis function and $\eta_j$ denotes its weight. The quality of the approximation depends on the choice of basis functions, which should be able to characterize the optimal value function (Powell 2007). Three types of basis functions are designed to represent the expected immediate cost and the expected future cost considering the special features of the problem.

In order to design a basis function for characterizing the expected future cost, note that, if the number of service part $i$ available at the beginning of a period is less than the number of machines that need service part $i$, a shortage cost will occur. If the number of service part $i$ available is greater than the number of machines that need service, a holding cost will occur. Let $1_{\{\cdot\}}$ be the indicator function. If each service part could be installed on each machine (i.e., full substitution graph), the number of machines that need service part $i$ at the beginning of a period will be $\sum_{m \in \bar{F}(i)} 1_{\{y_m=0\}}$, and any deviation from it results in a shortage or holding cost. Therefore, we set $|x_i - \sum_{m \in \bar{F}(i)} 1_{\{y_m=0\}}|$ for all $i \in \mathcal{I}$ as the first type of basis function, where $|\cdot|$ denotes the absolute value function. The function $|\cdot|$ is symmetric, implying that shortage and holding costs are treated equally. In the settings relevant to this study, however, the shortage costs are much higher than holding costs. Therefore, to consider the difference between costs, more weight is assigned to the failed machines, and the basis function is adjusted by $|x_i - \sum_{m \in \bar{F}(i)} 1_{\{y_m=0\}}| w_i$ with an added weight $w_i$. Our computational results show that adding parameter $w_i$ significantly improves the quality of the solutions.

To design a basis function for characterizing the immediate cost, note that, if machine $m$ fails at the beginning of a period, an immediate cost may arise. Therefore, we set $1_{\{y_m=0\}}$ for all $m \in \mathcal{M}$ as the second type of basis functions. The third basis function seeks to exploit the dependency of failure rates on the matching of parts to machines. Recall that $p_{im}$ denotes the probability of machine $m$ failure in a period if part $i$ is installed. Therefore, the failure rate for machine $m$ is estimated to be $\sum_{i \in F(m)} p_{im}^m 1_{\{y_m=i\}}$, which is the third basis function. We set $\hat{v}$ by

$$\hat{v}(s) := \alpha_0 + \sum_{i \in \mathcal{I}} \alpha_i |x_i - \sum_{m \in F(i)} 1_{\{y_m=0\}}| + \sum_{m \in \mathcal{M}} \beta_m 1_{\{y_m=0\}} + \sum_{m \in \mathcal{M}} \zeta_m \sum_{i \in F(m)} p_{im}^m 1_{\{y_m=i\}}. \quad (3)$$

Let $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_I)$, $\beta = (\beta_1, \beta_2, \ldots, \beta_M)$, $\zeta = (\zeta_1, \ldots, \zeta_M)$, and assume that $\alpha_i \geq 0$, $\forall i \in \mathcal{I}$, which allows the construction of an MILP formulation for the policy improvement step in Proposition 1.
The policy improvement step at iteration $n$ of the approximate policy iteration algorithm involves solving

$$\pi^n(s) := \arg \min_{(o,a) \in A(s)} \left\{ c(s, o, a) + \lambda \mathbb{E}_{(o,a)}(\hat{v}^n(s')|s) \right\},$$

where $\mathbb{E}_{(o,a)}(\cdot)$ is the expectation with respect to randomness in machines’ possibility of failure for action $(o,a)$. Let $v^n(s)$ be the value of (4) by setting the policy to $\pi^n(s)$, i.e.,

$$v^n(s) := \min_{(o,a) \in A(s)} \left\{ c(s, o, a) + \lambda \mathbb{E}_{(o,a)}(\hat{v}^n(s')|s) \right\}.$$

Next, we show that a relaxation of formulation (5) is equivalent to solving an MILP, which is used for the policy improvement step.

**Proposition 1.** For fixed $(\alpha, \beta, \zeta)$ and $s$, $v^R(s)$ is a relaxation of $v^n(s)$, i.e., $v^R(s) \leq v^n(s)$, where $v^R(s)$ is the optimal value of the following mixed integer linear program

$$v^R(s) = f(s, \alpha, \beta, \zeta) + \min \sum_{i \in I} (c_i^0 o_i + \lambda \alpha_i z_i) - \sum_{i \in I} \sum_{m \in Y} (c_i^m + c_i^m + \lambda(1 - p_i^m)(\beta_m - \zeta_m p_i^m)) a_i^m$$

s.t.

$$\sum_{i \in F(i)} a_i^m \leq x_i, \ \forall i,$$  

$$\sum_{i \in F(i)} a_i^m \leq 1, \ \forall m \in Y,$$  

$$- \sum_{m \in F(i)} a_i^m - w_i \sum_{j \in I} \sum_{m \in F(j) \cap Y} (p_j^m - 1) a_j^m + o_i - z_i \leq -x_i + w_i \sum_{m \in F(i) \cap Y} 1 + w_i \sum_{m \in F(i) \cap Y} p_{ym}^m, \ \forall i,$$  

$$w_i \sum_{m \in F(i) \cap Y} 1 - w_i \sum_{m \in F(i) \cap Y} p_{ym}^m, \ \forall i,$$  

$$a_i^m = 0, \ \forall i, m \in M \setminus Y, m \notin F(i),$$  

$$o_i, z_i \geq 0, \ a_i \in \mathbb{Z}_+, \ a_i^m \in \{0, 1\} \ \forall i, m,$$

and $f(s, \alpha, \beta, \zeta)$ is a constant term independent of action.

Proposition 1 provides $\hat{v}^n$-improving decision rules for a fixed state $s$. However, solving formulation (6) for each state may not be feasible because of the problem’s large state
space. Therefore, in our Monte Carlo simulation, to evaluate a policy, we use Proposition 1 upon visiting a state on the fly. That is, we solve formulation (6) only for states observed in the simulation. In the setting of interests, our computational experiments demonstrate that solving formulation (6) for a state is instantaneous.

Next, we develop an algorithmic approach to estimate \((\alpha, \beta, \zeta)\) to derive high-quality solutions. Consider an appropriately large finite horizon, and initialize \((\alpha, \beta, \zeta) = (\alpha^0, \beta^0, \zeta^0)\) to find \(\hat{v}^0(s)\) for states in \(\hat{S}\), where \(\hat{S}\) is a subset of \(S\). We construct \(\hat{S}\) by sampling states that are more likely to be visited by the “optimal” policy. The results that de Farias and Roy (2004) presented for the linear programming approach to ADP suggest that, by sampling constraints (states) reliable solutions may be determined. Consider a distribution \(\{\varphi(s) : s \in S\}\) such that \(\varphi(s) > 0\) for all \(s \in S\). As noted by de Farias and Roy (2004), \(\varphi\) regulates the quality of approximation across \(S\), and can therefore be used to target certain regions of the state space where one aims to obtain better approximations. See Appendix B for the detailed construction of \(\varphi(\cdot)\).

Let \(\pi^0\) be a myopic policy relative to \(\hat{v}^0(s)\). In each iteration of the approximate policy iteration, the policy should be evaluated. Let \(\hat{v}^n(\cdot)\) represent the value function approximation at iteration \(n\). For the policy evaluation step, calculate \(\hat{v}^n(\cdot)\). The study presented here proposes the following procedure for this calculation. Start from an initial state \(s\); use Monte Carlo to simulate the system, and upon observing a state, find actions to take by solving the MILP in (6); and calculate the total discounted cost for that realization of the system. Let \(C^r(s)\) be the total discounted cost of the realization of the system, starting from state \(s\) in replication \(r\). Consider \(C^r(s)\) as the simulated value function for state \(s\) in replication \(r\). Let \(R_s\) be the number of replications of the Monte Carlo simulation for each state \(s\). To estimate \((\alpha, \beta, \zeta)\) we solve the following optimization problem

\[
\min_{(\alpha, \beta, \zeta)} \sum_{s \in \hat{S}} R_s \sum_{r=1}^{R_s} \varphi(s) \left( C^r(s) - \hat{v}(s) \right)^2
\]

\[\text{s.t.}\]

\[\alpha_i \geq 0, \quad \forall i \in I,\]

which minimizes the (adjusted) squared error between the approximate value function and the simulated value function. We use quadratic programming solver of Gurobi Optimizer to solve formulation (7), which is a convex optimization problem. Computational experiments detailed in Section 5 show that solving formulation (7) in our settings is instantaneous.
Finally, we summarize our algorithmic approach: in each step of the algorithm, we simulate the system and at each period use formulation (6) to determine which service part to assign and what to order. We compute the total discounted cost for each replication, i.e., $C_r(s)$ and use formulation (7) to tune $(\alpha, \beta, \zeta)$. This procedure continues until convergence in some norm is achieved. Algorithm 1 formalizes this approach.

**Algorithm 1** Approximate policy iteration

Set $n = 0$, $\epsilon > 0$, and $(\alpha, \beta, \zeta) = (\alpha^0, \beta^0, \zeta^0)$.

while $|\hat{v}^n(s) - \hat{v}^{n-1}(s)| < \epsilon$ or $n \neq 0$ do

Policy improvement: Find a myopic policy induced by $\hat{v}^n(s)$ (i.e., $(\alpha^n, \beta^n, \zeta^n)$) by solving formulation (6).

Policy evaluation: Use Monte Carlo to simulate the system; find actions for each state visited by the simulation by calculating formulation (6); and determine the total discounted cost for each initial state $s$ and replication $r$.

Projection: Use $C_r(s)$ from the Monte Carlo simulation and solve formulation (7) to estimate $(\alpha^{n+1}, \beta^{n+1}, \zeta^{n+1})$ for the next iteration.

Bertsekas and Tsitsiklis (1996, Proposition 6.2) provided a performance guarantee for the approximate policy iteration algorithm. However, the conditions necessary for the theoretical performance guarantee of the algorithm are difficult to verify in practice. In fact, these conditions require that the error induced by the regression counterpart of formulation (7) is exactly known and the expectations are calculated exactly, which is not the case in our settings. Therefore, the approach presented here constructs a lower bound to the problem via a relaxation technique.

### 3.2. Lower Bound

Finding lower bounds for stochastic dynamic programs (in minimization settings) is difficult in general. Brown et al. (2010) developed a bounding technique based on information-relaxation and duality. Their technique requires solving a deterministic optimization problem that uses a sample path of uncertainty realization, which cannot be done in the setting presented here because of the latter problem’s complexity. Instead, this section proposes a different technique to compute lower bounds.
We consider a relaxation of our formulation which can be solved in exact form. Pursuant to this goal, we consider two layers of relaxations to construct the bounding formulation. First, we assume that the substitution feasibility graph \( G = (M, I) \) is biclique, i.e., each service part \( i \) can be used on any machine \( m \). Second, we assume that (1) the probability of failure for each machine in the bounding formulation \( p_{LB} \) is equal to the original problem’s minimum probability of failure, i.e., \( p_{LB} = \min_{i \in I, m \in M} \{p_{m}^{i}\} \); (2) the ordering cost for each service part in the bounding formulation \( c_{LB}^{o} \) equals the minimum of ordering costs in the original problem, i.e., \( c_{LB}^{o} = \min_{i \in I} \{c_{i}^{o}\} \); (3) the holding cost for each service part during a period in the bounding formulation \( c_{LB}^{h} \) equals the minimum of holding costs in the original problem, i.e., \( c_{LB}^{h} = \min_{i \in I} \{c_{i}^{h}\} \); (4) the shortage cost in the bounding system \( c_{LB}^{s} \) is the minimum of each machine’s failed cost in the original problem, i.e., \( c_{LB}^{s} = \min_{m \in M} \{c_{m}\} \). Therefore, by this construction, an optimal solution to the bounding system provides a lower bound on the optimal value function of the original problem, because (i) any feasible assignment and ordering action in the original problem is also feasible in the bounding system, (ii) the probability of machine failure in the bounding system is less than or equal to that of the original problem, and (iii) the cost incurred from any event accounted for in the bounding system is less than or equal to that in the original problem.

Next, we provide an infinite-horizon MDP formulation of the bounding system. The bounding system’s construction suggests that service parts and machines are homogeneous. Therefore, let \( x'(t) \in \{0, 1, \ldots M\} \) (by Remark 1) and \( y'(t) \in \{0, 1, \ldots M\} \) denote the number of service parts available and the number of failed machines at the beginning of period \( t \), respectively. Let \( s'(t) := (x'(t), y'(t)) \in S' \) denote the state of the system at period \( t \), and \( S' \) denote the state space whose size is polynomial in the bounding system, i.e., \(|S'| = M^2\). (Recall that the state space of the original problem is exponential even after an inventory capacity constraint is added.) Because of the homogeneity of service parts and machines in the bounding system, the order of assignments becomes irrelevant. Define \( a'(t) \) as the number of service parts assigned to failed machines and \( o'(t) \) as the number of service parts ordered at period \( t \). The action space is represented by

\[
A'(s') = \left\{ (o'(t), a'(t)) : a'(t) \leq x'(t); a'(t) \leq y'(t); o'(t) \leq M - x'(t); o'(t), a'(t) \in \mathbb{Z}_+ \right\}.
\]
The size of the action space is also polynomial, i.e., \(|A'(s')| \leq M^2\). In addition, \(x'(t + 1) = x'(t) - a'(t) + o'(t)\) and it is straightforward to show that \(a'(t) = \min\{x'(t), y'(t)\}\).

Let \(Y\) denote a binomial random variable with parameters \(M\) and \(p_{LB}\). The transition probabilities are given by

\[
P\{y'(t+1)|y'(t), o'(t), a'(t)\} = \begin{cases} P\{Y = y'(t+1) - y'(t) + a'(t)\} & \text{if } y'(t+1) \geq y'(t) - a'(t), \\ 0 & \text{otherwise.} \end{cases}
\]

The expected immediate cost for the bounding system is given by

\[
c'(s', o', a') = c_{LB}^o o' + c_{LB}^h (x' - a') + c_{LB}^m (y' - a').
\]

The optimal value function of the bounding system, \(v'(s')\), satisfies

\[
v'(s') = \min_{(o', a') \in A'(s')} \{c'(s', o', a') + \lambda \mathbb{E}\{v'(s'')\}\}, \forall s' \in S', \tag{8}
\]

where \(s''\) denotes the state of the system at the beginning of the next period. Note that, because demand in each period is a sequence of non-i.i.d. random variables, the available order-up-to policies may not be optimal. Therefore, formulation (8) is solved using the standard value iteration algorithm (Puterman 2005, Section 6.3).

The quality of the lower bound designed in this section depends on (at least) two factors: (i) the density of the substitution feasibility graph in the original problem, (ii) the variance of input parameters \(c_i^o \forall i\), \(c_i^h \forall i\), and that of \(c_m \forall m\) in the original problem. For example, if most of the service parts can be installed on most of the machines, the structure of the bounding system will be similar to the original problem and one may expect a good bound. Also, if the variance of input parameters in the original model is small, the homogeneity assumption of machines and service parts in the bounding system may be more likely to hold. The computational results presented in Section 5 confirm these observations.

4. Benchmark Policies

This section details two benchmark policies designed to compare their performance with the policies suggested by the ADP approach. The first benchmark uses a two-stage stochastic programming formulation of the problem in a rolling horizon fashion, i.e., in each period, the decision-maker solves a decision problem for the current and the next period. The second class of benchmarks follows a base-stock policy in ordering service items and solves an assignment problem to assign service parts to failed machines.
4.1. Two-Stage Stochastic Programming Benchmark

Consider a rolling horizon single-period version of the problem. In the single-period version, the decision-maker observes the state of machines $y$ and the inventory of service parts $x$ at the beginning of the period. Then, the decision-maker places orders for service parts that will be received at the end of the period, when the states of the machines are observed, and the appropriate service parts are selected for failed machines based on the service parts’ availability. Therefore, in the first stage, the decision-maker decides how many orders to place for each service part, and in the second stage, after observing the random failures and receiving orders, the decision-maker decides how to assign service parts to machines. This problem is similar to the multi-dimensional newsvendor problem with substitution and correlated demands, where the substitution is determined by the decision maker, not the customer.

Let $\xi := (\xi_1, \xi_2, \ldots, \xi_M)$ be the vector of machine failure realizations, where $\xi_m$ is 1 if machine $m$ fails in a period and 0, otherwise. If no service part is installed on machine $m$ in the beginning of the period, set $\xi_m = 1$. Let $Q(x, y, o, \xi)$ be the cost for an initial inventory level $x$, with a machine state $y$, a given order of $o$, and failure realization $\xi$. The decision-maker at the beginning of a period solves for

$$
\min_{o} \sum_{i=1}^{I} c^o_i o_i + \mathbb{E}_{\xi}\{Q(x, y, o, \xi)\} \\
\text{s.t.} \\
o_i \geq 0, o_i \in \mathbb{Z}_+, \forall i \in \mathcal{I},
$$

where

$$
Q(x, y, o, \xi) = \min_a \sum_{m \in \mathcal{M}} c^m(\xi_m - \sum_{i \in F(m)} a^m_i) + \sum_{i \in \mathcal{I}} c^h_i (x_i + o_i - \sum_{m \in F(i)} a^m_i) \\
\text{s.t.} \\
\sum_{i \in F(m)} a^m_i \leq \xi_m, \forall m \in \mathcal{M}, \\
\sum_{m \in F(i)} a^m_i \leq x_i + o_i, \forall i \in \mathcal{I}, \\
a^m_i \in \{0, 1\} \forall i, m.
$$

Since the constraints in formulation (10) are of assignment type, the binary variables can be replaced with their continuous counterparts (the corresponding matrix is totally unimodular) and therefore, the $L$-shaped method solves formulation (9). In our computational
study we solved formulation (9) exactly. However, the random variable \( \xi \) has \( 2^M \) realizations, and for large \( M \), solving formulation (9) becomes computationally challenging. One may use Monte Carlo methods such as sample average approximation or stochastic decomposition methods to solve formulation (9) (Birge and Louveaux 2011).

4.2. Heuristic Benchmarks

This section introduces a class of heuristics that follow a base-stock policy in ordering service parts because base-stock policies have shown to be optimal under fairly general conditions and are frequently used in practice: see Miyaoka and Hausman (2004) and the references therein. For assigning service parts to failed machines, this class of heuristics solve the assignment problem (10), which considers the possibility of substitution. In order to calculate how many of service part \( i \) to order at each period \( (o_i^H) \), we adapted the approach presented in Veinott (1965a,b), which finds the base-stock levels for a periodic inventory system of a single item. Under the heuristic policy, let \( x_i^H \) denote the inventory level of service part \( i \) at the beginning of the period; \( y_i^H \) denote the number of failed machines that can use service part \( i \) at the beginning of the period; \( c_o^i \) denote the ordering cost of service part \( i \); \( c_h^i \) denote the holding cost of service part \( i \); and \( \hat{c}_i \) denote the penalty cost for unavailability of service part \( i \). Thus, the optimal order for service part \( i \) in heuristic, \( o_i^H \), is computed by

\[
G_i(x_i^H - y_i^H + o_i^H) = \frac{\hat{c}_i - c_o^i}{c_i + c_h^i},
\]

where \( G_i(\cdot) \) denotes the cumulative distribution function (cdf) for the demand of service part \( i \) in the period. In order to adapt this approach, the demand should be estimated in our context, as well as the penalty cost. To that end, we assume that the heuristic policy does not consider the possibility of substitution when computing the ordering decision of service parts. Therefore, in order to estimate the demand for a service part \( i \), we consider machines belonging to \( \bar{F}(i) \). In this construction, one machine may be considered in calculating the ordering decision of several service parts. In particular, machine \( m \) is considered to estimate the demand of all service parts \( i \in F(m) \). This construction is likely to overestimate the demand for each service part, especially in dense substitution feasibility graphs. Section 5.2 investigates the implications of this overestimation in a variety of problem settings. Because the probability of machine failure may be different for each machine, we further assume
that machines are homogenous for the ordering phase. Thus, the number of machines that
need service part $i$ (fail during one period) has a binomial distribution with parameters
$n_i = |\bar{F}(i)|$ and $\hat{p}_i$, i.e., $G_i(\cdot) \sim \text{binomial}(n_i, \hat{p}_i)$. Implementing this framework requires a
calibration for $\hat{c}_i$ and $\hat{p}_i$. Estimating $\hat{c}_i$ and $\hat{p}_i$ is determined by one of three scenarios:
(i) optimistic scenario, (ii) robust scenario, and (iii) expected scenario. In the optimistic
scenario, set $\hat{c}_i = \min_{m \in \bar{F}(i)} c^m$, and $\hat{p}_i = \min_{m \in \bar{F}(i)} P_{y_m}^m$. In the robust scenario, set $\hat{c}_i = \max_{m \in \bar{F}(i)} c^m$, and $\hat{p}_i = \max_{m \in \bar{F}(i)} P_{y_m}^m$. In the expected scenario, set $\hat{c}_i = \frac{\sum_{m \in \bar{F}(i)} c^m}{n_i}$, and
$\hat{p}_i = \frac{\sum_{m \in \bar{F}(i)} P_{y_m}^m}{n_i}$. We let $o_{i}^{\text{OP}}$, $o_{i}^{\text{RO}}$, and $o_{i}^{\text{EX}}$ denote, respectively, the optimal order for
optimistic, robust, and expected scenarios. The next proposition formalizes the natural
relationship among the amounts ordered for each scenario.

**Proposition 2.** For a state $s$, $o_{i}^{\text{OP}} \leq o_{i}^{\text{EX}} \leq o_{i}^{\text{RO}}$ for all $i \in I$.  

Finally, this heuristic works in the Monte Carlo simulation as follows. At the beginning
of period $t$, the decision-maker observes the state of the system $s(t)$ and places orders for
each service part determined by formulation (11). Then, a realization of $\xi$ is generated by
the Monte Carlo simulation, and the system moves to the new state $s(t+1)$. The decision-
maker assigns service parts to machines according to formulation (10), and this heuristic
policy proceeds until the end of the planning horizon.

5. **Computational Experiments**

The quality of the proposed upper and lower bounds is tested, along with benchmark
policies, on a diverse set of problem instances. Then, the performance of ADP policies is
analyzed to derive managerial insights.

5.1. **Experimental Setup**

Two classes of problem instances are considered to test the performance of the ADP and
heuristics, as well as the lower bound. The first problem instance called “designed setup”
considers three service parts and six machines. Each machine has a native service part,
which has the lowest failure probability. The problem instance is designed such that the
more expensive a service part, the higher its reliability. The data on reliability, holding
cost, purchasing cost, and penalty cost for the designed setup is reported in Table 1.

A variety of “random instances” is also developed to test the performance of different
policies for a wide range of problems. In particular, the random test problems are divided
Table 1 Parameter values for the designed setup

<table>
<thead>
<tr>
<th>i</th>
<th>(m)</th>
<th>(c^i)</th>
<th>(c^h)</th>
<th>(p^m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>0.05</td>
<td>0.13</td>
</tr>
<tr>
<td>2</td>
<td>900</td>
<td>95</td>
<td>0.12</td>
<td>0.07</td>
</tr>
<tr>
<td>3</td>
<td>800</td>
<td>90</td>
<td>0.17</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>(c^m)</td>
<td>20,000</td>
<td>19,750</td>
<td>19,500</td>
</tr>
</tbody>
</table>

into two groups based on the density (\(\delta\)) of the substitution feasibility graph. In a fully dense graph, each service part can be installed on any of the machines, but in a sparse graph, only two-thirds of the service parts can be used on a machine, or \(\delta \approx 67\%\). The sparse graphs are constructed such that each machine is connected to at least one service part. The random instances are also divided into two groups based on the service parts’ reliability. On average, service parts fail with 7.5% probability in the reliable cases and with 17.5% probability in the less reliable cases. Also, in each class of problems, several combinations of the number of service parts and machines is considered. Table 2 reports the distribution of each random parameter for this set of experiments. The actual data values are available at http://burak.people.clemson.edu/research.html.

Table 2 Parameter values and distributions for random instances

<table>
<thead>
<tr>
<th>(I)</th>
<th>(M)</th>
<th>(p^m)</th>
<th>(c^h)</th>
<th>(c^o)</th>
<th>(c^m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\in {3, 4, 5, 6})</td>
<td>(\in {6, 12})</td>
<td>(U(0.05, 0.10)) for settings with high reliability</td>
<td>(U(0.15, 0.20)) for settings with low reliability</td>
<td>(U(50, 150))</td>
<td>(U(500, 1500))</td>
</tr>
</tbody>
</table>

Note: \(U(a, b)\) denotes the uniform distribution between \(a\) and \(b\).

In our experiments, planning horizon \(T\) is set to a large value (i.e., \(\lambda^T < \epsilon\)) to ensure a good approximation for the infinite horizon MDPs. The sampling size for solving formulation (7) is set to \(|\hat{S}| = 100\), which balances the solution quality and computational time. Increasing the sample size to 200 and 500 did not significantly improve the quality of solutions. The number of replications for each initial state in the Monte Carlo simulation is set to \(R_s = 10\). The number of ADP iterations in Algorithm 1 is set to \(n = 10\). For example, Figure 2 shows the typical performance of the ADP algorithm in each iteration for a given initial state. As can be seen, a significant cost reduction in the first iteration is achieved.
by solving formulation (7). A similar behavior is observed in applying approximate policy
iteration in other applications, e.g., see Maxwell et al. (2010). Recall that in developing the
first type of basis functions, parameter $w_i$ is used to consider the fact that the penalty cost
is much higher than the holding cost. Therefore, $w_i \propto \bar{c}_m / \bar{c}_i$, where $\bar{c}_m$ denotes the average
penalty cost for the machines that can use service part $i$. In our computational study, we
set $w_i = \gamma \bar{c}_m / \bar{c}_i$, where parameter $\gamma$ is calibrated by numerical experimentation, i.e., a variety
of values are tested and the one with the highest quality of solution is selected.

![Figure 2](image_url)  

5.2. Numerical Results

For the designed setup, Table 3 reports the results for the optimistic heuristic (H$^{OP}$), robust
heuristic (H$^{RO}$), expected heuristic (H$^{EX}$), two-stage stochastic program (H$^{SP}$), lower bound
(LB), and ADP in terms of performance and computational time. Recall that simulating
any of these feasible policies provide an upper to the optimal value function and because the
ADP policy provides the tightest bound in almost all cases tested, we use it to calculate the
optimality gap. The “average optimality gap” for a policy $H$ in Table 3 represents $100\% \times
(V_H - V_{ADP}) / V_{ADP}$, where $V_H$ is the estimated value function (expected total discounted
cost) obtained by simulating heuristic $H$. The average computational time (CPU time)
for a policy in Table 3 is reported for one state and one replication by averaging over all
replications and over 50 initial states which are also the most visited states. In particular,
the mean and standard deviation (in parenthesis) of the optimality gap as well as the
computational time are reported. The computational time for lower bound is calculated
by dividing the time required for the value iteration to optimality by the number of states.
All experiments are run on an Optiplex 9020 with 32GB of RAM and an i7-4770 running at 3.9 GHz with 4 cores using Gurobi Optimizer 7.0.

### Table 3: Results for the designed setup

<table>
<thead>
<tr>
<th>Policy</th>
<th>Avg. optimality gap (%)</th>
<th>Avg. computational time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H^{OP})</td>
<td>36.5 (4.5)</td>
<td>0.95</td>
</tr>
<tr>
<td>(H^{RO})</td>
<td>53.2 (5.8)</td>
<td>0.96</td>
</tr>
<tr>
<td>(H^{EX})</td>
<td>52.4 (6.7)</td>
<td>0.96</td>
</tr>
<tr>
<td>(H^{SP})</td>
<td>38.6 (4.8)</td>
<td>4.89</td>
</tr>
<tr>
<td>LB</td>
<td>-0.8 (0.3)</td>
<td>0.01</td>
</tr>
<tr>
<td>ADP</td>
<td>0.0</td>
<td>1.33</td>
</tr>
</tbody>
</table>

Note: The average optimality gap for each policy is calculated with respect to ADP. The numbers in parenthesis show the standard deviation.

Results in Table 3 suggest that the ADP policy is near-optimal with an optimality gap of less than one percent. The optimality gap of the best available benchmark is around 36%. The rolling horizon two-stage stochastic program performs slightly worse than the optimistic heuristic, which is the best heuristic in terms of performance. Our analysis shows that the reason for this suboptimality for base-stock heuristics is that they carry too much inventory, thus a high holding cost. The two-stage stochastic program performs poorly because it does not keep enough inventory, thus a high penalty cost. Among base-stock heuristics the optimistic heuristic performs the best as the amount of demand overestimation is minimal. In terms of assigning service parts to machines, both heuristics solve formulation (10), which considers only the holding cost not the reliability. However, the ADP policy solves formulation (6), which considers reliability in assigning service parts to machines, as well as cost.

The results for random instances are summarized in Tables 4-5. The numbers reported for each policy and instance are the optimality gaps with respect to ADP, i.e., for a policy \(H\) the tables report \(100\% \times (V_H - V_{ADP})/V_{ADP}\). Note that 50 of most visited states are used as the initial state for each policy and the corresponding mean and standard deviation for these initial states are reported. Specifically, Table 4 reports the average optimality gaps and CPU times for less reliable service parts on dense and sparse graphs, and Table 5 reports those for reliable service parts.

Results for the random instances convey a similar message as the designed setup. The optimality gap between ADP and lower bound shows that the ADP policies are efficient with the largest gap of 11%. The rolling horizon two-stage stochastic program is usually better than base-stock policies significantly. The gap between ADP and \(H^{SP}\) ranges between 1% to 26%. Among base-stock heuristics the optimistic one usually is the best, but the
The gap between ADP and $H^{OP}$ is large. In addition, the performance of base-stock heuristics is better in sparse graphs compared to dense ones because the amount of demand overestimation in sparse graphs is smaller, i.e., the base-stock heuristics keep less inventory. In terms of CPU time, ADP and base-stock heuristics are fast, but the CPU time of the two-stage stochastic program exponentially increased, i.e., from two seconds in settings with six machines to 300 seconds with 12 machines for one state and one replication, on average.

<table>
<thead>
<tr>
<th>$I$</th>
<th>$M$</th>
<th>$H^{OP}$ Avg. optimality gap for sparse graphs (%)</th>
<th>$H^{RO}$ Avg. optimality gap for sparse graphs (%)</th>
<th>$H^{EX}$ Avg. optimality gap for sparse graphs (%)</th>
<th>$H^{SP}$ Avg. optimality gap for sparse graphs (%)</th>
<th>$H^{LB}$ Avg. optimality gap for sparse graphs (%)</th>
<th>$H^{OP}$ Avg. CPU time for sparse graphs (sec.)</th>
<th>$H^{RO}$ Avg. CPU time for sparse graphs (sec.)</th>
<th>$H^{EX}$ Avg. CPU time for sparse graphs (sec.)</th>
<th>$H^{SP}$ Avg. CPU time for sparse graphs (sec.)</th>
<th>$H^{LB}$ Avg. CPU time for sparse graphs (sec.)</th>
<th>ADP Avg. CPU time for sparse graphs (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>1.6 (2.3)</td>
<td>10.4 (2.5)</td>
<td>10.0 (2.6)</td>
<td>11.5 (3.8)</td>
<td>-4.3 (1.8)</td>
<td>0.7</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>0.01</td>
<td>1.1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>21.4 (1.3)</td>
<td>39.8 (3.6)</td>
<td>39.3 (3.6)</td>
<td>19.0 (4.7)</td>
<td>-2.5 (1.2)</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>2.0</td>
<td>0.01</td>
<td>1.5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>16.2 (6.3)</td>
<td>23.2 (7.8)</td>
<td>22.5 (7.3)</td>
<td>1.1 (2.0)</td>
<td>-2.4 (1.5)</td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
<td>3.9</td>
<td>0.01</td>
<td>1.2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>72.4 (9.9)</td>
<td>71.5 (10.2)</td>
<td>71.8 (10.0)</td>
<td>6.4 (2.6)</td>
<td>-5.6 (2.4)</td>
<td>1.1</td>
<td>1.0</td>
<td>1.0</td>
<td>2.6</td>
<td>0.03</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>12.5 (1.3)</td>
<td>13.9 (1.5)</td>
<td>12.6 (1.2)</td>
<td>9.8 (4.8)</td>
<td>-5.6 (0.5)</td>
<td>1.0</td>
<td>1.1</td>
<td>0.7</td>
<td>180.8</td>
<td>0.04</td>
<td>1.3</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>31.0 (7.2)</td>
<td>34.0 (6.7)</td>
<td>31.8 (5.2)</td>
<td>17.4 (7.2)</td>
<td>-7.8 (2.1)</td>
<td>1.4</td>
<td>1.2</td>
<td>0.8</td>
<td>323.8</td>
<td>0.06</td>
<td>1.5</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>32.2 (4.8)</td>
<td>36.3 (4.9)</td>
<td>33.5 (5.1)</td>
<td>4.4 (3.6)</td>
<td>-5.7 (2.4)</td>
<td>1.0</td>
<td>1.0</td>
<td>0.8</td>
<td>417.5</td>
<td>0.09</td>
<td>1.2</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>45.1 (2.7)</td>
<td>52.1 (4.9)</td>
<td>49.0 (3.6)</td>
<td>14.1 (7.2)</td>
<td>-10.9 (2.6)</td>
<td>1.2</td>
<td>1.2</td>
<td>1.8</td>
<td>398.1</td>
<td>0.13</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Note: The average optimality gap for each policy is calculated with respect to ADP. The numbers in parenthesis show the standard deviation.

5.3. Policy Insights

This section provides insights on the ADP policy’s decisions in a variety of settings. Analyzing the settings described in Section 5.2 reveals that the inventory ordering policy generated by the ADP approach resembles a base-stock policy with occasional deviations (Figure 3). As can be seen from Figure 3, the heuristics $H^{OP}$, $H^{RO}$, and $H^{EX}$ are base-stock policies with $H^{OP}$ maintaining the lowest inventory position and $H^{RO}$ the highest, as expected. The ADP policy, however, deviates from a base-stock policy at time $t$. These rare deviations occur when the demand (i.e., the number of failed machines) exceeds supply (i.e., the
Table 5  Results for random graphs with reliable service parts

<table>
<thead>
<tr>
<th>$I$</th>
<th>$M$</th>
<th>$\text{Avg. optimality gap for dense graphs (%)}$</th>
<th>$\text{Avg. CPU time for dense graphs (sec.)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\text{H}^{\text{OP}}$</td>
<td>$\text{H}^{\text{RO}}$</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>36.3 (6.3)</td>
<td>64.1 (8.2)</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>53.8 (10.2)</td>
<td>79.9 (13.1)</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>56.7 (11.1)</td>
<td>106.0 (16.9)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>132.3 (24.4)</td>
<td>172.1 (27.1)</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>60.7 (8.3)</td>
<td>75.3 (9.4)</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>86.8 (12.1)</td>
<td>105.5 (14.0)</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>109.9 (15.7)</td>
<td>125.6 (16.4)</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>137.4 (20.7)</td>
<td>154.3 (21.2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$I$</th>
<th>$M$</th>
<th>$\text{Avg. optimality gap for sparse graphs (%)}$</th>
<th>$\text{Avg. CPU time for sparse graphs (sec.)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\text{H}^{\text{OP}}$</td>
<td>$\text{H}^{\text{RO}}$</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9.1 (4.0)</td>
<td>11.2 (3.9)</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>19.4 (4.7)</td>
<td>27.2 (4.7)</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>46.1 (9.3)</td>
<td>51.1 (8.6)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>65.0 (11.8)</td>
<td>71.0 (11.6)</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>13.2 (2.4)</td>
<td>7.8 (4.6)</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>14.0 (1.6)</td>
<td>22.8 (2.4)</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>27.9 (4.9)</td>
<td>36.3 (6.4)</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>48.2 (10.3)</td>
<td>61.6 (10.2)</td>
</tr>
</tbody>
</table>

Note: The average optimality gap for each policy is calculated with respect to ADP. The numbers in parenthesis show the standard deviation.

on-hand inventory) at which point the ADP policy places a large order which increases the inventory position. Figure 3 also reveals that the two-stage stochastic programming approach, $H^{\text{SP}}$, is a policy that seeks to fulfill the demand in an effort to minimize inventory cost.

![Figure 3  ADP and heuristic policies](image-url)

Our detailed analysis of the results shows that the heuristics $H^{\text{OP}}$, $H^{\text{RO}}$, and $H^{\text{EX}}$ carry higher levels of inventory which minimizes penalty cost but increases holding cost. On
the other hand, $H_{SPI}$ carries minimum inventory which results in lower holding but higher penalty costs. The ADP approach strikes a balance between these two extremes resulting in a policy that performs better than all of these heuristics. Furthermore, in dense graphs, the base-stock in ADP policy is an aggregate level, i.e., the total number of different service parts resembles a base stock. However, in sparse graphs, the base-stock is for each service part. In assigning service parts to machines the heuristics solve the second-stage assignment problem in the two-stage stochastic program, which only considers cost and neglects reliability. However, the ADP policy considers both reliability and cost in the MILP formulation for assignment. Therefore, the ADP policy uses native service parts if available whereas heuristics select the least expensive service parts. The combination of keeping good inventory levels along with good assignments makes the ADP policy superior compared to available benchmarks.

Note that the insights presented for ADP policies are based on scaled cost and reliability estimates of those observed in these equipment and systems’ real data. Considering other cost and reliability estimates may change the policy insights. Nonetheless, the optimization framework presented in this study can derive high-quality solutions to make better replenishment and replacement decisions.

6. Sensitivity Analysis

This section seeks to investigate the robustness of the heuristic and ADP policies by testing their performance under several milder conditions. In particular, we test the performance of $H_{HP}$, $H_{R}$, $H_{EX}$, and ADP by considering the possibility of service part replacement of a working machine, different lead times, and age-dependent failure rates.

6.1. Service Part Replacement for a Working Machine

In the modeling formulation, we assumed that when a machine is working, the decision-maker does not reassign a service part to it. This assumption is aligned with the practice of the motivating example because reassigning a service part may result in discontinuing production, thus an opportunity cost. However, if a non-native service part is installed on a machine and a native one becomes available, replacing the current service part with the native one would increase the reliability of the machine. This section seeks to estimate the price of such an assumption via simulation. In particular, in simulating the ADP policy, in order to find actions, we still solve formulation (6) with parameters tuned from the designed
setup in Section 5. However, at the beginning of each period, we remove all service parts from the machines and add it to the inventory. By this approach the ADP policy is able to assign native service parts to machines at each time period. A similar approach is used for the heuristics: the heuristics still use formulations (11) and (10) to place orders and assign service parts to machines, but at each period service parts are removed from machines and added to the inventory for assignment. Our computational results for the designed setup show that by allowing service replacement the performance of $H^{OP}$, $H^{RO}$, $H^{EX}$, and ADP change by 14.4%, 12.2%, 12.6%, and -0.9%, respectively. These results suggest that allowing replacement increases the average total cost for the heuristics whereas the ADP costs decrease slightly. Our analysis shows that this is because the ADP policies without service replacement option assign the native service parts to machines when available, and the percentage of time that a non-native part is installed and a native one becomes available is negligible. Thus, the ADP policies are more robust to the replacement assumption. On the other hand, the assignment decisions for the heuristic policies are primarily driven by the holding cost rather than reliability. Therefore, even if a native part is installed in previous periods, by allowing replacement the heuristics assign the least expensive service part, which may not be native anymore. Thus, the performance of the heuristics actually declined when replacement is allowed.

6.2. Lead Time

Our modeling framework assumes that the lead time for each service part is one period. In practice though, manufactures may face longer lead times, as well as different lead times across service parts. This section investigates the change in the optimality gap by considering different lead times for service parts. In particular, the designed setup in Section 5.1 is considered, where the lead time of service parts 1, 2, and 3 is one, two, and three periods, respectively. The ADP policy still uses formulation (6) to place orders and assign service parts to machines, which assumes that the lead time is one, and the heuristics still use (11) and (10) for order placements and assignments (i.e., the state space does not consider the ordering history). Recall that the average performance gaps between the ADP policy and $H^{OP}$, $H^{RO}$, $H^{EX}$ are, respectively, 36.5%, 53.2%, 52.4% when the lead time is one period (see Table 3). When the lead times are changed, the gaps are increased to 43.2%, 70.9%, and 62.1%, respectively. These results suggest that the ADP policies are more robust to changes in lead time compared to conventional base-stock policies frequently used in practice.
6.3. Age-Dependent Reliability

The failure rate of service parts on machines is assumed to be constant over time in the modeling formulation. However, the reliability of service parts may deteriorate over time. The literature on failure models in dynamic environments where degradation is analyzed is quite rich; see, for example, Singpurwalla (1995) and the references therein, where a survey of modeling approaches based on stochastic processes is provided. In addition, these reliability models are used for replacement decisions; see, for example, Kurt and Kharoufeh (2010) and the references therein. In order to incorporate the deterioration of service parts, an age-dependent probability of failure is considered. Let $\tau^m$ denote the age of the current service part on machine $m$ and $\tau := (\tau^m : \forall m)$. If machine $m$ does not have a service part, set $\tau^m$ to null. Let $p_i^m(\tau)$ denote the probability of failure of service part $i$ on machine $m$ in a period when the service part is installed for $\tau^m$ time periods. The failure probability is then modified to $p_i^m(\tau) = \min\{p_i^m \exp (g_i^m \tau^m), 1\}$, where $g_i^m > 0$ denotes the deterioration rate of service part $i$ on machine $m$ and can be calibrated based on historical data. In our implementation, we set $g_i^m = 2\%$ for all $i$ and $m$. The ADP policy is adapted as follows: it still uses formulation (6), i.e., the state space does not consider the age of the service parts, but it replaces $p_i^m$ with $p_i^m(\tau)$ in the formulation. Similarly, the heuristics use formulations (11) and (10) for order placements and assignments in the simulation, but $p_i^m$ is replaced with $p_i^m(\tau)$. The simulation model also uses $p_i^m(\tau)$ as the failure rate. Our numerical results show that the average performance gaps between the ADP policy and $H^{OP}$, $H^{RO}$, $H^{EX}$ respectively become 34.7%, 52.5%, and 48.9%. The ADP policy still significantly outperforms the base-stock heuristics but the gap compared to the original setup is slightly decreased. Recall that our results showed that the base-stock levels in heuristics are higher than an ideal one. Therefore, the base-stock may be closer to the ideal base-stock level when machines fail with higher rates. Thus, a decrease in the performance gap between ADP and the base-stock heuristics.

7. Conclusions

To apply sustainable supply chain design principles to the semiconductor industry, this work studied a dynamic inventory management problem of equipment for air-pollution control systems. Substitution of equipment on air-pollution control systems is possible, and the future reliability of the systems depends on which equipment is selected. In assigning
the equipment that require replacement to the corresponding air-pollution control systems when the native equipment is unavailable, the decision-maker has the option of selecting an alternative equipment. However, this choice may result in lower reliability, and the probability that the air-pollution control system will require a replacement in the future may increase. The decision-maker also has the option of replenishing the needed equipment in the presence of lead time.

This problem was formulated as a stochastic dynamic program, which was not amenable to exact solutions because of the curse of dimensionality. The approximate policy iteration framework was adapted to find high-quality solutions. Pursuant to this goal, the value function was replaced by an affine combination of nonlinear basis functions. This study demonstrates that, by this choice of value functions, a relaxation of the policy improvement step is equivalent to solving an MILP. This step enabled the design of an iterative procedure that systematically improves the quality of the value function approximation by solving a convex optimization problem. In order to assess the quality of solutions produced by the approximate policy iteration, a lower bound was developed on the optimal value function using a relaxation of the problem that was amenable to exact solutions. Moreover, several heuristic policies were designed, based on a rolling horizon single-period version of the problem, as well as base-stock inventory policies. In the heuristic based on the two-stage stochastic programming formulation, at each time period, an exact solution to the formulation is found for order placements and assignments. The other class of heuristics were based on following a base-stock strategy in ordering service parts and solving the second-stage problem in the stochastic programming formulation, which was an assignment problem, to determine replacement decisions. Three heuristics are developed based on demand estimation at a period, as well as penalty cost.

The performance of the ADP policies was tested, along with heuristic ones, on a variety of problem instances. Analysis of the policies revealed some insights on replenishment and replacement decisions. ADP policies generally resemble base-stock policies in the way they replenish equipment. In replacing the service parts for air-pollution control systems, ADP policies assign the native equipment when available. In case it is not available, ADP policies rank each service part’s reliability for replacement and assign the highest ranked appropriate choice. Overall, the results showed that ADP policies are near-optimal with the average optimality gap ranging from 1.1% to 10.9%.
References


Appendix A: Proofs

Proof of Proposition 1: Substituting formulations (1) and (3) into formulation (5) results in

\[ v^m(s) = \min_{(o,a) \in A(s)} \left\{ \sum_{i \in I} c_i^o a_i + \sum_{i \in I} c_i^a (x_i - w_i) + \sum_{m \in M} c^m (1 - \sum_{i \in I} a_i^m) + \sum_{m \in M} \lambda \alpha \sum_{i \in I} a_i^m \mid \sum_{m \in M} \beta_m 1_{\{y_m = 0\}} \right\} \]

\[ = f_1(s, \alpha) + \min_{(o,a) \in A(s)} \left\{ \sum_{i \in I} c_i^o a_i - \sum_{i \in I} \sum_{m \in M} c_i^m a_i^m - \sum_{i \in I} \sum_{m \in Y} c^m a_i^m + \sum_{i \in I} \lambda \alpha \sum_{i \in I} a_i^m \right\} \]

where \( f_1(s, \alpha) \) is a term independent of action, and \( s' \) denotes the state in the next period. Note that expectations are conditioned on the current state \( s \). Since calculating the expectation \( \mathbb{E}(|x'_i - w_i \sum_{m \in F(i)} 1_{\{y_m = 0\}}|) \) is intractable in closed form, we settle for relaxing \( v^m(s) \). Because the absolute value function \(|\cdot|\) is convex, by Jensen’s inequality

\[ \mathbb{E}(|x'_i - w_i \sum_{m \in F(i)} 1_{\{y_m = 0\}}|) \geq |\mathbb{E}(x'_i - w_i \sum_{m \in F(i)} 1_{\{y_m = 0\}})|. \]

Moreover,

\[ P(y_m' = 0 | s) = (1 - \sum_{i \in I} a_i^m) + \sum_{i \in I} a_i^m p_i^m, \quad \forall m \in \bar{Y}, \]

\[ P(y_m = 0 | s) = p_i^m, \quad \forall m \in \mathcal{M} \setminus \bar{Y}, \]

\[ P(y_m = i | s) = (1 - p_i^m) 1_{\{y_m = i\}} + (1 - p_i^m) a_i^m, \quad \forall m \in \mathcal{M}, i \in F(m), \]

and

\[ \mathbb{E}(x'_i - w_i \sum_{m \in F(i)} 1_{\{y_m = 0\}} | s) = x_i + \alpha_i - \sum_{m \in F(i)} a_i^m - w_i \sum_{m \in F(i)} P(y_m' = 0 | s) \]

\[ = x_i + \alpha_i - \sum_{m \in F(i)} a_i^m - w_i \sum_{m \in F(i) \setminus Y} \left( (1 - \sum_{j \in I} a_j^m) + \sum_{j \in I} a_j^m p_j^m \right) - w_i \sum_{m \in F(i) \setminus \bar{Y}} p_i^m. \]

Therefore, by rearranging terms and considering \( A(s) \) one can construct the following mathematical program

\[ v^R(s) = f(s, \alpha, \beta, \zeta) + \min_{(o,a) \in A(s)} \left\{ \sum_{i \in I} c_i^o a_i - \sum_{i \in I} \sum_{m \in Y} (c_i^a + c^m + \lambda \beta_m (1 - p_i^m)) a_i^m + \sum_{i \in I} \sum_{m \in M} c_i^a a_i^m + \sum_{i \in I} \sum_{m \in M} (\lambda \alpha_m p_i^m (1 - p_i^m)) a_i^m + \sum_{i \in I} \sum_{m \in \mathcal{M} \setminus \bar{Y}} \lambda \alpha_i \mathbb{E}(x'_i - w_i \sum_{m \in F(i)} 1_{\{y_m = 0\}}) \right\} \]

s.t.

\[ \sum_{m \in F(i) \setminus Y} a_i^m \leq x_i, \quad \forall i, \]
\[\sum_{i \in F(m)} a_i^m \leq 1, \forall m \in \bar{Y},\]
\[a_i^m = 0, \forall i, m \in \mathcal{M} \backslash \bar{Y},\]
\[a_i \geq 0, a_i \in \mathbb{Z}_+, a_i^m \in \{0, 1\} \forall i, m.\]

Let \(z_i = |\mathbb{E}(x_i' - w_i) \sum_{m \in F(i)} 1(|y_m' = 0)|.\) Since \(\alpha \geq 0,\) the above nonlinear program is equivalent to the following mixed integer linear program

\[v^R(s) = f(s, \alpha, \beta, \zeta) + \min \sum_{i \in \mathcal{I}} (c_i^o a_i + \lambda a_i) - \sum_{i \in \mathcal{I}} \sum_{m \in \bar{Y}} (c_i^b + c^m + \lambda \beta_m (1 - p_i^m) - \lambda \zeta_m p_i^m (1 - p_i^m)) a_i^m\]

s.t.
\[\sum_{m \in F(i) \cap \bar{Y}} a_i^m \leq x_i, \forall i,\]
\[\sum_{i \in F(m)} a_i^m \leq 1, \forall m \in \bar{Y},\]
\[z_i \geq x_i + a_i - \sum_{m \in F(i)} a_i^m - w_i \sum_{j \in \mathcal{I}} \left(1 - \sum_{j \in \mathcal{I}} a_j^m\right) + \sum_{j \in \mathcal{I}} (1 - \sum_{j \in \mathcal{I}} a_m) p_j^m - w_i \sum_{m \in F(i) \cap \bar{Y}} p_{ym}, \forall i,\]
\[-z_i \leq x_i + a_i - \sum_{m \in F(i) \cap \bar{Y}} a_i^m - w_i \sum_{j \in \mathcal{I}} \left(1 - \sum_{j \in \mathcal{I}} a_j^m\right) + \sum_{j \in \mathcal{I}} (1 - \sum_{j \in \mathcal{I}} a_m^m) p_j^m - w_i \sum_{m \in F(i) \cap \bar{Y}} p_{ym}, \forall i,\]
\[a_i^m = 0, \forall i, m \in \mathcal{M} \backslash \bar{Y},\]
\[a_i, z_i \geq 0, a_i \in \mathbb{Z}_+, a_i^m \in \{0, 1\} \forall i, m.\]

Rearranging the terms constructs formulation (6).

**Proof of Proposition 2:** Let \(n_i = |\mathcal{F}(i)|.\) For each \(i\) we have

\[\hat{p}_i^{OP} = \min_{m \in F(i)} p_{ym}^m \leq p_i^{EX} = \frac{\sum_{m \in F(i)} p_{ym}}{n_i} \leq \hat{p}_i^{RO} = \max_{m \in F(i)} p_{ym}^m.\] (A-3)

Let \(X^{OP} \sim \text{binomial}(n_i, \hat{p}_i^{OP}), X^{EX} \sim \text{binomial}(n_i, \hat{p}_i^{EX}),\) and \(X^{RO} \sim \text{binomial}(n_i, \hat{p}_i^{RO}).\) Formulation (A-3) leads to

\[X^{OP} \preceq_{st} X^{EX} \preceq_{st} X^{RO},\] (A-4)

where \(\preceq_{st}\) denotes stochastic ordering. We know that \(G_i(.) = \frac{\hat{c}_i - c^m}{n_i + c^m}\) is increasing in \(\hat{c}_i\) for \(\hat{c}_i > 0\) and

\[0 < \hat{c}_i^{OP} = \min_{m \in F(i)} c^m \leq \hat{c}_i^{EX} = \frac{\sum_{m \in F(i)} c^m}{n_i} \leq \hat{c}_i^{RO} = \max_{m \in F(i)} c^m.\]

Therefore,

\[G_i(x_i^H - y_i^H + o_i^{OP}) \leq G_i(x_i^H - y_i^H + o_i^{EX}) \leq G_i(x_i^H - y_i^H + o_i^{RO}).\] (A-5)

Finally, formulations (A-4) and (A-5) result in

\[o_i^{OP} \leq o_i^{EX} \leq o_i^{RO}.\]
Appendix B: Sampling States

As noted by de Farias and Roy (2004), $\varphi$ regulates the quality of the approximation across $S$, and can therefore be used to target certain regions of the state space where one aims to obtain better approximations. In that regard, we would like to obtain better approximations in the states that are most likely to be visited in the near future when the “optimal” policy is used. For a policy $\pi$, define the distribution $\varphi_\pi$ by

$$
\varphi_\pi(s) := (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t P_\pi\{s^t = s | s^0\}, \quad s \in S.
$$

We would like to use $\varphi^*_\pi$ in the objective function of formulation (7), and also in sampling $\hat{S}$ through $\psi$, as prescribed in de Farias and Roy (2004, Theorem 3.1., p.469). Unfortunately, one does not have prior access to $\pi^*$. Let $\pi^0$ denote an initial policy. Finding $\varphi^{\pi_0}$ is computationally intractable, thus, we settle for approximating it using its empirical counterpart, $\hat{\varphi}^{\pi_0, T'}$, where

$$
\hat{\varphi}^{\pi_0, T'}(S) := (1 - \lambda) \sum_{t=0}^{T'} \lambda^t 1\{s^t(\omega) = s\}, \quad s \in S, \pi \in \mathcal{P},
$$

where $\{s^t(\omega) : t \geq 0\}$ represents the outcome of a Monte Carlo simulation and $T'$ denotes the simulation budget. (In our numerical study we take the average over many replications.) We use this simulation run to select $\hat{S}$ as well: Computing $\psi$ is infeasible, thus we approximate it as $\hat{\varphi}^{\pi_0, T'}$. The underlying motivation is that if on most simulation runs, states are visited at most once, thus we approximate $P_\pi\{s^t = s | s^0 = s'\} \approx \sum_{r=1}^{R} 1\{s^t(\omega_r) = s\}/R$, where $R$ is the number of replications. This results in $\varphi \approx \psi$. 