Solving Truly Massive Budgeted Monotonic POMDPs with Oracle-Guided Meta-Reinforcement Learning

Manav Vora[1], Michael N Grussing[2], Melkior Ornik[1]

[1]Department of Aerospace Engineering and Coordinated Science Laboratory, University of Illinois Urbana Champaign, Urbana, USA
[2]Engineer Research and Development Center, US Army Corps of Engineers, Champaign, USA
mkvora2@illinois.edu, michael.n.grussing@erdc.dren.mil, mornik@illinois.edu

Abstract

Monotonic Partially Observable Markov Decision Processes (POMDPs), where the system state progressively decreases until a restorative action is performed, can be used to model sequential repair problems effectively. This paper considers the problem of solving budget-constrained multi-component monotonic POMDPs, where a finite budget limits the maximal number of restorative actions. For a large number of components, solving such a POMDP using current methods is computationally intractable due to the exponential growth in the state space with an increasing number of components. To address this challenge, we propose a two-step approach. Since the individual components of a budget-constrained multi-component monotonic POMDP are only connected via the shared budget, we first approximate the optimal budget allocation among these components using an approximation of each component POMDP's optimal value function which is obtained through a random forest model. Subsequently, we introduce an oracle-guided meta-trained Proximal Policy Optimization (PPO) algorithm to solve each of the independent budget-constrained single-component monotonic POMDPs. The oracle policy is obtained by performing value iteration on the corresponding monotonic Markov Decision Process (MDP). This two-step method provides scalability in solving truly massive multi-component monotonic POMDPs. To demonstrate the efficacy of our approach, we consider a real-world maintenance scenario that involves inspection and repair of an administrative building by a team of agents within a maintenance budget. Our results show that the proposed method significantly improves average component survival times compared to baseline policies, thereby highlighting its potential for practical applications in large-scale maintenance problems. Finally, we perform a computational complexity analysis for a varying number of components to show the scalability of the proposed approach.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) provide an efficient framework for modeling a wide range of real-world sequential decision-making problems (Cassandra 1998; Bravo, Leiras, and Cyrino Oliveira 2019). Numerous methods have been developed to compute optimal policies for POMDPs, including Monte-Carlo methods (Katt, Oliehoek, and Amato 2017), reinforcement learning approaches (Singh et al. 2021), and various approximation techniques (Kearns, Mansour, and Ng 1999). One notable

Figure 1: Condition of infrastructure component over time. (a) Line plot showing component condition over time for 100 runs. The red x marks denote the time step when condition reaches 0. (b) Violin-plot showing distribution of component condition for different time steps.

application of POMDPs is in infrastructure maintenance or sequential repair problems, where the system state represents the condition of infrastructure components, which is only partially observable (Bhattacharya et al. 2021). Over time, the state of these components stochastically deteriorates unless a restorative action, such as a repair, is performed. Figure 1 shows this stochastic decrease and also shows the probability distribution of the condition of a sample infrastructure component for different time steps. The state evolution of the POMDP, thus, exhibits monotonic properties (Miehling and Teneketzis 2020) and we define such POMDPs as monotonic POMDPs. Previously, Bhattacharya et al. (2020) proposed an optimal inspection and repair scheduling policy for a pipeline, which is a single-component POMDP. However, infrastructure systems naturally have multiple components (Daulat et al. 2024).

In this paper, we address the challenge of computing approximately optimal policies for budget-constrained multi-component monotonic POMDPs. Each component POMDP operates independently in terms of transition probabilities, but they are collectively constrained by the shared budget. There has been substantial work on solving budget constrained POMDPs (Lee et al. 2018; Undurti and How 2010; Khonji, Jasour, and Williams 2019). However, the complexity of these algorithms is often exponential in the number of states of the POMDP, and consequently would be exponential in the number of components for a multi-component

POMDP. Thus, they become computationally intractable for multi-component POMDPs with a large number of components. In Vora et al. (2023), the authors propose a welfare-maximization method for solving budget-constrained multi-component POMDPs. However, the method in that paper requires generating optimal policies for multiple budget values for every component POMDP to get the optimal budget allocation, i.e., optimal distribution of the shared budget among the components POMDPs. Hence, it cannot be scaled to a large number of components.

To address this scalability issue, we propose a computationally efficient meta-reinforcement learning approach for solving budget-constrained multi-component monotonic POMDPs. First, we approximate the optimal distribution of budget among the individual component POMDPs using a random forest budget allocation strategy, thereby decoupling the problem into independent components. The random forest model is trained to approximate the value function of a component POMDP as a function of the budget, leveraging the concavity of the value function to frame the budget allocation as a concave maximization problem, which guarantees a computationally feasible solution. Subsequently, we apply the oracle policy-guided meta-trained Proximal Policy Optimization (PPO) algorithm, trained across various components and budget values, to compute the approximately optimal policy for each component-budget pair. The oracle policy for the meta-PPO is derived from value iteration applied to the corresponding component MDP. To validate the efficacy of our proposed approach, we examine a maintenance scenario involving an administrative building with 1000 infrastructure components and a fixed maintenance budget. We compare the performance of our approach, in terms of the survival time of the components, against the oracle policy, a currently-used baseline policy, and a vanilla meta-PPO algorithm. We also perform a computational complexity analysis of our algorithm for varying number of components and show that the proposed approach is linear in the number of components.

In summary, the key contributions of this paper are:

1. We develop an oracle policy-guided meta-PPO algorithm for solving budget-constrained single-component monotonic POMDPs.

2. We introduce a random forest budget allocation strategy for optimally distributing the budget among the component POMDPs of a budget-constrained multi-component POMDP.

3. We demonstrate the efficacy of our approach through a large-scale real-world infrastructure maintenance scenario with 1000 infrastructure components.

4. We show the scalability of our algorithm through a computational complexity analysis for varying number of components.

2 Preliminaries and Related Work

2.1 Partially Observable Markov Decision Processes

A discrete-time finite-horizon Partially Observable Markov Decision Process (POMDP) (Cassandra, Kael-bling, and Littman 1994) M is defined by the 7-tuple $(\mathcal{S}, A, T, \Omega, O, R, H)$, which denotes the state space, action space, state transition function, observation space, observation function, reward function and planning horizon, respectively. In a POMDP, the agent does not have direct access to the true state of the environment. Instead, the agent may maintain a *belief state*, representing a probability distribution over $\mathcal{S}$. This belief is updated based on the received observation using Bayes' rule (Araya et al. 2010).

2.2 POMDP Solution Methods

Computing optimal policies for a POMDP is generally PSPACE-complete (Mundhenk et al. 2000; Vlassis, Littman, and Barber 2012). Thus, to address the computational intractability of solving POMDPs, various approximation methods have been widely used (Poupart and Boutilier 2002; Pineau et al. 2003; Roy, Gordon, and Thrun 2005). Several reinforcement learning approaches have also been developed for computing approximate POMDP solutions (Az-izzadenesheli, Lazaric, and Anandkumar 2016; Igl et al. 2018). However these methods become computationally intractable when faced with the high dimensionality and shared resource constraints of budget-constrained multi-component monotonic POMDPs such as those considered in this paper.

2.3 Consumption MDPs and Budgeted POMDPs

The integration of budget or resource constraints into Markov Decision Processes (MDPs) has been previously studied under the frameworks of Consumption MDPs (Blahoudek et al. 2020) and Budgeted POMDPs (Vora et al. 2023). However, the algorithm proposed in Blahoudek et al. (2020) assumes full observability of the state and hence cannot be applied to budget-constrained POMDPs. A solution for budget-constrained multi-component POMDPs is presented in Vora et al. (2023). However, the method in this paper requires repeated computations of optimal policies for different budget values for all component POMDPs and hence is not scalable to a budget-constrained multi-component POMDP with a large number of components.

3 Problem Formulation

In this paper, we consider a weakly-coupled multi-component monotonic POMDP with a total budget. A weakly-coupled multi-component POMDP refers to a system where the individual component POMDPs have independent transition probabilities but are interconnected through a shared budget B. This shared budget introduces a weak coupling between the components, as the allocation of budget to one component affects the available budget for the others. The state space for an n-component monotonic budget-constrained POMDP is given by $\mathcal{S} = \prod_{i=1}^{n} \mathcal{S}_i$, where $\mathcal{S}_i$ represents the state space for component i, and $i \in \{1, \ldots, n\}$. The action space is given by $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}_i$, where the action space for component i is $\mathcal{A}_i = \{d^i, q^i, m^i\}$. Each action incurs a fixed cost. The state at time instant k is an n-tuple, $s_k = (s_k^1, s_k^2, \cdots, s_k^n)$, where $s_k^i \in \mathcal{S}_i = \{0, 1, \ldots, \bar{s}\}$ denotes the state of component i,

Figure 2: Architectural overview of the proposed approach.

and $\bar{s} \in \mathbb{N}_0$ is the maximum possible value of s_k^i. Here, $\mathbb{N}_0$ denotes the set of non-negative integers. Similarly, the action at time k is given by $a_k = (a_k^1, a_k^2, \cdots, a_k^n)$ and the cost associated with this action is given by $c_{a_k} = \sum_{i=1}^n c_{a_k^i}$, where $c_{a_k^i}$ represents the cost associated with each action a_k^i. The transition function for the multi-component POMDP is:

$$T(s_k, a_k, s_{k+1}) = \prod_{i=1}^n T_i(s_k^i, a_k^i, s_{k+1}^i).$$

The transition probability function for each component i is:

$$T^i(s_k^i, a_k^i, s_{k+1}^i) = \begin{cases} p_1^i(s_k^i, a_k^i, s_{k+1}^i), & \text{if } a_k^i = m^i \text{ and} \\ & s_k^i \leq s_{k+1}^i \leq \bar{s}, \\ p_2^i(s_k^i, a_k^i, s_{k+1}^i), & \text{if } a_k^i \in \{d^i, q^i\} \\ & \text{and } s_{k+1}^i \leq s_k^i, \\ 1, & \text{if } a_k^i \in A^i \text{ and} \\ & s_{k+1}^i = 0 = s_k^i, \\ 0, & \text{otherwise.} \end{cases}$$

Here, action m^i is a restorative action that increases the state value, with the increase being upper bounded by $\bar{s}$. In contrast, actions d^i and q^i decrease the state value. Moreover, $s_k^i = 0$ is an absorbing state for all k, i. Finally, the observation probability function for each component follows the model from Vora et al. (2023), where action q_i gives true state information and the other two actions provide no information about the true state.

3.1 Problem Statement

The primary objective of this paper is to determine a policy π^* for this multi-component monotonic POMDP over a horizon H, that maximizes the sum of expectations of the individual times before reaching the absorbing state for each component, while adhering to the total budget B. We denote this maximal time k by $T_{max} = \sum_{i=1}^n T_{max}^i$, where T_{max}^i denotes the corresponding maximal time for component i. Mathematically, the problem can be formulated as:

$$\max_\pi \left(\sum_{i=1}^n \mathbb{E}[T_{max}^i(\pi)] \right)$$

$$\text{s.t.} \sum_{k=0}^H c_{a_k}(\pi) \leq B. \tag{1}$$

In this formulation, π represents the policy, and both T_{max}^i and c_{a_k} depend on π. For simplicity, we will not explicitly denote this dependence in the remainder of this paper. There are many other possible formulations of the objective of the problem statement like a *maxmin* formulation:

$$\max_\pi \min_i \mathbb{E}[T_{max}^i(\pi)]. \tag{2}$$

In this paper we consider the formulation given by (1).

4 Solution Approach

In this section, we present our methodology for solving a budget-constrained multi-component monotonic POMDP. Figure 2 presents an architectural overview of our proposed approach. We first input information about all n component POMDPs into a pre-trained random forest regressor to get the T_{max}^i for each component i as a function of the budget allocated to the component. Next, we propose an appropriate allocation of the shared budget B among the component POMDPs by solving a constrained maximization problem. We then compute the oracle policy for each component POMDP and budget pair through value iteration applied to the corresponding component MDP. Finally, using these oracle policies and a meta-trained reinforcement learning (RL) agent, we solve each component POMDP with respect to the allocated budget and consequently propose a policy for the multi-component POMDP. Note that an alternate allocation strategy could involve redistributing the budget at every time step during planning. However, such a method would be computationally more expensive, due to the repeated computation of the allocation, as compared to our proposed a priori budget distribution approach.

4.1 Oracle-Guided RL for Budget-Constrained Monotonic POMDPs

To derive the optimal policy for a budget-constrained single-component monotonic POMDP, we propose an oracle policy-guided reinforcement learning algorithm. Adherence to the budget constraints is achieved using the budget-constrained POMDP (bPOMDP) framework proposed in Vora et al. (2023). In a bPOMDP, the state at each time step includes an additional fully observable component representing the cost incurred up to that point.

The oracle policy is denoted as π_{oracle} and is obtained by solving the corresponding MDP using value iteration.

For a single-component monotonic POMDP with budget B, the corresponding MDP has an action space $\mathcal{A}_{\text{MDP}} = \{d, m\}$, identical transition probabilities as the POMDP, and *full observability of the state*. We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017) in conjunction with the oracle policy to solve the single-component budget-constrained POMDP. Doing so allows us to treat the problem as that of determining the optimal inspection policy. At each time step, the PPO agent decides whether to perform inspection or not. If the agent decides to take the inspection action, it receives accurate information about the true state of the POMDP. On the other hand, if the agent decides not to inspect, then the action is chosen according to the oracle policy. Hence, the action space for the PPO agent is reduced to $\{q, \neg q\}$. Since the full state is not observable in a POMDP, we utilize the belief b_s for planning. The agent's belief of the true state is updated at each time step using a particle filter approach. For our work, we empirically observe that using the expected belief $\bar{b}_s$ and the variance of the belief $\sigma_{b_s}^2$ suffices for planning.

Hence, for the proposed oracle policy-guided PPO agent, the state at time instant k is given by the vector $[\bar{b}_{s_k}, c_k, \sigma_{b_{s_k}}^2]$. Furthermore, the reward function is defined as follows:

$$R(s_k, c_k, a_k) = \begin{cases} r_1 < 0, & \text{if } c_k > B, \\ r_2 < 0, & \text{if } \lfloor \bar{b}_{s_k} \rfloor = 0, \\ r_3 = \frac{k}{H} - \alpha |\bar{b}_{s_k} - s_k|, & \text{if } \bar{b}_{s_k}, c_k > 0, \end{cases}$$

where $|r_1| > |r_2| > |r_3|$ for all k, $0 < \alpha < 1$ and $\lfloor . \rfloor$ denotes the floor function. This reward function imposes substantial negative rewards for exceeding the budget B and allowing the state s_k to reach 0. Additionally, at each time step, the agent receives a positive reward proportional to the time step for maintaining s_k above zero and incurs a penalty proportional to the absolute error between the expected belief and the true state. As a result, the agent gets higher rewards for keeping $s_k > 0$ for a longer time and is heavily penalized when the expected belief deviates significantly from the true state. It is crucial to note that during training, the agent relies solely on the observed reward signals, without access to the true state.

4.2 Random Forest Approach for Optimal Budget Allocation

In the previous subsection, we detailed the methodology for deriving the optimal policy for a single-component budget-constrained monotonic POMDP. We now extend this to an n-component POMDP described in Section 3. Each component in this multi-component POMDP operates independently in terms of transition probabilities, but they share a common total budget, rendering them weakly coupled. While reinforcement learning algorithms have made significant advances, they often face challenges when scaling to the extremely large state and action spaces characteristic of multi-component systems (Sutton and Barto 2018). In the case of a budget-constrained n-component POMDP, directly applying a single-component solution approach would result in an exponentially large state space and action space,

making such an approach impractical for large-scale problems. To address this scalability issue, we propose an a priori budget distribution strategy to decouple the components, thereby reducing the size of the state and action spaces. Under certain assumptions, the expected maximal time to reach the absorbing state, $\mathbb{E}[T_{\max}]$, for a budget-constrained single-component monotonic POMDP, is concave with respect to the allocated budget b (Vora et al. 2023). For a given component i and allocated budget b^i, let $\tilde{T}_{\max}^i$ be the approximate expected maximal time as a function of the budget, modeled using an exponential function:

$$\tilde{T}_{\max}^i = \alpha^i e^{\beta^i b^i} + \gamma^i, \tag{3}$$

where $\alpha^i, \beta^i, \gamma^i \in \mathbb{R}$ are constants. While many other concave functions could be used to model $\tilde{T}_{\max}^i$, we empirically observe that the exponential function provides a good fit for the data. We use a random forest regressor (Breiman 2001) to estimate the parameters of this exponential function. The training dataset for this model is obtained via nonlinear least squares regression on multiple $(\mathbb{E}[T_{\max}], b)$ pairs for various budget-constrained single-component monotonic POMDPs. The input to this model includes specific statistics related to the POMDP's transition function, which are the expected time to reach state 0 without repairs, $\mathbb{E}[T]$, and the variance of this expected time, $\sigma_{\mathbb{E}[T]}^2$, as well as the cost of the various actions.

For an n-component POMDP with total budget B, let the budget allocated to the i-th component be b^i, and the approximate expected maximal time to reach the absorbing state for this budget be $\tilde{T}_{\max}^i$. We then formulate the following constrained maximization problem:

$$\max_{b^i} \sum_{i=1}^{n} \tilde{T}_{\max}^i$$
$$\text{s.t.} \sum_{i=1}^{n} b^i \leq B. \tag{4}$$

Solving (4) yields the approximately optimal budget allocation among the individual components. Since we assume $\tilde{T}_{\max}^i$ to be concave in b^i, the optimization problem becomes a concave maximization problem and is thus guaranteed to have a computationally tractable solution.

4.3 Optimal Policy for Multi-Component Monotonic POMDPs

We now integrate the approaches described in the previous subsections to compute the optimal policy for an n-component POMDP, where n is substantially large. Utilizing the random forest regressor from Section 4.2, we efficiently approximate $\mathbb{E}[T_{\max}]$ for each component i. Additionally, we meta-train the oracle-guided PPO agent by constantly updating the policy network's parameters over a randomly selected a subset of components and budget values. This approach allows the agent to generalize across components. This meta-trained agent is then utilized to derive the optimal policy π^{i^*} for each component i, following the optimal budget allocation obtained from (4). Consequently, the overall

Figure 3: Performance comparison of oracle policy, oracle-guided meta-PPO, realistic baseline and vanilla meta-PPO. (a) Comparison of $\hat{T}_{max}$ values for all 1000 components across different budget values allocated to each component. (b) Comparison of average number of repairs performed by the agent under each of the four policies. (c) Comparison of average total cost incurred by the agent over the planning horizon for each of the four policies.

policy for the multi-component POMDP is:

$$\pi^*(s_k, a_k) = (\pi^{1^*}(s_k^1, a_k^1), \pi^{2^*}(s_k^2, a_k^2), \cdots, \pi^{n^*}(s_k^n, a_k^n)).$$

While this policy is not guaranteed to be globally optimal for the entire multi-component POMDP, we empirically observe that it performs well in practice while respecting the budget constraints. We validate this approach by evaluating its performance on real-world data in the subsequent section.

5 Implementation and Evaluation

In this section, we evaluate the efficacy of the proposed methodology for determining the optimal policy for a very large multi-component budget-constrained POMDP. Specifically, we compare our approach against existing methods in the context of a multi-component building maintenance scenario managed by a team of agents. We also perform a computational complexity analysis of the proposed approach, for varying number of components.

We consider an administrative building comprising 1000 infrastructure components, including roofing elements, water fountains, lighting systems, and boilers. Each component's health is quantified by the Condition Index (CI) (Grussing, Uzarski, and Marrano 2006), which ranges from 0 to 100. For each infrastructure component, we utilize historical CI data to generate the transition probabilities for the corresponding POMDP, modeled using the Weibull distribution (Grussing, Uzarski, and Marrano 2006). We use the `weibull_min` class from the `scipy.stats` module in Python to simulate the CI transitions over time. While a seed can be set using the `random_state` parameter in `weibull_min` for reproducibility, we did not set one to preserve the stochastic nature of the CI transitions. The condition index deteriorates stochastically over time, influenced by various factors, and can only be accurately assessed through explicit inspections, which incur a cost. A component is considered to have failed when its CI falls below a failure threshold, which is assumed to be 0. Components can be repaired to increase their CI. The building is allocated a maintenance budget of $B = 500,000$ units for a

given horizon of 100 decision steps. At the beginning of the horizon, the CI of all components is 100. The objective of the agents is to maximize the time until failure of the components by efficiently allocating the budget among the components and performing repairs and inspections as needed. The replacement costs (ranging from 50 to 500 units) and inspection costs (ranging from 1 to 5 units) of these components are derived from industry averages. Consistent with the approach described in Section 4.1, we model this objective as a POMDP (with $\alpha = 10^{-3}$ in the reward function). This POMDP has roughly 10^{2000} states and 3^{1000} actions.

5.1 Analysis of Maintenance Policy

We begin by analyzing the performance of the maintenance policy derived using the proposed oracle-guided meta-PPO strategy for a single-component POMDP representing a component i of the 1000 components. This policy is compared with the performance of the oracle policy on the corresponding component MDP. Since the oracle policy has full observability of the state, it is expected to always perform better than the proposed approach. Additionally, we evaluate two baseline policies:

1. A heuristic policy often used in practice (Lam and Yeh 1994; Straub 2004) where the agent performs inspections at regular intervals and repairs the component when its expected belief about the Condition Index (CI) falls below a predefined threshold. We chose an inspection interval of 5 steps and a repair threshold of 15 after extensive experiments with intervals ranging from 1 to 10 steps and repair thresholds from 5 to 50.

2. A vanilla meta-PPO agent that is trained on the same subset of component-budget pairs as the oracle-guided agent, but without an oracle policy.

Both the oracle-guided meta-PPO and vanilla meta-PPO are trained for 2M time steps each, with an Adam step-size of 10^{-4}, a minibatch size 128, policy update horizon of $T = 4096$ and discount factor 0.95. All other hyperparameters follow those used in Schulman et al. (2017). We perform 100 simulations for this component to obtain the corre-

sponding $T_{\max}^i$ values, which are then averaged over the runs for a given budget value allocated to the component. This process is repeated for all 1000 components and the run-averaged $T_{\max}^i$ values are then averaged across components. We compare this average denoted by $\hat{T}_{max}$ for 11 different budget values ranging from 0 to 5000 units, along with the average number of repairs performed by the agent and the average cost incurred over the planning horizon. Figure 3 illustrates a comparison of these metrics for all four policies. We observe that the proposed approach significantly outperforms the baselines. The oracle-guided meta-PPO agent nearly matches the performance of the oracle policy for all 3 metrics, presumably due to the low inspection costs of the components. If inspection costs were significantly higher, the agent's performance would likely diverge from the oracle policy, which is an expected outcome given the budget constraints. We also infer that the vanilla meta-PPO agent has only learnt to not violate the budget constraint by not performing any repairs. These results demonstrate the value of incorporating an oracle policy into the training of a reinforcement learning agent.

5.2 Analysis of Budget Allocation

Next, we demonstrate the effectiveness of our random forest-based budget allocation strategy. We compare it with a baseline approach that allocates budgets proportional to the ratio of a component's replacement cost to its $\mathbb{E}[T]$. For a component i, we model its $\mathbb{E}[T_{\max}^i]$ using $\tilde{T}_{\max}^i$ as given in (3). The parameters α^i and γ^i can be estimated directly by considering the boundary conditions: γ^i is estimated by substituting $b^i = 0$, representing the scenario where no budget is available, and α^i is determined by substituting $b^i = \infty$, corresponding to the scenario of unlimited budget, where the supremum of $T_{\max}^i$ ($\sup_{b^i} T_{\max}^i = H = 100$) is reached. We then train a random forest regressor to estimate parameter β^i. The training dataset is created by performing non-linear least squares regression on 11 distinct (T_{max}^i, b^i) pairs each for 800 components. These pairs correspond to the run-averaged $T_{\max}^i$ values and the respective budget values b^i from Section 5.1. The input to the random forest model is a vector consisting of the shape and scale factors of the Weibull distribution, which represent $\mathbb{E}[T]$ and $\sigma_{\mathbb{E}[T]}^2$, along with the replacement and inspection costs for a given component i. If a different distribution were used to model the transition probability, we would similarly extract the parameters, $\mathbb{E}[T]$ and $\sigma_{\mathbb{E}[T]}^2$, for inclusion in the input vector.

Figure 4 shows the prediction performance of the random forest model for a test dataset of 200 components which were not encountered during training. We see that most points on the plot are very close to the perfect prediction line and bad predictions are few in number (29 out of 200 for error threshold of 10^{-4}). The random forest model achieves a mean squared error (MSE) = 1.8×10^{-8} for this test dataset. Note that the non-linear least squares regressor constrains β^i to be ≤ 0 and hence for some components we observe that $\beta^i = 0$. We use this trained random forest model to estimate $\tilde{T}_{max}^i$ for all 1000 components. Finally, using these approximated expressions, we solve the

Figure 4: Performance of random forest model for predicting the value of parameter β for a test dataset of 200 components. The horizontal axis represents parameter values obtained via non-linear least squares and vertical axis represents predicted values. The dotted line represents the $y = x$ line, i.e., perfect predictions.

constrained maximization problem described in (4) to obtain the appropriate budget allocation for the components. We quantify the performance of the random forest budget

Approach	T_{max}
Random Forest Budget Allocation	22,009.5
Baseline Budget Allocation	16,445.4

Table 1: T_{max} (in steps), averaged over 100 runs, achieved under random forest and baseline budget allocations.

Figure 5: Performance comparison of random forest-based budget allocation and baseline budget allocation for all 1000 components for an overall budget of 500,000 units.

allocation and the baseline budget allocation algorithms by calculating the $T_{max} = \sum_i T_{max}^i$ and averaging it over 100 runs. For a fair comparison, these values are obtained using the oracle-guided meta-PPO approach for both allocation schemes. Table 1 shows the T_{max} values achieved by both allocation approaches. The random forest budget allocation vastly outperforms the baseline approach. Furthermore, Figure 5 presents violin plots showing the distribution of the T_{max}^i values achieved under the proposed and baseline

Number of Components	Random Forest	Budget Split	Value Iteration	Meta-PPO
1	0.9724	0.9046	113.7227	2.8885
2	0.8870	0.8314	116.3281	3.0858
5	0.8719	0.8207	135.3953	4.7940
10	0.8762	0.8132	280.4909	9.5495
20	0.9534	0.8997	451.2948	16.2575
50	0.9449	0.8916	1208.1000	33.7387
100	0.9324	0.9171	2389.5641	64.6809
500	0.9575	1.2226	10269.1037	313.9742
1000	0.9599	1.6232	20612.1734	627.7477

Table 2: Time taken (in seconds) for running each process with varying numbers of components, averaged over 10 runs.

budget allocations for all 1000 components. We observe that there are more components with higher T^i_{max} values for the random forest budget allocation approach. Preliminary experiments on alternative objective formulations, such as the *maxmin* approach given by (2), also indicate that the proposed method consistently outperforms the baseline.

5.3 Analysis of Time Complexity

Finally, we analyze the time complexity of our proposed approach for varying number of components N. As mentioned earlier, our method comprises of four major steps:

1. **Random Forest** regression for estimating $\tilde{T}^i_{max}$ for each component i.

2. **Budget Allocation** among components via constrained optimization.

3. **MDP Value Iteration** for each component-budget pair to obtain the corresponding oracle policy.

4. **Oracle-Guided Meta-PPO** to approximately solve each component POMDP.

Table 2 presents the times taken for running each of the four processes, with different number of components. The time complexity experiments were performed in Python on a laptop running MacOS with an M2 chip @3.49GHz CPU and 8GB RAM. The times taken for random forest and budget allocation steps are negligible compared to those for performing value iteration and generating optimal policies through meta-PPO. The value iteration is applied to each component independently and hence scales linearly with the number of components. Similarly, Step 4 involves applying the pre-trained policy to each component separately and thus is also linear in the number of components. Consequently, we expect that the time complexity of our algorithm is linear in the number of components, i.e., $O(n)$. This expectation is confirmed by the log-log plot of computational complexity shown in Figure 6. Our algorithm's performance is thus significantly faster as compared to existing POMDP solvers which would be exponential in the number of states and thus doubly exponential in the number of components (Silver and Veness 2010), (Pineau et al. 2003). If the problem is approached directly as a single POMDP, it will have a prohibitively vast state space of approximately 10^{2000} states. Previous work by Vora et al. (2023) demonstrated that standard methods indeed become computationally intractable after a few components due to this combinatorial explosion.

Figure 6: Log-log plot of computational complexity of the proposed approach for varying numbers of components.

6 Conclusions

In this paper we introduce a framework to provide computationally efficient solutions, scalable for large scale budget-constrained multi-component monotonic POMDPs. Noting that the coupling is expressed solely through the shared budget, our method functions by first decomposing the large n-component POMDP into n independent single-component POMDPs using a random forest budget allocation of the shared budget. Next, we compute the oracle policy using value iteration for each component POMDP and budget pair. Finally, using this oracle policy together with a meta-trained PPO agent, we solve each component POMDP. Consequently, we obtain the approximately optimal policy for the multi-component POMDP. The experimental evaluations conducted for a real-world scenario of large scale infrastructure maintenance show that the proposed approach outperforms existing baselines and almost matches the performance of the oracle policy, which relies on information not available to the planner in a partially observable setting. Furthermore, computational complexity analysis demonstrates the scalability of our framework. Future work will focus on extending the framework's capabilities to more dynamic budget allocation schemes and more complicated hierarchical budget constraints.

References

Araya, M.; Buffet, O.; Thomas, V.; and Charpillet, F. 2010. A POMDP extension with belief-dependent rewards. In *23rd Advances in Neural Information Processing Systems*.

Azizzadenesheli, K.; Lazaric, A.; and Anandkumar, A. 2016. Reinforcement learning of POMDPs using spectral methods. In *Conference on Learning Theory*, 193–256.

Bhattacharya, S.; Badyal, S.; Wheeler, T.; Gil, S.; and Bertsekas, D. 2020. Reinforcement learning for POMDP: Partitioned rollout and policy iteration with application to autonomous sequential repair problems. *IEEE Robotics and Automation Letters*, 5(3): 3967–3974.

Bhattacharya, S.; Kailas, S.; Badyal, S.; Gil, S.; and Bertsekas, D. 2021. Multiagent rollout and policy iteration for POMDP with application to multi-robot repair problems. In *Conference on Robot Learning*, 1814–1828.

Blahoudek, F.; Brázdil, T.; Novotný, P.; Ornik, M.; Thangeda, P.; and Topcu, U. 2020. Qualitative controller synthesis for consumption Markov decision processes. In *International Conference on Computer Aided Verification*, 421–447.

Bravo, R. Z. B.; Leiras, A.; and Cyrino Oliveira, F. L. 2019. The use of UAVs in humanitarian relief: An application of POMDP-based methodology for finding victims. *Production and Operations Management*, 28(2): 421–440.

Breiman, L. 2001. Random forests. *Machine Learning*, 45: 5–32.

Cassandra, A. R. 1998. A survey of POMDP applications. In *AAAI Fall Symposium on Planning with Partially Observable Markov Decision Processes*.

Cassandra, A. R.; Kaelbling, L. P.; and Littman, M. L. 1994. Acting Optimally in Partially Observable Stochastic Domains. In *12th AAAI National Conference on Artificial Intelligence*, 1023–1028.

Daulat, S.; Rokstad, M. M.; Klein-Paste, A.; Langeveld, J.; and Tscheikner-Gratl, F. 2024. Challenges of integrated multi-infrastructure asset management: A review of pavement, sewer, and water distribution networks. *Structure and Infrastructure Engineering*, 20(4): 546–565.

Grussing, M. N.; Uzarski, D. R.; and Marrano, L. R. 2006. Condition and reliability prediction models using the Weibull probability distribution. In *Applications of Advanced Technology in Transportation*, 19–24.

Igl, M.; Zintgraf, L.; Le, T. A.; Wood, F.; and Whiteson, S. 2018. Deep variational reinforcement learning for POMDPs. In *International Conference on Machine Learning*, 2117–2126.

Katt, S.; Oliehoek, F. A.; and Amato, C. 2017. Learning in POMDPs with Monte Carlo tree search. In *International Conference on Machine Learning*, 1819–1827.

Kearns, M.; Mansour, Y.; and Ng, A. 1999. Approximate planning in large POMDPs via reusable trajectories. In *12th Advances in Neural Information Processing Systems*.

Khonji, M.; Jasour, A.; and Williams, B. C. 2019. Approximability of Constant-horizon Constrained POMDP. In *International Joint Conferences on Artificial Intelligence*, 5583–5590.

Lam, C. T.; and Yeh, R. 1994. Optimal maintenance-policies for deteriorating systems under various maintenance strategies. *IEEE Transactions on Reliability*, 43(3): 423–430.

Lee, J.; Kim, G.-H.; Poupart, P.; and Kim, K.-E. 2018. Monte-Carlo tree search for constrained POMDPs. In *31st Advances in Neural Information Processing Systems*.

Miehling, E.; and Teneketzis, D. 2020. Monotonicity properties for two-action partially observable Markov decision processes on partially ordered spaces. *European Journal of Operational Research*, 282(3): 936–944.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM*, 47(4): 681–720.

Pineau, J.; Gordon, G.; Thrun, S.; et al. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conferences on Artificial Intelligence*, 1025–1032.

Poupart, P.; and Boutilier, C. 2002. Value-directed compression of POMDPs. In *15th Advances in Neural Information Processing Systems*.

Roy, N.; Gordon, G.; and Thrun, S. 2005. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23: 1–40.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv*.

Silver, D.; and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *23rd Advances in Neural Information Processing Systems*.

Singh, G.; Peri, S.; Kim, J.; Kim, H.; and Ahn, S. 2021. Structured world belief for reinforcement learning in POMDP. In *International Conference on Machine Learning*, 9744–9755.

Straub, D. 2004. *Generic approaches to risk based inspection planning for steel structures*. 284. vdf Hochschulverlag AG.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Undurti, A.; and How, J. P. 2010. An online algorithm for constrained POMDPs. In *2010 IEEE International Conference on Robotics and Automation*, 3966–3973.

Vlassis, N.; Littman, M. L.; and Barber, D. 2012. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory*, 4(4): 1–8.

Vora, M.; Thangeda, P.; Grussing, M. N.; and Ornik, M. 2023. Welfare Maximization Algorithm for Solving Budget-Constrained Multi-Component POMDPs. *IEEE Control Systems Letters*, 7: 1736–1741.