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

Anonymous authors
Paper under double-blind review

Abstract

Many real-world decision problems, ranging from asset-maintenance scheduling to portfolio rebalancing, can be naturally modelled as budget-constrained multi-component monotonic Partially Observable Markov Decision Processes (POMDPs): each component's latent state degrades stochastically until an expensive restorative action is taken, while all assets share a fixed intervention budget. For a large numbers of assets, deriving an optimal policy for this joint POMDP is computationally intractable. To tackle this challenge, we prove that the value function of the associated belief-MDP is *budget-concave*, which allows an efficient two-step approach to finding a near-optimal policy. First, we approximate the optimal cross-component budget split via a random-forest surrogate of each single-component value function. Second, we solve each resulting budget-constrained single-component POMDP with an oracle-guided meta-trained Proximal Policy Optimization (PPO) policy: value-iteration on the fully observable counterpart yields an oracle that shapes the PPO update and greatly accelerates learning. We validate our method through experiments in two disparate domains: (i) preventive maintenance for a large-scale building infrastructure containing 1,000 components, and (ii) portfolio risk management under debit-only loss-budget constraints, where each asset's latent budget depletes with market losses and can only be replenished through costly recapitalization. Results show that our method consistently achieves longer component survival times and enhanced portfolio viability than both baseline heuristics and vanilla PPO. Furthermore, our approach maintains linear scalability in solution time with respect to the number of components.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) offer a principled framework for sequential decision making under uncertainty regarding the true state of the system (Cassandra, 1998; Bravo et al., 2019). Solving POMDPs is computationally challenging, leading to the development of various solvers, including Monte-Carlo tree search (Katt et al., 2017), reinforcement-learning variants (Singh et al., 2021), and diverse approximation schemes (Kearns et al., 1999).

Many application domains share a *monotonic* structure, where the latent state of individual components degrades stochastically over time unless a costly restorative action is taken. Canonical examples include online advertising (Boutilier & Lu, 2016), inventory replenishment (Shin & Lee, 2015), and sequential repair or maintenance scheduling for physical assets (Miehling & Teneketzis, 2020; Bhattacharya et al., 2021). Figure 1 shows this stochastic decline and the probability distribution of a sample component's condition at successive time steps. While prior work, such as (Bhattacharya et al., 2020), has addressed optimal policies for single-component systems, real-world systems—from building portfolios to exchange-traded-fund (ETF) baskets—naturally involve *many* such components (Daulat et al., 2024).

In this paper, we address the challenge of computing approximately optimal policies for budget-constrained multi-component monotonic POMDPs. We assume that each component POMDP operates independently in terms of transition probabilities, but they are collectively constrained by the shared budget. Substantial work has been done to solve budget-constrained POMDPs (Lee et al., 2018; Undurti & How, 2010; Khonji

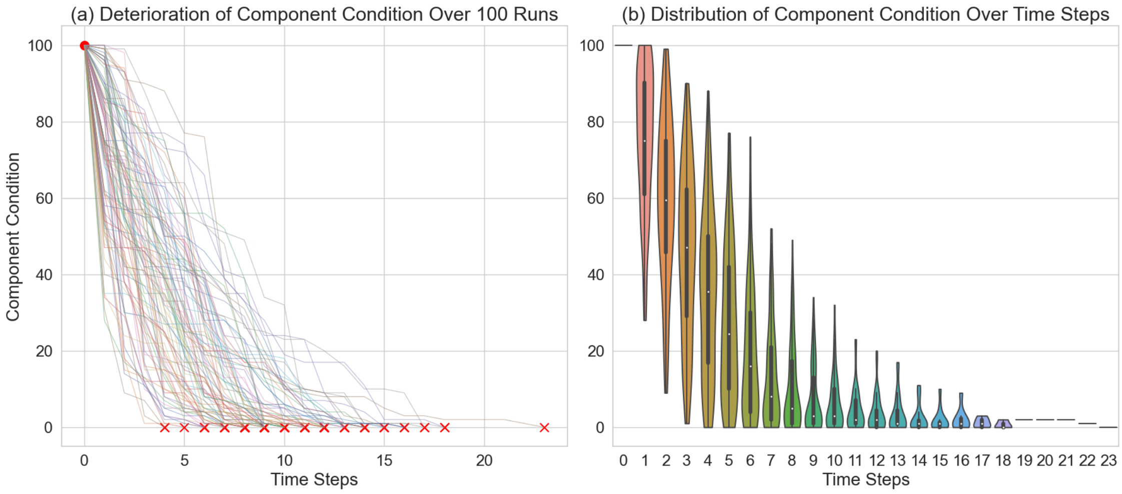

Figure 1: Condition of infrastructure component over time. (a) Line plot showing component condition over time for 100 runs. The red x marks denote the time step when condition reaches 0. (b) Violin-plot showing distribution of component condition for different time steps.

et al., 2019). However, the complexity of these algorithms is often exponential in the number of states of a single POMDP. For a multi-component POMDP, where the overall state space is the Cartesian product of individual component state spaces, this complexity consequently becomes exponential in the number of components. Thus, these methods become computationally intractable for multi-component POMDPs with a large number of components. A key challenge in solving budget-constrained multi-component POMDPs is how to optimally allocate the shared budget across the multiple 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. Hence, it cannot be scaled to a large number of components.

Our insight. The primary computational bottleneck in solving budget-constrained multi-component POMDPs is the *coupling* induced by the shared budget. If that budget could be split *a-priori*, the joint POMDP would factor into n independent, single-component problems solvable in parallel. To enable this decomposition, we prove that the optimal value function of a *single* monotonic POMDP is **concave** in its allocated budget. This budget-concavity lets us decouple first, optimise second:

(1) *Budget allocation.* We maximize a concave surrogate of the value function, estimated with a random-forest regressor, to distribute the global budget across components; and

(2) *Component policies.* With budgets fixed, we learn a near-optimal policy for each component–budget pair using an *oracle-guided, meta-trained* Proximal Policy Optimization (PPO) agent, where the oracle is obtained by value iteration on the fully observable counterpart.

The result is a scalable solution whose runtime grows linearly with the number of components while retaining strong performance guarantees.

Contributions.

1. **Theory.** We prove budget-concavity of the optimal value function for monotonic POMDPs. While prior works implicitly assume and use this budget-concavity, our work provides the first general structural guarantee that formally links budget availability to expected return.

2. **Algorithms.** We introduce (i) a random-forest budget-allocation module that exploits concavity for fast global optimization, and (ii) an oracle-guided meta-PPO solver for each single-component POMDP.

3. **Empirical evidence.** On two domains—preventive maintenance of a 1000-component building and portfolio loss-budget management with recapitalization—we outperform baseline heuristics and vanilla PPO, whilst solution time of the proposed approach scales *linearly* in the number of components.

4. **Complexity analysis.** We provide a detailed runtime study confirming linear growth in wall-clock time as components increase from $n = 10$ to $n = 1000$.

The remainder of the paper is organized as follows. Section 2 surveys related work on budget-constrained POMDPs and large-scale maintenance or portfolio problems. Section 3 formalises the budget-constrained multi-component monotonic POMDP. Section 4 presents our solution pipeline: **(i)** Subsection 4.1 proves budget–concavity of the single-component value function; **(ii)** Subsection 4.2 exploits this structure to allocate the global budget via a random-forest surrogate; and **(iii)** Subsection 4.3 derives an oracle-guided meta-PPO policy for each component and composes them into the overall controller. Section 5 reports empirical results on infrastructure maintenance and financial loss-budget management, and Section 6 concludes with key findings and future directions.

2 Preliminaries and Related Work

2.1 Partially Observable Markov Decision Processes

A discrete-time finite-horizon Partially Observable Markov Decision Process (POMDP) (Cassandra et al., 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 et al., 2012). Thus, to address the computational intractability of solving POMDPs, various approximation methods have been widely used (Poupart & Boutilier, 2002; Pineau et al., 2003; Roy et al., 2005). Several reinforcement learning approaches have also been developed for computing approximate POMDP solutions (Azizzadenesheli et al., 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.

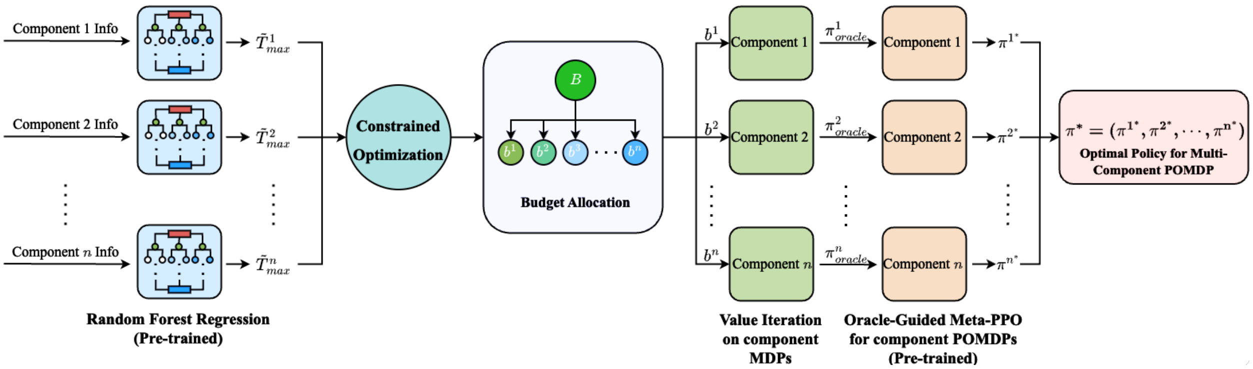

Figure 2: Architectural overview of the proposed approach.

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, 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:

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

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:

$$\begin{aligned} &\max_{\pi} \left(\sum_{i=1}^{n} \mathbb{E}[T_{max}^i(\pi)] \right) \\ &\text{s.t.} \sum_{k=0}^{H} c_{a_k}(\pi) \leq B. \end{aligned} \tag{2}$$

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:

$$\max_{\pi} \min_{i} \mathbb{E}[T_{max}^i(\pi)]. \tag{3}$$

In this paper we consider the formulation given by (2).

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. The key idea is to *decouple first, optimize second.* Allocating the shared budget *as a first step of* planning shrinks the original large joint POMDP into n independent single-component POMDPs. Each of these single-component POMDPs now operates with its own fixed budget cap, which is determined by the initial allocation. This transformation converts a problem that is intractable for $n \gg 1$ into n modest ones that can be solved in parallel. We organize the section accordingly:

- **Structural Result: Budget Concavity** (Section 4.1): We prove that each component's value function is concave in its budget.
- **Stage 1: Budget Allocation** (Section 4.2): Leveraging concavity, we fit a random-forest surrogate of the value function and solve a tractable concave maximization problem to distribute the shared global budget across components.
- **Stage 2: Oracle-Guided Meta-PPO** (Section 4.3): With budgets fixed, we learn near-optimal policies for each component–budget pair (with respect to that component's allocated budget and local POMDP) using an oracle-guided, meta-trained PPO agent, then compose these into the overall multi-component policy.

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 than our proposed approach due to the repeated computation of the allocation.

4.1 Budget–Concavity of the Value Function

We first show that the optimal value function of a single-component belief-MDP—derived from a monotonic POMDP with random, nonnegative action costs—is concave in the budget variable B for any finite planning horizon $H \geq 0$.

Setting

We consider a POMDP defined by the tuple $\langle S, A, T, R, \Omega, O, \gamma, \mathcal{C} \rangle$, where:

- S is a finite state space and A is a finite action space, as defined in Section 3.

- $T(s'|s,a)$ is the transition kernel (1); $R(s,a)$ is the reward function.
- Ω is the observation space; $O(o|s',a)$ is the observation model as defined in Section 3.
- $\gamma \in [0,1)$ is the discount factor.
- $\mathcal{C}(s,a)$ is the distribution of a random, nonnegative cost incurred by taking action a in state s. The specific cost, c, is drawn from this distribution.

Following standard practice in POMDP literature (Cassandra, 1998), we reformulate this POMDP as a belief-MDP with state (b,B), where $b \in \Delta(S)$ is the belief (posterior distribution over hidden states), and $B \geq 0$ is a remaining budget. The reward at belief b under action a is:

$$\rho(b,a) := \sum_{s\in S} b(s)R(s,a),$$

and the budget evolves as $B \mapsto B - c$, where c is a realization from the random cost $C(s,a)$ under the current belief. To prove the budget-concavity of the value function for this belief-MDP, we first establish two foundational properties concerning concavity. These lemmas demonstrate how concavity is preserved under common mathematical operations relevant to dynamic programming.

Lemma 1 (Concavity under Affine Shift)**.** *If $f(x)$ is concave on an interval I, then $f(x-l)$ is concave on the interval $\{y \mid y = x + l, x \in I\}$ for any constant l.*

Proof. This lemma is a standard result in convex analysis (Boyd & Vandenberghe, 2004). □

Lemma 2 (Expectation Preserves Concavity)**.** *Let $f(B,\xi)$ be concave in B for every realization ξ. If ξ is a random variable following an arbitrary probability distribution, then $\mathbb{E}_\xi[f(B,\xi)]$ is also concave in B.*

Proof. Fix any $B_1, B_2 \in \mathbb{R}$ and any $\lambda \in [0,1]$. Let $g(B) = \mathbb{E}_\xi[f(B,\xi)]$. Then

$$g(\lambda B_1 + (1-\lambda)B_2) = \mathbb{E}_\xi\left[f\big(\lambda B_1 + (1-\lambda)B_2, \xi\big)\right].$$

Since $f(B,\xi)$ is concave in B for each ξ, we have

$$f\big(\lambda B_1 + (1-\lambda)B_2, \xi\big) \geq \lambda f(B_1,\xi) + (1-\lambda) f(B_2,\xi)$$

for all ξ. Taking expectations on both sides yields

$$\mathbb{E}_\xi\left[f(\lambda B_1 + (1-\lambda)B_2, \xi)\right] \geq \lambda \mathbb{E}_\xi\left[f(B_1,\xi)\right] + (1-\lambda)\mathbb{E}_\xi\left[f(B_2,\xi)\right],$$

that is,

$$g(\lambda B_1 + (1-\lambda)B_2) \geq \lambda g(B_1) + (1-\lambda) g(B_2).$$

Thus, g is concave in B. □

Having established these fundamental properties regarding the preservation of concavity under affine shifts and expectations, we will now prove that the optimal value function of a monotonic POMDP is concave with respect to the available budget.

Theorem 3 (Budget Concavity)**.** *For any fixed belief $b \in \Delta(S)$ and horizon $H \geq 0$, the value function $V_H(b,B)$ is concave in B on $[0,\infty)$.*

Proof. We proceed by mathematical induction on H.

Base Case ($H = 0$). At horizon zero, there are no rewards:

$$V_0(b,B) = 0 \quad \text{for all } b \in \Delta(S),\ B \geq 0.$$

Function V_0 is thus trivially concave.

Inductive Hypothesis. Suppose that for some $H \geq 0$, the function $V_H(b, B)$ is concave in B for every belief b.

Inductive Step. We aim to prove that $V_{H+1}(b, B)$ is concave in B for all b. The Bellman equation in the belief-MDP is:

$$V_{H+1}(b, B) = \max_{a \in A} \left\{ \rho(b, a) + \gamma \, \mathbb{E}_{o,c|b,a} \left[V_H(b', B - c) \right] \right\},$$

where b' is the updated belief after taking action a and observing o. The expectation $\mathbb{E}_{o,c|b,a}$ is taken over the random observation o and cost c given the current belief b and chosen action a.

Define the inner expectation as:

$$g(a, b, B) := \mathbb{E}_{o,c|b,a} \left[V_H(b', B - c) \right].$$

Apply Lemma 1 and inductive hypothesis to assert that $B \mapsto V_H(b', B - c)$ is concave for each (o, c). Then Lemma 2 implies that $g(B)$, being the expectation over such functions, is also concave.

Therefore, the Q-value

$$Q_{H+1}(b, B, a) := \rho(b, a) + \gamma g(B)$$

is concave in B for each a.

Finally, the value function is

$$V_{H+1}(b, B) = \max_{a \in A} Q_{H+1}(b, B, a),$$

which is the pointwise maximum of finitely many concave functions, and hence concave itself. □

4.1.1 Relating $\mathbb{E}[T_{\max}]$ to the Value Function

We proved the budget-concavity of the value function $V_H(s, B)$ in Theorem 3. In our problem setting (2), however, we aim to maximize the expected time to failure $E[T_{\max}]$. Specifically, in many practical applications such as preventive maintenance or portfolio management, the objective can be naturally framed as maximizing the expected time until a critical failure occurs or a budget is exhausted. We now show how the concavity property extends to $E[T_{\max}]$, which serves as the objective function for our initial budget allocation stage.

Lemma 4 (Expected-time equivalence)**.** *Consider the reward function*

$$R(s, a) = \begin{cases} 1, & s \neq 0, \\ 0, & s = 0, \end{cases} \tag{4}$$

. Let $V(s, B)$ be the corresponding optimal value function. Denote by $\mathbb{E}[T_{\max}(B)]$ the expected time to reach the absorbing state $s = 0$ under the optimal budget-feasible policy. Then

$$V(s, B) = \mathbb{E}[T_{\max}(B)].$$

Proof. Under reward scheme (4) each non-absorbing step contributes exactly 1 to the return; steps in state 0 contribute 0. Hence, for any budget-feasible policy π,

$$\text{Total reward} = \mathbb{E}\Big[\sum_{t=0}^{H} \mathbf{1}\{s_t \neq 0\}\Big] = \mathbb{E}[T_{\max}(\pi)].$$

Taking the maximum over all budget-feasible policies yields $V(s, B) = \mathbb{E}[T_{\max}(B)]$. □

Corollary 5. *Because $B \mapsto V(s, B)$ is concave by Theorem 3, the expected absorption time $\mathbb{E}[T_{\max}(B)]$ is also concave in the allocated budget B.*

4.2 Random Forest Approach for Optimal Budget Allocation

By Corollary 5, the expected maximal survival time $\mathbb{E}[T_{\max}(B)]$ is a concave function of the budget allocated to a single component. This structural property lets us treat budget splitting across n components as a *concave maximization* problem—one that is both tractable and amenable to surrogate modeling. Each component evolves independently but competes for the shared budget, rendering the components 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 & Barto, 2018). To address this scalability issue, our remedy is an *a-priori* budget distribution that decouples the system. Concretely, for component i we approximate the concave map $B \mapsto \mathbb{E}[T^i_{\max}(B)]$ by the exponential surrogate

$$\widetilde{T}^i_{\max}(B) \;=\; \alpha^i \, e^{\beta^i B} + \gamma^i, \tag{5}$$

where $(\alpha^i, \beta^i, \gamma^i)$ are constants. While many other concave functions could be used to model $\widetilde{T}^i_{\max}$, we empirically observe that the exponential function provides a good fit for the data (see Appendix A). 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 non-linear 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, $\sigma^2_{\mathbb{E}[T]}$, as well as the various actions costs.

Let b^i denote the budget assigned to component i and $\widetilde{T}^i_{\max}$ its surrogate survival time. The allocation problem becomes

$$\begin{aligned} &\max_{b^{1:n}} \; \sum_{i=1}^{n} \widetilde{T}^i_{\max}(b^i) \\ &\text{s.t.} \; \sum_{i=1}^{n} b^i \;\leq\; B, \qquad b^i \;\geq\; 0 \;\; \forall i, \end{aligned} \tag{6}$$

a concave maximization with linear constraints. Because each surrogate in (5) is concave, the problem is globally tractable and we solve it with off-the-shelf convex optimizers. Solving (6) yields the approximately optimal budget allocation among the individual components. The next subsection shows how an oracle-guided meta-PPO agent learns the individual component policies given this budget allocation.

4.3 Oracle-Guided RL for a Budget-Constrained Single Component

Given the per-component budgets b^i obtained in Section 4.2, we now derive a near-optimal control policy for each single-component budget-constrained monotonic POMDP. We adopt the budget-augmented POMDP (bPOMDP) formalism of Vora et al. (2023), in which the state includes an additional, fully-observable coordinate that tracks cumulative cost.

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 then train a Proximal Policy optimization (PPO) agent (Schulman et al., 2017) that *queries* this oracle selectively: at each time step it chooses either to inspect (q) or to defer $(\neg q)$, in which case the action recommended by the oracle is executed. 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 $\sigma^2_{b_s}$ suffices for planning.

Hence, for the proposed oracle policy-guided PPO agent, the state at time instant k is given by the vector $[\bar{b}_{s_k}, c_k, \sigma^2_{b_{s_k}}]$. 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.

4.4 Optimal Policy for Multi-Component Monotonic POMDPs

We now integrate the approaches described in Section 4.2 and Section 4.3 to compute the optimal policy for an n-component POMDP, where n is substantially large. Utilizing the random forest regressor, we efficiently approximate $\mathbb{E}[T_{\max}]$ for each component i. Additionally, we meta-train the oracle-guided PPO agent by continuously updating the policy network's parameters over a randomly selected 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 (6). Consequently, the overall policy for the multi-component POMDP is:

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

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 empirically validate our proposed framework on two disparate domains. The first domain, which we call the *infrastructure scenario*, involves preventive maintenance for a large-scale building comprising 1000 independent components whose latent condition stochastically degrades over time; our goal is to allocate a finite maintenance budget to maximize the expected survival time of all components. The second domain, the *financial loss-budget scenario*, addresses portfolio risk management using daily price data for S&P 500 constituents, where each asset is endowed with a debit-only loss budget that depletes under negative returns and can be replenished only through costly recapitalization. In the infrastructure scenario, we compare our oracle-guided meta-PPO approach against baseline heuristics, vanilla PPO, and an idealized oracle policy, reporting results on survival time, cost efficiency, and computational scalability across a range of budget levels. In the financial loss-budget scenario, we focus on analyzing the learned recapitalization policy and assessing the generalizability and window robustness of proposed oracle-guided meta-PPO.

5.1 Implementation and Evaluation for Infrastructure Scenario

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 et al., 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 et al., 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 4.3, 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.

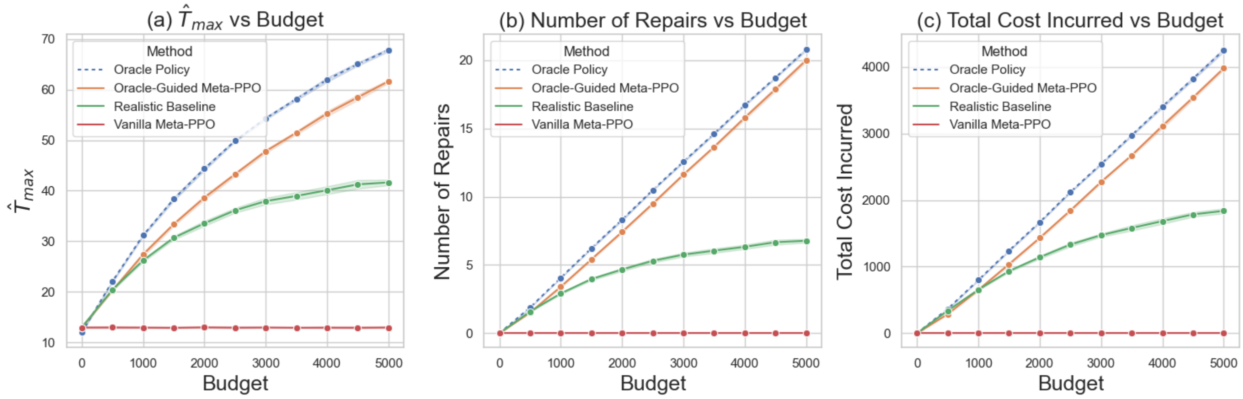

Figure 3: Performance comparison of oracle policy, oracle-guided meta-PPO, realistic baseline and vanilla meta-PPO. (a) Comparison of $\hat{T}_{max}$ values for all 1000 components across different budget values allocated to each component. (b) Comparison of average number of repairs performed by the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

5.1.1 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 & 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 stepsize 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 corresponding $T^i_{\max}$ 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 $T^i_{\max}$ 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.

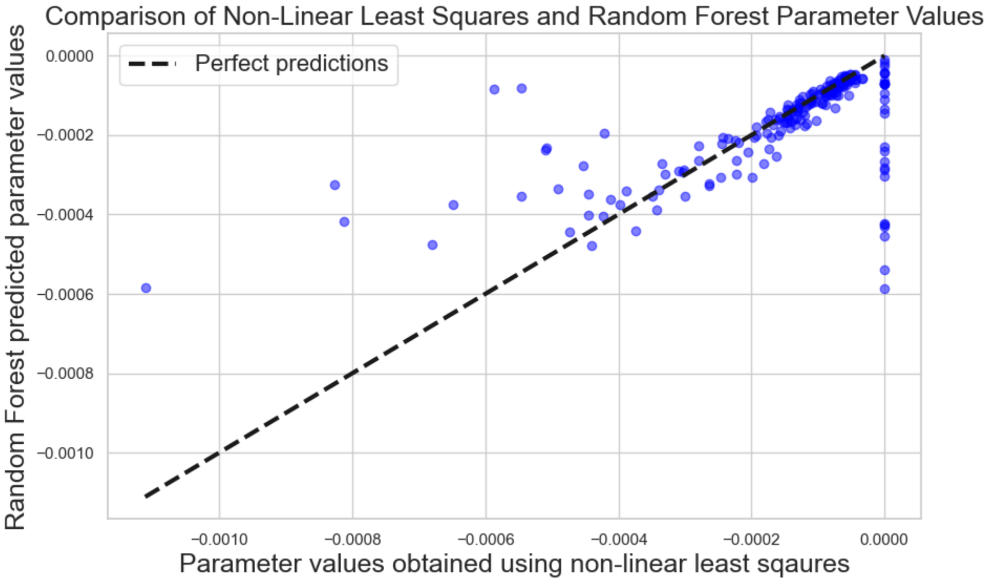

Figure 4: Performance of random forest model for predicting the value of parameter β for a test dataset of 200 components. The horizontal axis represents parameter values obtained via non-linear least squares and vertical axis represents predicted values. The dotted line represents the $y = x$ line,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

5.1.2 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 $\mathbb{E}[T^i_{\max}]$ using $\tilde{T}^i_{\max}$ as given in (5) (see Appendix A for justification of this exponential form). 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 $T^i_{\max}$ ($\sup_{b^i} T^i_{\max} = H = 100$) 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^i_{max}, b^i) pairs each for 800 components. These pairs correspond to the run-averaged $T^i_{\max}$ values and the respective budget values b^i from Section 5.1.1. 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 $\sigma^2_{\mathbb{E}[T]}$, along with the replacement and inspection costs for a given component i. If a different distribution was used to model the transition probability, we would similarly extract the parameters, $\mathbb{E}[T]$ and $\sigma^2_{\mathbb{E}[T]}$, 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 $\tilde{T}^i_{max}$ for all 1000 components. Finally, using these approximated expressions, we solve the constrained maximization problem described in (6) to obtain the appropriate budget allocation for the components. We quantify the performance of the random forest budget allocation and the baseline budget allocation algorithms by calculating the $T_{max} = \sum_i T^i_{max}$ 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: Maximum time $T_{\max}$ (steps), averaged over 100 runs, under random forest and baseline budget allocations.

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

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^i_{max} values achieved under the proposed and baseline 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 (3), also indicate that the proposed method consistently outperforms the baseline.

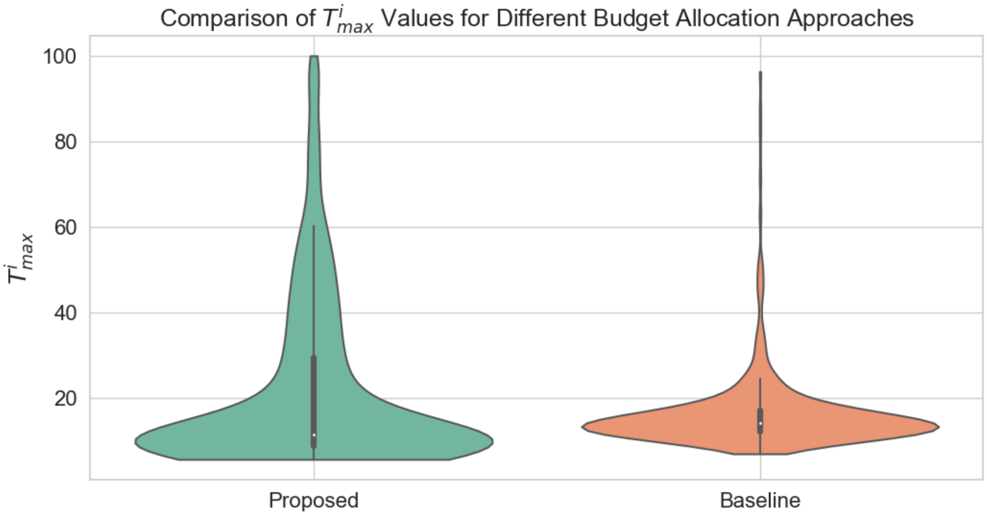

Figure 5: Performance comparison of random forest-based budget allocation and baseline budget allocation for all 1000 components for an overall budget of 500,000 units.

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

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

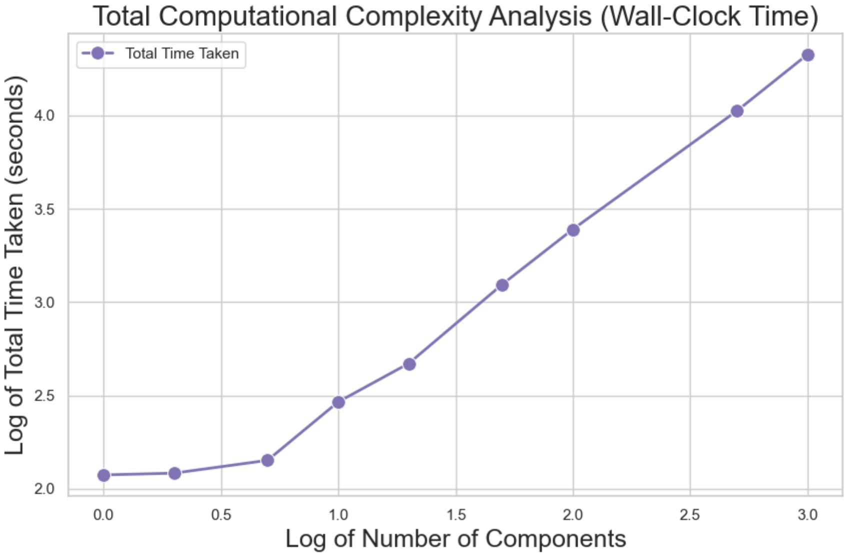

Figure 6: Log-log plot of computational complexity of the proposed approach for varying numbers of components.

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 & 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.

5.2 Implementation and Evaluation for Financial Loss-Budget Management Scenario

Our second experimental scenario addresses a portfolio risk management task. We use daily price data for the S&P 500 constituent stocks over a two-year window, reserving the final $T = 120$ trading days for evaluation and using earlier data for training. The core component of the POMDP is an unobserved latent state defined per component as a **debit-only loss budget** (**health**), $s_t \in [0, 100]$. Each component receives a small loss budget: on a day with a negative return, s_t is debited proportionally and decreases; on a non-negative day, s_t is unchanged; the state does not self-recover. Health increases only when the agent executes **recapitalize**. All actions draw from one shared, limited budget, and actions are taken when drift relative to the per-component no-loss floor becomes meaningful. This design is practice-inspired for two reasons. First, because we manage a large number of components, governance and our own policy favor a conservative stance: we avoid repeatedly allocating budget to components with recent serial losses, so the health is debit-only and does not auto-replenish. Second, it follows the risk-budgeting workflow described in Benham & Bebee (2024)—set a budget ex ante, allocate and monitor against a benchmark, and treat material drift as a trigger for action. To make the benchmark operational, we instantiate a per-component **no-loss floor**: losses are deviations that consume the per-component budget; gains are consistent with the floor and do not raise limits by themselves; replenishment occurs only through **recapitalize**. For training and evaluation, components are assumed independent.

Actions and Costs: The agent's action space $\mathcal{A} = \{\text{defer, inspect, recapitalize}\}$ manages the per-component loss budget (health). All actions draw from a shared, limited budget B and follow a strict cost hierarchy $c_{\text{recapitalize}} > c_{\text{inspect}} > c_{\text{defer}}$:

- **Defer:** Continue with the current position. Incurs a low, continuous cost c_{defer} each step. Health remains subject to depletion by negative returns.
- **Inspect:** Pay c_{inspect} to obtain a precise observation of the hidden health s_t for the selected component.
- **Recapitalize:** Pay the high cost $c_{\text{recapitalize}}$ to rebuild health by resetting s_t to 100. This is the only action that increases health.

Objective and Failure Condition. The agent's objective is to learn a policy π that maximizes its **survival time**. An absorbing failure state is triggered immediately if any component's health is exhausted, i.e., $s_t \leq 0$. For each day the agent survives, it receives a reward of +1. This setup forces the agent to learn a sophisticated policy that balances the continuous drain from defer costs and market losses against the high, discrete costs of inspection and recapitalization, in order to prolong its survival.

5.2.1 Analysis of Recapitalization Policy

We evaluate our approach on a **stock-level loss-budget** management task constructed from the S&P 500 universe. Starting from 500 constituents, we retain the subset with at least 80% daily-price coverage over the preceding three years, yielding 471 components. As in the infrastructure experiment, we reserve the final $T = 120$ trading days for evaluation and use earlier data for model training.

State, actions, and costs. Each component j is modeled as a single-component monotonic POMDP with an unobserved, debit-only **loss-budget (health)** $s_t^j \in [0, 100]$. Negative returns debit s_t^j proportionally; non-negative returns leave s_t^j unchanged; the state does not self-recover. The action set is $\mathcal{A} = \{\text{defer}, \text{inspect}, \text{recapitalize}\}$ with a strict cost hierarchy $c_{\text{recapitalize}} > c_{\text{inspect}} > c_{\text{defer}}$. A global budget B_{tot} is shared across all components. Table 3 presents the values of the various parameters used for the experiments.

Policies compared. We compare four policies:

Table 3: Cost and budget settings for the stock-level scenario.

Parameter	Value
Total budget B_{tot}	15,000
Recapitalization cost c_{RECAP}	10.0
Inspection cost c_{INSP}	0.5
Defer cost c_{DEF}	0.2
Number of components	471
Evaluation horizon T	120 days

1. **Oracle**: full observability of the health s_t; **recapitalize** whenever $s_t < 20$ (no inspection cost).
2. **Oracle-guided meta-PPO**: the agent chooses **inspect** vs. **defer**; upon **defer**, it executes the oracle's suggested restorative/default control; upon **inspect**, it pays c_{inspect} to reduce belief uncertainty. The agent learns when to buy observations and when to accept uncertainty.
3. **Baseline (Heuristic)**: fixed **inspect** every 5 trading days; if the observed $s_t < 20$, take **recapitalize**; otherwise **defer**.
4. **Vanilla meta-PPO**: trained on the same component–budget pairs as the oracle-guided agent but without oracle shaping.

Budget allocation. We allocate B_{tot} across the 471 **components** using the same random-forest surrogate procedure as in the maintenance experiment: for each component i we fit a concave surrogate for the map $B \mapsto \mathbb{E}[T^i_{\max}(B)]$ and solve a tractable concave maximization to obtain per-component budgets.

Training details. Both **oracle-guided meta-PPO** and **vanilla meta-PPO** are trained for 2×10^6 timesteps with Adam step size 10^{-4}, minibatch size 128, PPO horizon $T_{\text{PPO}} = 2048$, and discount factor 0.95. For each component and policy we run 100 simulations and report the component-level average $T^i_{\max}$; we then average across all 471 components to obtain $\hat{T}_{\max}$.

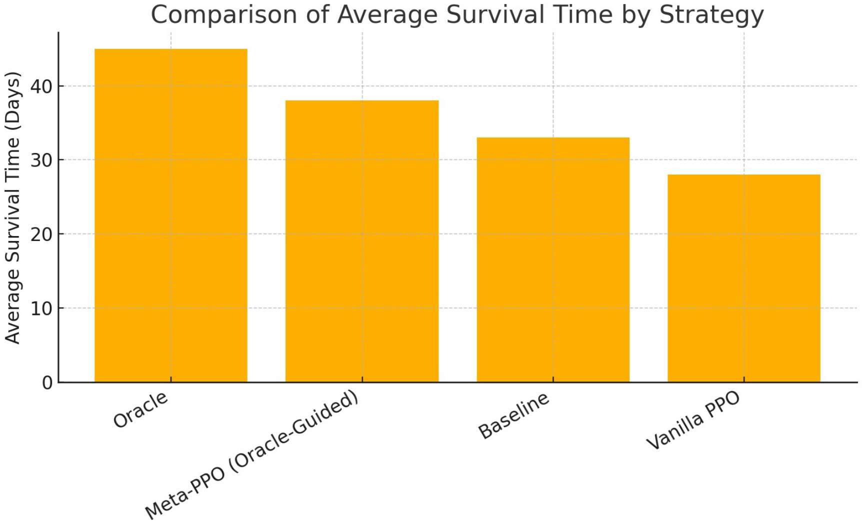

Figure 7: S&P 500 stock-level scenario: average survival time $\hat{T}_{\max}$ under a shared budget $B_{\text{tot}} = 15{,}000$ across 471 components. Observed ordering: **Oracle** > **Oracle-guided meta-PPO** > **Baseline** > **Vanilla meta-PPO**.

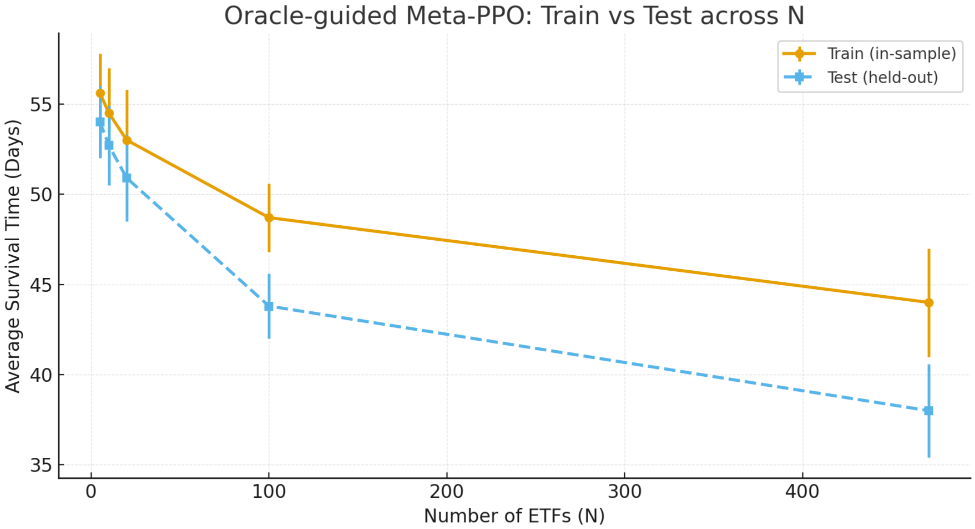

Figure 8: Oracle-guided meta-PPO: train vs. test across component set size N. The y-axis is average survival time (days); the x-axis is the number of components.

Findings. Under the shared budget $B_{\text{tot}} = 15{,}000$ across 471 components and a 120-day evaluation horizon, we observe a consistent ordering (see Figure 7): **Oracle** > **Oracle-guided meta-PPO** > **Baseline** > **Vanilla meta-PPO**. The **Vanilla meta-PPO** tends to conserve budget and rarely recapitalizes, yielding the lowest survival time. The **Baseline** performs periodic inspections (every 5 trading days) and recapitalizes below the threshold but spends budget indiscriminately and misses urgent cases. By contrast, the **Oracle-guided meta-PPO** learns when to inspect versus defer and when to act, allocating budget to higher-value opportunities; it reliably outperforms the Baseline and closes a substantial portion of the gap to the Oracle upper bound.

5.2.2 Generalizability and Window Robustness of the Oracle-guided Meta-PPO

Design. We vary the number of **components** $N \in \{5, 10, 20, 100, 471\}$. For each N, the policy is trained on rolling 120-day train windows and evaluated both in-sample (train) and on a held-out test window. We report average survival time (days) over r=5 seeds; error bars denote ±1 standard deviation across seeds.

Findings. (1) Both train and test curves decrease as N grows, reflecting budget dilution and increased problem complexity. (2) Train performance is consistently above test with a modest generalization gap that tends to widen at larger N. (3) Variability is non-negligible and generally larger at higher N.

Takeaway. As can be seen from Figure 8, the oracle-guided meta-PPO exhibits window robustness: trends are consistent across train windows, and the train-to-test drop remains moderate. At small N, the effective exploration/interaction budget is limited, which can hinder learning; as N increases, richer allocation opportunities make better use of the oracle guidance even though absolute survival time declines under a fixed total budget.

6 Conclusions

We proposed a scalable framework for solving *budget-constrained multi-component monotonic POMDPs*. Our chief theoretical contribution is a proof that the single-component value function is **concave in budget**, which underpins an efficient two-step solution strategy. First, a random-forest surrogate exploits that concavity to distribute the shared budget across components, thereby decomposing the large n-component POMDP into n independent single-component POMDPs. Second, an *oracle-guided, meta-trained PPO* agent—shaped by value iteration on the fully observable counterpart—learns a near-optimal policy for each component–budget pair. Comprehensive experiments on two disparate domains confirm the framework's generality. For a 1000-component building-maintenance task, our method significantly prolongs component survival relative to baseline heuristics and approaches the performance of the oracle policy. On an ETF portfolio-rebalancing problem with draw-down–risk budgets, the same algorithm consistently preserves portfolio viability and outperforms vanilla PPO and the equal-weight baseline. Across both settings, empirical runtimes grow *linearly* with the number of components, validating the scalability predicted by our complexity analysis. Future work will focus on extending the framework's capabilities to more dynamic budget allocation schemes and more complicated hierarchical budget constraints.

References

Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In *23rd Advances in Neural Information Processing Systems*, 2010.

Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Reinforcement learning of POMDPs using spectral methods. In *Conference on Learning Theory*, pp. 193–256, 2016.

Frank Benham and Colin Bebee. Risk budgeting primer. Technical report, Meketa Investment Group, 2024. URL https://meketa.com/wp-content/uploads/2024/10/MEKETA_Risk-Budgeting-Primer.pdf.

Sushmita Bhattacharya, Sahil Badyal, Thomas Wheeler, Stephanie Gil, and Dimitri Bertsekas. 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, 2020.

Sushmita Bhattacharya, Siva Kailas, Sahil Badyal, Stephanie Gil, and Dimitri Bertsekas. Multiagent rollout and policy iteration for pomdp with application to multi-robot repair problems. In *Conference on Robot Learning*, pp. 1814–1828, 2021.

František Blahoudek, Tomáš Brázdil, Petr Novotnỳ, Melkior Ornik, Pranay Thangeda, and Ufuk Topcu. Qualitative controller synthesis for consumption markov decision processes. In *International Conference on Computer Aided Verification*, pp. 421–447, 2020.

Craig Boutilier and Tyler Lu. Budget allocation using weakly coupled, constrained markov decision processes. In *UAI*, 2016.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Raissa Zurli Bittencourt Bravo, Adriana Leiras, and Fernando Luiz Cyrino Oliveira. The use of uavs in humanitarian relief: An application of pomdp-based methodology for finding victims. *Production and Operations Management*, 28(2):421–440, 2019.

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

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

Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *12th AAAI National Conference on Artificial Intelligence*, pp. 1023–1028, 1994.

Shamsuddin Daulat, Marius Møller Rokstad, Alex Klein-Paste, Jeroen Langeveld, and Franz Tscheikner-Gratl. Challenges of integrated multi-infrastructure asset management: A review of pavement, sewer, and water distribution networks. *Structure and Infrastructure Engineering*, 20(4):546–565, 2024.

Michael N. Grussing, Donald R. Uzarski, and Lance R. Marrano. Condition and reliability prediction models using the Weibull probability distribution. In *Applications of Advanced Technology in Transportation*, 2006.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pp. 2117–2126, 2018.

Sammie Katt, Frans A Oliehoek, and Christopher Amato. Learning in pomdps with monte carlo tree search. In *International Conference on Machine Learning*, pp. 1819–1827, 2017.

Michael Kearns, Yishay Mansour, and Andrew Ng. Approximate planning in large pomdps via reusable trajectories. In *12th Advances in Neural Information Processing Systems*, 1999.

Majid Khonji, Ashkan Jasour, and Brian C Williams. Approximability of constant-horizon constrained pomdp. In *International Joint Conferences on Artificial Intelligence*, pp. 5583–5590, 2019.

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

Jongmin Lee, Geon-Hyeong Kim, Pascal Poupart, and Kee-Eung Kim. Monte-carlo tree search for constrained pomdps. In *31st Advances in Neural Information Processing Systems*, 2018.

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

Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon markov decision process problems. *Journal of the ACM*, 47(4):681–720, 2000.

Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conferences on Artificial Intelligence*, pp. 1025–1032, 2003.

Pascal Poupart and Craig Boutilier. Value-directed compression of pomdps. In *15th Advances in Neural Information Processing Systems*, 2002.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

Joohyun Shin and Jay H Lee. Mdp formulation and solution algorithms for inventory management with multiple suppliers and supply and demand uncertainty. In *Computer Aided Chemical Engineering*, volume 37, pp. 1907–1912. Elsevier, 2015.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *23rd Advances in Neural Information Processing Systems*, 2010.

Gautam Singh, Skand Peri, Junghyun Kim, Hyunseok Kim, and Sungjin Ahn. Structured world belief for reinforcement learning in POMDP. In *International Conference on Machine Learning*, pp. 9744–9755, 2021.

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

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

Aditya Undurti and Jonathan P How. An online algorithm for constrained pomdps. In *IEEE International Conference on Robotics and Automation*, pp. 3966–3973, 2010.

Nikos Vlassis, Michael L Littman, and David Barber. On the computational complexity of stochastic controller optimization in pomdps. *ACM Transactions on Computation Theory*, 4(4):1–8, 2012.

Manav Vora, Pranay Thangeda, Michael N. Grussing, and Melkior Ornik. Welfare maximization algorithm for solving budget-constrained multi-component pomdps. *IEEE Control Systems Letters*, 7:1736–1741, 2023.

A Function Approximation of $\mathbb{E}[T_{\max}]$

We model $\mathbb{E}[T^i_{\max}]$ as an exponential function of the budget allocated to component i. The choice of an exponential function is motivated by its ability to capture the saturation in $\mathbb{E}[T^i_{\max}]$ values at higher budget levels, a result of the finite planning horizon H. Additionally, the exponential model accounts for non-zero $\mathbb{E}[T^i_{\max}]$ even when the budget is zero.

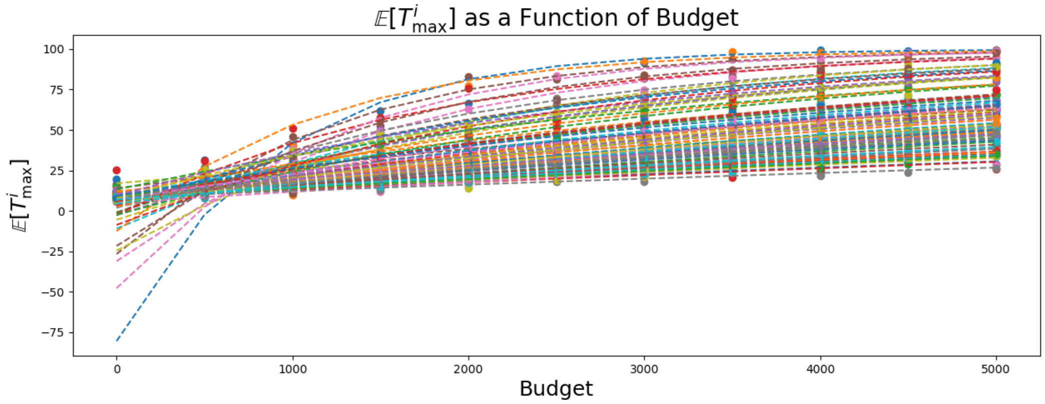

Figure 9: Exponential $\tilde{T}^i_{\max}$ curves obtained using non-linear least-squares regression.

To validate the accuracy of this exponential model for $\mathbb{E}[T^i_{\max}]$, we conducted non-linear least squares regression on 100 infrastructure components. Figure 9 illustrates the curves obtained through this regression, where $\mathbb{E}[T^i_{\max}]$ is modeled as an exponential function. The results indicate that the exponential function provides a strong approximation for $\mathbb{E}[T^i_{\max}]$, with an average coefficient of determination $R^2_{mean} = 0.899$.