

Scalable Finite Adaptability via Polyhedral Partition and Learning

Zolykha Rezaei *

Ningji Wei †

Eojin Han ‡

June 3, 2026

Abstract

We study finite adaptability for decision-making under uncertainty, where a small set of candidate solutions is prepared in advance and the best response is selected after uncertainty is realized. While existing methods have made significant progress on exact formulations, scalability remains a persistent challenge due to (i) the combinatorial nature of assigning decisions to uncertainty realizations, and (ii) the joint optimization of uncertainty set partition and subsequent decisions. We propose a framework that makes the partition of the uncertainty set explicit and uses polyhedral partitions as the basis for policy design. Under mild regularity conditions and for general risk measures, we show that such policies converge to the optimal fully adjustable policy as the number of regions increases. Building on this result, we develop a parametric partition framework that allows flexible policy design with tractable reformulations for both robust and stochastic finite adaptability problems. To improve scalability, we introduce an approximate-learn-parallel framework that integrates partition learning with parallel optimization while preserving solution robustness. Computational experiments on classical testbeds in both robust and stochastic settings show that the proposed method scales to larger instances and yields competitive policy performance.

1 Introduction

Decision-making under uncertainty arises naturally across many domains, including logistics, production planning, finance, and emergency response, and continues to grow in importance. In these settings, the decision maker often needs to navigate the trade-off between implementation readiness and response optimality. An *ex ante* approach commits to a solution before uncertainty is resolved, ensuring that a vetted plan is available for immediate deployment across all plausible scenarios. An *ex post* approach defers the decision until uncertainty is revealed, enabling an optimal response at the cost of compressing the time available for computation, communication, and deployment. In practice, contingency planning offers a balanced middle ground: rather than committing to a single solution or postponing decisions entirely, it prepares a menu of candidate solutions in advance, allowing rapid adaptation to realized conditions while preserving implementation readiness.

*Department of Industrial, Manufacturing & Systems Engineering, Texas Tech University, Lubbock, TX 79409 (email: zrezaei@ttu.edu).

†Department of Industrial, Manufacturing & Systems Engineering, Texas Tech University, Lubbock, TX 79409 (email: ningji.wei@ttu.edu).

‡Information Technology, Analytics and Operations Mendoza College of Business, University of Notre Dame, Notre Dame, IN 46556 (email: ehan3@nd.edu).

This idea of contingency planning has been formalized in the optimization literature in several closely related forms. In robust optimization, a widely studied formulation is K -adaptability, where K candidate decisions are optimized against worst-case uncertainty. When such candidates serve as recourse decisions in two-stage or multistage problems, the contingency plan provides a tractable approximation to fully adaptive recourse [10, 33, 45]. A related stream studies min–max–min formulations without first-stage variables, focusing entirely on preparing and selecting among candidate responses [20, 29, 36]. Most of these works fix the number of candidate decisions K in advance; when this restriction is relaxed, a complementary line of research constructs partitions of the uncertainty set dynamically, refining both the regions and the associated responses adaptively [12, 40]. Similar ideas have also been explored in stochastic programming [22, 38], where the focus is often on selecting among candidate responses without first-stage decisions (min– \mathbb{E} –min) under a finite support. Because the partition in this setting takes the form of a scenario-to-policy assignment rather than over a continuous domain, the techniques developed in robust and stochastic settings have largely evolved independently. Finite adaptability has also been studied in distributionally robust optimization [32, 34], where the support of an ambiguity set is partitioned and a contingent decision is assigned to each region.

Despite differences in modeling perspectives, these formulations share a common core: the uncertainty set is partitioned into regions, either implicitly or explicitly, where each region is associated with one decision, a problem structure we collectively refer to as *finite adaptability*. This paper focuses on both the robust (min–max–min) and stochastic (min– \mathbb{E} –min) settings without first-stage decisions. A detailed comparison with existing formulations is provided in Table 1.

Significant progress has been made in developing solution methods for finite adaptability, including mixed integer reformulations [33], branch-and-bound algorithms [45], and iterative partition refinement schemes [12, 40]. Despite these advances, scalability remains a persistent challenge, driven by three interrelated sources of complexity. First, assigning scenarios to candidate responses induces a combinatorial structure whose size grows rapidly with both the number of scenarios and the number of candidate decisions. Second, when the uncertainty set is continuous, this assignment extends to a semi-infinite setting, requiring robustness over infinitely many realizations within each region. Third, the partitioning of the uncertainty set and the optimization of the associated decisions are tightly coupled, necessitating the joint optimization of all K responses alongside the partition itself, even though solving a single such problem can already be computationally demanding. As a result, state-of-the-art exact methods can encounter difficulty beyond moderate problem sizes. To the best of our knowledge, for capital budgeting problems with a four-dimensional uncertainty set and 30 projects, existing exact approaches reach a two-hour time limit on most test instances with $K = 4$ candidate decisions [33, 45].

This paper aims to advance the scalability of finite adaptability, guided by two observations. First, any finite adaptability model induces an implicit partition of the uncertainty set, and making this partition explicit enables new formulations, optimization strategies, and algorithmic design. Second, because finite adaptability is itself an approximation of the fully adjustable policy, a contingency plan with a larger number of responses, even constructed approximately, can outperform an exact solution restricted to fewer responses. Building on these observations, three contributions follow.

On the theoretical side, we establish convergence of polyhedral finite-adaptable policies to the fully adjustable benchmark as the number of regions grows, under mild regularity conditions and for general risk measures. This result motivates the use of polyhedral partitions as the basis of our

Table 1 Overview of finite adaptability in robust optimization, stochastic programming and distributionally robust optimization.

Reference	Type	K	Model	Uncertainty	Recourse Variables
Bertsimas and Caramanis [10]	RO	Fixed	Multistage	General	General
Hanasusanto et al. [33]	RO	Fixed	Two-stage	General	Binary
Bertsimas and Georghiou [13]	RO	Fixed	Multistage	General	General
Bertsimas and Dunning [12]	RO	Not fixed	Multistage	General	General
Postek and den Hertog [40]	RO	Not fixed	Multistage	General	General
Buchheim and Kurtz [20]	RO	Fixed	Min-max-min	Objective	Binary
Subramanyam et al. [45]	RO	Fixed	Two-stage	General	General
Buchheim and Prunte [22]	SP	Fixed	Min- \mathbb{E} -min	Objective	Binary
Malaguti et al. [38]	SP	Fixed	Min- \mathbb{E} -min	General	Continuous
Donnini et al. [25]	SP	Fixed	Two-stage	General	General
Hanasusanto et al. [34]	DRO	Fixed	Two-stage	General	Binary
Han et al. [32]	DRO	Fixed	Two-stage	RHS	General
This paper	RO / SP	Not Fixed	Min-max-min / Min- \mathbb{E} -min	General	General

Notes. RO, SP, and DRO denote robust optimization, stochastic programming, and distributionally robust optimization. The K column indicates whether candidate policies are fixed or determined adaptively. “General” in the uncertainty column indicates uncertainty beyond the objective or right-hand side; in the recourse column, it indicates applicability to both integer and continuous variables.

framework. On the formulation side, we introduce a parametric polyhedral partition framework that accommodates multiple partition schemes, each inducing a distinct policy class with different formulation and tractability properties. We derive both a master-subproblem procedure and a piecewise dualization reformulation, the latter exploiting the explicit polyhedral structure of each region. On the computational side, we develop a learning-based approach that combines partition learning with parallel optimization while preserving solution robustness, where the learned partition can be further refined through a tailored reformulation. Computational experiments on classical testbeds show that the approach scales to larger instances and yields competitive policy performance. To our knowledge, the resulting framework is the first applicable to finite adaptability in both robust and stochastic settings. Beyond scalability, making the partition explicit yields an additional benefit: each response is optimized within its own region rather than solely against the worst case, preserving robustness guarantees while improving performance on non-worst-case scenarios.

1.1 Literature Review

Adjustable robust optimization is a general framework that bridges *ex ante* and *ex post* optimization under uncertainty, seeking decision rules that adapt solutions to realized uncertainty. First introduced by Ben-Tal et al. [7] in robust optimization, the framework has since attracted theoretical studies on performance analysis and solution methods [3, 7, 14, 17, 18, 39, 47] as well as diverse applications across inventory management [1, 5, 19], supply chain design [44], power systems [41], and transportation [37]. In both robust optimization and stochastic programming, different families of decision rules have been studied for their trade-off between policy performance and computational tractability, including affine [7, 15, 16, 23, 35], lifted affine [26, 28], piecewise-affine [9, 31, 46],

polynomial [4, 49], and piecewise-constant policies [10], the last of which is also known as finite adaptability. Dual decision rules, which apply to Lagrangian multipliers rather than primal variables, have also been developed in both robust and stochastic settings [24, 35]. We refer interested readers to [27, 50] for comprehensive surveys.

Among these decision rule families, finite adaptability, which corresponds to piecewise-constant policies, is especially well suited to discrete decision spaces and yields policies that are interpretable in practice. Since its introduction by Bertsimas and Caramanis [10], the framework has been extended along multiple directions within robust optimization. For two-stage problems, Hanasusanto et al. [33] derived mixed integer formulations for binary recourse, while Subramanyam et al. [45] proposed an exact partition-based branch-and-bound algorithm for mixed integer settings. In multistage contexts, Bertsimas and Georghiou [13] studied the design of near-optimal decision rules and showed that restricted adaptability can recover much of the benefit of full adjustability. A complementary line of work focuses on adaptive partitioning schemes that do not fix the number of candidate policies *a priori*, including the frameworks of Bertsimas and Dunning [12] and Postek and den Hertog [40]. A closely related perspective arises in min–max–min robust optimization, where the absence of first-stage decisions leads naturally to a finite policy selection problem; this viewpoint was formalized by Buchheim and Kurtz [20, 21] and subsequently extended to broader settings [29, 36].

Beyond robust optimization, finite adaptability has been extended to stochastic and distributionally robust settings. In stochastic optimization, Buchheim and Prunte [22] study the min– \mathbb{E} –min paradigm for combinatorial problems under objective uncertainty, while Malaguti et al. [38] analyze its complexity and develop exact algorithms. Donnini et al. [25] provide a more general treatment by studying two-stage stochastic programs with uncertainty in both objectives and constraints, establishing approximation guarantees and proposing exact partition-based methods. In distributionally robust optimization, Hanasusanto et al. [34] consider the problem setting with binary recourse, and Han et al. [32] extend the framework to structured partitions in two-stage settings. Table 1 summarizes this literature across optimization paradigms, model structures, and uncertainty settings.

1.2 Positioning Our Work

Many existing methods for finite adaptability have been developed from an *algebraic* perspective, following the logic of selecting the best decision after observing uncertainty. Despite their exactness or approximation guarantees, the computational scalability of these methods remains a challenge. In contrast, alternative approaches take a *geometric* perspective, viewing finite adaptability as a partitioning problem, i.e., identifying a mapping from the uncertainty space to a finite number of decisions. Our work lies in this latter stream by learning effective partitions of the uncertainty space to obtain *computationally scalable* solutions for larger and more realistic instances. As our computational results will show, the two perspectives exhibit complementary strengths in terms of solution quality: algebraic approaches are particularly well suited to problems with objective-only uncertainty, where exact worst-case evaluation largely determines the objective value, while partition-based approaches become preferable when uncertainty also affects constraints, where the geometric structure of the partition yields better objective values alongside faster computation.

The remainder of this paper is organized as follows. Section 2 introduces finite adaptability and analyzes the asymptotic performance of polyhedral policies. Section 3 develops the parametric polyhedral partition framework, including three partition schemes and exact reformulations. Section 4

presents a scalable learning-and-parallel-optimization framework, and Section 5 extends it to stochastic programming. Section 6 reports experiments on classical robust and stochastic testbeds, and Section 7 concludes. Throughout this paper, bold lowercase letters (e.g., $\mathbf{x} \in \mathbb{R}^n$) denote vectors and bold uppercase letters (e.g., $\mathbf{A} \in \mathbb{R}^{m \times n}$) denote matrices. The symbol $\mathbf{1}$ denotes the all-ones vector, with dimension understood from context. We write \odot for element-wise multiplication, $\langle \cdot, \cdot \rangle$ for the inner product and $[q] = \{1, \dots, q\}$.

2 Preliminaries on Finite Adaptability

This section introduces finite adaptability and analyzes the approximation performance of polyhedron-based finite-adaptable policies. We consider a decision problem under uncertainty with solution space \mathcal{Y} , a convex and compact uncertainty set Ξ , and a cost function $f(\mathbf{y}, \boldsymbol{\xi})$ taking values in $\mathbb{R} \cup \{\pm\infty\}$. We adopt extended-value conventions for infeasibility and unboundedness, and assume that all problems with finite value admit optimal solutions. The general *fully adjustable problem* is

$$\min_{\pi(\cdot) \in \mathcal{Y}^\Xi} \mathcal{R}_\Xi(f(\pi(\boldsymbol{\xi}), \boldsymbol{\xi})), \quad (1)$$

where \mathcal{Y}^Ξ is the policy space containing all functions from Ξ to \mathcal{Y} , and the functional $\mathcal{R}_\Xi(\cdot)$, termed the *risk measure* [2, 42, 43], aggregates objective values across uncertainty realizations in Ξ and represents a chosen risk or robustness criterion, such as the worst-case operator $\mathcal{R}_\Xi(f(\mathbf{y}, \boldsymbol{\xi})) := \max_{\boldsymbol{\xi} \in \Xi} f(\mathbf{y}, \boldsymbol{\xi})$ in robust optimization, or the expectation $\mathcal{R}_\Xi(f(\mathbf{y}, \boldsymbol{\xi})) := \mathbb{E}_{\boldsymbol{\xi} \sim \mathbb{P}}[f(\mathbf{y}, \boldsymbol{\xi})]$ under a given probability measure \mathbb{P} in stochastic programming. It satisfies three basic properties: (i) *monotonicity*, i.e., for any $f, g : \Xi \rightarrow \mathbb{R}$, $f \leq g$ implies $\mathcal{R}_\Xi(f) \leq \mathcal{R}_\Xi(g)$; (ii) *normalization*, i.e., $\mathcal{R}_\Xi(0) = 0$; and (iii) *translation equivalence*, i.e., $\mathcal{R}_\Xi(f + \alpha) = \mathcal{R}_\Xi(f) + \alpha$ for any constant α . The following lemma derives two additional properties of \mathcal{R}_Ξ that will be used later.

Lemma 1. *Every risk measure \mathcal{R}_Ξ also satisfies the following two properties: (i) worst-case upper bound: $\mathcal{R}_\Xi(f) \leq \max_{\boldsymbol{\xi} \in \Xi} f(\boldsymbol{\xi})$; (ii) 1-Lipschitz under ∞ -norm: $|\mathcal{R}_\Xi(f) - \mathcal{R}_\Xi(g)| \leq \|f - g\|_\infty$.*

Due to the monotonicity of \mathcal{R}_Ξ , the optimal policy of (1) can be obtained via pointwise optimization $\pi(\boldsymbol{\xi}) \in \arg \min_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}, \boldsymbol{\xi})$. While this characterization is conceptually simple, solving (1) exactly is intractable in general [7, 8]. Finite adaptability is one policy class that has been introduced to balance adjustability and tractability [10]. It can be viewed as a form of contingency planning, in which a finite set of candidate decisions is precomputed to prepare for the realized uncertainty. This approach can be generally formulated as

$$\min_{\mathbf{y} \in \mathcal{Y}^\mathcal{K}, \sigma(\cdot) \in \mathcal{K}^\Xi} \mathcal{R}_\Xi(f(\mathbf{y}_{\sigma(\boldsymbol{\xi})}, \boldsymbol{\xi})), \quad (2)$$

where $\mathcal{K} := [K]$, $\mathbf{y} = (\mathbf{y}_k)_{k \in \mathcal{K}} \in \mathcal{Y}^\mathcal{K}$ denotes a collection of K candidate solutions from \mathcal{Y} , and the policy $\sigma(\cdot)$ selects a solution index based on the realized uncertainty $\boldsymbol{\xi} \in \Xi$. The following observation shows that any such K -adaptability problem is equivalent to identifying an optimal partition of Ξ , since each policy $\sigma \in \mathcal{K}^\Xi$ induces a unique partition $\{\sigma^{-1}(k)\}_{k \in \mathcal{K}}$ of Ξ .

Observation 1. Problem (2) can be equivalently formulated as the following

$$\min_{\substack{\mathbf{y}:=(\mathbf{y}_k)_{k \in \mathcal{K}} \in \mathcal{Y}^\mathcal{K} \\ \mathcal{P}:=(\Xi_k)_{k \in \mathcal{K}} \in \Pi_K(\Xi)}} \mathcal{R}_\Xi(f(\mathbf{y}_{\sigma_{\mathcal{P}}(\boldsymbol{\xi})}, \boldsymbol{\xi})), \quad (3)$$

where $\Pi_K(\Xi)$ denotes the collection of K -partitions of Ξ , and $\sigma_{\mathcal{P}}(\xi) := \min\{k \in \mathcal{K} \mid \xi \in \Xi_k\}$ selects the smallest index k such that ξ belongs to Ξ_k .

From an optimization standpoint, allowing empty regions or overlaps between distinct sets Ξ_k and $\Xi_{k'}$ does not affect either solution validity or policy implementation in (3), provided that $f(\mathbf{y}_k, \xi) = f(\mathbf{y}_{k'}, \xi)$ for all $\xi \in \Xi_k \cap \Xi_{k'}$. Moreover, any such collection can be refined into a genuine partition of Ξ by applying the tie-breaking rule encoded in $\sigma_{\mathcal{P}}(\cdot)$. Since these overlaps and refinements are immaterial to the behavior of the optimization problem yet introduce unnecessary technical complications, we adopt the following convention.

Remark 1. Throughout the paper, we use the term *partition* to refer to a collection of sets whose union equals the underlying space Ξ , while allowing empty parts and boundary overlaps.

Although the two formulations are equivalent, (3) makes the partitioning of the uncertainty set explicit within the optimization problem. Interestingly, prior work on K -adaptability in linear settings has shown that the partition $\{\Xi_k\}$ induced by an optimal solution admits a polyhedral structure. For example, the result of Hanasusanto et al. [33] implies that under objective uncertainty, an optimal partition can always be chosen to be polyhedral, whereas Han et al. [32] characterizes an optimal structure of partitions whose regions are representable as finite unions of polyhedra under right-hand side uncertainty. The following proposition provides a criterion under which a generalized version of this result can be established.

Proposition 1. *Suppose that $f(\mathbf{y}, \cdot)$ is affine in ξ for every $\mathbf{y} \in \mathcal{Y}$. Then there exists an optimal partition $(\Xi_k)_{k \in \mathcal{K}}$ for (3) such that each Ξ_k is the intersection of Ξ with a polyhedron. More generally, if $f(\mathbf{y}, \cdot)$ is piecewise affine in ξ with finitely many polyhedral pieces for every $\mathbf{y} \in \mathcal{Y}$, then there exists an optimal partition $(\Xi_k)_{k \in \mathcal{K}}$ for (3) such that each Ξ_k is the intersection of Ξ with a finite union of polyhedra.*

This proposition has two immediate implications. First, for K -adaptability problems with a polyhedral uncertainty set Ξ and objective uncertainty only, where the objective function is affine in ξ , there exists an optimal partition that is polyhedral. Second, if the problem additionally involves constraint uncertainty, with all constraint functions affine in ξ , then the problem admits an optimal polyhedral partition consisting of $K' \geq K$ polyhedra. This is because these constraints can be incorporated into the objective via indicator functions, which induce a piecewise-affine structure and, consequently, a further refinement of the partition. In what follows, we refer to finitely adaptable policies with polyhedral partitions as *polyhedral finite-adaptable policies*.

For more general settings, the following theorem establishes that, as $K \rightarrow \infty$, polyhedral finite-adaptable policies converge to the performance of the fully adjustable benchmark under mild regularity conditions. For any policy $\pi : \Xi \rightarrow \mathcal{Y}$, we define $\phi_{\pi}(\xi) := f(\pi(\xi), \xi)$ as its value function, and define $\mathcal{R}_{\Xi}(\phi_{\pi})$ as its aggregated value under \mathcal{R}_{Ξ} .

Theorem 1. *Suppose $\Xi \subseteq \mathbb{R}^n$ is compact. Let $\pi(\xi) \in \arg \min_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}, \xi)$ be the optimal fully adjustable policy. If, for every $\mathbf{y} \in \mathcal{Y}$, the function $f(\mathbf{y}, \cdot)$ is upper semicontinuous on Ξ , then there exists a family of polyhedral finite-adaptable policies $(\pi_{\epsilon})_{\epsilon > 0}$ such that their value functions $\phi_{\pi_{\epsilon}}$ converge pointwise to ϕ_{π} as $\epsilon \rightarrow 0$. Moreover, if $f(\mathbf{y}, \cdot)$ is Lipschitz continuous on Ξ for every $\mathbf{y} \in \mathcal{Y}$, then the aggregated values also converge as $\lim_{\epsilon \downarrow 0} \mathcal{R}_{\Xi}(\phi_{\pi_{\epsilon}}) = \mathcal{R}_{\Xi}(\phi_{\pi})$.*

Motivated by the approximation capabilities of polyhedral finite-adaptable policies established by these results, the remainder of the paper focuses on solution methods for this polyhedral finite-adaptable policy class. For the clarity of exposition, we focus on problems satisfying the following

structural assumptions: (i) the uncertainty set Ξ is polyhedral; and (ii) both the objective function and the constraints are affine in ξ . Most techniques developed under this setting extend naturally to more general convex settings. Moreover, when Ξ is discrete, any partition of a convex outer approximation (e.g., its convex hull or LP relaxation) induces a corresponding partition of Ξ , which enables our framework to be applied in discrete uncertainty settings as well.

3 Finite Adaptability with Parametric Polyhedral Partition

In this section, we study the following *robust* K -adaptability problem in linear settings. Its extension to stochastic optimization will be discussed in Section 5.

$$\begin{aligned}
& \min_{\substack{\mathbf{y} := (\mathbf{y}_k)_{k \in \mathcal{K}} \in \mathcal{Y}^{\mathcal{K}} \\ \mathcal{P} := (\Xi_k)_{k \in \mathcal{K}} \in \Pi_K(\Xi)}} z \\
& \text{s.t.} \quad \langle \mathbf{Q}(\mathbf{y}_k), \xi \rangle + \mathbf{q}(\mathbf{y}_k) \leq z, \quad \forall k \in \mathcal{K}, \xi \in \Xi_k \\
& \quad \quad \langle \mathbf{Q}_j(\mathbf{y}_k), \xi \rangle + \mathbf{q}_j(\mathbf{y}_k) \leq 0, \quad \forall k \in \mathcal{K}, j \in J, \xi \in \Xi_k.
\end{aligned} \tag{4}$$

The index set J labels the constraints, and the polyhedral uncertainty set $\Xi := \{\xi \in \mathbb{R}^n \mid \mathbf{B}\xi \leq \mathbf{b}\}$ is assumed to be compact and nonempty. The mappings $\mathbf{Q}, \mathbf{Q}_j : \mathcal{Y} \rightarrow \mathbb{R}^n$ and $\mathbf{q}, \mathbf{q}_j : \mathcal{Y} \rightarrow \mathbb{R}$ are functions defined over the solution space \mathcal{Y} . Hence, this problem seeks to jointly optimize a K -partition $\mathcal{P} = (\Xi_k)_{k \in \mathcal{K}}$ of the uncertainty set and a collection of responses $\mathbf{y} = (\mathbf{y}_k)_{k \in \mathcal{K}}$, such that each response \mathbf{y}_k is robust with respect to all scenarios in the corresponding region Ξ_k . We address this via the *parametric polyhedral partition* framework introduced below.

3.1 Parametric Polyhedral Partition

To ensure $\mathcal{P} = (\Xi_k)_{k \in \mathcal{K}}$ forms a partition of Ξ with a finite number of parameters to optimize, we begin with the following definition.

Definition 1 (Parametric Polyhedral Partition). Let \mathbb{R}^ℓ be a target space equipped with a predefined partition $\mathcal{L} = (\mathcal{L}_k)_{k \in \mathcal{K}}$, where each region is defined as $\mathcal{L}_k := \{\mathbf{x} \in \mathbb{R}^\ell \mid \mathbf{B}_k \mathbf{x} \leq \mathbf{b}_k\}$, and let $\xi \mapsto \mathbf{T}_\theta \xi + \boldsymbol{\tau}_\theta$ be a parametric affine mapping with $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$. We refer to the tuple $(\mathcal{L}, \mathbf{T}, \boldsymbol{\tau})$ as a *partition scheme*. Then, the sets $\mathcal{P}_\theta = (\Xi_k^\theta)_{k \in \mathcal{K}}$, with

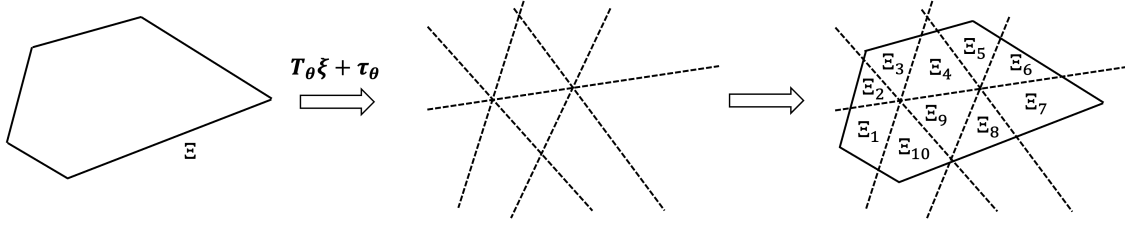
$$\Xi_k^\theta := \{\xi \in \Xi \mid \mathbf{B}_k(\mathbf{T}_\theta \xi + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_k\},$$

is called the *parametric polyhedral partition* induced by $\boldsymbol{\theta}$ under the scheme $(\mathcal{L}, \mathbf{T}, \boldsymbol{\tau})$.

Remark 2. Although the parameters $(\mathbf{B}_k, \mathbf{b}_k)$ may appear redundant, as their role can be absorbed into $(\mathbf{T}_\theta, \boldsymbol{\tau}_\theta)$, enforcing partition constraints directly on $(\mathbf{T}_\theta, \boldsymbol{\tau}_\theta)$ is difficult in general. Prefixing a partition through an explicit design of $(\mathbf{B}_k, \mathbf{b}_k)$ addresses this.

Figure 1 illustrates this framework. The global partition \mathcal{L} of the target space can be specified explicitly via \mathbf{B}_k and \mathbf{b}_k (e.g., orthants or box partitions), thereby ensuring that the induced collection \mathcal{P}_θ forms a partition. The parametric mapping $(\mathbf{T}, \boldsymbol{\tau})$ governs the expressiveness of the resulting policy class and enables a finite-dimensional optimization over partitions. Moreover, since the dimensions d and ℓ of the parameter and target spaces are user-specified design choices, the framework offers flexibility to balance policy granularity and computational tractability. The following proposition ensures that the collection \mathcal{P}_θ is indeed a partition of Ξ .

Figure 1: Illustration of the parametric polyhedral partition framework.



Note. Let $\mathcal{L} = (\mathcal{L}_k)_{k \in \mathcal{K}}$ denote a fixed polyhedral partition of the target space \mathbb{R}^2 , where each part is $\mathcal{L}_k = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{B}_k \mathbf{x} \leq \mathbf{b}_k\}$. For a given parameter $\boldsymbol{\theta}$, the affine map $\mathbf{x} = \mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta$ embeds the uncertainty set $\Xi = \{\boldsymbol{\xi} \in \mathbb{R}^2 \mid \mathbf{B} \boldsymbol{\xi} \leq \mathbf{b}\}$ into the target space. The preimages of the parts \mathcal{L}_k under this map induce a partition of Ξ into polyhedral regions $\Xi_k^\theta = \{\boldsymbol{\xi} \in \Xi \mid \mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_k\}$. As $\boldsymbol{\theta}$ varies, the image of Ξ moves relative to the fixed partition \mathcal{L} , thereby inducing a $\boldsymbol{\theta}$ -dependent polyhedral partition of Ξ .

Proposition 2. *For any partition scheme $(\mathcal{L}, \mathbf{T}, \boldsymbol{\tau})$, the induced collection \mathcal{P}_θ constitutes a partition of the uncertainty set Ξ for every $\boldsymbol{\theta} \in \Theta$.*

Under a given partition scheme $(\mathcal{L}, \mathbf{T}, \boldsymbol{\tau})$, Problem (4) can be rewritten as

$$\min_{\substack{\mathbf{y} := (\mathbf{y}_k)_{k \in \mathcal{K}} \in \mathcal{Y}^{\mathcal{K}} \\ \boldsymbol{\theta} \in \Theta}} z \tag{5a}$$

$$\text{s.t.} \quad \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}(\mathbf{y}_k) \leq z, \quad \forall k \in \mathcal{K}, \boldsymbol{\xi} \in \Xi_k^\theta \tag{5b}$$

$$\langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}_j(\mathbf{y}_k) \leq 0, \quad \forall k \in \mathcal{K}, j \in J, \boldsymbol{\xi} \in \Xi_k^\theta, \tag{5c}$$

where each region $\Xi_k^\theta := \{\boldsymbol{\xi} \in \mathbb{R}^n \mid \mathbf{B} \boldsymbol{\xi} \leq \mathbf{b}, \mathbf{B}_k \mathbf{T}_\theta \boldsymbol{\xi} \leq \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta\}$ is a parametric polytope. Different choices of partition schemes give rise to distinct formulations and, consequently, distinct classes of policies. In the following subsection, we introduce three specific partition schemes.

3.2 Three Partition Schemes

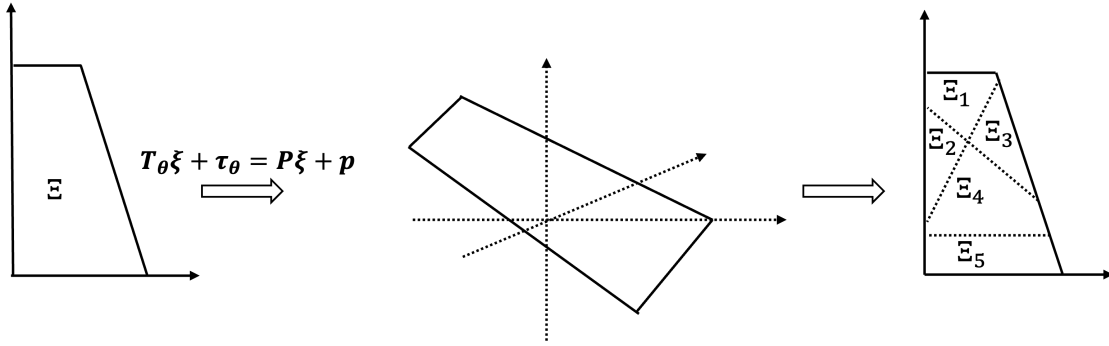
This subsection introduces an orthant- and two tree-based partition schemes. These approaches are also closely related to classical learning methods such as support vector machines (SVMs) and decision trees (DTs), enabling learning-based algorithms developed later in Section 4.

3.2.1 Orthant-Based Partition Scheme

Figure 2 illustrates the orthant-based partition scheme. An uncertainty set $\Xi \subseteq \mathbb{R}^2$ is embedded into the target space \mathbb{R}^3 via the transformation $\mathbf{P} \boldsymbol{\xi} + \mathbf{p}$. The orthants in \mathbb{R}^3 partition $\mathbf{P} \Xi + \mathbf{p}$ and naturally induce a corresponding partition of Ξ in the original space. Moreover, the resulting partition depends on the parameter $\boldsymbol{\theta} = (\mathbf{P}, \mathbf{p})$. The following definition formalizes this construction.

Definition 2 (Orthant-Based Partition Scheme). Let the target space be \mathbb{R}^ℓ , and let $\mathcal{K} := [2^\ell]$ index the 2^ℓ orthants, each identified by a sign vector $\mathbf{s}_k \in \{1, -1\}^\ell$. The associated orthant-based partition $\mathcal{L} = (\mathcal{L}_k)_{k \in \mathcal{K}}$ is defined by $\mathcal{L}_k := \{\mathbf{x} \in \mathbb{R}^\ell \mid \mathbf{s}_k \odot \mathbf{x} \geq \mathbf{0}\}$. Let $\boldsymbol{\theta} = (\mathbf{P}, \mathbf{p})$, the associated affine mapping is defined by $\mathbf{T}_\theta(\boldsymbol{\xi}) + \boldsymbol{\tau}_\theta := \mathbf{P} \boldsymbol{\xi} + \mathbf{p}$. The induced parametric polyhedral partition is $\Xi_k^\theta := \{\boldsymbol{\xi} \in \Xi \mid \mathbf{s}_k \odot (\mathbf{P} \boldsymbol{\xi} + \mathbf{p}) \geq \mathbf{0}\}$ for each $k \in \mathcal{K}$.

Figure 2: Illustration of the orthant-based partition scheme.



Note. An uncertainty set $\Xi \subseteq \mathbb{R}^2$ is embedded into the target space \mathbb{R}^3 via $P\xi + p$. The orthants of \mathbb{R}^3 partition the image $P\xi + p$, and the inverse mapping induces a parametric partition of Ξ into five regions, separated by the preimages of the orthant boundaries.

As illustrated in Figure 2, although the orthants are mutually orthogonal in the target space, the induced partition on Ξ is highly flexible, since each row of the parameters P and p defines an affine hyperplane in the original space that separates Ξ . This partitioning mechanism is similar to SVM, which uses multiple hyperplanes to divide the feature space into distinct regions.

3.2.2 Decision Tree Partition Scheme

This partition scheme is motivated by decision trees in machine learning, where a sequence of coordinate-wise thresholding rules induces a hierarchical partition of the input domain. In our setting, the same principle is used to construct a polyhedral partition of Ξ by splitting along selected coordinates at optimizable thresholds, as defined below.

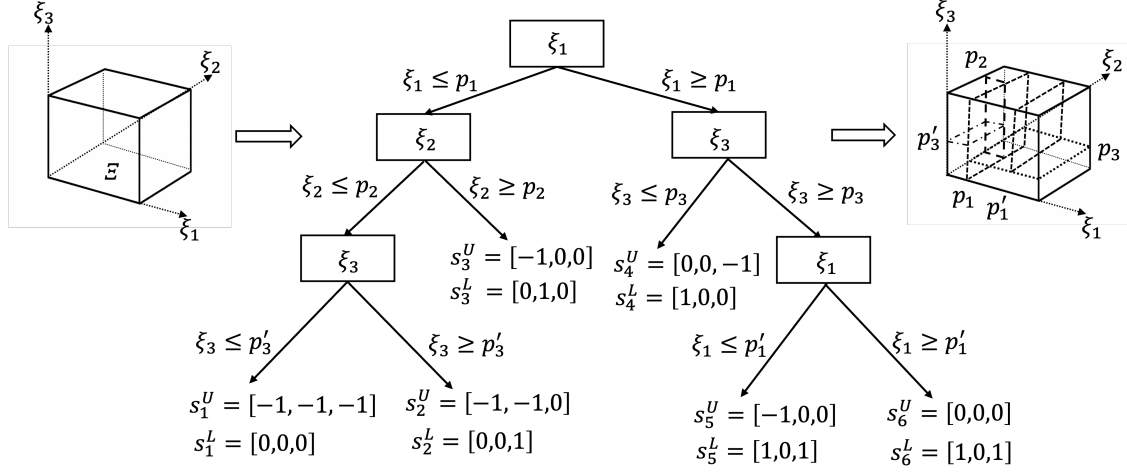
Definition 3 (Decision Tree Partition Scheme). Given a tree structure with leaf set \mathcal{K} , we associate a vector $\mathbf{s}_k = (\mathbf{s}_k^U, \mathbf{s}_k^L)$ with each $k \in \mathcal{K}$ to encode the splitting decisions along the path from the root to that leaf, defined componentwise by

$$(\mathbf{s}_k^U)_i = \begin{cases} -1, & \text{if } \xi_i \text{ is bounded above at } k \\ 0, & \text{otherwise} \end{cases}, \quad (\mathbf{s}_k^L)_i = \begin{cases} 1, & \text{if } \xi_i \text{ is bounded below at } k \\ 0, & \text{otherwise} \end{cases}.$$

Let the target space be \mathbb{R}^ℓ with $\ell = 2n$. The decision tree partition $\mathcal{L} = (\mathcal{L}_k)_{k \in \mathcal{K}}$ is defined by $\mathcal{L}_k := \{\mathbf{x} \in \mathbb{R}^\ell \mid \mathbf{s}_k \odot \mathbf{x} \geq 0\}$. Let $\boldsymbol{\theta} := (\mathbf{p}^U, \mathbf{p}^L)$, the associated affine transformation is defined by $T_{\boldsymbol{\theta}}\boldsymbol{\xi} + \boldsymbol{\tau}_{\boldsymbol{\theta}} := (\boldsymbol{\xi} - \mathbf{p}^U, \boldsymbol{\xi} - \mathbf{p}^L)$. For every $k \in \mathcal{K}$, the induced parametric partition becomes $\Xi_k^{\boldsymbol{\theta}} := \{\boldsymbol{\xi} \in \Xi \mid \mathbf{s}_k^U \odot (\boldsymbol{\xi} - \mathbf{p}^U) \geq 0, \mathbf{s}_k^L \odot (\boldsymbol{\xi} - \mathbf{p}^L) \geq 0\}$.

Figure 3 depicts how a predefined tree structure induces a parametric polyhedral partition: the splitting structure is fixed by the tree, while the split thresholds \mathbf{p}^U and \mathbf{p}^L remain as parameters. As $\boldsymbol{\xi}$ is routed through the tree according to comparisons with thresholds in \mathbf{p}^U and \mathbf{p}^L , each leaf $k \in \mathcal{K}$ corresponds to a region $\Xi_k^{\boldsymbol{\theta}} \subseteq \Xi$ defined by the resulting system of inequalities.

Figure 3: Illustration of the decision tree partition scheme.



Note. Given the domain $\Xi \subseteq \mathbb{R}^3$, a fixed decision tree routes each realization ξ via coordinate-wise threshold tests, producing a leaf $k \in \mathcal{K}$ with sign label $\mathbf{s}_k := (\mathbf{s}_k^U, \mathbf{s}_k^L)$. This label encodes the root-to-leaf decisions and yields a polyhedral description of each leaf as $\Xi_k^\theta = \{\xi \in \Xi \mid \mathbf{s}_k^U \odot (\xi - \mathbf{p}^U) \geq 0, \mathbf{s}_k^L \odot (\xi - \mathbf{p}^L) \geq 0\}$. Varying θ induces different partitions on Ξ .

3.2.3 General Tree Partition Scheme

The decision tree partition scheme is relatively restrictive for two reasons: (i) the target space must have dimension $\ell = 2n$, where n denotes the dimension of the uncertainty space; (ii) the splitting thresholds must be applied coordinate-wise. The following general tree partition scheme relaxes both conditions at the cost of more decision variables.

Definition 4 (General Tree Partition Scheme). Given a tree structure defined in a target space \mathbb{R}^ℓ for some even number ℓ with leaf set \mathcal{K} . Let \mathbf{s}_k indicate the splitting decision associated with the leaf $k \in \mathcal{K}$. The general tree partition $\mathcal{L} = (\mathcal{L}_k)_{k \in \mathcal{K}}$ is defined by $\mathcal{L}_k := \{\mathbf{x} \in \mathbb{R}^\ell \mid \mathbf{s}_k \odot \mathbf{x} \geq 0\}$. Let $\theta := (\mathbf{P}, \mathbf{p}^U, \mathbf{p}^L)$, the associated affine transformation is defined by $\mathbf{T}_\theta \xi + \boldsymbol{\tau}_\theta := (\mathbf{P}\xi - \mathbf{p}^U, \mathbf{P}\xi - \mathbf{p}^L)$. Then, for every $k \in \mathcal{K}$, the induced parametric partition is $\Xi_k^\theta := \{\xi \in \Xi \mid \mathbf{s}_k^U \odot (\mathbf{P}\xi - \mathbf{p}^U) \geq 0, \mathbf{s}_k^L \odot (\mathbf{P}\xi - \mathbf{p}^L) \geq 0\}$.

Under this partition scheme, the thresholding rules at each split of the given tree can be written as $\mathbf{s}_k^U \odot \mathbf{P}\xi \geq \mathbf{s}_k^U \odot \mathbf{p}^U$ for upper bounds and $\mathbf{s}_k^L \odot \mathbf{P}\xi \geq \mathbf{s}_k^L \odot \mathbf{p}^L$ for lower bounds. This formulation allows Ξ to be first affinely mapped into the target space before applying the thresholding rules encoded in \mathbf{p}^U and \mathbf{p}^L . A closely related construction was introduced in the machine learning literature through optimal classification trees [11].

3.3 Exact Solution Methods

In this subsection, we derive two exact reformulations of (5) under a given partition scheme: (i) a master-subproblem procedure that iteratively generates adversarial scenarios and enforces robustness over the resulting finite scenario set, and (ii) a piecewise dualization approach that

yields a single-level formulation by exploiting the explicit polyhedral description of each Ξ_k^θ . Both reformulations provide algorithmic foundations for the scalable method proposed in Section 4.

3.3.1 Master–Subproblem Method

To address the semi-infinite constraint system in (5) indexed by the scenarios from Ξ_k^θ , this method maintains a finite scenario set and solves a corresponding relaxation of Problem (5) as a master problem. Given a master solution $(\mathbf{y}, \boldsymbol{\theta})$, robustness is assessed by solving separation subproblems over each partition region to identify worst-case scenarios, which are then added back to the master problem for a new iteration. The procedure terminates once $(\mathbf{y}, \boldsymbol{\theta})$ is certified both feasible and optimal by all subproblems. The next theorem states the resulting reformulation and establishes its correctness.

Theorem 2. *Let I be the index set of scenarios in $\{\boldsymbol{\xi}_i\}_{i \in I} \subseteq \Xi$, define binary variables $\lambda_{k,i} \in \{0, 1\}$ to indicate whether $\boldsymbol{\xi}_i$ is assigned to region Ξ_k^θ . For sufficiently large constants M , M_j , and M' , Problem (5) can be reformulated as the following master and subproblems:*

$$z^{\text{mst}} := \min_{\substack{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}, z \in \mathbb{R} \\ \boldsymbol{\theta} \in \Theta, \boldsymbol{\lambda} \in \{0, 1\}^{\mathcal{K} \times I}}} z \quad (6a)$$

$$s.t. \quad \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi}_i \rangle + \mathbf{q}(\mathbf{y}_k) \leq z + M(1 - \lambda_{ki}), \quad \forall i \in I, \forall k \in \mathcal{K}, \quad (6b)$$

$$\langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi}_i \rangle + \mathbf{q}_j(\mathbf{y}_k) \leq M_j(1 - \lambda_{ki}), \quad \forall j \in J, \forall k \in \mathcal{K}, \forall i \in I, \quad (6c)$$

$$\mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi}_i + \boldsymbol{\tau}_\theta) - \mathbf{b}_k \leq M'(1 - \lambda_{ki})\mathbf{1}, \quad \forall i \in I, \forall k \in \mathcal{K}, \quad (6d)$$

$$\sum_{k \in \mathcal{K}} \lambda_{ki} = 1, \quad \forall i \in I. \quad (6e)$$

Given any solution $(\mathbf{y}, \boldsymbol{\theta})$ from (6), define the following subproblems for every $k \in \mathcal{K}$ and $j \in J$

$$z_k^{\text{opt}} := \max_{\boldsymbol{\xi} \in \Xi} \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}(\mathbf{y}_k) \quad (7a) \quad z_{kj}^{\text{fea}} := \max_{\boldsymbol{\xi} \in \Xi} \langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}_j(\mathbf{y}_k) \quad (7b)$$

$$s.t. \quad \mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_k, \quad s.t. \quad \mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_k,$$

The procedure halts when $z^{\text{mst}} = \max_{k \in \mathcal{K}} z_k^{\text{opt}}$ and $\max_{k \in \mathcal{K}, j \in J} z_{kj}^{\text{fea}} \leq 0$ under some incumbent $(\mathbf{y}, \boldsymbol{\theta})$.

Constraint (6e) ensures that each scenario $\boldsymbol{\xi}_i$ is assigned to exactly one part Ξ_k^θ . Constraint (6d) then enforces membership: whenever $\lambda_{ki} = 1$, we must have $\boldsymbol{\xi}_i \in \Xi_k^\theta$, while for $k' \neq k$ the corresponding constraints are made redundant via a sufficiently large constant $M' > 0$. For example, because the orthant partition is invariant under positive scaling of (\mathbf{P}, \mathbf{p}) , we can impose a normalization on (\mathbf{P}, \mathbf{p}) to ensure bounded big- M constants. Constraints (6b) and (6c) ensure that the associated constraints are active whenever $\lambda_{ki} = 1$. The constants M and M_j must be tailored to the specific problem so that these constraints become redundant when $\lambda_{ki} = 0$.

Although there are $K + K|J|$ subproblems, each is a relatively simple optimization problem and can be solved in parallel. The primary computational burden lies in the master problem: each newly generated scenario introduces $K(1 + |J| + \ell)$ additional constraints and K binary variables. This growth is driven in part by (6d), which simultaneously optimizes the polyhedral partition parameters $\boldsymbol{\theta}$ and the scenario-to-region assignments $\boldsymbol{\lambda}$. Later in Section 4, we will decouple these two tasks to obtain a more scalable solution framework.

3.3.2 Piecewise Dualization Method

Because (5) provides an explicit polyhedral description of each region Ξ_k^θ , we can dualize the inner maximization problems region by region. By strong duality, this yields dual representations of the worst-case objective value and each worst-case constraint over Ξ_k^θ . Aggregating these piecewise dual characterizations gives the following reformulation.

Theorem 3. *Given each part Ξ_k^θ nonempty, (5) can be equivalently reformulated as follows.*

$$\min_{\substack{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}, z \in \mathbb{R} \\ \boldsymbol{\theta} \in \Theta, \boldsymbol{\pi} \geq \mathbf{0}, \boldsymbol{\mu} \geq \mathbf{0}}} z \quad (8a)$$

$$s.t. \quad \mathbf{q}(\mathbf{y}_k) + \langle \mathbf{b}, \boldsymbol{\pi}_k \rangle + \langle \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta, \boldsymbol{\mu}_k \rangle \leq z, \quad \forall k \in \mathcal{K}, \quad (8b)$$

$$\mathbf{B}^\top \boldsymbol{\pi}_k + \mathbf{T}_\theta^\top \mathbf{B}_k^\top \boldsymbol{\mu}_k = \mathbf{Q}(\mathbf{y}_k), \quad \forall k \in \mathcal{K}, \quad (8c)$$

$$\mathbf{q}_j(\mathbf{y}_k) + \langle \mathbf{b}, \boldsymbol{\pi}_{kj} \rangle + \langle \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta, \boldsymbol{\mu}_{kj} \rangle \leq 0, \quad \forall k \in \mathcal{K}, j \in J, \quad (8d)$$

$$\mathbf{B}^\top \boldsymbol{\pi}_{kj} + \mathbf{T}_\theta^\top \mathbf{B}_k^\top \boldsymbol{\mu}_{kj} = \mathbf{Q}_j(\mathbf{y}_k), \quad \forall k \in \mathcal{K}, j \in J. \quad (8e)$$

If the optimal value of this reformulation is finite, then, by duality, the induced primal partition is well-defined, and the corresponding polyhedral partition can be recovered explicitly from the optimal parameter $\boldsymbol{\theta}^*$. This reformulation is bilinear and hence generally nonconvex, due to multiplications between the partition parameters $\boldsymbol{\theta}$ and the dual variables $\boldsymbol{\mu}$ in the constraints. While such bilinear programs can be solved by commercial solvers, this approach is typically not scalable. A key observation, however, is that if a high-quality initial partition parameter $\boldsymbol{\theta}_0$ can be computed efficiently, then alternating optimization over $(\mathbf{y}, z, \boldsymbol{\pi}, \boldsymbol{\mu})$ and $\boldsymbol{\theta}$ can refine $\boldsymbol{\theta}_0$ and converge to a locally optimal partition [30]. This is because all bilinear terms in (8) stem from products of $\boldsymbol{\theta}$ and the other decision variables. We develop this idea further in the next section to obtain a scalable finite adaptability algorithm.

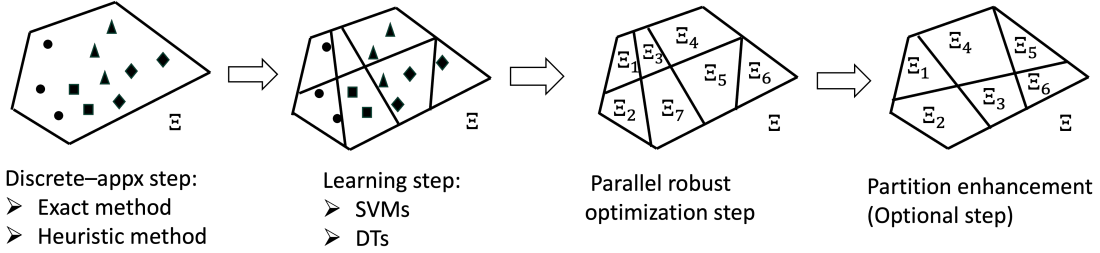
4 A Scalable Learning-Based Framework for Finite Adaptability

The exact solution methods developed in the previous section simultaneously optimize the recourse decisions and the partition parameters. While both reformulations are tractable in principle (i.e., cut generation for (6)–(7) and bilinear program for (8)), they become computationally expensive even at moderate scales.

To enable scalability, this section proposes a learning-based framework that decouples partition construction from decision optimization. As illustrated in Figure 4, the framework proceeds in three steps, with an optional enhancement step. We refer to this procedure as the *approximate-learn-parallel* (ALP) framework, summarized below.

- *Step 1: Discrete approximation.* Given a finite set of representative scenarios $\{\boldsymbol{\xi}_i\}_{i \in I}$ from Ξ , this step constructs or approximates an optimal K -partition over $\{\boldsymbol{\xi}_i\}_{i \in I}$. Upon completion, each scenario $\boldsymbol{\xi}_i$ is assigned a label $k \in \mathcal{K}$.
- *Step 2: Partition learning.* This step trains a classifier (e.g., an SVM or a DT) on the dataset $\{(\boldsymbol{\xi}_i, k_i)\}_{i \in I}$. We then extract the parameter $\boldsymbol{\theta}$ from the trained classifier, inducing an explicit polyhedral partition $\{\Xi_k^\theta\}_{k \in \mathcal{K}'}$. The resulting number of parts $|\mathcal{K}'|$ is often larger than K .

Figure 4: Four-step scalable approximate-learn-parallel (ALP) framework for finite adaptability.



Note. First, a finite scenario set is initialized and labeled using either the exact or heuristic method. Second, a classifier, such as an SVM or a decision tree, is trained to learn a partition rule from these labels. Third, robust optimization subproblems are solved independently over the resulting regions. Finally, the partition can be further enhanced using the dualization reformulation (8).

- *Step 3: Robust optimization (parallel).* This step solves, in parallel, a robust optimization problem over each learned region Ξ_k^θ , yielding region-wise solutions \mathbf{y}_k for every $k \in \mathcal{K}'$.
- *Step 4: Partition enhancement (optional).* Use θ from Step 2 as the initial partition parameter in the alternating optimization scheme for the bilinear program (8), yielding a policy associated with a locally improved partition.

Unlike standard uses of learning models for prediction, the trained model in Step 2 serves to extract the underlying partition. The resulting solution therefore preserves robustness. The details of each step are presented in the subsequent subsections.

4.1 Discrete Approximation

The goal of this step is to obtain an informative discrete partition that enables Step 2 to learn a high-quality partition over the entire Ξ . This requires (i) the samples $\{\xi_i\}_{i \in I}$ to be representative, and (ii) the resulting partition over $\{\xi_i\}_{i \in I}$ to reflect a good approximation of the underlying partition structure induced by the robust objective and constraints.

Regarding (i), the choice of representative samples depends on the problem class. In a robust setting with a budget uncertainty set $\Xi = \{\xi \in [0, 1]^n \mid \langle \mathbf{1}, \xi \rangle \leq B\}$ for some $B \in (0, n)$, if the objective function is convex in ξ , then it suffices to sample from the extreme points of Ξ , since for any fixed \mathbf{y} an optimal worst-case ξ exists at an extreme point. If the objective is further entrywise increasing, then the worst case lies on the dominating boundary $\{\xi \mid \langle \mathbf{1}, \xi \rangle = B\}$, and sampling can be restricted to the extreme points of this boundary. In contrast, in stochastic programming, the scenarios $\{\xi_i\}_{i \in I}$ should be sampled from an underlying distribution to represent the expectation.

Once the sample set $\{\xi_i\}_{i \in I}$ is generated, the corresponding discrete partition problem can be obtained from (6) by dropping the explicit partition constraint (6d). When (Q, Q_j, q, q_j) admit simple expressions in \mathbf{y}_k , this formulation can be solved using a standard MIP solver. However, since Step 1 is inherently approximative, solving this formulation to optimality is unnecessary. We therefore introduce a more efficient heuristic method.

4.1.1 Alternating Assignment–Optimization (AAO)

Let $f(\mathbf{y}_k, \boldsymbol{\xi}_i) := \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi}_i \rangle + \mathbf{q}(\mathbf{y}_k)$ and $g_j(\mathbf{y}_k, \boldsymbol{\xi}_i) := \langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi}_i \rangle + \mathbf{q}_j(\mathbf{y}_k)$. Let $\delta_{g(\mathbf{y}_k, \boldsymbol{\xi}_i) \leq 0}(\mathbf{y}_k, \boldsymbol{\xi}_i)$ denote the indicator function of any given constraint $g(\mathbf{y}_k, \boldsymbol{\xi}_i) \leq 0$, i.e., it equals $+\infty$ upon violation and 0 otherwise. Problem (6) without (6d) is equivalent to the following general assignment problem with a nonlinear cost function,

$$\min_{\boldsymbol{\lambda} \in \{0,1\}^{\mathcal{K} \times I}} \min_{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}} \max_{\substack{i \in I \\ k \in \mathcal{K}}} h(\mathbf{y}_k, \boldsymbol{\xi}_i; \lambda_{ki}) \quad (9a)$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} \lambda_{ki} = 1, \quad \forall i \in I, \quad (9b)$$

where $h(\mathbf{y}_k, \boldsymbol{\xi}_i; \lambda_{ki}) := f(\mathbf{y}_k, \boldsymbol{\xi}_i) - M(1 - \lambda_{ki}) + \sum_{j \in J} \delta_{\lambda_{ki} g_j(\mathbf{y}_k, \boldsymbol{\xi}_i) \leq 0}(\mathbf{y}_k, \boldsymbol{\xi}_i)$. Then, the proposed AAO method aims to alternate between (i) the assignment problem under fixed $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$ and (ii) K independent scenario-based robust optimization problems under fixed $\boldsymbol{\lambda}$. Both steps utilize parallel computation for efficient implementation, with details presented below.

- *Initialization:* Heuristically identify K initial solutions $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$, e.g., by solving K iterations of the associated optimization problem with solution elimination cuts, or through problem-specific algorithms such as K -shortest paths for the robust shortest path problem. Then, alternate between the following two steps until the objective value no longer decreases.
- *Assignment (parallel):* For each $\boldsymbol{\xi}_i$, evaluate $\{h(\mathbf{y}_k, \boldsymbol{\xi}_i; 1)\}_{k \in \mathcal{K}}$ in parallel and set $\lambda_{ki} = 1$ for the index k attaining the smallest value.
- *Optimization (parallel):* Let $I_k := \{i \in I \mid \lambda_{ki} = 1\}$. In parallel over $k \in \mathcal{K}$, solve the scenario-based robust optimization problem $\min_{\mathbf{y}_k \in \mathcal{Y}} \max_{i \in I_k} h(\mathbf{y}_k, \boldsymbol{\xi}_i; 1)$.

We repeat these two steps until the objective value converges. This finite convergence requires both steps to solve their respective subproblems to optimality. The optimization step clearly yields an optimal policy $(\mathbf{y}_k)_{k \in \mathcal{K}}$ for a fixed $\boldsymbol{\lambda}$. The following proposition shows that the assignment step likewise computes an optimal assignment $\boldsymbol{\lambda}$ for a fixed policy $(\mathbf{y}_k)_{k \in \mathcal{K}}$.

Proposition 3. *The assignment step produces a solution $\boldsymbol{\lambda}$ that is optimal for (9) under any fixed policy $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$.*

Since both steps solve their respective subproblems to optimality, the following proposition establishes finite convergence of the AAO algorithm.

Proposition 4. *The AAO algorithm terminates in finitely many iterations with a convergent objective value.*

4.2 Partition Learning

After the discrete approximation step, we obtain a labeled dataset of scenario-to-part assignments $\{(\boldsymbol{\xi}_i, k_i)\}_{i \in I}$. This step then trains a classifier to recover a polyhedral partition of Ξ . We focus on two learning models that align with the partition schemes in Section 3. First, a linear multi-class SVM (implemented via a one-vs-rest strategy) learns a collection of separating hyperplanes, which can be extracted to construct an orthant-based partition scheme (Definition 2). Second, a decision-tree classifier induces a tree-based partition scheme (Definition 3) by extracting the branching rules of the learned tree. Other learners that induce polyhedral partitions (e.g., XGBoost) can also be used in this step, provided that the associated partition scheme is designed accordingly.

4.3 Parallel Robust Optimization

Given the learned partition $\{\Xi_k^\theta\}_{k \in \mathcal{K}'}$ from Step 2, we compute the final finite-adaptable decisions by solving one robust optimization problem per region. Since the partition parameter θ is fixed, these subproblems are independent and can be solved in parallel. For each $k \in \mathcal{K}'$, the associated robust optimization is

$$\min_{\mathbf{y} \in \mathcal{Y}} \max_{\xi \in \Xi_k^\theta} \langle \mathbf{Q}(\mathbf{y}), \xi \rangle + q(\mathbf{y}) \quad (10a)$$

$$\text{s.t. } \langle \mathbf{Q}_j(\mathbf{y}), \xi \rangle + q_j(\mathbf{y}) \leq 0, \quad \forall \xi \in \Xi_k^\theta, j \in J, \quad (10b)$$

which can be solved using standard techniques in robust optimization [6]. Let \mathbf{y}_k^* denote the optimal solution for the k th subproblem. Collecting $\{\mathbf{y}_k^*\}_{k \in \mathcal{K}'}$ yields a finite-adaptable policy that selects \mathbf{y}_k^* whenever the realized uncertainty ξ falls into region Ξ_k^θ .

4.4 Partition Enhancement

This optional step aims to obtain improved policies at the expense of additional computational effort. Starting from the partition parameter θ obtained in Step 2, we initialize the bilinear formulation (8) and then alternate between two updates: (i) fixing the partition parameter θ and solving (8) for the remaining variables $(\mathbf{y}, z, \pi, \mu)$; and (ii) fixing $(\mathbf{y}, z, \pi, \mu)$ and updating the partition parameter θ by solving the corresponding linearized subproblem induced by (8). In each update, the bilinear terms reduce to linear ones since one block of variables is held fixed, allowing each subproblem to be solved efficiently using standard optimization solvers. Repeating this procedure yields successive refinements of the induced partition and improved objective values (up to local optimality).

Aligned with other partition-based methods [12, 40], the ALP framework does not provide an approximation guarantee on the partition quality for finite K . Theorem 1 nevertheless establishes that polyhedral finite-adaptable policies can approximate the fully adjustable benchmark arbitrarily closely as K grows, motivating larger K as a practical strategy. The optional Step 4 further refines the learned partition via alternating optimization, converging to a locally optimal partition.

5 Extension to Stochastic Programming

Since the parametric polyhedral partition is fundamentally a support partitioning technique rather than one tied to a specific risk measure \mathcal{R}_Ξ , the solution methods we develop readily extend to other risk measures. This section illustrates this flexibility through K -adaptable stochastic programming.

Under a given partition scheme $(\mathcal{L}, \mathcal{T}, \tau)$, let $h(\mathbf{y}, \xi)$ denote the objective function, where all constraints are incorporated as indicator functions taking the value zero if satisfied and $+\infty$ otherwise. Then, the associated K -adaptable stochastic program can be expressed as follows

$$\min_{\substack{\mathbf{y}:=(\mathbf{y}_k)_{k \in \mathcal{K}} \in \mathcal{Y}^{\mathcal{K}} \\ \theta \in \Theta}} \sum_{k \in \mathcal{K}} q_k^\theta \mathbb{E} \left[h(\mathbf{y}_k, \xi) \mid \xi \in \Xi_k^\theta \right],$$

where q_k^θ is the probability measure of region Ξ_k^θ under the given probability measure over Ξ . Using the standard sample average approximation (SAA), this problem can be reformulated as follows

under a set of i.i.d. samples $\{\boldsymbol{\xi}_i\}_{i \in I}$.

$$\min_{\substack{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}, \mathbf{z} \in \mathbb{R}^{|I|} \\ \boldsymbol{\theta} \in \Theta, \boldsymbol{\lambda} \in \{0,1\}^{\mathcal{K} \times I}}} \frac{1}{|I|} \sum_{i \in I} z_i \quad (11a)$$

$$\text{s.t.} \quad h(\mathbf{y}_k, \boldsymbol{\xi}_i) \leq z_i + M(1 - \lambda_{ki}), \quad \forall i \in I, \forall k \in \mathcal{K}, \quad (11b)$$

$$\mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi}_i + \boldsymbol{\tau}_\theta) - \mathbf{b}_k \leq M'(1 - \lambda_{ki})\mathbf{1}, \quad \forall i \in I, \forall k \in \mathcal{K}, \quad (11c)$$

$$\sum_{k \in \mathcal{K}} \lambda_{ki} = 1, \quad \forall i \in I. \quad (11d)$$

Similarly, the ALP framework can be applied to this reformulation to improve scalability.

5.1 ALP Framework for Adaptable SP

We adapt the approximate-learn-parallel (ALP) framework in Section 4 to SP as follows. In Step 1, given the i.i.d. samples $\{\boldsymbol{\xi}_i\}_{i \in I}$, we solve (11) without (11c), either exactly or via the AAO method using the same initialization, but with a modified assignment–optimization loop.

- *Assignment (parallel)*: For each $\boldsymbol{\xi}_i$, evaluate $\{h(\mathbf{y}_k, \boldsymbol{\xi}_i)\}_{k \in \mathcal{K}}$ in parallel and set $\lambda_{ki} = 1$ for the index k attaining the smallest value.
- *Optimization (parallel)*: Let $I_k := \{i \in I \mid \lambda_{ki} = 1\}$. In parallel over $k \in \mathcal{K}$, solve the scenario-based stochastic optimization problem $\min_{\mathbf{y}_k \in \mathcal{Y}} \sum_{i \in I_k} h(\mathbf{y}_k, \boldsymbol{\xi}_i)$.

Similar to the robust counterpart, this modified assignment–optimization loop yields an optimal partition $\boldsymbol{\lambda}$ for a fixed response $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$ and, conversely, yields optimal responses for a fixed partition $\boldsymbol{\lambda}$. Once a local optimum is reached by alternating between these two steps, we obtain a labeled dataset $\{(\boldsymbol{\xi}_i, k_i)\}_{i \in I}$. Step 2 then applies the same learning procedure to extract the partition parameter $\boldsymbol{\theta}$. In Step 3, fixing $\boldsymbol{\theta}$, we solve the region-wise stochastic program $\min_{\mathbf{y} \in \mathcal{Y}} \mathbb{E}[h(\mathbf{y}, \boldsymbol{\xi}) \mid \boldsymbol{\xi} \in \Xi_k^\theta]$ in parallel for each $k \in \mathcal{K}'$, thereby obtaining the final finite-adaptable SP policy.

Remark 3. The SAA reformulation (11) provides a valid approximation when the problem admits complete recourse. In the absence of this property, sampling alone cannot guarantee the feasibility of the optimized policy $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$. In this case, the explicit polyhedral partition learned in Step 2 of the ALP framework enables a hybrid formulation in Step 3, where the objective is approximated via i.i.d. samples while the constraints are enforced as robust optimization conditions. This yields robust policies with optimized expected performance, enabled by the explicit partition from Step 2.

6 Numerical Experiments

This section reports the computational performance of the ALP framework on three classical testbeds for robust finite adaptability from the literature [12, 33, 45]: (i) the shortest path problem with uncertain arc lengths (objective uncertainty), (ii) the capital budgeting problem with uncertain costs and profits (objective and single-constraint uncertainty), and (iii) the project management problem with uncertain task durations (size-dependent constraint uncertainty). Section 6.4 then applies the stochastic extension of the ALP framework to the capital budgeting problem. Consistent with the complementarity noted in Section 1.2, the experimental results reveal a clear pattern: the proposed approach yields stronger objective values when uncertainty affects constraints, while

algebraic approaches hold an advantage under objective-only uncertainty, where exact worst-case evaluation largely dictates solution quality.

All experiments were conducted on a 2021 MacBook Pro equipped with an Apple M1 Max chip and 64 GB of memory, running macOS Sonoma (version 14.6.1). Implementations were written in Python 3.13, with mixed integer programs solved using Gurobi 12.0. The constraint feasibility tolerance was set to $\epsilon = 10^{-3}$ for accepting incumbent solutions; all other solver parameters were kept at their default values. We evaluate four variants of the ALP framework, distinguished by the algorithms employed in the first two steps: SVM–E: exact MIP in Step 1, SVM classifier in Step 2; DT–E: exact MIP in Step 1, decision tree classifier in Step 2; SVM–H: AAO heuristic in Step 1, SVM classifier in Step 2; DT–H: AAO heuristic in Step 1, decision tree classifier in Step 2. For all these variants, Step 3 solves the region-wise robust (or stochastic) subproblems over the learned partition $\{\Xi_k^\theta\}_{k \in \mathcal{K}'}$ in parallel, yielding solutions $\{\mathbf{y}_k\}_{k \in \mathcal{K}'}$.

For each instance, we test multiple values of $K \in \{2, 3, 4, 6, 8, 10\}$ and analyze the associated performance metrics. Specifically, we record the performance improvement (%), defined explicitly for each problem class, along with both the total parallel and sequential runtimes. In the parallel setting, the runtime of each parallel step (i.e., Step 1 of SVM–H and DT–H, Step 3 in all variants) is measured as the maximum completion time across all subroutines; in the sequential setting, it is measured as their sum. A time limit of 1200 seconds is imposed on Step 1 of all ALP variants. For the robust K -adaptability problems, we compare these variants against the baseline heuristic developed in [45], which, to the best of our knowledge, is among the most scalable algorithms currently available for general robust K -adaptability problems. We applied their publicly available code to our generated instances with a time limit of 1800 seconds per value of K . Since this baseline algorithm solves a sequence of robust optimization problems with K incremented by one at each iteration, its total runtime can exceed the per- K limit of 1800 seconds.

We also evaluate the partition enhancement step (Step 4 of ALP) on each instance, reporting the additional improvement and the corresponding runtime. A five-minute time limit is imposed on each alternating optimization subroutine.

6.1 Shortest Paths

This subsection studies the shortest path problem under the same setup as in Subramanyam et al. [45]. We analyze the results from three aspects: (i) scalability, assessed through performance improvement and runtime compared with the baseline algorithm; (ii) additional improvement enabled by the partition enhancement step (Step 4 of ALP); and (iii) the sensitivity of performance improvement and runtime to K and the number of scenarios used in Step 1 of ALP.

Let $G = (V, A)$ be a directed graph with node set V , arc set A , a source $s \in V$, and a terminal $t \in V$. The nominal arc lengths are denoted by $\mathbf{d}_0 \in \mathbb{R}_+^{|A|}$, and the realized lengths under a deviation scenario $\boldsymbol{\xi}$ are given by $\mathbf{d}(\boldsymbol{\xi}) = (1 + \frac{1}{2}\boldsymbol{\xi}) \odot \mathbf{d}_0$, where $\boldsymbol{\xi}$ belongs to the budgeted uncertainty set $\Xi = \{\boldsymbol{\xi} \in [0, 1]^{|A|} \mid \langle \mathbf{1}, \boldsymbol{\xi} \rangle \leq B\}$ for a fixed budget $B > 0$. The decision maker selects a menu of candidate s – t paths *here-and-now*, each represented by an incidence vector from the solution space

$$\mathcal{Y} = \left\{ \mathbf{y} \in \{0, 1\}^{|A|} \mid \sum_{(i,j) \in A} y_{ij} - \sum_{(j,i) \in A} y_{ji} \geq \mathbb{I}[i = s] - \mathbb{I}[i = t], \forall i \in V \right\},$$

where $\mathbb{I}[\cdot]$ returns 1 if the condition is satisfied and 0 otherwise. After the uncertain arc lengths are realized, the least costly path among the candidates is implemented. This gives rise to the

classic K -adaptability formulation $\min_{\mathbf{y} \in \mathcal{Y}^K} \max_{\boldsymbol{\xi} \in \Xi} \min_{k \in \mathcal{K}} \langle \mathbf{d}(\boldsymbol{\xi}), \mathbf{y}_k \rangle$. Since the partition learned in Step 2 of ALP may produce more regions than the initial K , the resulting policy can exploit finer uncertainty partitions than a standard K -adaptability solution.

We consider graph sizes $N \in \{20, 50, 60, \dots, 100\}$ and generate 10 random directed graphs for each N as follows. Node coordinates are sampled independently and uniformly from $[0, 10]^2$, and the nominal weight of each arc $(i, j) \in A$ is set to the Euclidean distance between its endpoints. The source s and terminal t are chosen as the pair of nodes with the maximum Euclidean distance. Starting from the complete directed graph on V , we construct the arc set A by removing the $\lfloor 0.7(N^2 - N) \rfloor$ arcs with the largest nominal weights while preserving s - t connectivity. The uncertainty budget is fixed at $B = 3$ for all instances.

6.1.1 Scalability Comparison.

Table 2 summarizes the computational performance of the ALP framework (from Step 1 to Step 3, without Step 4) on the shortest path instances. For each graph size, we run every variant with $K \in \{2, 3, 4, 6, 8, 10\}$ on ten randomly generated graphs and report the results associated with the K that yields the best average objective value z^* . The column *Improvement* is computed as $(z_{\text{rob}}^* - z^*) / (z_{\text{rob}}^* - z_{\text{adj}}^*) \times 100$, where z_{rob}^* and z_{adj}^* denote the optimal values of the static robust optimization and the fully adjustable policy, respectively; an improvement of 100% indicates that the fully adjustable value is attained. The column *Runtime* reports the average total wall-clock time under full parallelization. The corresponding sequential runtimes, which assume all steps are solved in sequence without parallelization, are reported in Table 9 in Appendix B. For most instances, these sequential runtimes remain within approximately 120% of the parallel counterparts.

Across all instance sizes, the baseline algorithm achieves the highest improvements, exceeding 95% in every case, likely owing to its iterative scheme being particularly well suited to problems with objective-only uncertainty. Among the ALP variants, SVM-H performs best, with improvements ranging from 15% to 29%.

In terms of runtime, the ALP variants are considerably more efficient than the baseline, with the largest instances solved within 300 seconds, whereas the baseline runtime grows more rapidly with instance size. Among the ALP variants, the heuristic versions (SVM-H and DT-H) are generally

Table 2 Performance comparison of ALP variants and the baseline algorithm on the shortest path problem.

Nodes (#)	Edges (#)	Improvement (%)					Runtime (seconds)				
		SVM-E	DT-E	SVM-H	DT-H	Baseline	SVM-E	DT-E	SVM-H	DT-H	Baseline
20	114	22.98	10.13	28.87	17.82	100.00	30.59	30.64	5.25	5.24	0.05
50	735	20.17	10.65	19.99	9.96	99.15	157.03	26.05	32.75	32.45	11.44
60	1062	19.98	8.28	17.70	9.93	98.41	152.45	35.75	97.80	98.15	180.14
70	1449	18.48	9.81	19.35	11.13	97.88	46.01	46.18	140.23	134.03	521.63
80	1896	15.91	5.95	19.01	9.42	97.40	61.28	61.30	69.69	74.21	1336.45
90	2403	12.65	4.20	17.02	5.07	96.93	227.56	227.30	221.15	222.23	1693.11
100	2970	12.51	2.09	15.41	4.03	96.45	253.17	173.17	108.07	108.81	2377.88

Notes. Improvement is computed as $(z_{\text{rob}}^* - z^*) / (z_{\text{rob}}^* - z_{\text{adj}}^*) \times 100$, where z^* , z_{rob}^* , and z_{adj}^* denote the optimal values of the tested algorithm, static robust optimization, and fully adjustable optimization, respectively. Step 1 uses 1500 scenarios for the exact MIP and 8000 for the AAO heuristic. For ALP, parallelized steps are timed by the maximum completion time across parallel tasks; corresponding sequential runtimes are reported in Appendix B.

Table 3 Effect of partition enhancement (Step 4) on the shortest path problem.

Nodes (#)	Edges (#)	Additional Improvement (%)				Additional Time (seconds)			
		SVM-E	DT-E	SVM-H	DT-H	SVM-E	DT-E	SVM-H	DT-H
20	114	+28.57	+40.87	+18.91	+29.92	+2.04	+11.64	+2.70	+5.94
50	735	+25.10	+27.50	+20.38	+25.83	+54.18	+66.36	+44.80	+32.86
60	1062	+16.78	+27.84	+17.49	+20.57	+106.85	+97.93	+156.26	+65.54
70	1449	+21.77	+27.55	+12.01	+22.31	+282.40	+263.85	+345.23	+157.70
80	1896	+21.08	+22.80	+18.28	+25.53	+544.79	+147.42	+1312.33	+215.77
90	2403	+22.76	+24.80	+18.19	+25.03	+1530.08	+543.55	+868.60	+352.83
100	2970	+12.06	+27.51	+14.20	+18.44	+3067.45	+823.65	+1273.90	+626.02

Notes. Additional improvement and runtime from applying alternating optimization to the bilinear program (8), warm-started with the Step 2 partition, for each ALP variant. For example, at (20, 114), SVM-E attains a total improvement of $22.98 + 28.57 = 51.55$ in $30.59 + 2.04 = 32.63$ seconds.

faster than their exact counterparts (SVM-E and DT-E), particularly on larger instances. These results indicate that for robust shortest path problems, ALP provides a computationally efficient alternative that scales to larger networks, though it attains lower improvement than the baseline.

6.1.2 Effect of Partition Enhancement.

Table 3 reports the effect of the partition enhancement step (Step 4), which warm-starts the alternating optimization of the bilinear program (8) from the partition parameter θ obtained in Step 2. For each instance size, we report the additional improvement in percentage points and the corresponding additional runtime in seconds. For example, at the configuration (20, 114), SVM-E achieves $22.98 + 28.57 = 51.55$ total performance improvement with total runtime of $30.59 + 2.04 = 32.63$ seconds.

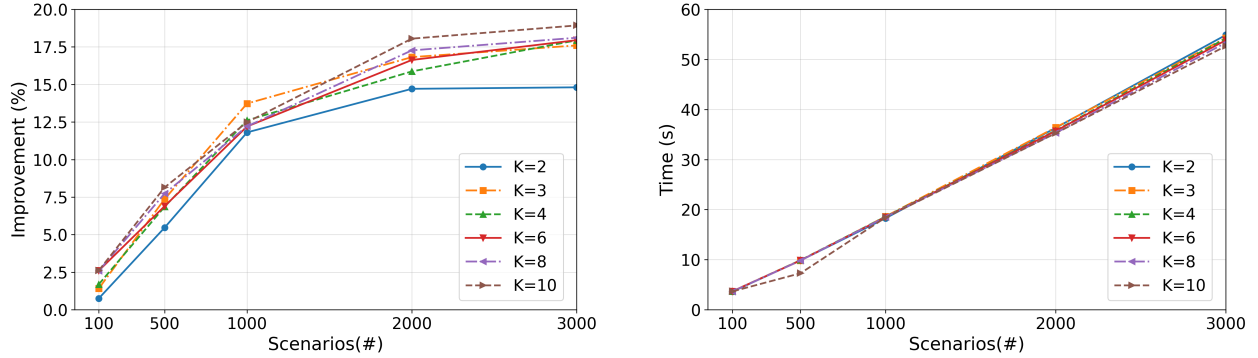
Across all graph sizes, Step 4 consistently improves the policies produced by ALP, yielding additional improvements ranging from 12 to 40 percentage points for every variant. Among all variants, DT-E achieves the largest incremental gains, potentially because its partition after Step 2 leaves more room for refinement. In terms of runtime, DT-based partitions lead to a more efficient Step 4, as the corresponding bilinear formulation (8) involves fewer partition parameters and is therefore smaller than its SVM-based counterpart. These results indicate that Step 4 can effectively refine the learned partition and improve solution quality at the cost of additional computational time.

6.1.3 Sensitivity to K and Scenario Count.

Figure 5 illustrates the sensitivity of ALP to the policy size K and the number of scenarios used in Step 1. We use SVM-H for this analysis, as it offers the best trade-off between improvement and efficiency among the four variants. The left plot reports the improvement and the right plot the runtime, both averaged over all tested graph sizes.

The improvement curves in Figure 5 indicate that increasing K generally leads to higher improvement, with $K = 2$ serving as a lower bound across all scenario counts and $K = 10$ achieving the best performance in most cases. Moreover, improvement increases with the number of scenarios, suggesting that a large sample size enables the learning of a better partition. The runtime curves,

Figure 5: Sensitivity to K and scenario count on the shortest path problem.



Note. Experiment results of SVM-H are used for demonstration. The associated improvement percentage and runtime are averaged over all graph sizes.

on the other hand, show that computation time grows primarily with the number of scenarios and depends only weakly on K , as evidenced by the closely clustered lines. In summary, ALP delivers stronger performance as both K and the scenario count increase, while computational cost scales steadily with the scenario count.

6.2 Capital Budgeting

Following the problem setup as in Subramanyam et al. [45], this subsection examines the capital budgeting problem. A company allocates a fixed budget B among N projects whose cost vector $\mathbf{c}(\boldsymbol{\xi})$ and profit vector $\mathbf{r}(\boldsymbol{\xi})$ depend on a latent factor $\boldsymbol{\xi} \in \Xi := [-1, 1]^4$ via $\mathbf{c}(\boldsymbol{\xi}) = (\mathbf{1} + \frac{1}{2}\boldsymbol{\Phi}\boldsymbol{\xi}) \odot \mathbf{c}_0$ and $\mathbf{r}(\boldsymbol{\xi}) = (\mathbf{1} + \frac{1}{2}\boldsymbol{\Psi}\boldsymbol{\xi}) \odot \mathbf{r}_0$ for given nominal vector \mathbf{c}_0 and \mathbf{r}_0 , and factor-loading matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$. A discount factor $\rho = 0.8$ captures the fraction of potential profit realized upon investment. The finite-adaptable policy then optimizes over K candidate investment plans, selecting the most suitable one after observing $\boldsymbol{\xi}$, defined as $\max_{\mathbf{y} \in \mathcal{Y}} \min_{\boldsymbol{\xi} \in \Xi} \max_{k \in \mathcal{K}} \{\rho \langle \mathbf{r}(\boldsymbol{\xi}), \mathbf{y}_k \rangle \mid \langle \mathbf{c}(\boldsymbol{\xi}), \mathbf{y}_k \rangle \leq B\}$, where $\mathcal{Y} = \{0, 1\}^N$ and each $\mathbf{y}_k \in \mathcal{Y}$ indicates a project selection. For each problem size $N \in \{5, 30, 50, 100, 1000, 3000, 5000\}$, we generate 10 random instances. Nominal costs \mathbf{c}_0 are drawn uniformly from $[0, 10]^N$, with $\mathbf{r}_0 = \mathbf{c}_0/5$. Each row of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ is sampled uniformly from the unit simplex in \mathbb{R}^4 , and the budget is set to $B = \frac{1}{2}\langle \mathbf{c}_0, \mathbf{1} \rangle$.

6.2.1 Scalability Comparison.

Table 4 summarizes the computational performance of the proposed ALP framework (from Step 1 to Step 3, without Step 4) on the capital budgeting instances. For each problem size N , we run every variant with $K \in \{2, 3, 4, 6, 8, 10\}$ on ten randomly generated instances and report the results associated with the K that yields the best average objective value z^* . Improvement is computed as $(z^* - z_{\text{rob}}^*)/z_{\text{rob}}^* \times 100$, differing from Table 2 because the fully adjustable problem becomes difficult to solve when \mathcal{Y} is discrete. Note that improvement can exceed 100% under this metric since it is a maximization problem. Runtime reports the average total wall-clock time under full parallelization. The corresponding sequential runtimes, reported in Table 10 in Appendix B, remain within roughly

Table 4 Performance comparison of ALP variants and the baseline algorithm on the capital budgeting problem.

Projects (#)	Improvement (%)					Runtime (seconds)				
	SVM-E	DT-E	SVM-H	DT-H	Baseline	SVM-E	DT-E	SVM-H	DT-H	Baseline
5	82.19	48.53	41.45	41.54	127.02	4.29	16.78	0.21	0.14	0.15
30	116.47	60.41	83.52	59.87	102.00	1240.56	1203.27	0.55	0.38	1912.99
50	118.30	58.99	85.06	63.68	82.46	1220.19	1206.09	20.96	0.58	8696.41
100	118.16	64.77	83.27	62.59	–	1204.71	1213.52	33.74	0.94	–
1000	118.38	63.81	76.43	46.07	–	1260.47	1253.39	5.97	5.88	–
3000	119.17	59.72	65.12	48.84	–	1304.91	1360.52	18.69	18.07	–
5000	117.49	60.91	74.19	49.40	–	1387.28	1471.68	30.12	30.07	–

Notes. Improvement is computed as $(z^* - z_{\text{rob}}^*)/z_{\text{rob}}^* \times 100$, where z^* and z_{rob}^* denote the optimal values of the tested algorithm and static robust optimization, respectively. Step 1 uses 500 scenarios for both the exact MIP and the AAO heuristic. For ALP, parallelized steps are timed by the maximum completion time across parallel tasks; sequential runtimes are reported in Appendix B.

200% of their parallel counterparts across most instances.

Unlike the shortest path case, ALP attains both higher improvement and faster computation across all instances except the smallest ($N = 5$). Even at $N = 50$, the baseline achieves 82.46% improvement in over two hours, whereas SVM-H reaches 85.06% in under 25 seconds. In terms of improvement, SVM-based methods outperform their DT counterparts, and exact MIP variants outperform their AAO heuristic counterparts. The heuristic variants, however, exhibit superior scalability, solving all instances in under one minute. Among all four ALP variants, SVM-H strikes the best balance between solution quality and computational efficiency.

6.2.2 Effect of Partition Enhancement.

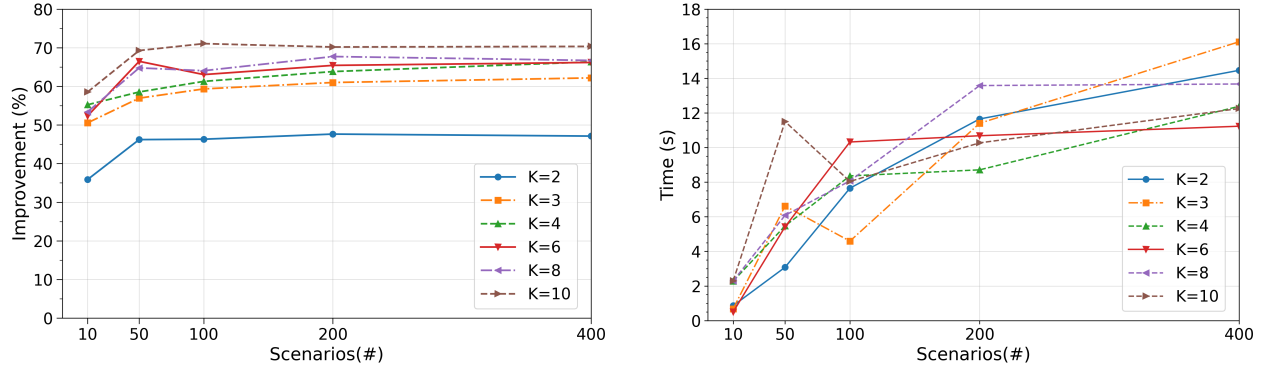
Table 5 reports the effect of the optional partition enhancement step (Step 4) in terms of additional improvement and the corresponding additional runtime. This partition enhancement step yields

Table 5 Effect of partition enhancement (Step 4) on the capital budgeting problem.

Projects (#)	Additional Improvement (%)				Additional Time (seconds)			
	SVM-E	DT-E	SVM-H	DT-H	SVM-E	DT-E	SVM-H	DT-H
5	+0.91	+34.90	+0.00	+32.00	+0.16	+0.22	+8.37	+0.31
30	+1.00	+9.45	+0.32	+7.41	+2.51	+0.96	+1.51	+1.36
50	+0.93	+7.23	+0.22	+3.36	+15.72	+0.56	+2.24	+4.17
100	+0.45	+8.16	+0.01	+8.64	+36.95	+12.40	+433.41	+10.59
1000	+0.01	+2.42	+2.03	+14.10	+21.59	+16.64	+634.60	+7.39
3000	+0.15	+6.07	+0.00	+11.37	+62.07	+36.75	+547.46	+ 26.03
5000	+0.03	+20.09	+0.00	+23.07	+91.48	+96.78	+787.11	+ 50.10

Notes. Additional improvement and runtime obtained by applying alternating optimization to the bilinear program (8), warm-started with the Step 2 partition, for each ALP variant.

Figure 6: Sensitivity to K and scenario count on the capital budgeting problem.



Note. Experiment results of SVM-H are used for demonstration. The associated improvement percentage and runtime are averaged over all instance sizes.

positive improvements across all tested instances, with the same trend of DT-based variants obtaining notably larger gains than their SVM counterparts. All instances are solved within minutes, and DT-H exhibits better efficiency on the larger instances. Overall, Step 4 provides a practical post-processing tool for improving finite-adaptable policies at modest additional computational cost, particularly when paired with DT-based variants.

6.2.3 Sensitivity to K and Scenario Count.

Figure 6 illustrates the sensitivity of ALP to the policy size K and the scenario count in Step 1 for the capital budgeting problem. As in the shortest path experiment, SVM-H is used for demonstration. The left and right plots report the achieved improvement and the corresponding runtime, respectively, both averaged over all instances.

As before, improvement increases with both K and the scenario count, with $K = 2$ and $K = 10$ providing lower and upper bounds, respectively, across all scenario counts. Unlike the shortest path case, however, the gains are more pronounced at smaller scenario counts: improvement rises rapidly as the number of scenarios increases from 10 to around 100, then plateaus, suggesting that additional scenarios yield only marginal returns. This behavior likely stems from the uncertainty dimension being fixed at four regardless of the problem size. Runtime grows primarily with the scenario count, while the dependence on K remains mild.

6.3 Project Management

This subsection evaluates the algorithms on the project management problem [45, 48]. Let $G = (V, A)$ be a directed acyclic graph parameterized by $m \in \mathbb{Z}_+$, with node set $V := [3m + 1]$ and arc set $A := \{(3l + 1, 3l + p), (3l + p, 3l + 4) \mid l = 0, \dots, m, p = 2, 3\}$. Nodes represent project tasks and arcs represent precedence relations. Each task $i \in V$ has uncertain duration $d_i(\xi)$ given by $d_{3l+2}(\xi) = \xi_{l+1}$, $d_{3l+3}(\xi) = 1 - \xi_{l+1}$ for $l = 0, \dots, m - 1$, and $d_{3l+1}(\xi) = 0$ for $l = 0, \dots, m$, with uncertainty set $\Xi = \{\xi \in \mathbb{R}_+^m \mid \|\xi - \mathbf{1}/2\|_1 \leq 1/2\}$. The sink node $N := 3m + 1$ represents project completion. A schedule $\mathbf{y} \in \mathcal{Y} = \mathbb{R}_+^N$ assigns start time y_i to each task i . The K -adaptability

Table 6 Performance comparison of ALP variants and the baseline algorithm on the project management problem.

m	Projects (#)	Improvement (%)					Runtime (seconds)				
		SVM-E	DT-E	SVM-H	DT-H	Baseline	SVM-E	DT-E	SVM-H	DT-H	Baseline
3	10	0.00	18.03	0.07	16.23	0.00	1211.21	1080.53	0.93	0.91	0.00
8	25	1.21	8.77	8.19	11.93	0.00	1212.12	467.18	1.89	1.74	0.13
10	31	0.81	7.02	7.14	9.77	0.00	1236.96	1207.55	2.28	2.07	0.12
20	61	2.93	1.61	4.22	4.87	0.00	1273.80	1215.18	4.22	3.86	0.32
30	91	3.25	0.73	2.99	3.21	0.00	1310.20	1310.09	6.09	5.71	1.32
40	121	3.20	0.82	2.31	2.35	0.00	1348.77	1318.73	8.11	7.07	1.67
50	151	3.68	0.61	1.88	1.89	0.00	1312.53	1256.97	10.32	8.98	1.39

Notes. Improvement is computed as $(z_{\text{rob}}^* - z^*)/z_{\text{rob}}^* \times 100$, where z^* and z_{rob}^* denote the optimal values of the tested algorithm and static robust optimization. Step 1 uses 2000 scenarios for all variants. For ALP, parallelized steps are timed by the maximum completion time across parallel tasks; sequential runtimes are reported in Appendix B. Baseline runtimes are not bolded as no improvement is achieved.

problem minimizes the worst-case project makespan:

$$\min_{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}} \max_{\xi \in \Xi} \min_{k \in \mathcal{K}} \{(\mathbf{y}_k)_N \mid (\mathbf{y}_k)_j - (\mathbf{y}_k)_i \geq d_i(\xi), \forall (i, j) \in A\},$$

We consider network sizes with $m \in \{3, 8, 10, 20, 30, 40, 50\}$. As shown in Wiesemann et al. [48, Example 2.2], the optimal value of the static robust problem is m . Unlike the previous problems, the number of constraints with right-hand-side uncertainty grows with the problem size.

6.3.1 Scalability Comparison.

Table 6 reports the scalability results. The corresponding sequential runtimes are reported in Table 11 in Appendix B. In terms of performance improvement, all ALP variants outperform the baseline across all instances. The baseline yields no improvement, which may be attributed to the presence of multiple uncertain constraints that make partition refinement more difficult. Among the four variants, DT-based methods perform best on small and medium instances, while SVM-E achieves the greatest improvement on large instances. In terms of runtime, the heuristic variants complete within seconds, while their exact counterparts often reach the time limit at Step 1 of ALP.

6.3.2 Effect of Partition Enhancement.

Table 7 reports the effect of the optional partition refinement step (Step 4) on the project management instances. Step 4 provides additional improvement mainly for the DT-based variants, while the SVM-based variants show little or no gain. This difference arises because SVM-based partitions typically contain more regions, resulting in larger formulations that cannot be optimized within the five-minute time limit. In contrast, DT-based partitions keep the problem size smaller, allowing Step 4 to remain computationally effective, with DT-E obtaining the largest gains for most instance sizes. DT-based variants also require less additional runtime than their SVM counterparts, again due to the smaller problem sizes.

Table 7 Effect of partition enhancement (Step 4) on the project management problem.

m	Projects (#)	Additional Improvement (%)				Additional Time (seconds)			
		SVM-E	DT-E	SVM-H	DT-H	SVM-E	DT-E	SVM-H	DT-H
3	10	+0.32	+52.05	+0.00	+25.69	+0.53	+0.80	+8.37	+0.90
8	25	+0.22	+6.99	+0.00	+6.30	+3.07	+2.02	+76.27	+4.82
10	31	+0.00	+8.53	+0.00	+2.16	+110.41	+5.18	+91.14	+7.15
20	61	+0.00	+4.26	+0.00	+0.00	+283.54	+22.10	+319.70	+2.06
30	91	+0.00	+4.25	+0.00	+1.15	+237.02	+36.82	+748.01	+7.44
40	121	+0.00	+3.29	+0.00	+1.86	+73.12	+48.96	+296.60	+15.48
50	151	+0.00	+1.56	+0.00	+2.32	+93.62	+90.19	+509.51	+17.33

Notes. Additional improvement and runtime obtained by applying alternating optimization to the bilinear program (8), warm-started with the Step 2 partition, for each ALP variant.

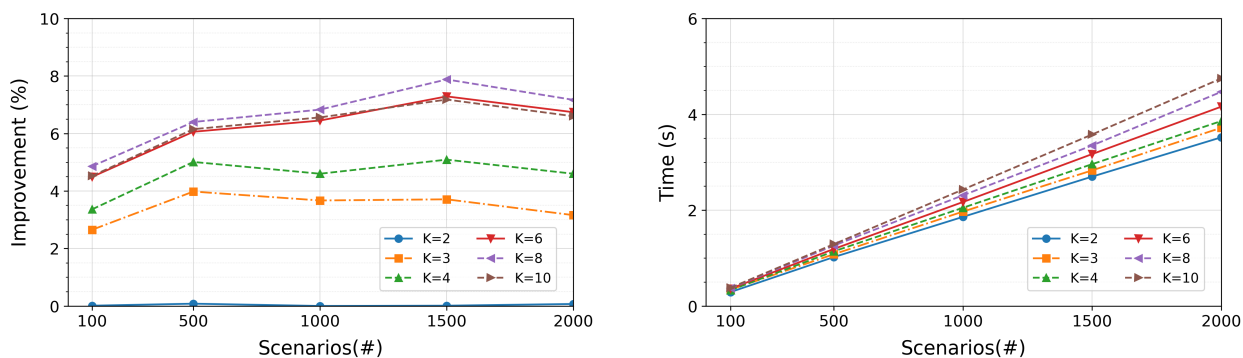
6.3.3 Sensitivity to K and Scenario Count.

Figure 7 illustrates the associated sensitivity analysis using DT-H results. The left and right plots report the achieved improvement and the corresponding runtime, respectively, both averaged over all instance sizes. As before, larger values of K and scenario count generally lead to better performance. However, the rate of improvement is more moderate than in previous testbeds, and at $K = 2$ the improvement is near zero. Runtime grows steadily with the number of scenarios, while the dependence on K remains relatively mild.

6.4 ALP in Stochastic Programming

This subsection evaluates the stochastic version of ALP, introduced in Section 5, on the capital budgeting problem. Given their scalable performance in the robust setting, we focus on the SVM-H and DT-H variants. For each problem size $N \in \{5, 30, 50, 100, 1000, 3000, 5000\}$, five instances

Figure 7: Sensitivity to K and scenario count on the project management problem.



Note. Experiment results of DT-H are used for demonstration. The associated improvement percentage and runtime are averaged over all instance sizes.

Table 8 Performance comparison of the stochastic extension of ALP variants on the capital budgeting problem.

Projects (#)	Improvement (%)		Parallel Time (seconds)		Sequential Time (seconds)	
	SVM-H	DT-H	SVM-H	DT-H	SVM-H	DT-H
5	22.53	24.23	0.95	1.60	1.29	9.14
30	24.90	27.90	5.06	4.82	14.30	45.89
50	25.12	28.87	12.98	7.39	72.00	71.99
100	25.18	29.35	13.06	10.62	83.15	118.07
1000	24.84	29.15	81.63	79.91	251.76	1183.31
3000	22.29	28.28	233.58	235.18	832.39	3289.97
5000	22.54	27.98	385.23	398.66	1273.95	5083.45

Notes. Improvement is computed as $(z^* - z_{\text{sto}}^*)/z_{\text{sto}}^* \times 100$, where z^* and z_{sto}^* denote the optimal values of the tested algorithm and stochastic programming, respectively. Step 1 uses 2000 scenarios for the AAO heuristic. Parallel times report the maximum completion time across parallel tasks; sequential times report the sum of these task durations.

are generated following the setup in Subsection 6.2. We note that existing methods on stochastic K -adaptability [22, 25, 38] pursue exact solutions for discrete uncertainty sets, while the proposed ALP framework targets approximate finite-adaptable policies for general uncertainty sets with an emphasis on scalability.

Table 8 reports the computational results. For each problem size N , every variant is evaluated over $K \in \{2, 3, 4, 6, 8, 10\}$, and we report results for the K attaining the best average objective value z^* . Improvement is computed as $(z^* - z_{\text{sto}}^*)/z_{\text{sto}}^* \times 100$, where z_{sto}^* denotes the optimal stochastic programming value. Both parallel and sequential runtimes are presented.

Across all instance sizes, DT-H consistently achieves higher improvement than SVM-H, ranging from 24% to 29% compared with 22% to 25% for SVM-H. Moreover, improvement remains stable as N increases, indicating that the stochastic ALP framework maintains solution quality even on the largest instances. In terms of runtime, both variants remain computationally tractable as N increases. Under parallelization, they perform similarly, with DT-H slightly faster on small-to-medium instances and SVM-H on the largest, and all runtimes remaining below 400 seconds. In contrast, sequential runtimes grow noticeably with N , especially for DT-H, whose learning step refines the uncertainty set into a larger number of partitions. Overall, the stochastic extension of ALP achieves stable improvement while remaining efficient, particularly under parallel execution.

6.5 Summary of Numerical Insights

Beyond computational scalability, our experiments offer further insights into finite adaptability. When uncertainty affects only the objective (e.g., the shortest path problem), exact worst-case evaluation largely determines solution quality, and algebraic approaches such as [45] hold a natural advantage. When uncertainty also affects constraints, as in the capital budgeting and project management problems, the geometric perspective becomes preferable, since the proposed partition-based method yields stronger objective values and scales to substantially larger instances. These insights are also consistent with existing results, such as [33], which shows that worst-case evaluation becomes intractable under constraint uncertainty, and [32], which characterizes the non-convex structure of optimal partitions.

7 Conclusion

This paper developed a scalable framework for finite adaptability in both robust and stochastic settings. We established that polyhedral finite-adaptable policies converge to the fully adjustable benchmark under mild regularity conditions, motivating a parametric polyhedral partition framework that offers flexible design of partition schemes. Building on this, we proposed the ALP framework, which decouples partition construction from decision optimization by learning the partition from a discrete approximation and solving the resulting region-wise subproblems in parallel. The learned partition can be further refined through alternating optimization on a piecewise dualization reformulation. Computational experiments on shortest path, capital budgeting, and project management problems demonstrated that the proposed approach scales to instance sizes that are challenging for existing approaches, while producing solutions that are independently optimized within each region.

Several directions for future exploration emerge naturally from this work. First, extending the current framework to two-stage or multistage settings would broaden its applicability to a wider class of problems. Second, the partition-based structure is not inherently tied to a specific risk measure, suggesting natural extensions to distributionally robust optimization. Third, investigating richer partition schemes, such as those induced by ensemble trees or neural network classifiers, may improve partition quality while preserving the tractability of the region-wise subproblems.

References

- [1] Amir Ardestani-Jaafari and Erick Delage. Robust optimization of sums of piecewise linear functions with application to inventory problems. *Operations Research*, 64(2):474–494, 2016.
- [2] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.
- [3] Pranjal Awasthi, Vineet Goyal, and Brian Y Lu. On the adaptivity gap in two-stage robust linear optimization under uncertain packing constraints. *Mathematical Programming*, 173(1): 313–352, 2019.
- [4] Dimitra Bampou and Daniel Kuhn. Scenario-free stochastic programming with polynomial decision rules. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5330–5335, 2011.
- [5] Chaithanya Bandi, Eojin Han, and Omid Nohadani. Sustainable inventory with robust periodic-affine policies and application to medical supply chains. *Management Science*, 65(10):4636–4655, 2019.
- [6] Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization—methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.
- [7] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.
- [8] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.
- [9] Aharon Ben-Tal, Omar El Housni, and Vineet Goyal. A tractable approach for designing piecewise affine policies in two-stage adjustable robust optimization. *Mathematical Programming*, 182(1):57–102, 2020.
- [10] Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.
- [11] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7): 1039–1082, 2017.
- [12] Dimitris Bertsimas and Iain Dunning. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research*, 64(4):980–998, 2016.
- [13] Dimitris Bertsimas and Angelos Georghiou. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, 63(3):610–627, 2015.
- [14] Dimitris Bertsimas and Vineet Goyal. On the power of robust solutions in two-stage stochastic and adaptive optimization problems. *Mathematics of Operations Research*, 35(2):284–305, 2010.
- [15] Dimitris Bertsimas and Vineet Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical Programming*, 134(2):491–531, 2012.

- [16] Dimitris Bertsimas, Dan A. Iancu, and Pablo A. Parrilo. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, 2010.
- [17] Dimitris Bertsimas, Vineet Goyal, and Xu Andy Sun. A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. *Mathematics of Operations Research*, 36(1):24–54, 2011.
- [18] Dimitris Bertsimas, Vineet Goyal, and Brian Y Lu. A tight characterization of the performance of static solutions in two-stage adjustable robust linear optimization. *Mathematical Programming*, 150(2):281–319, 2015.
- [19] Dimitris Bertsimas, Shimrit Shtern, and Bradley Sturt. A data-driven approach to multistage stochastic linear optimization. *Management Science*, 69(1):51–74, 2023.
- [20] Christoph Buchheim and Jannis Kurtz. Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163(1):1–23, 2017.
- [21] Christoph Buchheim and Jannis Kurtz. Complexity of min–max–min robustness for combinatorial optimization under discrete uncertainty. *Discrete Optimization*, 28:1–15, 2018.
- [22] Christoph Buchheim and Jonas Prunte. K-adaptability in stochastic combinatorial optimization under objective uncertainty. *European Journal of Operational Research*, 277(3):953–963, 2019.
- [23] Xin Chen, Melvyn Sim, Peng Sun, and Jiawei Zhang. A linear decision-based approximation approach to stochastic programming. *Operations Research*, 56(2):344–357, 2008.
- [24] Maryam Daryalal, Merve Bodur, and James R Luedtke. Lagrangian dual decision rules for multistage stochastic mixed-integer programming. *Operations Research*, 72(2):717–737, 2024.
- [25] Federica Donnini, Marianna De Santis, and Jannis Kurtz. K-adaptability for two-stage stochastic optimization. *Optimization Online*, 2025.
- [26] Angelos Georghiou, Wolfram Wiesemann, and Daniel Kuhn. Generalized decision rule approximations for stochastic programming via liftings. *Mathematical Programming*, 152(1):301–338, 2015.
- [27] Angelos Georghiou, Daniel Kuhn, and Wolfram Wiesemann. The decision rule approach to optimization under uncertainty: methodology and applications. *Computational Management Science*, 16(4):545–576, 2019.
- [28] Angelos Georghiou, Angelos Tsoukalas, and Wolfram Wiesemann. A primal–dual lifting scheme for two-stage robust optimization. *Operations Research*, 68(2):572–590, 2020.
- [29] Marc Goerigk, Jannis Kurtz, and Michael Poss. Min–max–min robustness for combinatorial problems with discrete budgeted uncertainty. *Discrete Applied Mathematics*, 285:707–725, 2020.
- [30] Luigi Grippo and Marco Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.
- [31] Eojin Han and Omid Nohadani. Nonlinear decision rules made scalable by nonparametric liftings. *Management Science*, 71(4):3449–3471, 2025.

- [32] Eojin Han, Chaithanya Bandi, and Omid Nohadani. On finite adaptability in two-stage distributionally robust optimization. *Operations Research*, 71(1):88–106, 2023.
- [33] Grani A. Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.
- [34] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage distributionally robust binary programming. *Operations Research Letters*, 44(1):6–11, 2016.
- [35] Daniel Kuhn, Wolfram Wiesemann, and Angelos Georghiou. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209, 2011.
- [36] Jannis Kurtz. Approximation guarantees for min-max-min robust optimization and k -adaptability under objective uncertainty. *SIAM Journal on Optimization*, 34(2):1538–1563, 2024.
- [37] Hongcheng Liu, Ke Han, Vikash V Gayah, Terry L Friesz, and Tao Yao. Data-driven linear decision rule approach for distributionally robust optimization of on-line signal control. *Transportation Research Part C: Emerging Technologies*, 59:260–277, 2015.
- [38] Enrico Malaguti, Michele Monaci, and Jonas Prunte. K-adaptability in stochastic optimization. *Mathematical Programming*, 196(1):567–595, 2022.
- [39] Ahmadreza Marandi and Dick den Hertog. When are static and adjustable robust optimization problems with constraint-wise uncertainty equivalent? *Mathematical Programming*, 170(2):555–568, 2018.
- [40] Krzysztof Postek and Dick den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574, 2016.
- [41] Anubhav Ratha, Anna Schwele, Jalal Kazempour, Pierre Pinson, Shahab Shariat Torbaghan, and Ana Virag. Affine policies for flexibility provision by natural gas networks to power systems. *Electric Power Systems Research*, 189:106565, 2020.
- [42] R. Tyrrell Rockafellar. Coherent approaches to risk in optimization under uncertainty. In *OR Tools and Applications: Glimpses of Future Technologies*, pages 38–61. INFORMS, 2007.
- [43] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2021.
- [44] David Simchi-Levi, Nikolaos Trichakis, and Peter Yun Zhang. Designing response supply chain against bioattacks. *Operations Research*, 67(5):1246–1268, 2019.
- [45] Anirudh Subramanyam, Chrysanthos E. Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming*, 184(1):177–219, 2020.
- [46] Simon Thoma, Maximilian Schiffer, and Wolfram Wiesemann. A note on piecewise affine decision rules for robust, stochastic, and data-driven optimization. *Operations Research*, 2026.

- [47] Ningji Wei and Peter Zhang. Adjustability in robust linear optimization. *Mathematical Programming*, 208(1):581–628, 2024.
- [48] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust resource allocations in temporal networks. *Mathematical Programming*, 135(1):437–471, 2012.
- [49] Daniel Woolnough, V. Jeyakumar, and G. Li. Exact conic programming reformulations of two-stage adjustable robust linear programs with new quadratic decision rules. *Optimization Letters*, 15:25–44, 2021.
- [50] Ihsan Yanikoglu, Bram L Gorissen, and Dick den Hertog. A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813, 2019.

A Mathematical Proofs

Lemma 1. *Every risk measure \mathcal{R}_Ξ also satisfies the following two properties: (i) worst-case upper bound: $\mathcal{R}_\Xi(f) \leq \max_{\xi \in \Xi} f(\xi)$; (ii) 1-Lipschitz under ∞ -norm: $|\mathcal{R}_\Xi(f) - \mathcal{R}_\Xi(g)| \leq \|f - g\|_\infty$.*

Proof. Let $M := \max_{\xi \in \Xi} f(\xi)$, the following holds due to the three properties of risk measures.

$$f \leq M \implies \mathcal{R}_\Xi(f) \leq \mathcal{R}_\Xi(M) = \mathcal{R}_\Xi(0 + M) = \mathcal{R}_\Xi(0) + M = M.$$

For the second statement, let $\delta := \|f - g\|_\infty$, then $g - \delta \leq f \leq g + \delta$ by the definition of ∞ -norm. By monotonicity and translation equivalence,

$$\mathcal{R}_\Xi(g) - \delta = \mathcal{R}_\Xi(g - \delta) \leq \mathcal{R}_\Xi(f) \leq \mathcal{R}_\Xi(g + \delta) = \mathcal{R}_\Xi(g) + \delta,$$

i.e., $|\mathcal{R}_\Xi(f) - \mathcal{R}_\Xi(g)| \leq \delta$. □

Proposition 1. *Suppose that $f(\mathbf{y}, \cdot)$ is affine in ξ for every $\mathbf{y} \in \mathcal{Y}$. Then there exists an optimal partition $(\Xi_k)_{k \in \mathcal{K}}$ for (3) such that each Ξ_k is the intersection of Ξ with a polyhedron. More generally, if $f(\mathbf{y}, \cdot)$ is piecewise affine in ξ with finitely many polyhedral pieces for every $\mathbf{y} \in \mathcal{Y}$, then there exists an optimal partition $(\Xi_k)_{k \in \mathcal{K}}$ for (3) such that each Ξ_k is the intersection of Ξ with a finite union of polyhedra.*

Proof. Let $\mathbf{y} = (\mathbf{y}_k)_{k \in \mathcal{K}}$ be an optimal solution of (3), the partition problem reduces to assigning each ξ to the best response \mathbf{y}_k . Since the aggregation operator \mathcal{R} is monotone, each Ξ_k can be defined as

$$\Xi_k = \{\xi \in \Xi \mid f(\mathbf{y}_k, \xi) - f(\mathbf{y}_{k'}, \xi) \leq 0 \ \forall k' \neq k\}.$$

When f is affine in ξ , this is the intersection of a finite number of half-spaces with Ξ . When f is piecewise-affine in ξ with finite polyhedral pieces, we can construct a polyhedral refinement of the domain so that the function $f(\mathbf{y}_k, \xi) - f(\mathbf{y}_{k'}, \xi)$ restricted to each piece is affine. □

Theorem 1. *Suppose $\Xi \subseteq \mathbb{R}^n$ is compact. Let $\pi(\xi) \in \arg \min_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}, \xi)$ be the optimal fully adjustable policy. If, for every $\mathbf{y} \in \mathcal{Y}$, the function $f(\mathbf{y}, \cdot)$ is upper semicontinuous on Ξ , then there exists a family of polyhedral finite-adaptable policies $(\pi_\epsilon)_{\epsilon > 0}$ such that their value functions ϕ_{π_ϵ} converge pointwise to ϕ_π as $\epsilon \rightarrow 0$. Moreover, if $f(\mathbf{y}, \cdot)$ is Lipschitz continuous on Ξ for every $\mathbf{y} \in \mathcal{Y}$, then the aggregated values also converge as $\lim_{\epsilon \downarrow 0} \mathcal{R}_\Xi(\phi_{\pi_\epsilon}) = \mathcal{R}_\Xi(\phi_\pi)$.*

Proof. To prove the pointwise convergence, for every $\epsilon > 0$, we construct a surrogate value function ψ_ϵ and a polyhedral finite-adaptable policy π_ϵ with the following properties:

- (a) For every $\epsilon > 0$, $\psi_\epsilon \geq \phi_{\pi_\epsilon}$;
- (b) For every $\xi \in \Xi$, $\limsup_{\epsilon \downarrow 0} \psi_\epsilon(\xi) \leq \phi_\pi(\xi)$.

Since ϕ_{π_ϵ} is the value function of some policy π_ϵ , we obtain $\phi_{\pi_\epsilon} \geq \phi_\pi$ due to the optimality of π . On the other hand, (a) and (b) imply $\limsup_{\epsilon \downarrow 0} \phi_{\pi_\epsilon} \leq \phi_\pi$. Together, the following holds

$$\phi_\pi(\xi) \leq \liminf_{\epsilon \downarrow 0} \phi_{\pi_\epsilon}(\xi) \leq \limsup_{\epsilon \downarrow 0} \phi_{\pi_\epsilon}(\xi) \leq \phi_\pi(\xi),$$

which will imply $\lim_{\epsilon \downarrow 0} \phi_{\pi_\epsilon}$ converges pointwise to ϕ_π .

We first construct the surrogate value function ψ_ϵ . Let $B_{\epsilon, \boldsymbol{\xi}} := \{\boldsymbol{\xi}' \in \Xi \mid \|\boldsymbol{\xi}' - \boldsymbol{\xi}\|_\infty < \epsilon\}$ denote the open ϵ -ball under the ∞ -norm centered at $\boldsymbol{\xi}$, viewed as an open set in the subspace topology induced on Ξ . Let $\bar{B}_{\epsilon, \boldsymbol{\xi}}$ be the associated closed ϵ -ball, we define

$$\psi_\epsilon(\boldsymbol{\xi}) := \min_{\mathbf{y} \in \mathcal{Y}} \max_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon, \boldsymbol{\xi}}} f(\mathbf{y}, \boldsymbol{\xi}'),$$

so that its pointwise evaluation at $\boldsymbol{\xi}$ is the value of the robust optimization under the uncertainty set $\bar{B}_{\epsilon, \boldsymbol{\xi}}$.

To construct the polyhedral K -adaptable policy π_ϵ , we begin by considering the open cover $\{B_{\epsilon/4, \boldsymbol{\xi}}\}_{\boldsymbol{\xi} \in \Xi}$ of Ξ . By compactness of Ξ , there exists a finite subcover $\{B_{\epsilon/4, \boldsymbol{\xi}_i}\}_{i \in [m]}$ that covers Ξ . Since each closed ball $\bar{B}_{\epsilon/4, \boldsymbol{\xi}_i}$ is a hypercube under ∞ -norm, finite intersections of these sets are polyhedral. By successively intersecting and subtracting these hypercubes, the finite family $\{\bar{B}_{\epsilon/4, \boldsymbol{\xi}_i}\}_{i \in [m]}$ can be refined into a polyhedral partition $\{\Xi_k\}_{k \in \mathcal{K}_\epsilon}$ of Ξ , with overlaps occurring only on boundaries. Moreover, by construction, each partition element Ξ_k is entirely contained in at least one hypercube $\bar{B}_{\epsilon/4, \boldsymbol{\xi}_i}$. For each $\boldsymbol{\xi} \in \Xi$, let $\sigma(\boldsymbol{\xi}) := \min\{k \in \mathcal{K}_\epsilon \mid \boldsymbol{\xi} \in \Xi_k\}$. We then define

$$\pi_\epsilon(\boldsymbol{\xi}) \in \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}} \max_{\boldsymbol{\xi}' \in \Xi_{\sigma(\boldsymbol{\xi})}} f(\mathbf{y}, \boldsymbol{\xi}'), \quad \tilde{\psi}_\epsilon(\boldsymbol{\xi}) := \min_{\mathbf{y} \in \mathcal{Y}} \max_{\boldsymbol{\xi}' \in \Xi_{\sigma(\boldsymbol{\xi})}} f(\mathbf{y}, \boldsymbol{\xi}').$$

so that this policy selects for each $\boldsymbol{\xi}$ a solution that is robust with respect to the uncertainty set $\Xi_{\sigma(\boldsymbol{\xi})}$. Since each Ξ_k is constructed to be a polyhedron, and all scenarios from the same Ξ_k are associated with the same response by definition, π_ϵ is indeed a polyhedral finite-adaptable policy. Moreover, due to the robustness nature of this policy (i.e., the worst-case upper bound property from Lemma 1), its value function satisfies

$$\phi_{\pi_\epsilon}(\boldsymbol{\xi}) = f(\pi_\epsilon(\boldsymbol{\xi}), \boldsymbol{\xi}) \leq \tilde{\psi}_\epsilon(\boldsymbol{\xi}).$$

It remains to verify (a) and (b) under this construction. To show (a), we fix some $\boldsymbol{\xi} \in \Xi$. Since $\Xi_{\sigma(\boldsymbol{\xi})}$ is contained in some hypercube $\bar{B}_{\epsilon/4, \boldsymbol{\xi}_i}$, then the maximum ∞ -norm distance between $\boldsymbol{\xi}$ and any other point in $\Xi_{\sigma(\boldsymbol{\xi})}$ is at most 0.5ϵ , which implies $\Xi_{\sigma(\boldsymbol{\xi})} \subseteq \bar{B}_{\epsilon, \boldsymbol{\xi}}$. Hence, $\phi_{\pi_\epsilon}(\boldsymbol{\xi}) \leq \tilde{\psi}_\epsilon(\boldsymbol{\xi}) \leq \psi_\epsilon(\boldsymbol{\xi})$. For (b), we will show

$$\limsup_{\epsilon \downarrow 0} \psi_\epsilon(\boldsymbol{\xi}) = \limsup_{\epsilon \downarrow 0} \min_{\mathbf{y} \in \mathcal{Y}} \max_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon, \boldsymbol{\xi}}} f(\mathbf{y}, \boldsymbol{\xi}') \leq \min_{\mathbf{y} \in \mathcal{Y}} \limsup_{\epsilon \downarrow 0} \max_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon, \boldsymbol{\xi}}} f(\mathbf{y}, \boldsymbol{\xi}') \leq \min_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}, \boldsymbol{\xi}) = \phi_\pi(\boldsymbol{\xi}).$$

The first and last equalities hold by definition. For the first inequality, define $g_\epsilon(\mathbf{y}) := \max_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon, \boldsymbol{\xi}}} f(\mathbf{y}, \boldsymbol{\xi}')$, we obtain

$$\begin{aligned} \forall \epsilon > 0, \mathbf{y} \in \mathcal{Y}, \min_{\mathbf{y}' \in \mathcal{Y}} g_\epsilon(\mathbf{y}') \leq g_\epsilon(\mathbf{y}) &\implies \forall \mathbf{y} \in \mathcal{Y}, \limsup_{\epsilon \downarrow 0} \min_{\mathbf{y}' \in \mathcal{Y}} g_\epsilon(\mathbf{y}') \leq \limsup_{\epsilon \downarrow 0} g_\epsilon(\mathbf{y}) \\ &\implies \limsup_{\epsilon \downarrow 0} \min_{\mathbf{y}' \in \mathcal{Y}} g_\epsilon(\mathbf{y}') \leq \min_{\mathbf{y} \in \mathcal{Y}} \limsup_{\epsilon \downarrow 0} g_\epsilon(\mathbf{y}). \end{aligned}$$

For the second inequality, fix $\mathbf{y} \in \mathcal{Y}$, let $\epsilon_n := 1/n$ for $n \in \mathbb{N}$. Then, the optimal solutions $\boldsymbol{\xi}_n \in \operatorname{argmax}_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon_n, \boldsymbol{\xi}}} f(\mathbf{y}, \boldsymbol{\xi}')$ form a convergent sequence to the center $\boldsymbol{\xi}$. Since $f(\mathbf{y}, \cdot)$ is upper semicontinuous on Ξ , we obtain

$$\limsup_{\epsilon \downarrow 0} \max_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon, \boldsymbol{\xi}}} f(\mathbf{y}, \boldsymbol{\xi}') = \limsup_{\boldsymbol{\xi}_n \rightarrow \boldsymbol{\xi}} f(\mathbf{y}, \boldsymbol{\xi}_n) \leq f(\mathbf{y}, \boldsymbol{\xi}),$$

which verifies the second inequality and concludes the pointwise convergence.

Now, suppose $f(\mathbf{y}, \cdot)$ is L -Lipschitz for every $\mathbf{y} \in \mathcal{Y}$. Then, for every $\epsilon > 0$ and $\boldsymbol{\xi} \in \Xi$, our construction implies

$$0 \leq \phi_{\pi_\epsilon}(\boldsymbol{\xi}) - \phi_\pi(\boldsymbol{\xi}) \leq \psi_\epsilon(\boldsymbol{\xi}) - \phi_\pi(\boldsymbol{\xi}) \leq \max_{\boldsymbol{\xi}' \in \bar{B}_{\epsilon, \boldsymbol{\xi}}} f(\mathbf{y}^*, \boldsymbol{\xi}') - f(\mathbf{y}^*, \boldsymbol{\xi}) = f(\mathbf{y}^*, \boldsymbol{\xi}^*) - f(\mathbf{y}^*, \boldsymbol{\xi}),$$

where \mathbf{y}^* is the optimizer of $\phi_\pi(\boldsymbol{\xi})$. Since $f(\mathbf{y}^*, \cdot)$ is L -Lipschitz and the distance between the optimizer $\boldsymbol{\xi}^*$ and $\boldsymbol{\xi}$ is bounded by $\|\boldsymbol{\xi}^* - \boldsymbol{\xi}\|_\infty \leq \epsilon$ due to the closed ϵ -ball, the above gap is bounded by ϵL . Thus, $\lim_{\epsilon \downarrow 0} \|\phi_{\pi_\epsilon} - \phi_\pi\|_\infty = 0$. According to Lemma 1, \mathcal{R}_Ξ is 1-Lipschitz, which implies $\lim_{\epsilon \downarrow 0} |\mathcal{R}_\Xi(\phi_{\pi_\epsilon}) - \mathcal{R}_\Xi(\phi_\pi)| = 0$. This completes the proof. \square

Proposition 2. *For any partition scheme $(\mathcal{L}, \mathbf{T}, \boldsymbol{\tau})$, the induced collection \mathcal{P}_θ constitutes a partition of the uncertainty set Ξ for every $\theta \in \Theta$.*

Proof. Fix an arbitrary $\theta \in \Theta$ and let the affine map $f_\theta : \Xi \rightarrow \mathbb{R}^\ell$ by $f_\theta(\boldsymbol{\xi}) := \mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta$ for all $\boldsymbol{\xi} \in \Xi$. Using f_θ , we can write $\Xi_k^\theta = \{\boldsymbol{\xi} \in \Xi \mid f_\theta(\boldsymbol{\xi}) \in \mathcal{L}_k\} = \Xi \cap f_\theta^{-1}(\mathcal{L}_k)$, where $f_\theta^{-1}(\mathcal{L}_k) := \{\boldsymbol{\xi} \in \Xi \mid f_\theta(\boldsymbol{\xi}) \in \mathcal{L}_k\}$ denotes the preimage. We first prove that $\bigcup_{k \in \mathcal{K}} \Xi_k^\theta = \Xi$. Let $\boldsymbol{\xi} \in \Xi$ be arbitrary. Then $f_\theta(\boldsymbol{\xi}) \in \mathbb{R}^\ell$, and since $\mathcal{L} = (\mathcal{L}_k)_{k \in \mathcal{K}}$ is a partition of \mathbb{R}^ℓ in the sense of Remark 1, we have $\bigcup_{k \in \mathcal{K}} \mathcal{L}_k = \mathbb{R}^\ell$. Hence there exists some $k \in \mathcal{K}$ such that $f_\theta(\boldsymbol{\xi}) \in \mathcal{L}_k$, which by the definition of Ξ_k^θ implies $\boldsymbol{\xi} \in \Xi_k^\theta$. Since $\boldsymbol{\xi} \in \Xi$ was arbitrary, this shows that $\Xi \subseteq \bigcup_{k \in \mathcal{K}} \Xi_k^\theta$. Conversely, for every $k \in \mathcal{K}$ we have $\Xi_k^\theta \subseteq \Xi$ by construction, and therefore $\bigcup_{k \in \mathcal{K}} \Xi_k^\theta \subseteq \Xi$. Combining both inclusions yields $\bigcup_{k \in \mathcal{K}} \Xi_k^\theta = \Xi$. This proves that \mathcal{P}_θ is at least a cover of Ξ .

Finally, we show that this partition on Ξ can only overlap on the boundaries. Suppose otherwise, there exists some $\boldsymbol{\xi} \in \Xi$ that satisfies $\mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) < \mathbf{b}_k$ and $\mathbf{B}_{k'}(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_{k'}$ for some $k \neq k'$, which contradicts the construction that (\mathcal{L}_k) is a partition that only overlaps on the boundaries. \square

Theorem 2. *Let I be the index set of scenarios in $\{\boldsymbol{\xi}_i\}_{i \in I} \subseteq \Xi$, define binary variables $\lambda_{k,i} \in \{0, 1\}$ to indicate whether $\boldsymbol{\xi}_i$ is assigned to region Ξ_k^θ . For sufficiently large constants M , M_j , and M' , Problem (5) can be reformulated as the following master and subproblems:*

$$z^{\text{mst}} := \min_{\substack{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}, z \in \mathbb{R} \\ \theta \in \Theta, \lambda \in \{0, 1\}^{\mathcal{K} \times I}}} z \tag{6a}$$

$$\text{s.t.} \quad \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi}_i \rangle + \mathbf{q}(\mathbf{y}_k) \leq z + M(1 - \lambda_{ki}), \quad \forall i \in I, \forall k \in \mathcal{K}, \tag{6b}$$

$$\langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi}_i \rangle + \mathbf{q}_j(\mathbf{y}_k) \leq M_j(1 - \lambda_{ki}), \quad \forall j \in J, \forall k \in \mathcal{K}, \forall i \in I, \tag{6c}$$

$$\mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi}_i + \boldsymbol{\tau}_\theta) - \mathbf{b}_k \leq M'(1 - \lambda_{ki}) \mathbf{1}, \quad \forall i \in I, \forall k \in \mathcal{K}, \tag{6d}$$

$$\sum_{k \in \mathcal{K}} \lambda_{ki} = 1, \quad \forall i \in I. \tag{6e}$$

Given any solution (\mathbf{y}, θ) from (6), define the following subproblems for every $k \in \mathcal{K}$ and $j \in J$

$$z_k^{\text{opt}} := \max_{\boldsymbol{\xi} \in \Xi} \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}(\mathbf{y}_k) \tag{7a} \quad z_{kj}^{\text{fea}} := \max_{\boldsymbol{\xi} \in \Xi} \langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}_j(\mathbf{y}_k) \tag{7b}$$

$$\text{s.t.} \quad \mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_k, \quad \text{s.t.} \quad \mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) \leq \mathbf{b}_k,$$

The procedure halts when $z^{\text{mst}} = \max_{k \in \mathcal{K}} z_k^{\text{opt}}$ and $\max_{k \in \mathcal{K}, j \in J} z_{kj}^{\text{fea}} \leq 0$ under some incumbent (\mathbf{y}, θ) .

Proof. Problem (5) can be expressed equivalently by introducing indicator variables $\lambda_k(\boldsymbol{\xi}) \in \{0, 1\}$ that encode the assignment of each scenario $\boldsymbol{\xi}$ to a partition region. Specifically, $\lambda_k(\boldsymbol{\xi}) = 1$ indicates

that $\boldsymbol{\xi}$ belongs to region k , and the big- M constraints enforce the corresponding objective and feasibility requirements only for the active region. This yields the following semi-infinite formulation:

$$\begin{aligned}
& \min_{\substack{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}, z \in \mathbb{R} \\ \boldsymbol{\theta} \in \Theta, \boldsymbol{\lambda}}} z \\
& \text{s.t.} \quad \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}(\mathbf{y}_k) \leq z + M(1 - \lambda_k(\boldsymbol{\xi})), \quad \forall k \in \mathcal{K}, \forall \boldsymbol{\xi} \in \Xi_k^\theta, \\
& \quad \langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}_j(\mathbf{y}_k) \leq M_j(1 - \lambda_k(\boldsymbol{\xi})), \quad \forall j \in J, \forall k \in \mathcal{K}, \forall \boldsymbol{\xi} \in \Xi_k^\theta, \\
& \quad \mathbf{B}_k(\mathbf{T}_\theta \boldsymbol{\xi} + \boldsymbol{\tau}_\theta) - \mathbf{b}_k \leq M'(1 - \lambda_k(\boldsymbol{\xi}))\mathbf{1}, \quad \forall k \in \mathcal{K}, \forall \boldsymbol{\xi} \in \Xi_k^\theta, \\
& \quad \sum_{k \in \mathcal{K}} \lambda_k(\boldsymbol{\xi}) = 1, \quad \forall \boldsymbol{\xi} \in \Xi_k^\theta, \\
& \quad \lambda_k(\boldsymbol{\xi}) \in \{0, 1\}, \quad \forall k \in \mathcal{K}, \forall \boldsymbol{\xi} \in \Xi_k^\theta.
\end{aligned}$$

This is a semi-infinite linear program, for which a standard approach is cut generation. For a master incumbent $(\mathbf{y}, \boldsymbol{\theta})$, the feasibility certificate $\max_{k \in \mathcal{K}, j \in J} z_{kj}^{\text{fea}} \leq 0$ guarantees that $(\mathbf{y}, \boldsymbol{\theta})$ satisfies the original constraints in (5). Moreover, the condition $z^{\text{mst}} = \max_{k \in \mathcal{K}} z_k^{\text{opt}}$ certifies optimality: the master relaxation yields a valid lower bound, while (7a) provides an upper bound once (7b) verifies feasibility across all regions. \square

Theorem 3. *Given each part Ξ_k^θ nonempty, (5) can be equivalently reformulated as follows.*

$$\min_{\substack{\mathbf{y} \in \mathcal{Y}^{\mathcal{K}}, z \in \mathbb{R} \\ \boldsymbol{\theta} \in \Theta, \boldsymbol{\pi} \geq \mathbf{0}, \boldsymbol{\mu} \geq \mathbf{0}}} z \tag{8a}$$

$$\text{s.t.} \quad \mathbf{q}(\mathbf{y}_k) + \langle \mathbf{b}, \boldsymbol{\pi}_k \rangle + \langle \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta, \boldsymbol{\mu}_k \rangle \leq z, \quad \forall k \in \mathcal{K}, \tag{8b}$$

$$\mathbf{B}^\top \boldsymbol{\pi}_k + \mathbf{T}_\theta^\top \mathbf{B}_k^\top \boldsymbol{\mu}_k = \mathbf{Q}(\mathbf{y}_k), \quad \forall k \in \mathcal{K}, \tag{8c}$$

$$\mathbf{q}_j(\mathbf{y}_k) + \langle \mathbf{b}, \boldsymbol{\pi}_{kj} \rangle + \langle \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta, \boldsymbol{\mu}_{kj} \rangle \leq 0, \quad \forall k \in \mathcal{K}, j \in J, \tag{8d}$$

$$\mathbf{B}^\top \boldsymbol{\pi}_{kj} + \mathbf{T}_\theta^\top \mathbf{B}_k^\top \boldsymbol{\mu}_{kj} = \mathbf{Q}_j(\mathbf{y}_k), \quad \forall k \in \mathcal{K}, j \in J. \tag{8e}$$

Proof. Fix $k \in \mathcal{K}$ and $\mathbf{y}_k \in \mathcal{Y}$. Constraint (5b) is equivalent to

$$\max_{\boldsymbol{\xi} \in \Xi_k^\theta} \langle \mathbf{Q}(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}(\mathbf{y}_k) \leq z,$$

where $\Xi_k^\theta = \{\boldsymbol{\xi} \in \mathbb{R}^n \mid \mathbf{B}\boldsymbol{\xi} \leq \mathbf{b}, \mathbf{B}_k \mathbf{T}_\theta \boldsymbol{\xi} \leq \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta\}$. Since Ξ_k^θ is assumed nonempty, strong duality applies. Dualizing the above linear program with dual variables $\boldsymbol{\pi}_k \geq \mathbf{0}$ and $\boldsymbol{\mu}_k \geq \mathbf{0}$ gives

$$\mathbf{B}^\top \boldsymbol{\pi}_k + \mathbf{T}_\theta^\top \mathbf{B}_k^\top \boldsymbol{\mu}_k = \mathbf{Q}(\mathbf{y}_k), \quad \mathbf{q}(\mathbf{y}_k) + \langle \mathbf{b}, \boldsymbol{\pi}_k \rangle + \langle \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta, \boldsymbol{\mu}_k \rangle \leq z.$$

Applying the same dualization procedure to each constraint j in (5c) as

$$\max_{\boldsymbol{\xi} \in \Xi_k^\theta} \langle \mathbf{Q}_j(\mathbf{y}_k), \boldsymbol{\xi} \rangle + \mathbf{q}_j(\mathbf{y}_k) \leq 0$$

with dual variables $\boldsymbol{\pi}_{kj} \geq \mathbf{0}$ and $\boldsymbol{\mu}_{kj} \geq \mathbf{0}$ gives

$$\mathbf{B}^\top \boldsymbol{\pi}_{kj} + \mathbf{T}_\theta^\top \mathbf{B}_k^\top \boldsymbol{\mu}_{kj} = \mathbf{Q}_j(\mathbf{y}_k), \quad \mathbf{q}_j(\mathbf{y}_k) + \langle \mathbf{b}, \boldsymbol{\pi}_{kj} \rangle + \langle \mathbf{b}_k - \mathbf{B}_k \boldsymbol{\tau}_\theta, \boldsymbol{\mu}_{kj} \rangle \leq 0.$$

Combining all these constraints leads to the claimed reformulation (8). Since strong duality holds for all the dualization steps, this reformulation is exact. \square

Proposition 3. *The assignment step produces a solution λ that is optimal for (9) under any fixed policy $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$.*

Proof. Fix a policy $\{\mathbf{y}_k\}_{k \in \mathcal{K}}$. For any pair (k, i) such that $g_j(\mathbf{y}_k, \xi_i) > 0$ for some $j \in J$, assigning $\lambda_{k,i} = 1$ renders the objective value of (9) equal to $+\infty$. Hence, any optimal solution λ must satisfy $\lambda_{k,i} = 0$ for such pairs whenever this is feasible. If no feasible assignment exists, then every λ yields an objective value of $+\infty$, and the claim holds trivially.

Otherwise, $\lambda_{k,i}$ can take the value 1 only for pairs (k, i) satisfying all constraints. The problem reduces to

$$\begin{aligned} \min_{\lambda \in \{0,1\}^{\mathcal{K} \times I}} \quad & \max_{i \in I, k \in \mathcal{K}} f(\mathbf{y}_k, \xi_i) - M(1 - \lambda_{ki}) \\ \text{s.t.} \quad & \sum_{k \in \mathcal{K}} \lambda_{ki} = 1, \quad \forall i \in I, \end{aligned}$$

where $f(\mathbf{y}_k, \xi_i)$ are fixed constants. Consider any assignment λ that, for some $i \in I$, selects an index k that does not minimize $f(\mathbf{y}_k, \xi_i)$. Reassigning ξ_i to a minimizing index k weakly decreases the selected value for scenario i while leaving the selected values for all other scenarios unchanged. Consequently, the overall maximum objective value does not increase. Repeating this argument for all such scenarios yields an optimal assignment λ that assigns each ξ_i to an index k attaining $\min_{k \in \mathcal{K}} f(\mathbf{y}_k, \xi_i)$, which coincides with the assignment produced by the assignment step. \square

Proposition 4. *The AAO algorithm terminates in finitely many iterations with a convergent objective value.*

Proof. After initialization, the solution $((\mathbf{y}_k)_{k \in \mathcal{K}}, \lambda)$ remains feasible and its objective value is nonincreasing, since both steps are minimizations. Because the number of distinct assignments λ is finite, some assignment must recur, say at iterations t and $t' > t$. The nonincreasing property then implies that all iterates between t and t' share the same objective value. Since both steps are deterministic, the same cycle repeats from iteration t' onward, so the objective value remains constant and the algorithm terminates with a convergent objective value. \square

B Additional Experimental Results

This appendix reports comparisons of parallel and sequential runtimes for the ALP variants on the shortest path, capital budgeting, and project management problems, as detailed in Tables 9, 10, and 11, respectively.

Across the three problems, sequential runtimes remain reasonably close to their parallel counterparts in many cases, indicating that the computational bottleneck lies in non-parallelized steps. For the shortest path problem, sequential runtimes are typically within 120% of the parallel runtimes, with the heuristic variants (SVM-H and DT-H) remaining the fastest in most cases at under 250 seconds. A similar pattern holds for the capital budgeting problem, where sequential runtimes are within approximately 200% of the parallel runtimes. For the project management problem, the sequential runtimes of the exact variants remain relatively close to their parallel counterparts, with SVM-E and DT-E staying within approximately 130% and 102% of their corresponding parallel runtimes, respectively. In contrast, the heuristic variants exhibit larger sequential-to-parallel ratios, although they remain faster than the exact variants in both parallel and sequential implementations. Taken together, these results show that the robust ALP framework, particularly in its heuristic versions, scales efficiently across all three problem classes and retains its practical runtime advantage even without full parallelization.

Table 9 Shortest path problem: parallel and sequential runtimes.

Nodes (#)	Edges (#)	Parallel Time (seconds)				Sequential Time (seconds)			
		SVM-E	DT-E	SVM-H	DT-H	SVM-E	DT-E	SVM-H	DT-H
20	114	30.59	30.64	5.25	5.24	30.71	31.10	9.33	7.33
50	735	157.03	26.05	32.75	32.45	160.75	27.53	34.92	34.26
60	1062	152.45	35.75	97.80	98.15	166.27	38.56	108.62	103.54
70	1449	46.01	46.18	140.23	134.03	46.45	50.17	144.70	141.10
80	1896	61.28	61.30	69.69	74.21	61.91	66.79	81.80	82.45
90	2403	227.56	227.30	221.15	222.23	238.59	234.89	242.92	232.50
100	2970	253.17	173.17	108.07	108.81	264.54	180.83	135.51	123.57

Notes. Parallel times are measured as the maximum completion time among all parallel tasks, while the corresponding sequential times are computed as the sum of these task durations.

Table 10 Capital budgeting problem: parallel and sequential runtimes.

Projects (#)	Parallel Time (seconds)				Sequential Time (seconds)			
	SVM-E	DT-E	SVM-H	DT-H	SVM-E	DT-E	SVM-H	DT-H
5	4.29	16.78	0.21	0.14	4.33	16.78	1.30	1.22
30	1240.56	1203.27	0.55	0.38	1240.74	1203.66	1.14	1.27
50	1220.19	1206.09	20.96	0.58	1224.05	1206.70	22.02	1.86
100	1204.71	1213.52	33.74	0.94	1236.10	1214.39	47.14	2.99
1000	1260.47	1253.39	5.97	5.88	1262.36	1258.53	14.74	15.30
3000	1304.91	1360.52	18.69	18.07	1310.54	1376.43	35.23	44.05
5000	1387.28	1471.68	30.12	30.07	1396.08	1496.94	58.99	72.55

Notes. Parallel times are measured as the maximum completion time among all parallel tasks, while the corresponding sequential times are computed as the sum of these task durations.

Table 11 Project management problem: parallel and sequential runtimes.

m	Projects (#)	Parallel Time (seconds)				Sequential Time (seconds)			
		SVM-E	DT-E	SVM-H	DT-H	SVM-E	DT-E	SVM-H	DT-H
3	10	1211.21	1080.53	0.93	0.91	1217.27	1082.68	4.87	3.70
8	25	1212.12	467.18	1.89	1.74	1212.43	467.72	38.83	5.89
10	31	1236.96	1207.55	2.28	2.07	1283.02	1208.20	54.49	6.97
20	61	1273.80	1215.18	4.22	3.86	1424.19	1243.74	183.79	13.03
30	91	1310.20	1310.09	6.09	5.71	1669.82	1320.42	384.66	19.48
40	121	1348.77	1318.73	8.11	7.07	1736.96	1333.88	673.56	24.53
50	151	1312.53	1256.97	10.32	8.98	1363.37	1279.89	1042.22	31.77

Notes. Parallel times are measured as the maximum completion time among all parallel tasks, while the corresponding sequential times are computed as the sum of these task durations.