Data-driven Policies For Two-stage Stochastic Linear Programs

Chhavi Sharma*[1] and Harsha Gangammanavar†[1]

[1]Department of Operations Research and Engineering Management, Southern Methodist University, Dallas TX

Abstract

A stochastic program typically involves several parameters, including deterministic first-stage parameters and stochastic second-stage elements that serve as input data. These programs are re-solved whenever any input parameter changes. However, in practical applications, quick decision-making is necessary, and solving a stochastic program from scratch for every change in input data can be computationally costly. This work addresses this challenge for two-stage stochastic linear programs (2-SLPs) with varying right-hand sides for the first-stage constraints. We construct a Piecewise Linear Difference-of-Convex (PLDC) policy by leveraging optimal bases from previous solves. This PLDC policy retains optimal solutions for previously encountered parameters and provides high-quality solutions for new right-hand-side vectors. Our proposed policy directly applies to the extensive form of the 2-SLP. When stage decomposition algorithms, such as the L-Shaped and Stochastic Decomposition, are applied to solve the 2-SLPs, we develop L-Shaped- and Stochastic-Decomposition-guided static procedures to train the policy. We also develop a sequential procedure that iteratively tracks the quality of the learned policy and incorporates new basis information to improve it. We assess the performance of our policy through analytical and numerical techniques. Our compelling experimental results show that the policy prescribes solutions that are feasible and optimal for a significant percentage of new instances.

History: First submission: March 2026; Current version: March 17, 2026

1 Introduction

In this work, we study the two-stage stochastic linear programming (2-SLP) problem of the following form

$$\begin{aligned} \min \;\; & f(x) \coloneqq c^\top x + \mathbb{E}_{\mathbb{P}}[Q(x, \tilde{\xi})] && \text{(1a)} \\ \text{s.t.} \;\; & x \in \mathcal{X} \coloneqq \{x \in \mathbb{R}^{d_x} \mid Ax = b, x \geqslant 0\}, \end{aligned}$$

*chhsh17@gmail.com
†harsha@smu.edu

where $Q(x, \xi)$ is the value of a second-stage linear program,

$$Q(x, \xi) = \min_{y \in \mathbb{R}^{d_y}} \; q^\top y \tag{1b}$$

$$\text{s.t. } Wy = h(\xi) - T(\xi)x, \; y \geqslant 0. \tag{1c}$$

The decisions in 2-SLP are made in two stages. The decision x, known as a first-stage decision, is made before the random variable $\tilde{\xi}$ is realized. After implementing the decision x, a scenario of $\tilde{\xi}$ is observed, and the second-stage decision y is determined. Without loss of generality, we assume that A is of full row rank. The random vector represents the stochastic quantities that model the second stage, viz., $h \in \mathbb{R}^{m_2}$, and $T \in \mathbb{R}^{m_2 \times d_x}$. Throughout this article, we denote the random quantities using a tilde, such as $\tilde{\xi}$, and their observations without a tilde, as ξ. The 2-SLP has a diverse set of applications ranging from supply chain, power systems, telecommunications networks, pollution control, and chemical processes (Wallace and Ziemba, 2005). The above problem is parametrized by deterministic quantities $A \in \mathbb{R}^{m_1 \times d_x}, b \in \mathbb{R}^{m_1}$, and $c \in \mathbb{R}^{d_x}$, and the probability space associated with the random vector $\tilde{\xi}$ that we denote by $(\Xi, \Sigma_\xi, \mathbb{P})$. Typically, these parameters represent the available data to instantiate the 2-SLP problem. In this work, we study a setting in which we repeatedly need to solve 2-SLPs with parameters that may change over time.

The field of stochastic programming (SP) has excellent methods to efficiently solve (1) for given parameter values $(c, b, A, \mathbb{P})$, such as the L-Shaped (Van Slyke and Wets, 1969), proximal bundle (Ruszczyński, 1986) and stochastic decomposition (SD) methods (Higle and Sen, 1994). However, repeatedly solving a 2-SLP can be computationally expensive, even with efficient algorithms. Indeed, the most challenging component in solving 2-SLP is iteratively computing an approximation of the expectation-valued recourse function $\mathbb{E}_{\mathbb{P}}[Q(x, \tilde{\xi})]$. Even for small scenario sizes, solving the second-stage subproblems to obtain high-quality approximations of the recourse function and using these approximations to identify a here-and-now decision can be time-consuming. In many applications where optimization problems must be solved repeatedly and quickly, identifying optimal solutions from scratch under stringent deadlines is impractical. We identify one such example later in this section. Fortunately, these frequently solved decision problems share a common optimization structure, in which only some parameters are updated. This setting raises an interesting question: How can we leverage information from prior executions of SP algorithms to quickly identify implementable solutions for 2-SLP as its parameters change over time? This work addresses the question of when the *right-hand sides of the first-stage constraints change*. Specifically, we model the first-stage right-hand-side vector as a random variable $\tilde{b}$ associated with a probability measure $(\mathcal{B}, \Sigma_b, \mathbb{Q})$, where $\mathcal{B} \subset \mathbb{R}^{m_1}$ is a compact set. If the 2-SLP problem is feasible for almost all $b \in \mathcal{B}$, then given the information collected over the past n executions of an SP algorithm to solve 2-SLP with possibly different observations of $\tilde{b}$, we aim to identify a *policy* that maps a right-hand-side vector to the optimal solution of the corresponding 2-SLP problem.

Recall that every 2-SLP with finite support of second-stage uncertainty can be equivalently formulated as a linear programming (LP) problem, commonly referred to as the extensive scenario form in the SP literature (Ruszczynski and Shapiro, 2003), whose size scales with the number of second-stage scenarios. For deterministic LP, Walkup and Wets (1969) has formally demonstrated that the optimal solution is a continuous piecewise linear (PL) function of its right-hand side. In the ideal case, we can use this function to obtain

an optimal solution whenever we encounter a new right-hand side. However, finding its closed-form expression requires identifying optimal basis matrices of the extensive-scenario LP for all possible right-hand sides. Such an endeavor is computationally expensive for large-scale stochastic programs that arise in real-world applications. Alternatively, we can solve the following optimization problem:

$$\begin{aligned} \min_{\phi \in \Phi_{\mathrm{PL}}} \ & \mathbb{E}_{\tilde{b}}[f(\phi(\tilde{b}))] \\ \text{s.t.} \ & A\phi(\tilde{b}) = \tilde{b}, \phi(\tilde{b}) \geqslant 0 \text{ for almost every } \tilde{b}, \end{aligned} \tag{2}$$

where Φ_{PL} denotes the family of vector-valued continuous piecewise linear functions defined from $\mathbb{R}^{m_1}$ to $\mathbb{R}^{d_x}$. Problem (2) has at least one solution because the optimal solution of (1) is a piecewise linear function of the right-hand side. A solution $\phi^\star$ of (2), hereafter referred to as an optimal policy, can be used to get the optimal solutions of 2-SLP problems with varying right-hand sides. Clearly, both the objective and the constraints in (2) are nonconvex, rendering the problem intractable. Constraint-penalized approaches can be used to solve (2) by incorporating the constraints into the objective function via a suitable penalty term. Then, we can optimize the resulting penalized problem over the class of piecewise-linear functions using the algorithms developed, for instance, by Zhang et al. (2023). However, penalization-based approaches are sensitive to the choice of the penalty parameter and may yield infeasible solutions. In either case, a further complication is an incomplete knowledge of the objective $f(\cdot)$ in 2-SLP, as it requires all the pieces of the polyhedral function $\mathbb{E}_{\mathbb{P}}[Q(x, \tilde{\xi})]$.

The 2-SLP problem has nice structural properties regarding the objective function and the relationship between the optimal basis and the right-hand sides. These properties can inform the number of pieces in the PL policy and allow us to leverage information from prior executions of a solution algorithm. Thus, we will bypass the intractable nonconvex problem (2) to design a PL policy that efficiently delivers high-quality solutions whenever a new right-hand side is observed.

1.1 Relations to Parametric Optimization and Decision Rules

The 2-SLP can be viewed as a parametric stochastic program, which under finite support, is equivalent to a parametric LP. Sensitivity analysis is a fundamental approach for post-optimality analysis of the current basis of a parametrized LP problem (Bertsimas and Tsitsiklis, 1997). Suppose a solver produces an optimal basis $\mathbf{B}$ of an LP problem at right-hand side b. Sensitivity analysis tells us that $\mathbf{B}$ remains optimal within a range of perturbations on b. Beyond this sensitivity range, a new basis yields an optimal solution. This tool is useful for understanding how changes to parameters affect the optimal basis. Our policy design builds upon this understanding to address the issue of repeatedly solving large-scale, complex stochastic programs.

The 2-SLP problems are #P-hard (Dyer and Stougie, 2006) and intractable even when seeking medium-accuracy solutions (Shapiro and Nemirovski, 2005). Several previous works have addressed parametric SP (Robinson and Wets, 1987; Dupačová, 1990; Royset et al., 2025). However, they stem from the stability analysis of optimal values and solutions under perturbations in the underlying distribution of second-stage uncertainty. Moreover,

these works are analytical in nature, and their adoption into a solution algorithm, while appropriate, has not yet been addressed to the best of our knowledge. Another related stream of work in SP concerns the decision rules (Kuhn et al., 2011; Bertsimas and Goyal, 2012; Georghiou et al., 2015; Bertsimas and Bidkhori, 2015; Georghiou et al., 2019), where the aim is to address the computational difficulty of solving SP by assuming that the second-stage decision is either a linear or a nonlinear function of uncertainty scenarios ξ. The SP problem is then solved by replacing the second-stage decision with an appropriate mapping. The setting we study in this paper is distinct from that addressed in previous works on decision rules. We aim to find decision rules (which we call policies) for 2-SLPs that help repeatedly solve perturbed SP problems. We do not impose any restrictions on the behavior of second-stage decisions in relation to the observed scenario. Moreover, we concentrate on the propagation of information across different executions of the 2-SLP algorithms.

A relevant study for our setting is the recent work (Royset and Lejeune, 2024) that designs decision rules for mixed-binary parametric optimization problems. This work lays a solid theoretical foundation for studying the asymptotic behavior of decision rules derived from training problems. However, two major difficulties prevent us from using their decision rules. The first is that it lacks guidance on how to solve the sequence of nonconvex training problems of the form (2), which is required to adopt their approach in our setting. The second bottleneck arises because their feasible decision rules are prescribed for problems where the constraints are independent of the parameters, whereas we deal with varying constraint parameters in our 2-SLP setting.

1.2 Contributions

In this paper, we design data-driven Piecewise Linear Difference-of-Convex (PLDC) policies to solve 2-SLP problems with adaptive right-hand sides of the first-stage constraints. These policies also apply to deterministic LP problems that must be solved repeatedly, with right-hand sides varying across instantiations. Since our goal is to devise a policy that delivers near-optimal or at least feasible solutions to 2-SLP problems, which are repeatedly solved in practice, the process of finding such a policy should be simpler than that of solving the 2-SLP problems from scratch. In this regard, we summarize the main contributions of this paper below:

1. We utilize the properties of the optimal solution map and difference-of-convex functions to design a linear training problem. A solution to this training problem generates the desired PLDC policy for deterministic LP problems. The training problem is constructed using data comprising right-hand sides and the optimal solution and optimal basic variables of the corresponding LP. We analytically show that our proposed policy yields an optimal solution whenever the right-hand-side vector lies in the convex hull of the right-hand sides in the training data. This part of the contributions is presented in section 2.

2. In section 3, we adopt the PLDC policy of deterministic LP to 2-SLP problems solved using the L-Shaped or SD methods as solution algorithms. We devise a static procedure that constructs a consolidated master problem by reusing the local

polyhedral approximations and terminal solutions generated by executions of the decomposition algorithm to solve a fixed number of 2-SLPs with different right-hand sides. Based on this consolidated problem, we formulate a training problem whose solution yields the parameters of the PLDC policy for 2-SLPs. It is also worth noting that the resulting PLDC policy-training problem can be solved using off-the-shelf LP solvers.

3. Next, in section 3.3, we develop a unified decomposition algorithm-guided sequential procedure using either L-Shaped or SD methods. This sequential procedure iteratively feeds information from new solves to train the policy iteratively and tracks its quality. We maintain a training dataset that includes the right-hand sides, optimal solutions, optimal values, and local approximations. Adding new information to the training data increases the size of the training problem, which we control by appending it to the training set from 2-SLPs that are infeasible or suboptimal under the current policy. We prove that the sequential procedure returns a viable policy in a finite number of rounds under reasonable assumptions.

4. We test our PLDC policies on multiple 2-SLP problems, ranging from small to large scale. This computational study demonstrates that our proposed policies retain the optimal solutions from previous solves and produce optimal solutions for most new instances. Experiments with the sequential procedure show that it arms the training process to efficiently incorporate new information, thereby improving the quality of the policy-prescribed solutions in an online manner.

The main goal of designing policies that prescribe optimal or near-optimal solutions to large-scale SP problems within a reasonable time is to enable the practice of operations research tools, particularly SP models and algorithms, in real-world settings. We identify one such application area below.

1.3 A Motivating Example

Economic dispatch (ED) is a key problem that power system operators solve to plan electricity generation from various sources, including hydro, thermal (e.g., coal or gas), and renewable (e.g., wind and solar). The goal is to determine the amount of electricity each generating unit should produce to minimize total operating cost. This planning problem comprises several constraints, including the system flow-balance equation; physical constraints on generators (capacity, ramping, etc.) and transmission lines (capacity, power flows, etc.); and water-conservation equations for the reservoirs. The stochastic variant of the ED problem can be cast as a 2-SLP (see Gangammanavar et al. (2015)), in which the first-stage decision is to determine the generation levels of slow-responding generators fueled, for instance, by coal and nuclear. Once the actual demand is revealed, the second stage determines the generation levels of fast-responding gas generators needed to align total generation with the realized demand and determine the power flows through the transmission network. This 2-SLP problem includes several parameters, such as electricity price, resource availability, and weather-related factors, which are integral to the model. In practice, the ED problem is solved several times a day. For instance, the California Independent System Operator must solve the ED problem every 5 minutes California

Independent System Operator (CAISO) (2022). At this timescale, the parameters change continuously due to market fluctuations and varying weather conditions. Thus, it becomes necessary to re-solve a 2-SLP problem whenever any of these parameters change, making the stochastic ED an excellent example of the decision setting we study in this paper. Similar decision settings also arise in applications involving production planning, scheduling, resource allocation, and inventory control. The current state-of-the-art algorithms are incapable of delivering high-quality solutions in the desired timeframe. For instance, the SD algorithm takes about 20 minutes to deliver a statistically optimal solution to an ED instance for a medium-sized power network, as reported in Gangammanavar et al. (2015). This experience motivates us to avoid solving these SP problems from scratch and instead explore policy designs that are easy to train and execute within the tight time bounds.

1.4 Notations and Background

Throughout this paper, we assume that the scalars, sets, vectors, and matrices in a collection are denoted by the indices in their superscripts. For example, vector b^i denotes the i-th right-hand-side vector, B^i denotes i-th matrix and I^i denotes i-th set of indices $\{i_1, i_2, \ldots\} \subseteq \{1, 2, \ldots, d_x\}$. We use subscripts to denote components of a vector, e.g., b^i_k denotes the k-th component of b^i. The boldface symbols $\mathbf{0}$ and $\mathbf{1}$ denote, respectively, vectors of zeros and ones of appropriate dimensions, as will be clear from the context. Let $\mathbf{0}_n$ denote a square matrix of size n with all zero entries. An optimal solution and the corresponding optimal value of (1) at a right-hand side b are denoted by $x^\star(b)$ and $v^\star(b)$ respectively. For any positive integer a, $[a]$ denotes a set of indices $\{1, 2, \ldots, a\}$. We denote the convex hull of a finite set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ by $\text{Conv}(\mathcal{S}) = \{\lambda_1 s_1 + \lambda_2 s_2 + \ldots + \lambda_n s_n \mid \sum_{i=1}^n \lambda_i = 1, \lambda_i \geqslant 0\}$. A continuous function $f : \mathbb{R}^m \to \mathbb{R}^d$ is piecewise linear if there exists a finite set of linear functions $f_i(x) = D^i x$, $i = 1, \ldots, K$ such that $f(x) \in \{f_1(x), \ldots, f_K(x)\}$ for all $x \in \mathbb{R}^m$ (Scholtes, 2012). A function $f : \mathbb{R}^m \to \mathbb{R}^d$ is called a difference-of-convex function if there exists convex functions $g_1 : \mathbb{R}^m \to \mathbb{R}^d$ and $g_2 : \mathbb{R}^m \to \mathbb{R}^d$ such that $f(x) = g_1(x) - g_2(x)$ for all $x \in \mathbb{R}^m$, where difference is defined componentwise (Le Thi and Pham Dinh, 2018).

2 Policy Design

In this section, we construct a policy for deterministic LP problems with adaptive right-hand sides. We focus on a deterministic LP to communicate the key ideas involved in policy design and construction to the reader without complicating the notation and adapt the steps to 2-SLP in the subsequent sections of the paper. To this effect, we consider the standard form of a deterministic LP problem: $\min \{c^\top x \mid x \in \mathcal{X}\}$. Notice that we retain the notations and form of the first-stage program in (1a). The optimal solutions of LP problems are extreme points of the feasible region, also called basic feasible solutions. We can partition the solution vector into basic variables, indexed by B and denoted by x_B, and nonbasic variables, indexed by N and denoted by x_N. The columns of the constraint matrix A corresponding to basic variables constitute the basis that we denote by $\mathbf{B}$. If x is an optimal solution, then we term the corresponding matrix $\mathbf{B}$ as the optimal basis. Using the basis matrix, the basic feasible solution is given by $x_B = \mathbf{B}^{-1} b,\ x_N = 0$.

When some or all entries of the right-hand-side vector b are changed, say to a new vector b', the perturbation theory of LP (Bertsimas and Tsitsiklis, 1997) says that the current basis $\mathbf{B}$ is optimal to b'. Typically, the post-optimality sensitivity analysis reveals the sensitivity range when each element is perturbed in turn. In particular, the basis $\mathbf{B}$ remains optimal as long as $\mathbf{B}^{-1}b'$ is nonnegative. Beyond a certain level of perturbation, a new basis matrix, say $\mathbf{B}'$, yields the optimal solution. Hence, the optimal solution of an LP problem is a continuous, piecewise-linear function of the right-hand-side vector. In other words, ϕ^* is a continuous, piecewise linear function. This result is well established in Walkup and Wets (1969) for parametric LPs.

More formally, following the Basis Decomposition Theorem of Walkup and Wets (1969), we can decompose the closed convex polyhedral cone $\text{pos}(A) = \{b \mid b = Ax, x \geqslant 0\}$ into a finite number, say $\bar{L}$, of closed convex polyhedra, which are referred to as cells. The m_1 columns of A corresponding to the edges of an m_1-dimensional cell $\mathcal{C}$ form an optimal basis for all $b \in \mathcal{C}$. If $\mathbf{B}(\mathcal{C})$ is the optimal basis corresponding to cell $\mathcal{C}$, then a linear piece given by $(\mathbf{B}(\mathcal{C}))^{-1}b$, which we term the optimal piece, recovers the optimal value of basic variables for all $b \in \mathcal{C}$. Moreover, since the $\bar{L}$ cells cover the entire $\text{pos}(A)$, the corresponding bases, along with the trivial zero function for non-basic variables, provide the means to construct the optimal piecewise linear policy ϕ^*. Since the number of cells is finite, the policy ϕ^* has a finite number of pieces.

2.1 Data-driven Policy Approximation

Obtaining a closed-form expression of the optimal piecewise policy ϕ^* requires discovering all possible bases, a number that scales exponentially in the problem dimension. This step is computationally prohibitive, particularly for large-scale LP models commonly encountered in real-world applications and in SP problems that we encounter in later sections. Alternatively, we can adopt a data-driven approach that uses the optimal bases obtained by solving LP problems for a finite set of right-hand-side parameters, say n. Let $\widehat{\mathcal{B}} := \{b^j\}_{j\in[n]}$ denote the finite collection of right-hand sides and $\widehat{\mathcal{X}} := \{x^\star(b^j)\}_{j\in[n]}$ the optimal solution vectors at these right-hand sides. Furthermore, let $\{\mathbf{B}^\ell\}_{\ell=1}^{L}$ denote the collection of optimal bases that we discover (notice $L \leqslant n$ and $L \leqslant \bar{L}$). We construct approximate cells by placing the right-hand sides $\widehat{\mathcal{B}}$ that have the same optimal basis $\mathbf{B}^\ell$ into a cell $\mathcal{C}^\ell$, which we define as $\mathcal{C}^\ell = \{b \in \mathbb{R}^{m_1} \mid x^\star_B(b) = (\mathbf{B}^\ell)^{-1}b\} \subseteq \widehat{\mathcal{B}}$. Figure 1 illustrates such cells for an LP problem with two equality constraints and five variables. The right-hand-side vectors belonging to the same cell are plotted using the same marker (and color).

Using the above elements, we construct a data-driven approximate policy $\hat{\phi} : \mathcal{B} \to \mathbb{R}^{d_x}$ such that for any $b \in \mathcal{B}$ as

$$\big(\hat{\phi}(b)\big)_k = \begin{cases} \big((\mathbf{B}^{\ell^*})^{-1}b\big)_k & \text{if } k \in B^{\ell^*} \\ 0 & \text{if } k \in N^{\ell^*}, \end{cases} \tag{3}$$

where $\ell^* \in \{\ell \mid (\mathbf{B}^\ell)^{-1}b \geqslant 0,\ \ell \in [L]\}$. Notice that the basis $\mathbf{B}^{\ell^*}$ is always optimal since the cost coefficients are unchanged. It is also worthwhile to note that $\mathbf{B}^{\ell^*}$ remains optimal for any $b \in \text{Conv}(\mathcal{C}^{\ell^\star})$, even if $b \notin \widehat{\mathcal{B}}$. On the contrary, since we may have discovered only a subset of optimal bases in our data-driven setting, we may encounter a case where no such

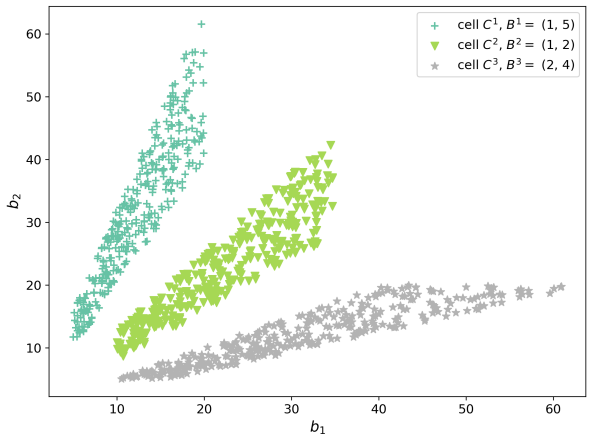

Figure 1: (Color online) Approximate cells for an LP problem with two constraints and five variables. b_1 and b_2 denote the right-hand-side values of the first and second constraints, respectively.

ℓ^* is found. As a consequence, all bases in the set $\{\mathbf{B}^\ell\}_{\ell=1}^L$ are infeasible to a right-hand side b. Therefore, $\hat{\phi}$ is only an approximation of the true policy $\phi^\star$.

While $\hat{\phi}$ is a viable policy, it requires considerable memory to store the dense inverses of the bases. Alternatively, one can store the column indices of the coefficient matrix A, and, upon observing a new b, compute the matrix inverse. Clearly, performing inverse computations and conducting feasibility checks for every new b also entails high costs. In what follows, we leverage the piecewise-linear relationship between the optimal solutions and the right-hand sides of the constraints to construct an alternative characterization of the policy $\hat{\phi}$. This characterization yields a functional form that does not require feasibility checks or the storage or computation of the bases' inverses.

To establish such a characterization, we exploit the fact that we can express a piecewise-linear function as a difference-of-convex function (Kripfganz and Schulze, 1987; Scholtes, 2012). That is, we can represent a piecewise linear vector-valued function $\phi(b)$ as $\phi(b) = \phi_1(b) - \phi_2(b)$ where $\phi_1 : \mathbb{R}^{m_1} \to \mathbb{R}^{d_x}$ and $\phi_2 : \mathbb{R}^{m_1} \to \mathbb{R}^{d_x}$ are convex piecewise-linear functions componentwise. In our context, we aim to discover the convex functions ϕ_1 and ϕ_2 that we can use to reconstruct the optimal policy ϕ^* or at least its data-driven approximation $\hat{\phi}$. We can deem such a reconstruction as a success if, when restricted to $\text{Conv}(\mathcal{C}^\ell)$, $\phi(b)$ recovers the piece $(\mathbf{B}^\ell)^{-1}b$ componentwise for basic variables and takes zero value for nonbasic variables. In this case, $(\phi(b))_k = (\phi_1(b))_k - (\phi_2(b))_k = ((\mathbf{B}^\ell)^{-1}b)_k$ for $b \in \text{Conv}(\mathcal{C}^\ell)$ and $k \in B^\ell$. A key observation is that ϕ_1 and ϕ_2 can each have multiple pieces and still yield a linear piece, since the difference between two piecewise linear functions can be linear. That is, a cell may have more than one piece in ϕ_1 and ϕ_2. However, we show in Lemma 2.1 below that there exist *convex* functions ϕ_1 and ϕ_2 such that, restricting them to $\text{Conv}(\mathcal{C}^\ell)$, they are linear functions. We prove such existence in terms of one subgradient for all right-hand sides belonging to a given cell $\mathcal{C}^\ell$.

Lemma 2.1. *Let $\phi(b) = \phi_1(b) - \phi_2(b)$ be a continuous piecewise-linear function, and let the right-hand side set $\widehat{\mathcal{B}}$ be partitioned into L cells. For every cell $\mathcal{C}^\ell$, there exists $u^\ell = [u_1^\ell; \ldots; u_{d_x}^\ell] \in \mathbb{R}^{d_x \times m_1}$, $v^\ell = [v_1^\ell; \ldots; v_{d_x}^\ell] \in \mathbb{R}^{d_x \times m_1}$ such that $u_k^\ell \in \partial(\phi_1(b))_k$ and $v_k^\ell \in \partial(\phi_2(b))_k$ for all $b \in Conv(\mathcal{C}^\ell)$ and $\ell \in [L]$.*

Our approach to proving this lemma is to fix a cell and show that every linear function can be represented as the difference of two linear functions. Moreover, we show that the collection of linear functions identified across all the cells forms piecewise-affine convex functions ϕ_1 and ϕ_2. We provide a complete proof of Lemma 2.1 in Appendix A.

From Lemma 2.1 and selecting a pair $(b^\ell, x^\star(b^\ell))$ from each cell $\mathcal{C}^\ell$, we obtain the following characterization of the policy:

$$\begin{aligned}(\phi(b))_k = &\max_{\ell\in[L]}\Big((u_k^\ell)^\top(b-b^\ell) + (x^\star(b^\ell))_k + z_k^\ell\Big)\\ &- \max_{\ell\in[L]}\Big((v_k^\ell)^\top(b-b^\ell) + z_k^\ell\Big).\end{aligned} \tag{4}$$

Here, the parameter vector $\{\theta_k\}_{k=1}^{d_x} = \{(u_k^{\ell(j)}, v_k^{\ell(j)}, z_k^j)_{j=1}^n\}_{k=1}^{d_x} \in \mathbb{R}^{(2m_1L+n)d_x}$ is a solution that satisfies the following system of inequalities:

$$\begin{aligned}&\langle u_k^{\ell(j)}, b^i - b^j\rangle + (x^\star(b^j))_k + z_k^j\\ &\qquad\leqslant (x^\star(b^i))_k + z_k^i \text{ for all } i,j\in[n],\end{aligned} \tag{5a}$$

$$\langle v_k^{\ell(j)}, b^i - b^j\rangle + z_k^j \leqslant z_k^i \text{ for all } i,j\in[n]. \tag{5b}$$

We call $\phi(b)$ a *data-driven* piecewise linear difference-of-convex (PLDC) policy as it is obtained using a finite number of data points $\{b^j, x^\star(b^j)\}_{j=1}^n$. Due to the PL nature of the optimal solution map and using Lemma 2.1, there exists at least one solution to the inequalities in (5), which we establish in Theorem 2.2 along with policy properties.

Theorem 2.2. *For a LP in the standard form, let $\widehat{\mathcal{B}}$ and $\widehat{\mathcal{X}}$ be given. Then, the inequalities in* (5) *have at least one feasible solution $\{\theta_k\}_{k=1}^{d_x} = \{(u_k^{\ell(j)}, v_k^{\ell(j)}, z_k^j)_{j=1}^n\}_{k=1}^{d_x}$ and the policy $\phi(b)$ in* (4) *constructed from $\{\theta_k\}_{k=1}^{d_x}$ satisfies the following:*

$$\phi(b) = x^\star(b) \text{ for all } b\in \mathit{Conv}(\mathcal{C}^\ell),\ \ell\in[L]. \tag{6}$$

Proof. Proof. We divide the proof into two parts. In the first part, we show the existence of a feasible solution. In the second part, we derive the property (6) of policy $\phi(b)$.

Feasible solution existence: We can write optimal solution map $\phi^\star(b)$ as $\phi^\star(b) = \phi_1^\star(b) - \phi_2^\star(b)$. Let $(\phi_2^\star(b^i))_k = z_k^i$ and $(\phi_2^\star(b^j))_k = z_k^j$ and consider

$$\begin{aligned}(\phi^\star(b^i))_k &= (\phi_1^\star(b^i))_k - (\phi_2^\star(b^i))_k\\ &\geqslant (\phi_1^\star(b^j))_k + \langle u_k^{\ell(j)}, b^i - b^j\rangle - (\phi_2^\star(b^i))_k\\ &= (\phi^\star(b^j))_k + \langle u_k^{\ell(j)}, b^i - b^j\rangle + z_k^j - z_k^i.\end{aligned} \tag{7}$$

The first inequality follows from the definition of subgradient. We recall that there exists a piecewise linear optimal solution map $\phi^\star$ such that $\phi^\star(b) = x^\star(b)$ for all $b\in\mathcal{B}$. Using this, we rewrite (7) as

$$\begin{aligned}&(x^\star(b^i))_k - (x^\star(b^j))_k\\ &\geqslant \langle u_k^{\ell(j)}, b^i - b^j\rangle - z_k^i + z_k^j \text{ for all } i,j\in[n].\end{aligned} \tag{8}$$

For the $\phi_2^\star$ function, we have

$$z_k^i \geqslant z_k^j + \langle v_k^{\ell(j)}, b^i - b^j\rangle. \tag{9}$$

From Lemma 2.1, there exists $u_k^{\ell(j)}$ and $v_k^{\ell(j)}$ satisfying (7) and (9); completing the proof of the first part.

Policy properties: Consider a cell $\mathcal{C}^\ell$. Then following the arguments of the proof of Lemma 2.1, ϕ_1 and ϕ_2 restricted to $\text{Conv}(\mathcal{C}^\ell)$ are linear functions. It implies that inequalities in (5) are tight for all $b^i, b^j \in \mathcal{C}^\ell$ i.e.,

$$\langle u_k^\ell, b^i - b^j \rangle + x^\star(b^j)_k + z_k^j = x^\star(b^i)_k + z_k^i$$
$$\langle v_k^\ell, b^i - b^j \rangle + z_k^j = z_k^i.$$

For any other $\tilde{\ell} \neq \ell$, we only have the inequalities in (5). This implies that $\max_{\ell' \in [L]} (u_k^{\ell'})^\top (b^i - b^{\ell'}) + x^\star(b^{\ell'})_k + z_k^{\ell'} = x^\star(b^i)_k + z_k^i$ and $\max_{\ell' \in [L]} (v_k^{\ell'})^\top (b^i - b^{\ell'}) + z_k^{\ell'} = z_k^i$ and hence $\phi(b^i) = x^\star(b^i)$. Consider a point $\hat{b}^\ell = \sum_{i=1}^{|\mathcal{C}^\ell|} \lambda^i b^{n_i} \in \text{Conv}(\mathcal{C}^\ell)$. Then we can clearly see that $(\phi_1(\hat{b}^\ell))_k = \sum_{i=1}^{|\mathcal{C}^\ell|} \lambda^i (x^\star(b^{n_i})_k + z_k^{n_i})$ and $(\phi_2(\hat{b}^\ell))_k = \sum_{i=1}^{|\mathcal{C}^\ell|} \lambda^i z_k^{n_i}$ and hence $\phi(\hat{b}^\ell) = x^\star(\hat{b}^\ell)$, completing the proof. □

The striking advantage of PLDC policy (4) is that it can be obtained using off-the-shelf solvers as it requires a solution feasible to constraints in (5). This efficiency can be further enhanced by observing that many entries in u^ℓ, v^ℓ, and z^ℓ may end up being zero when the number of nonbasic variables is large, and a few components of the right-hand-side vector are varying. Thus, we effectively restrict the solutions of (5a)-(5b) to the ones having minimum ℓ_1 norm. This step yields a convex optimization problem, which we refer to as the training problem.

$$\min_{u,v,z} \ \sum_{l=1}^{L} \sum_{k=1}^{d_x} \|u_k^\ell\|_1 + \|v_k^\ell\|_1 + \sum_{i=1}^{n} \|z^i\|_1 \tag{10a}$$
$$\text{s.t.} \ \langle u_k^{\ell(j)}, b^i - b^j \rangle + x^\star(b^j)_k + z_k^j$$
$$\leqslant x^\star(b^i)_k + z_k^i, \forall\ i, j \in [n],\ k \in [d_x], \tag{10b}$$
$$\langle v_k^{\ell(j)}, b^i - b^j \rangle + z_k^j \leqslant z_k^i, \forall\ i, j \in [n],\ k \in [d_x]. \tag{10c}$$

With the solution of this training problem, we can construct the PLDC policy (4). The resulting PLDC policy has several beneficial properties, which we identify below.

1. The policy recovers an optimal piece $(\mathbf{B}^\ell)^{-1}b$ for all b within the convex hull $\text{Conv}(\mathcal{C}^\ell)$. As the optimal solutions for past right-hand-side values are obtained after a significant investment of resources (viz., executing an SP algorithm), the ability to recover these solutions exactly is a minimal expectation that our policy design certainly meets. Note that an optimal piece $(\mathbf{B}^\ell)^{-1}b$ over an entire cell may not be recovered if the number of data points that belong to the cell is limited.

2. Choosing one subgradient per cell rather than a subgradient for every observation of the right-hand-side vector is one of the crucial features of our characterization, which alleviates the issue of holding a large number of pieces. Siahkamari et al. (2020) propose a policy that shares certain similarities with our PLDC policy (4), albeit for piecewise linear regression. In their design, they need one piece per data point. However, such an approach does not work in our problem setting and fails to recover the pieces $(\mathbf{B}^\ell)^{-1}b$. We elaborate on these differences in the behavior in Appendix D.

3. It is also worth recalling that the methodologies in Royset and Lejeune (2024) and Zhang et al. (2023) solve computationally expensive nonconvex problems sequentially to learn a piecewise linear function. In contrast to their approach, our training problem (10) is a convex optimization problem that can be reformulated into an LP and solved using off-the-shelf LP solvers. Moreover, our policy characterizes the optimal bases from previous solves in a functional form and prescribes the solution for a new right-hand side. Therefore, executing the policy to identify a solution to a new instance requires minimal computational effort and can be delivered within tight time bounds.

While the policy design procedure we present in this paper pertains to perturbations to the constraint right-hand sides, it also applies to the relatively easier setting of perturbations to the cost coefficients. In the next section, we adapt the training problem (10), and subsequently, the PLDC policy (4) to 2-SLPs with varying right-hand sides in the first stage.

3 PLDC Policy for Two-Stage Stochastic Linear Programming

In this section, we construct data-driven policies for 2-SLPs using the tools established in the previous section. Before presenting the policy design procedures, we state the assumptions for the 2-SLP models with a right-hand side belonging to $\mathcal{B}$.

A1. The solution set $\mathcal{X}$ of first-stage decisions is compact and set of scenarios Ξ is finite.

A2. The relatively complete recourse property is satisfied.

A3. The recourse function $Q(x, \xi) > -\infty$ almost surely.

A4. The recourse matrix (W) is deterministic, i.e., the fixed recourse property is satisfied.

The assumption A2. implies that the second-stage program is feasible for all $x \in \mathcal{X}$ and $\xi \in \Xi$, implying $Q(\cdot, \xi) < \infty$, almost surely. With A3., the dual feasible set of the second-stage is nonempty, almost surely.

The PLDC policy (4) directly applies to the extensive scenario form of a 2-SLP problem with finite support. Recall that the policy training dataset includes optimal solutions and the indices of their basic variables. Obtaining these requires solving extensive scenario forms, which becomes computationally intensive as the number of second-stage uncertainty scenarios increases, rendering the direct application of the policy design procedure in §2 computationally prohibitive. It is also well-known that the expected recourse function and consequently the first-stage objective $f(x)$ are polyhedral for a 2-SLP with finite support (Shapiro et al., 2021). Using this result, a 2-SLP is written in the decomposed form as

$$\begin{aligned} \min_{x \in \mathcal{X}, \eta} \ & c^\top x + \eta \\ \text{s.t. } & \eta \geqslant \alpha_j + \langle \beta_j, x \rangle \text{ for all } j \in \mathcal{J}, \end{aligned} \tag{11}$$

where $\mathcal{J}$ represents a bundle or the indices of pieces in the polyhedral function $\mathbb{E}_{\mathbb{P}}[Q(x,\tilde{\xi})]$. Notice that the above form is also an LP problem; hence, we can apply the PLDC policy (4). However, there are two challenges in applying the process described in §2. Firstly, the classical decomposition methods for solving 2-SLPs, such as the L-Shaped (Van Slyke and Wets, 1969), proximal bundle (Ruszczyński, 1986) and SD (Higle and Sen, 1994) find a local approximation of the polyhedral function. In other words, only a subset of $\mathcal{J}$ is discovered. We compute exact pieces of the polyhedral function in the L-Shaped method, and stochastic approximations of the pieces in the SD algorithm. While the convergence guarantees of the respective algorithms validate such approximations, the policy design procedures associated with a specific solution algorithm must account for the nature of the pieces computed by those algorithms. The second challenge lies in the policy's design steps. Recall that we need the indices of basic variables to construct the cells. Since decomposition algorithms generate local approximations, we can recover different parts of the polyhedral functions at distinct training data points. We have to account for these differences when choosing the basic variable indices that determine how to partition the right-hand sides of the training dataset into distinct cells. To address these challenges, we present tailored approaches for constructing the cells and the policy that account for the specific features of the solution algorithm (L-Shaped or SD) used to obtain the training data.

3.1 Description of Policy Design using L-Shaped

We refer the reader to Van Slyke and Wets (1969); Ruszczynski and Shapiro (2003); Shapiro et al. (2021) for a detailed description of the L-Shaped method and begin by directly discussing the relevant information collected and used from the L-Shaped runs to design the policy.

The L-Shaped method uses an outer linearization-based approximation of the first-stage objective function computed at a sequence of candidate solutions. The function approximation is a pointwise maximum of affine pieces. The slope and intercept of an individual piece are $\alpha = \sum_{\xi\in\Xi} p^{\xi}\langle \pi(\xi), h(\xi)\rangle$ and $\beta = -\sum_{\xi\in\Xi} p^{\xi}(\pi(\xi))^{\top}T(\xi)$, respectively, where p^{ξ} is the probability of scenario ξ and $\pi(\xi)$ is an optimal dual solution of scenario subproblem computed at a candidate solution. Note that the L-Shaped operates under a complete knowledge of the probability distribution $(p^{\xi})_{\xi\in\Xi}$, and therefore, the objective function approximation formed in the algorithm is deterministic. Moreover, using the optimal dual solutions of the subproblem, the affine pieces serve as exact supporting hyperplanes of the expectation-valued objective function along the candidate solution trajectory. If τ denotes the terminal iteration of an L-Shaped run, then the first-stage problem upon termination takes the form in (11), with a bundle $\mathcal{J}^{\tau} \subseteq \mathcal{J}$ of all affine pieces discovered in the τ iterations of the run. The optimal solution of the resultant problem satisfies:

$$\underbrace{\begin{bmatrix} A_B & \mathbf{0} & \mathbf{0}_{|J^{\tau}|} \\ -\boldsymbol{\beta} & \mathbf{1} & -\mathbf{I}_{|J^{\tau}|} \end{bmatrix}}_{:=\mathbf{B}} \begin{bmatrix} x_B^{\star} \\ \eta^{\star} \\ \zeta_B^{\star} \end{bmatrix} = \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}, \tag{12}$$

where A_B refers to the matrix A restricted to columns corresponding to basic variables in

$x^\star$ and $\zeta_B^\star$ are the optimal basic surplus variables corresponding to inequality constraints. We gather the intercept terms into $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{|J^\tau|})^\top$ and slope vectors into $\boldsymbol{\beta} = (-\boldsymbol{\beta}_1^\top; \ldots; -\boldsymbol{\beta}_{|J^\tau|}^\top)$. Notice that the bundle may include pieces generated in all the iterations or a subset of pieces resulting from a bundle management scheme. For instance, if we employ the regularized variant of the L-Shaped method, the bundle may contain only the pieces active at the terminal solution (Ruszczyński, 1986). Nonetheless, our PLDC policy design procedure applies regardless of the bundle management scheme utilized.

3.1.1 Creating Cells

In deterministic LPs, the basis size remains the same even as we vary the right-hand sides. Since we assume that A is of full rank, the resulting bases span the $\mathbb{R}^{m_1}$-dimensional space. In contrast, the terminal first-stage problem may have a different number of cuts in the bundle when 2-SLPs are solved with different right-hand sides. Consequently, the resulting bases may span spaces of different dimensions. Even when the bundle sizes are the same, the cuts within each bundle may differ, introducing distinct surplus variables (corresponding to the cuts included as inequality constraints in the terminal first-stage problem) into the optimal basis. Therefore, it is inappropriate to build cells using the basis in (12).

To address this challenge, we utilize a consolidated master problem. We denote by $\mathcal{A}_i$ the set of active cuts at the terminal solution of the 2-SLP with right-hand side $b = b^i$. We assemble the unique active cuts generated from solving 2-SLPs with n different right-hand sides into $\mathcal{A} = \cup_{i=1}^n \mathcal{A}_i$. Using the set $\mathcal{A}$, we build the consolidated master problem

$$\begin{aligned} \min_{x \in \mathcal{X}, \eta} \; & c^\top x + \eta \\ \text{s.t. } & \eta \geqslant \alpha_j + \langle \beta_j, x \rangle \text{ for all } (\alpha_j, \beta_j) \in \mathcal{A}. \end{aligned} \tag{CM}$$

Notice that $f(x) \geqslant \max_{(\alpha_j, \beta_j) \in \mathcal{A}} \{\alpha_j + \langle \beta_j, x \rangle\}$ for all $x \in \mathcal{X}$. Therefore, solving the consolidated master at $b^i \in \widehat{\mathcal{B}}$ recovers an optimal solution, although it may differ from the one at termination of the L-Shaped method. Alternatively, we can use the terminal solution $(x^\star(b^i), \eta^\star(b^i))$ to recover the basis of the consolidated master (CM) for all $i \in [n]$. Consequently, we identify the indices of the optimal basic variables and use them to create the collection of cells $\{\mathcal{C}^\ell\}_{\ell=1}^L$.

3.1.2 PLDC Policy

Through the master problem (CM) and associated cells, we modify our PLDC policy (4) for 2-SLP to

$$\begin{aligned} \phi(b) = & \max_{\ell \in [L]} u^\ell \left(\begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} b^\ell \\ \boldsymbol{\alpha} \end{pmatrix} \right) + \begin{pmatrix} x^\star(b^\ell) \\ \eta^\star(b^\ell) \end{pmatrix} + z^\ell \\ & - \max_{\ell \in [L]} v^\ell \left(\begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} b^\ell \\ \boldsymbol{\alpha} \end{pmatrix} \right) + z^\ell. \end{aligned} \tag{13}$$

Algorithm 1 Static Procedure: Decomposition Algorithm-guided Policy for 2-SLP

1: **Input:** $\widehat{\mathcal{B}} = \{b^i\}_{i=1}^n, \widehat{\mathcal{X}} = \{(x^\star(b^i), \eta^\star(b^i))\}_{i=1}^n$, set of cuts $\mathcal{A}$
2: Construct the consolidated master problem (CM) using $\mathcal{A}$
3: For every b^i, find the indices of basic variables, B^i, associated with $(x^\star(b^i), \eta^\star(b^i))$. Let the number of such sets be $L(\leqslant n)$
4: Gather right-hand sides having same basic variables and place them into the same cell, i.e., $\mathcal{C}^\ell = \{b^{i_1}, b^{i_2}, \ldots, b^{i_{m_\ell}}\}$ such that $B^{i_1} = B^{i_2} = \cdots\cdots = B^{i_{m_\ell}}$
5: Pick one right-hand side b^ℓ and corresponding optimal solution $(x^\star(b^\ell), \eta^\star(b^\ell))$ from each cell $\mathcal{C}^\ell$
6: Solve the training problem (14) and get its optimal solution (u^ℓ, v^ℓ, z^ℓ)
7: **Output:** Policy parameters $\{(u^\ell, v^\ell, z^\ell)\}_{\ell=1}^L$ and policy input $\{b^\ell, (x^\star(b^\ell), \eta^\star(b^\ell))\}_{\ell=1}^L$.

The parameters u^ℓ, v^ℓ and z^ℓ of policy (13) are obtained from solving the training problem

$$\min_{u,v,z} \sum_{l=1}^{L} \Big(\sum_{k=1}^{d_x+1} \|u_k^\ell\|_1 + \|v_k^\ell\|_1 \Big) + \sum_{i=1}^{n} \|z^i\|_1 \Big) \tag{14a}$$

$$\text{s.t.} \quad u^{\ell(j)} \left(\begin{pmatrix} b^i \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} b^j \\ \boldsymbol{\alpha} \end{pmatrix} \right) + \begin{pmatrix} x^\star(b^j) \\ \eta^\star(b^j) \end{pmatrix} + z^j$$

$$\leqslant \begin{pmatrix} x^\star(b^i) \\ \eta^\star(b^i) \end{pmatrix} + z^i \text{ for all } i, j \in [n], \tag{14b}$$

$$v^{\ell(j)} \left(\begin{pmatrix} b^i \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} b^j \\ \boldsymbol{\alpha} \end{pmatrix} \right) + z^j \leqslant z^i \text{ for all } i, j \in [n], \tag{14c}$$

where the inequalities are defined componentwise, $u^\ell, v^\ell \in \mathbb{R}^{(d_x+1)\times(m_1+|\mathcal{A}|)}$, and $z^\ell \in \mathbb{R}^{d_x+1}$. The policy $\phi(b)$ returns a first-stage decision x, hereafter referred to as a policy-prescribed solution, and an approximated value η of the polyhedral function.

The updated policy in (13) inherits the properties of (4) identified in the previous section, including the vital property unfolded in Theorem 2.2, i.e., $\phi(b) = (x^\star(b), \eta^\star(b))$ for all $b \in \text{Conv}(\mathcal{C}^\ell)$, $\ell \in [L]$. Moreover, for any $b \in \mathcal{B}$, if $\phi(b) = (\hat{x}(b), \hat{\eta}(b))$, the policy-prescribed solution and the approximate objective function value, if the policy recovers an optimal piece over a cell, the epigraph variable $\hat{\eta}(b)$ returned by the policy is the exact first-stage value at the policy-prescribed solution and is optimal over that cell. When the policy-prescribed solution is only feasible, the corresponding value $\hat{\eta}(b)$ may lie above the polyhedral function. In this case, we must estimate the objective function value as $\max_{(\alpha_j,\beta_j)\in\mathcal{A}}\{\alpha_j + \langle \beta_j, \hat{x}(b)\rangle\}$, which is a lower bound on the true value $f(\hat{x}(b))$. The entire policy construction procedure is summarized in Algorithm 1, which we call a static procedure because it uses a fixed number of right-hand sides.

When the scenario size is large, or the support is a continuous set, we can apply the L-Shaped method to the sample average approximation (SAA) of problem (1). In this case, our PLDC policy corresponds to the SAA problem. Several studies examine the quality of the SAA solution, and a variety of sequential procedures are used to measure it (for example, Bayraksan and Morton (2006, 2011)). These methods can also be used to assess the policy-prescribed SAA solutions. The connection between the PLDC policy of

SAA and the true policy is beyond the scope of this paper and warrants further research. Alternatively to the SAA, one can use the SD method as the solution algorithm to train the PLDC policy, which we describe next.

3.2 Description of Policy Design using Stochastic Decomposition

SD (Higle and Sen, 1994) is an elegant method for solving stochastic programs, even when the support is continuous. We provide a high-level description of SD sufficient to illustrate the policy construction process and refer the reader to Sen and Liu (2016) for a detailed exposition on the recent version.

Like L-shaped, SD is also an outer-linearization approach that maintains a bundle of affine functions to approximate the first-stage expectation-valued objective. SD exhibits three important features that make it suitable for 2-SP problems with either continuous support or a large number of scenarios. First, it incorporates concurrent sampling and optimization steps that operate on an iteratively updated sample-average function, thereby avoiding the computationally intensive approach of solving a sequence of SAA problems. Secondly, SD efficiently re-utilizes previously discovered solution information to compute lower-bounding affine pieces of the sample-average functions. In other words, SD does not solve all past scenario subproblems to optimality; thus, it speeds up each iteration. Moreover, the affine pieces generated by SD are stochastic in nature, and they support the expectation-valued objective function only asymptotically. Finally, SD employs a regularization term in the first-stage approximate problem at every iteration. It allows SD to retain only a finite number of affine pieces in the bundle, further reducing the computational burden. The regularization term uses a sequence of incumbent solutions, and the algorithm is guaranteed to generate a sequence that converges to an optimal solution with probability one. The terminal first-stage approximate problem takes the following form

$$\begin{aligned} \min_{x\in\mathcal{X},\eta} \; & f(x) = c^\top x + \eta + \frac{\varsigma}{2}\|x - \hat{x}^\tau\|^2 \\ \text{s.t. } & \eta \geqslant \hat{\alpha}_j + \langle \hat{\beta}_j, x\rangle \text{ for all } j \in J^\tau, \end{aligned} \tag{M^τ}$$

where $\hat{x}^\tau$ is the incumbent solution that meets the statistical optimality conditions. Due to the stochastic nature of SD, termination is based on a notion of statistical optimality that relies on estimates of primal-dual gap values, among other quantities (see Sen and Liu (2016) for details). Replicating SD runs with different seeds may yield different numbers of observations at termination, even for a 2-SLP with the same right-hand side. Consequently, the coefficients of the affine functions in the bundle J^τ are stochastic. Therefore, we must handle random optimal basis matrices, which poses a challenge for grouping the right-hand sides according to the random basic variables to construct the necessary cells needed for our policy design process.

3.2.1 Creating Cells

For a right-hand side b^i, let $\hat{x}^i$ be the incumbent solution after the termination of SD. The affine functions active at $\hat{x}^i$ are collected into the set $\mathcal{A}_i$. Since SD is invoked independently

for each right-hand side $b \in \widehat{\mathcal{B}}$, the number of second-stage observations used in the sample-average of the objective function can vary in each execution. Consequently, the collection $\mathcal{A} = \cup_{i=1}^{n} \mathcal{A}_i$ does not necessarily provide a valid lower bound on the first-stage objective. Thus, the consolidated master problem CM that we used to design the L-Shaped driven policy using $\mathcal{A}$ is not meaningful with respect to the original problem (1). We address this issue by constructing out-of-sample affine functions at all the identified incumbent solutions $\{\hat{x}^i\}_{i=1}^{n}$. The workflow for generating such affine functions is as follows.

1. Draw a sample $\{\xi^t\}_{t\in[N]}$ independently from the scenarios observed in n solves of 2-SLPs.

2. At an incumbent solution $\hat{x}^i$, solve the scenario subproblem for each $\xi \in \{\xi^t\}_{t\in[N]}$. Using the optimal dual solutions $\{\pi_t^i\}_{t\in[N]}$ of the of scenario subproblem, we construct affine functions with coefficients

$$\alpha_i = \frac{1}{N}\sum_{t=1}^{N}(\pi_t^i)^\top h(\xi^t); \quad \beta_i = -\frac{1}{N}\sum_{t=1}^{N}(\pi_t^i)^\top T(\xi^t). \tag{15}$$

   We refer to them as out-of-sample affine functions.

3. Collect the out-of-sample affine functions into a bundle $\bar{\mathcal{A}}$.

A few comments about the out-of-sample bundle are in order. Firstly, any piece in the bundle $\bar{\mathcal{A}}$ lower bounds an out-of-sample function approximation of the expected second-stage value function. That is

$$\frac{1}{N}\sum_{t=1}^{N} Q(x,\xi_t) \geqslant \alpha_i + \langle \beta_i, x\rangle, \text{ for all } (\alpha_i,\beta_i) \in \bar{\mathcal{A}}. \tag{16}$$

Secondly, the cardinality of $\bar{\mathcal{A}}$ is at most n as some of the affine functions might be repeated. Using the central limit theorem, it is evident that the left-hand side in (16) converges to the expected recourse value, almost surely, and therefore, the piecewise affine function defined as the pointwise maximum of the affine functions in $\bar{\mathcal{A}}$ provides a lower bound in expectation. Moreover, as the number of right-hand sides in $\widehat{\mathcal{B}}$ grows to infinity, bundle $\bar{\mathcal{A}}$ recovers all the affine pieces of polyhedral function $\mathbb{E}[Q(x,\tilde{\xi})]$ over $\mathcal{B}$. Finally, and most importantly for our purposes, we can formulate the consolidated master problem using affine functions over the bundle $\bar{\mathcal{A}}$. Thus, the SD-guided policy is obtained by invoking Algorithm 1 with $\mathcal{A}$ replaced by $\bar{\mathcal{A}}$. Since $\hat{x}^i$ is optimal with probability one, there exists N_i such that out-of-sample function value $c^\top\hat{x}^i + \frac{1}{N}\sum_{t=1}^{N} Q(\hat{x}^i,\xi_t)$ lies in ϵ-neighborhood of $v^\star(b^i)$ (Higle and Sen, 1991, 1994). As a result, the basis obtained using the out-of-sample bundle $\bar{\mathcal{A}}$ with $N \geqslant \max_i\{N_i\}$, and hence the policy provides solutions that are optimal with probability one.

Irrespective of the solution method employed, Algorithm 1 uses a fixed training sample size n to recover seen pieces of the solution trajectory. In practice, however, new right-hand sides may be observed sequentially, raising the question of how to feed the information learned from new solves into the policy and update it. In the next section, we develop a sequential procedure to address this question.

3.3 Sequential Procedure for Policy Design

In this section, we present a sequential procedure to train the PLDC policy. Suppose that a 2-SLP problem has been solved for a certain number of right-hand sides using a decomposition algorithm. These solves provide optimal bases and the high-quality estimates of the first-stage objective function at their respective terminal solutions. The sequential procedure is designed to assess the quality of the policy trained using the information collected from these solves, terminate if the policy-prescribed solutions are of sufficiently high quality, or continue the training process otherwise.

Our procedure performs multiple rounds sequentially, with each round comprising two steps. First, the current policy is evaluated on a new batch of right-hand sides. To perform this evaluation, we implement a decomposition algorithm to compare the first-stage objective value at the policy-prescribed solution with the optimal value. The second step appends new solve information, namely, the right-hand vectors, active cuts, corresponding optimal solutions, and estimated objective function values, to the training data. We incorporate only the information from solves whose solutions are rendered infeasible or suboptimal by the current policy. This choice allows us to maintain a training problem of manageable size by retaining only that information guaranteed to improve the quality of the policy. With the updated training data, the procedure solves the policy-training problem and produces an updated policy. The procedure proceeds to the next round and repeats the two steps above. We present a detailed description of the sequential policy update procedure below.

We start with an initial training data, $\widehat{\mathcal{B}}_0 = \{b^i\}_{i=1}^{n_0}, \widehat{\mathcal{X}}_0 = \{x^\star(b^i)\}_{i=1}^{n_0}$ and $\widehat{\mathcal{V}}_0 = \{v^\star(b^i)\}_{i=1}^{n_0}$. The set of corresponding active cuts is denoted by $\widehat{\mathcal{A}}_0$. If $\widehat{\mathcal{X}}_0$ and $\widehat{\mathcal{V}}_0$ are obtained using the L-Shaped method, $\widehat{\mathcal{A}}_0 = \mathcal{A}_0$; if they are obtained using SD, $\widehat{\mathcal{A}}_0 = \bar{\mathcal{A}}_0$. The static procedure in Algorithm 1 is invoked on this data to arrive at an initial policy ϕ^1. At round $t \geqslant 1$, we generate a batch of right-hand side observations $\widehat{\mathcal{B}}_t = \{b^1, b^2, \ldots, b^{n_t}\}$ independently from those observed in previous rounds. The policy is applied to each of these sampled right-hand sides, and the solution $\phi^t(b^i) = (\hat{x}^t(b^i), \hat{\eta}^t(b^i))$ is recorded. We execute the stage decomposition method (L-Shaped or SD) on each of the chosen right-hand sides and collect the solutions information $\widehat{\mathcal{X}} = \{x^\star(b^i)\}_{i=1}^{n_t}$ and $\widehat{\mathcal{V}} = \{v^\star(b^i)\}_{i=1}^{n_t}$, and the bundle of active cuts $\widehat{\mathcal{A}}_t$. Next, we compute the fraction of policy-prescribed solutions that are feasible to the corresponding 2-SLP instance. For this, we check the feasibility of the policy prescribed solution $\hat{x}^t(b^i)$ up to a tolerance level $\epsilon_{\mathcal{X}}$ (as is common practice in solvers). To assess this, we determine the fraction of infeasible instances as

$$R_{\mathcal{X}}(\phi^t) = 1 - \mathbb{E}_{\tilde{b}}[\mathbb{1}(A\hat{x}^t(\tilde{b}) = \tilde{b} + \epsilon_{\mathcal{X}}, \hat{x}^t(\tilde{b}) \geqslant 0)]. \tag{17}$$

We estimate the quantity in (17) and its $(1-\nu)$-confidence interval using the batch $\widehat{\mathcal{B}}_t$ as:

$$R_{\mathcal{X}}^{n_t}(\phi^t) = \ 1 - \frac{1}{n_t}\sum_{i=1}^{n_t} \mathbb{1}(A\hat{x}(b^i) = b^i + \epsilon_{\mathcal{X}}, \hat{x}^t(b) \geqslant 0), \tag{18a}$$

$$R_{\mathcal{X}}(\phi^t) \in \ \left(0, R_{\mathcal{X}}^{n_t} + z_{\nu/2}\frac{1}{\sqrt{4n_t}}\right). \tag{18b}$$

Whenever the interval length in (18b) is sufficiently small, we can conclude that ϕ^t generates feasible solutions to 2-SLP instances for almost every right-hand side with

high confidence. Hence, at a round t, if n_t is such that the interval length is below a threshold ρ, then we proceed to assessing the quality of the objective function value at the policy-prescribed solution.

We carry out the objective function value assessment by revisiting the nonconvex problem (2). We permit the user-defined errors in the policy-prescribed solution and rewrite problem (2) into the following form

$$\min_{\phi \in \hat{\Phi}_{\mathrm{PL}}} \left\{ p(\phi) := \mathbb{E}_{\tilde{b}}[\mathbb{1}\{f(\hat{x}(\tilde{b})) - v^\star(\tilde{b}) > \epsilon_f\}] \right\}, \tag{19}$$

where $\hat{\Phi}_{\mathrm{PL}} = \{\phi \in \Phi_{\mathrm{PL}} \mid \phi(b) = (\hat{x}(b), \hat{\eta}(b)),\ A\hat{x}(b) = b + \epsilon_{\mathcal{X}},\ \hat{x}(b) \geqslant 0 \text{ for almost every } b \in \mathcal{B}\}$. The value $p(\phi)$ represents the fraction of instances that are ϵ_f-away from their optimal value at the policy-prescribed solution. Since $\phi^\star$ is an element of $\hat{\Phi}_{\mathrm{PL}}$, in which case the expectation in (19) evaluates to zero, we have $p(\phi) \geqslant 0$ for all $\phi \in \hat{\Phi}_{\mathrm{PL}}$. Therefore, at round t, if $p(\phi^t) \leqslant \epsilon$ given that $\phi^t(b)$ is feasible for almost every $b \in \mathcal{B}$, we say that ϕ^t is a good quality policy. However, the value of $p(\phi^t)$ is unknown and we have to resort to sampling-based estimation. When the number of scenarios is manageable, we can compute $f(\hat{x}(b))$ and use the value $v^\star(b)$ returned by the L-Shaped method to obtain an estimate of $p(\phi^t)$ over the batch $\widehat{\mathcal{B}}_t$. For the large scenario size, the function value $f(\hat{x}(b))$ in (19) needs to be estimated. Moreover, in this case SD is our recommended choice of the algorithm and the value $v^\star(b)$ reported by SD is a random quantity. We therefore present the policy evaluation procedure separately for the L-Shaped and the SD.

3.3.1 Evaluating L-Shaped-Guided Policy

At round t, we begin by computing an estimate of (19) using the batch $\widehat{\mathcal{B}}_t$ as

$$p_{m_t} = \frac{1}{m_t} \sum_{j \in \mathcal{I}_{\mathcal{X}}} \mathbb{1}\{f(\hat{x}^t(b^j)) - v^\star(b^j) > \epsilon_f\}, \tag{20}$$

where $m_t = |\mathcal{I}_{\mathcal{X}}|$ and $\mathcal{I}_{\mathcal{X}} \subseteq [n_t]$ indexes instances where the policy-prescribed solution is feasible. Noting that $\mathbb{E}[p_{m_t}] = p(\phi^t)$ and $\mathrm{Var}(p_{m_t}) = \frac{p(\phi^t)(1-p(\phi^t))}{m_t} \leqslant \frac{1}{4m_t}$, we obtain the one-sided confidence interval on $p(\phi^t)$ at level $(1-\mu)$ as

$$p(\phi^t) \in \left(0, p_{m_t} + z_{\mu/2}\frac{1}{\sqrt{4m_t}}\right), \tag{21}$$

We conclude that when the feasible instances at the current policy $\phi^t(\cdot)$ meet the condition $p_{m_t} + z_{\mu/2}\frac{1}{\sqrt{4m_t}} \leqslant \epsilon$, we terminate the sequential procedure.

3.3.2 Evaluating SD-Guided Policy

When we are dealing with 2-SLPs with significantly large or continuous support, computing the gap $f(\hat{x}^t(b)) - v^\star(b)$ exactly, even for a given b, is not possible. For such problems, our proposed solution approach, SD, is terminated based on statistical optimality conditions.

Therefore, we only have an estimate of $v^\star(b)$ computed using a sample encountered during the execution of SD. The function value $f(\hat{x}^t(b))$ can only be estimated using sampling.

To examine the similarities between $f(x^\star(b))$ and $f(\hat{x}^t(b))$ for a given $b \in \widehat{B}_t$, we test the null hypothesis $\mathrm{H}_0(b) : f(x^\star(b)) = f(\hat{x}^t(b))$ against $\mathrm{H}_1(b) : f(x^\star(b)) \neq f(\hat{x}^t(b))$. We estimate $f(x^\star(b))$ and $f(\hat{x}^t(b))$ using the same sample of M observations, $\xi^1, \ldots, \xi^M$, generated independently from the observations used in the SD execution. Employing the common random numbers to estimate $f(x^\star(b))$ and $f(\hat{x}^t(b))$ ensures a low-variance estimation of the gap $f(x^\star(b)) - f(\hat{x}^t(b))$. Notice that the sample used can be the same as that used to generate the out-of-sample affine function in (15), in which case we only have to estimate $f(\hat{x}^t(b))$. If the scenario-specific objective function values are equal for all observations $\xi^1, \ldots, \xi^M$, then we trivially accept the null hypothesis. Otherwise, we compute the sample mean estimate of the gap $f(x^\star(b)) - f(\hat{x}^t(b))$, denoted by $\bar{G}$, and its sample standard deviation S_g. We reject the null hypothesis if the t-statistic, $\sqrt{M}\bar{G}/S_g$, is greater than $t_{M-1,\nu/2}$, where $t_{M-1,\nu/2}$ is the critical value at significance level ν and $M-1$ degrees of freedom. Otherwise, we accept the null hypothesis, which implies that the objective values at the policy-prescribed and SD solutions are not significantly different. Specifically, the difference $f(x^\star(b)) - f(\hat{x}^t(b))$ lies in the interval $\left(\bar{G} - t_{M-1,\nu/2}\frac{S_g}{\sqrt{M}}, \bar{G} + t_{M-1,\nu/2}\frac{S_g}{\sqrt{M}}\right)$ with $100(1-\nu)$ confidence. Thus, we conclude that the policy-prescribed solution is of good quality.

Analogous to p_{m_t} defined in section 3.3.1, we denote by $\hat{r} = \mathbb{E}_{\tilde{b}}[\mathbb{1}(\mathrm{H}_0(\tilde{b}) \text{ is rejected})]$, the fraction of instances for which the null hypothesis is rejected (the policy-prescribed solution is not acceptable). Its sample mean estimator is computed as

$$r_{m_t} = \frac{1}{m_t} \sum_{j \in \mathcal{I}_\mathcal{X}} \mathbb{1}(\mathrm{H}_0(b^j) \text{ is rejected}), \tag{22}$$

where $\mathcal{I}_\mathcal{X}$ is the set of instances feasible at the policy-prescribed solution. Then the one sided $100(1-\mu)$ confidence interval is given by

$$\left(0, r_{m_t} + z_{\mu/2}\frac{1}{\sqrt{4m_t}}\right), \tag{23}$$

With this, we conclude the evaluation of the policy-prescribed solution.

Using the interval in (18b) to determine the feasibility of the policy-prescribed solutions to the 2-SLP problem, and (21) (for L-Shaped), (23) (for SD) to determine the quality of the objective function values, we decide whether to terminate the sequential procedure or continue refining the policy. Specifically, if the lengths of these intervals fall within user-defined thresholds ($\rho \in (0,1)$ and $\epsilon > 0$), the procedure is terminated and outputs the current policy ϕ^t; otherwise, we append the suboptimal and infeasible instances into the previous training data, $\widehat{\mathcal{B}}_{t-1}, \widehat{\mathcal{X}}_{t-1}$ and add the set of cuts $\widehat{\mathcal{A}}_t$ into the bundle $\widehat{\mathcal{A}}_{t-1}$. The algorithm 1 is implemented on this updated data to get the policy ϕ^{t+1} and completes round t. We move to the next round $t+1$ with a new batch of samples of size $n_{t+1} = \varrho n_t$, where $\varrho > 1$, and repeat the above steps. The pseudocode of our sequential procedure explained above is presented in Algorithm 2.

The inclusion of the optimal instances shows that the policy embeds their optimal bases. Adding suboptimal and infeasible instances to the training data passes their information

Algorithm 2 Sequential Procedure: Decomposition Algorithm-guided Policy for 2-SLPs

1: **Input:** $n_0 \in \mathbb{Z}_+$, $\widehat{\mathcal{B}}_0 = \{b^i\}_{i=1}^{n_0}, \widehat{\mathcal{X}}_0 = \{(x^\star(b^i), \eta^\star(b^i))\}_{i=1}^{n_0}$, $\widehat{\mathcal{V}}_0 = \{v^\star(b^i)\}_{i=1}^{n_0}$, $\widehat{\mathcal{A}}_0$, $\epsilon > 0, \epsilon_{\mathcal{X}} > 0, \epsilon_f > 0$, $t = 1$, $\rho \in (0,1), \varrho > 1, n_1 = \lceil \varrho n_0 \rceil$, $z_{\mu/2}$-score of $100(1-\mu)$ confidence interval
2: **Initial policy:** Construct policy ϕ^1 using algorithm 1 with inputs $\widehat{\mathcal{B}}_0, \widehat{\mathcal{X}}_0$ and $\widehat{\mathcal{A}}_0$
3: Draw a sample $\widehat{\mathcal{B}}_t = \{b^1, b^2, \ldots, b^{n_t}\}$ independently and identically distributed
4: Compute solution $\phi^t(b^i) = (\hat{x}^t(b^i), \hat{\eta}^t(b^i))$ for each $b^i \in \widehat{\mathcal{B}}_t$
5: Calculate fraction of infeasible instances:

$$R_{\mathcal{X}}^{n_t}(\phi^t) = 1 - \frac{1}{n_t}\sum_{i=1}^{n_t} \mathbb{1}(A\hat{x}^t(b^i) = b^i + \epsilon_{\mathcal{X}}, \hat{x}^t(b^i) \geqslant 0) \tag{24}$$

6: **if** $R_{\mathcal{X}}^{n_t} + \frac{z_{\nu/2}}{2\sqrt{n_t}} \leqslant \rho$ **then**
7: Collect indices of feasible $\hat{x}^t(b^i)$ into set $\mathcal{I}_{\mathcal{X}} = \{i_1, i_2, \ldots, i_{m_t}\}$
8: Run L-Shaped (or SD) and obtain $\widehat{\mathcal{X}}_t = \{(x^\star(b^i), \eta^\star(b^i))\}_{i=1}^{n_t}$ and $\widehat{\mathcal{V}}_t = \{v^\star(b^i)\}_{i=1}^{n_t}$ along with bundle of cuts $\widehat{\mathcal{A}}_t$
9: Compute fraction of suboptimal instances: p_{m_t} using (20) for L-Shaped (or r_{m_t} using (22) for SD)
10: **if** $p_{m_t} + \frac{z_{\mu/2}}{2\sqrt{m_t}} \leqslant \epsilon$ $\left(\text{or } r_{m_t} + \frac{z_{\mu/2}}{2\sqrt{m_t}} \leqslant \epsilon\right)$ **then**
11: output ϕ^t and terminate the procedure
12: **else**
13: go to step 18
14: **end if**
15: **else**
16: go to step 18
17: **end if**
18: Collect indices of infeasible and suboptimal instances into $\mathcal{I}_{\mathcal{X}}^c \subseteq [n_t]$ and $\mathcal{I}_f \subseteq \mathcal{I}_{\mathcal{X}}$ respectively. Reset $\widehat{\mathcal{B}}_t \leftarrow \widehat{\mathcal{B}}_{t-1} \cup \{b^i\}_{i \in \mathcal{I}_{\mathcal{X}}^c \cup \mathcal{I}_f}, \widehat{\mathcal{X}}_t \leftarrow \widehat{\mathcal{X}}_{t-1} \cup \{(x^\star(b^i), \eta^\star(b^i)\}_{i \in \mathcal{I}_{\mathcal{X}}^c \cup \mathcal{I}_f}, \widehat{\mathcal{V}}_t \leftarrow \widehat{\mathcal{V}}_{t-1} \cup \{v^\star(b^i))\}_{i \in \mathcal{I}_{\mathcal{X}}^c \cup \mathcal{I}_f}$ and $\widehat{\mathcal{A}}_t \leftarrow \widehat{\mathcal{A}}_{t-1} \cup \widehat{\mathcal{A}}_t$
19: Invoke algorithm 1 with inputs $\widehat{\mathcal{B}}_t, \widehat{\mathcal{X}}_t, \widehat{\mathcal{A}}_t$
20: Compute policy ϕ^{t+1} using the output, $\{(u^\ell, v^\ell, z^\ell)\}_{\ell=1}^{L_t}$ and $\{b^\ell, (x^\star(b^\ell), \eta^\star(b^\ell))\}_{\ell=1}^{L_t}$, of algorithm 1
21: Set $n_{t+1} = \varrho n_t$, $t \leftarrow t+1$ and go to step 3

to the policy construction, and we include those bases in the policy's computation. This way, we sequentially add the new pieces to the policy, which significantly helps control the size of the training problem (14) and the number of pieces in the policy. It is reasonable to expect that our procedure in Algorithm 2 terminates after a finite number of rounds because the number of pieces in the optimal policy $\phi^\star$ is finite and each piece can be recovered using a finite number of right-hand sides. To formalize this claim, we define the stopping times at which the procedure attains the desired levels of feasibility and optimality as follows.

$$T_{\mathcal{X}}(\rho) = \inf_{t \geqslant 1}\{t \mid R_{\mathcal{X}}^{n_t}(\phi^t) + \frac{z_{\mu/2}}{\sqrt{4n_t}} \leqslant \rho\}, \tag{25a}$$

$$T_f(\epsilon) = \inf_{t \geqslant 1}\{t \mid p_{m_t}(\phi^t) + \frac{z_{\mu/2}}{\sqrt{4m_t}} \leqslant \epsilon\}. \tag{25b}$$

The stopping time $T_{\mathcal{X}}(\rho)$ is the round index at which the average number of infeasible instances with marginal error is below a threshold ρ, for the first time. Analogously, the stopping time $T_f(\epsilon)$ represents the first time where the average number of suboptimal instances with marginal error lies below a threshold ϵ. When Algorithm 2 employs SD, $T_f(\epsilon)$ can be modified by replacing $p_{m_t}(\phi^t)$ by $r_{m_t}(\phi^t)$. With these stopping times, we establish the finite round convergence of Algorithm 2 in Theorem 3.1 and defer its proof to Appendix B.

Theorem 3.1. *(Finite Round Convergence) Let $(\mathcal{B}, \Sigma_b, \mathbb{Q})$ be a probability space, where $\mathbb{Q}(b) > 0$, almost surely. Let the support set $\mathcal{B}$ be decomposed into $\bar{L}$ cells, each corresponding to an optimal basis. In every cell ℓ, suppose there exists a set $\{b^{\ell,i_0}, b^{\ell,i_1}, \ldots, b^{\ell,i_{m_1}}\}$ of $m_1 + 1$ vectors such that $b^{\ell,i_1} - b^{\ell,i_0}, \ldots, b^{\ell,i_{m_1}} - b^{\ell,i_0}$ are linearly independent. Let ϕ^t be a policy at round t of Algorithm 2. Then, the following results hold:*

1. *If Algorithm 2 invokes L-Shaped at each round, then $\mathbb{P}(T_{\mathcal{X}}(\rho) < \infty) = 1$ and $\mathbb{P}(T_f(\epsilon) < \infty) = 1$.*
2. *Suppose SD is deployed at each round of Algorithm 2. If the number of samples N used to construct out-of-sample function approximation (see (16)) is sufficiently large, then $\mathbb{P}(T_{\mathcal{X}}(\rho) < \infty) = 1$ and $\mathbb{P}(T_f(\epsilon) < \infty) = 1$.*

Theorem 3.1 states that the sequential procedure terminates in a finite number of rounds for L-Shaped and SD algorithms. This convergence is guaranteed under the assumption that every cell contains $m_1 + 1$ vectors such that the difference vectors are linearly independent. Such an assumption is crucial for recovering pieces of the optimal policy. The existence of this follows from the Basis Decomposition Theorem of Walkup and Wets (1969). The following remarks clarify this point.

Remark 3.1. From the Basis Decomposition Theorem, the edges of a closed cell $\mathcal{C}$ are the m_1 linearly independent columns of the constraint matrix A. These columns form an optimal basis $\mathbf{B}(\mathcal{C})$ of $\mathcal{C}$. Clearly, $\mathcal{C}$ contains these m_1 linearly independent columns of A. Let these columns be denoted by $a^{i_1}, \ldots, a^{i_{m_1}}$. Pick any $b^{i_0} \in \mathcal{C}$ such that $b^{i_0} \neq a^{i_j}$ for all $j \in [m_1]$. Then, compute $b^{i_0} + a^{i_j} =: b^{i_j}$. It is evident that $b^{i_1} - b^{i_0}, \ldots, b^{i_{m_1}} - b^{i_0}$ are linearly independent. Because b^{i_0} and a^{i_j} lie in a closed convex polyhedral cone $\mathcal{C}$ formed by the nonnegative linear combinations of the columns $a^{i_1}, \ldots, a^{i_{m_1}}$ of A, the sum vector b^{i_j} also belongs to $\mathcal{C}$. In essence, $\mathcal{C}$ contains $b^{i_0}, b^{i_0} + a^{i_1}, \ldots, b^{i_0} + a^{i_{m_1}}$ which in turn certifies the existence assumption stated in Theorem 3.1.

Remark 3.2. The sampling procedure in Algorithm 2 can be adapted so that cell $\mathcal{C}$ contains $b^{i_0}, b^{i_0} + a^{i_1}, \ldots, b^{i_0} + a^{i_{m_1}}$. For instance, suppose the procedure discovers a new basis $\mathbf{B}^{\ell}$ for some b'. We can utilize the corresponding columns of the matrix A to construct the set satisfying the required condition stated in Theorem 2. Note that the number of cells and the corresponding optimal basic variables are not known a priori over the set $\mathcal{B}$. As a result, we cannot construct cells with elements $b^{i_0}, b^{i_0} + a^{i_1}, \ldots, b^{i_0} + a^{i_{m_1}}$ beforehand. The right-hand sides are observed sequentially, and whenever we observe a new basis, we can use its column vectors to incorporate the required set into the cell.

4 Computational Study

In this section, we present the results from our numerical experiments where we apply the PLDC policy obtained from the static procedure in Algorithm 1 and the sequential procedure in Algorithm 2. We conduct these experiments on five well-known 2-SLP problems whose description is provided in Appendix C. All experiments were implemented in Julia. Based on the size of the training problem, we conducted experiments on two machines: (a) 64-bit Intel Core i7-1270P x 16 with 32 GB memory, and (b) AMD EPYC 7763 with 128 CPU cores and approximately 500GB of RAM. We begin by specifying our procedures, including the generation of right-hand sides, the number of perturbed constraints, and the feasibility and optimality tolerances. We then present the detailed results for the test problems.

4.1 Training-data generation and validation

A right-hand-side vector and associated optimal solution from each cell are input to our PLDC policy (13). For instances with only a few first-stage constraints (PGP2, CEP, and 20TERM), we generate input data by perturbing all first-stage right-hand sides. On the other hand, for the large-scale problems 4NODE and STORM, we select three and five constraints, respectively, and perturb their right-hand sides. Time series models are often used to estimate parameters that follow a particular trend, accounting for the effects of explanatory variables. In connection to this, we generate components of perturbed right-hand vectors using univariate linear time series model of order one, i.e., the i-th right-hand side of k-th constraint is computed as $b_k^i = a_{0,k}i + a_{1,k} + e_k^i$, where $a_{0,k}$ and $a_{1,k}$ are the time series parameters and $e_k^i \sim \mathcal{N}(0, \sigma_i^2)$. The parameters $a_{0,k}$ and $a_{1,k}$ are chosen in accordance with the right-hand sides of the perturbed constraints of respective instances. We simulate the time series for T time instants and utilize it in our static and sequential procedures. The Latin Hypercube Sampling (LHS) (McKay, 1992) is the second sampling approach employed in our experiments to generate right-hand sides. In this approach, we divide the range of each component of the right-hand-side vector into T non-overlapping intervals. For each component, one value is randomly chosen from each interval. These T values for each component are then randomly paired to create T vectors of dimension m_1. LHS uniformly covers the entire range of each component and ensures that values are sampled from each interval within that range. This sampling technique achieves good coverage of the sampling space while reducing the variance of the generated right-hand vectors.

For the generated right-hand sides, we solve the 2-SLP problem instances using the L-Shaped (or SD) method. The cuts active at the terminal solution of the L-Shaped method are stored for all right-hand sides. If 2-SLPs are solved with SD, the out-of-sample cuts are calculated and stored. With either collection of cuts, we construct the consolidated master problem (CM) and solve it using Gurobi at each right-hand side. We use the optimal basic feasible solution returned by the solver to construct the cells. The resulting training problem (14) is modeled in JuMP, a Julia modeling package, and solved with the Gurobi solver. A solution to the training problem provides the policy parameters. If a cell has more than one right-hand side, we pick one of the right-hand sides and the

corresponding optimal solution to construct the PLDC policy (13).

The policy-prescribed solution is acceptable when it is feasible and optimal within the user-defined tolerance. In light of this, we calculate the number of 2-SLP instances feasible under the policy-prescribed solutions and identify how many of these feasible instances are optimal. In our computational experiments, we solve a total of T 2-SLP instances and use n of them for training. The policy is deployed on the remaining $T - n$ validation instances to measure its performance. The gray rows in all tables represent the performance summary on validation instances, whereas the non-highlighted rows indicate the performance on training instances. While evaluating feasibility, we set the feasibility tolerance to 10^{-6} on the absolute feasibility gap, the standard criterion for most linear optimization solvers. The optimality of the feasible solution is assessed using the relative optimality gap, with tolerances of 0.0005 and 0.01 for small-scale and large-scale instances, respectively. It should be noted that the policy produces a pair consisting of a first-stage decision $\hat{x}$ and the value of the epigraph variable $\hat{\eta}$ (see the consolidated master problem (CM)). We verify the feasibility of the first-stage constraints at the solution $\hat{x}$. The decision $\hat{\eta}$ need not be a lower bound on the expected recourse function; therefore, we use the value of the polyhedral function at $\hat{x}$ derived from the cuts present in the consolidated master problem.

4.2 Numerical Results

The PLDC policy (13) prescribes an alternative way to obtain the solutions of a 2-SLP problem whenever a change is observed in the first-stage right-hand sides. Observe that policy construction is a one-time computational effort, whereas obtaining solutions whenever a parameter value of a complex 2-SLP changes requires intensive computations. Hence, using the PLDC policy, a functional characterized by sparse parameters is computationally cheaper. With this note, we present our findings from our numerical experiments conducted on 2-SLP problems described in Table 7. We start with a discussion on the policy obtained from the static procedure in Algorithm 1 with the L-Shaped method.

4.2.1 Static Procedure with L-Shaped

For PGP2 and CEP, we solve 800 instances using right-hand sides generated by a linear time-series model and the LHS. We use 80% of them for training and use the remaining 20% for validation. Table 1a illustrates the performance of the policy when right-hand sides are generated using a linear time series model. The non-highlighted rows reveal that the policy retains the optimal solutions of training instances that align with the policy's properties stated in Theorem 2.2. The policy performs well on validation instances, yielding feasible solutions for more than 99% of the problems, all of which are optimal. The number of cells reflects the distinct bases identified by the L-Shaped execution across the 640 training instances. The results show that the cell count is only a small fraction of the total number of instances, indicating that most right-hand sides have common optimal bases. Consequently, this demonstrates that one need not solve a stochastic program from scratch; instead, one can use the designed policy to obtain a high-quality solution for a new instance. The number of active cuts in Table 1a represents the size of the

Instance Name	Size	# of cells	# of active cuts	Feasible instances	Optimal instances	Feasibility gap (Maximum)	Optimality gap (Maximum)
PGP2	640	12	187	100%	100%	10^{-6}	0.0003
	160	–	–	99.3%	100%	9.5	0.0003
CEP	640	3	21	100%	100%	10^{-6}	0.0003
	160	–	–	100%	100%	10^{-6}	0.0003

(a) right-hand sides generated with linear time series model.

Instance Name	Size	# of cells	# of active cuts	Feasible instances	Optimal instances	Feasibility gap (Maximum)	Optimality gap (Maximum)
PGP2	640	72	416	99.5%	100%	3.6×10^{-6}	0.0005
	160	–	–	93.1%	98.6%	3.98	0.006
CEP	640	39	92	100%	100%	10^{-6}	0.0003
	160	–	–	95%	98.6%	231.1	0.0058

(b) right-hand sides generated with LHS.

Table 1: Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.0005.

consolidated set $\mathcal{A}$ and highlights that many instances can share the same optimal pieces.

We find a similar performance for right-hand sides generated via LHS, as evident in Table 1b. However, the percentage of feasible and optimal instances decreases due to the high variability in the right-hand side values. This fact is corroborated by the larger number of cells and active cuts when right-hand sides are generated using LHS, compared to when they are generated using the time-series model. While we should expect $\phi(b^i) = (x^\star(b^i), \eta^\star(b^i))$ for $b^i \in \widehat{\mathcal{B}}$, the training dataset, we observe a slight discrepancy in our results, which can be attributed to numerical approximation errors in solving the training problem. These errors are upto an order $\mathcal{O}$(tolerance level) in some of the results, e.g., the maximum feasibility gap of a training instance of PGP2 in Table 1b is $\mathcal{O}(10^{-6})$.

The runtime of L-Shaped applied to SAA for 4NODE is high because its second-stage subproblems are time-consuming to solve. So, we solve its SAA for 400 different right-hand sides generated from a linear time-series model, and we split the instances into training and validation sets with a $50\% - 50\%$ split. This process is repeated for four SAA replications. Table 2 shows that the policy performs well on 200 validation instances, with over 90% of these being feasible, and all are indeed optimal.

Our computational experience with the 20TERM and STORM instances has been noteworthy, yielding compelling observations. The objective of 20TERM exhibits high variability (Sen and Liu, 2016) and we observe a similar behavior in our numerical results. The active cuts differ significantly across training instances and lead to a considerable gap in the right-hand-side values of the training problem's constraints, as detailed in the Gurobi coefficient statistics report. This variability leads to an ill-conditioned problem, causing the solver to encounter numerical issues and terminate with the model status set to infeasible. However, the solver logs confirm that the training problem has at least one feasible solution. We address this numerical issue by relaxing the constraints of the training problem. We add a nonnegative auxiliary variable to the left-hand side of each constraint and minimize its magnitude in the objective function. For large-scale and unstable instances such as 20TERM and STORM, we solve the relaxed version of the training problem. These observations reveal another interesting phenomenon that encourages the use of the relaxed training problem. Solutions to SP problems with

Rep	Size	# of cells	# of active cuts	Feasible instances	Optimal instances	Feasibility gap (Maximum)	Optimality gap (Maximum)
1	200	8	1387	100%	100%	10^{-6}	0.00037
	200	–	–	90.7%	99.6%	0.54	0.18
2	200	8	1327	100%	100%	10^{-6}	0.00033
	200	–	–	91.5%	100%	1.27	0.00075
3	200	7	1274	100%	100%	10^{-6}	0.00067
	200	–	–	88.3%	100%	0.27	0.00038
4	200	6	1326	100%	100%	10^{-6}	0.00052
	200	–	–	98.9%	100%	0.62	0.00046

Table 2: Instance name: 4NODE, Right-hand sides generated with linear time series model. Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.01.

prohibitively large scenario sizes (or continuous support) are approximate, based on a chosen tolerance, and obtained using a potentially suboptimal basis matrix. Even for right-hand sides in the same cell, different bases may be identified, rendering the training problem infeasible. In these cases, the relaxed form of the training problem can be used to compute the policy efficiently.

We solve 300 instances of 20TERM and STORM and use 80% of them in the policy construction procedure. The size of the training problem for 20TERM and STORM grows rapidly with the number of data points. To demonstrate the policy's effectiveness, we begin with a small training size of 80 and progressively increase it to 240. The policy is computed independently for each training size. We use the same validation instances to compare the policies. The results are reported in Table 3. Given the complexity and scale of 20TERM instance, attaining optimality in 88.9% of the cases is good enough. A similar pattern is observed for STORM, where all feasible instances are optimal, and the maximum absolute feasibility gap is low. More than 99% feasible training instances with training size 240 indicate that the relaxed form of the training problem (14) is reliable for unstable and large-scale problems.

4.2.2 Sequential Procedure with L-Shaped

We analyze the policy obtained from the sequential procedure in Algorithm 2 for PGP2 and CEP. As noted in the previous section, the policy performs well in these instances when the right-hand sides are generated from a linear time-series model. We thus proceed with LHC sampling to generate right-hand sides, as this technique reveals intriguing properties of the sequential procedure.

We generate a pool of $5,000$ right-hand sides using LHC sampling. At every round of the procedure, n_t number of right-hand sides are drawn randomly with replacement from this pool. We start with an initial number of right-hand sides, $n_0 = 2$, and increase it by a factor of 1.1 at every round. We set $z_\mu = 1.96$ and use a tolerance of 10^{-4} for the confidence interval widths to ensure feasibility and optimality. We run the procedure for at least 20 rounds and terminate after 80 rounds. At termination, the policy is tested on a new set of 1000 instances. We report the results in Table 4. The results indicate

Instance Name	Size	# of cells	# of active cuts	Feasible instances	Optimal instances	Feasibility gap (Maximum)	Optimality gap (Maximum)
20TERM	80	29	3276	96.3%	100%	9.7	0.00037
	60	–	–	43.3%	61.5%	22.84	0.044
	160	52	6468	88.1%	99.3%	6.14	0.012
	60	–	–	63.3%	94.7%	52.28	0.053
	240	49	9114	99.2%	100%	1.2×10^{-6}	0.0004
	60	–	–	75%	88.9%	104.3	0.017
STORM	80	38	304	100%	100%	10^{-6}	0.0005
	60	–	–	25%	100%	0.21	0.0016
	160	42	465	99.4%	100%	0.0018	0.0006
	60	–	–	60%	100%	0.29	0.0048
	240	57	657	100%	100%	10^{-6}	0.0006
	60	–	–	63.3%	100%	0.7	0.0019

Table 3: Right-hand sides generated with linear time series model. Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.01.

Instance Name	Size	# of cells	# of active cuts	Feasible instances	Optimal instances	Feasibility gap (Maximum)	Optimality gap (Maximum)
PGP2	250	101	505	100%	100%	10^{-6}	0.0008
	1000	–	–	99% ($\pm$0.6%)	99.6% ($\pm$0.4%)	2.6	0.022
CEP	298	40	127	100%	100%	10^{-6}	0.00032
	1000	–	–	99.3% ($\pm$0.5%)	99.8% ($\pm$0.3%)	55.42	0.0028

Table 4: Right-hand sides generated with LHC sampling. Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.001.

that the policy performs well on unseen problems, with CI widths within 0.6%. The training problems utilize a significantly smaller number of data points than the total number of observations encountered in the procedure. For example, the final training problems for PGP2 and CEP use 250 and 298 instances, respectively, out of a total of $n_T \approx 15,000$ instances. This level of reduction is achieved because we append only the instances rendered infeasible or suboptimal by the policy available at the end of a given round. By doing so, we feed only new bases into the training problem while retaining the bases obtained in previous executions.

We examine the impact on the number of cells, the number of active cuts, the training data size, and the distribution of right-hand sides across cells as the procedure progresses. We present the outcome by plotting each of these quantities against the number of rounds, and it is depicted in Figure 2 for PGP2. The plots show that the fractions of feasible and optimal instances converge to 1 after around 40 rounds (see Figures 2a-2b). It indicates that the policy has recovered most of the optimal pieces over the support set $\mathcal{B}$. This behavior is further corroborated by the saturation of the number of cells after 60 rounds, as illustrated in Figure 2c. The size of the training set is also stabilized after 60 rounds, as illustrated in Figure 2c. We also present the proportion of right-hand sides in each cell (see Figure 2f), which shows that the second cell has the highest number of right-hand sides and most cells have less than 1% of the right-hand sides. Similar results are observed for the CEP instance and are presented in Figure 4 in Appendix E.

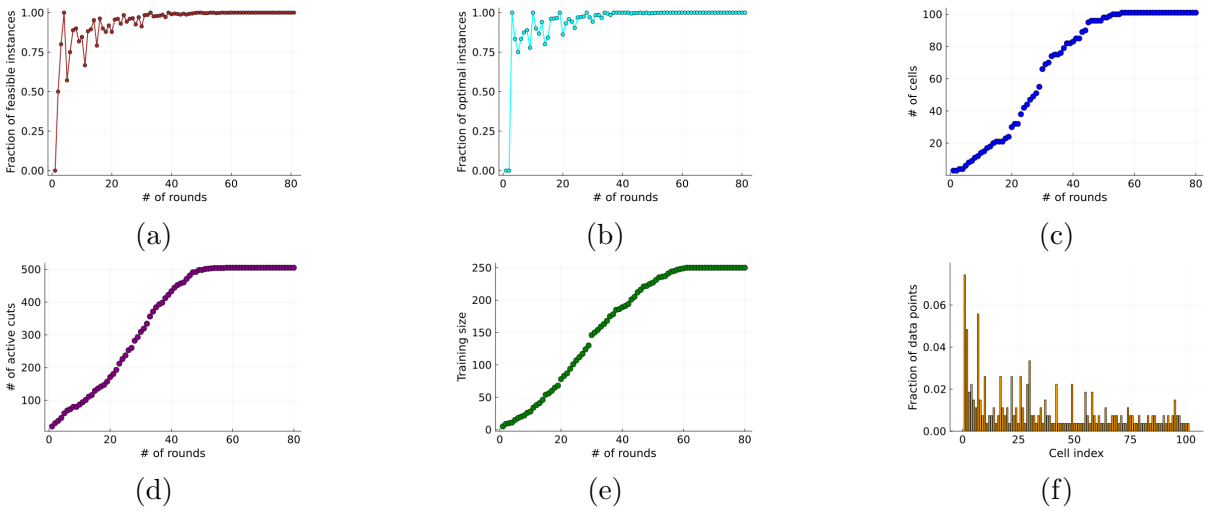

Figure 2: (Color online) Stage decomposition method: L-Shaped, Instance name: PGP2, Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.001, CI tolerance: 10^{-4}.

4.2.3 Performance of Stochastic Decomposition-Guided Policy

In this section, we assess the numerical performance of our PLDC policy when the 2-SLPs are solved using the SD method. We use the same four instances, PGP2, CEP, 20TERM, and STORM, described in Table 7 of the appendix. The right-hand sides for PGP2 and CEP are generated using LHC sampling, while those for 20TERM and STORM are obtained from a linear time-series model, as detailed earlier in this section. The support for second-stage uncertainty is prohibitively large in 20TERM and STORM to recover the exact policy. Therefore, the training problem (14) can be infeasible. Thus, we relax the constraints of (14) by adding slack variables to the constraints and minimizing their absolute values. The number of samples used in out-of-sample approximation is set to 2000 for all the instances. The sample size M for testing the null hypothesis is chosen so that the sample variance of the difference values is below 1%.

Static Procedure We solve all the instances for 500 distinct right-hand sides, out of which we allocate 400 for the training and 100 for the policy validation. Table 5 demonstrates that all the training instances of PGP2 and CEP are feasible at the policy-prescribed solutions, and the null hypothesis is accepted for all of them. More than 90% of the validation instances are feasible, and the null hypothesis is accepted for more than 80% of the feasible instances. We report the maximum relative mean difference value $\bar{D}$ across the instances in the last column of the Table 5, which is small for all the instances. This outcome shows that the policy delivers out-of-sample objective value comparable to that at the SD solution. For 20TERM, over 95% of training instances are feasible, with 97% feasibility in validation and policy-prescribed solution acceptance in 96.9% of cases. For STORM, 77% of instances are feasible with a very small maximum feasibility gap, and the policy is acceptable for 98.7% of these.

Sequential Procedure Using the experiment set up described in section 4.2.2, we implement the sequential procedure 2 with SD method. Over 98% of the validation instances are feasible at the policy-prescribed solutions with a marginal error of at most 0.5% for both PGP2 and CEP as illustrated in Table 6. The behavior of key quantities, such as the fraction of feasible and optimal validation instances, the number of cells, the

Instance Name	Size	# of cells	# of cuts	Feasible instances	Null hypothesis accepted	Feasibility gap (Maximum)	Relative mean value difference (Maximum)
PGP2	400	36	82	100%	100%	10^{-6}	10^{-14}
	100	–	–	94%	93.6%	6.33	0.079
CEP	400	34	40	100%	100%	10^{-6}	10^{-13}
	100	–	–	93%	83.9%	67.23	0.023
20TERM	400	3	400	97%	97.7%	0.87	0.04
	100	–	–	97%	96.9%	0.80	0.04
STORM	400	35	202	95%	99.7%	0.012	0.0005
	100	–	–	77%	98.7%	0.1	0.0003

Table 5: 2-SLPs are solved with SD, Feasibility tolerance: 10^{-6}, Null hypothesis: Mean value difference is 10^{-8}.

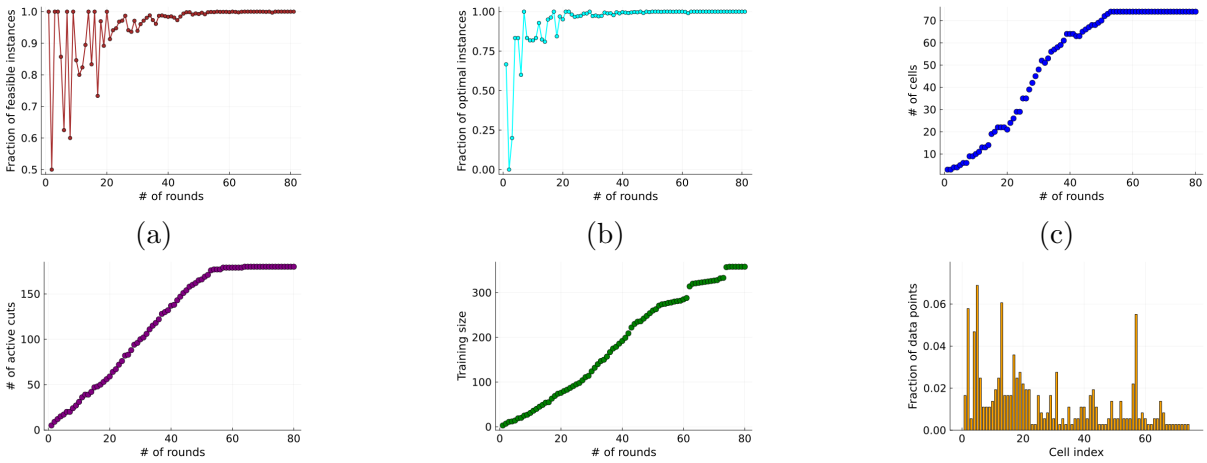

Figure 3: (Color online) Stage Decomposition Method: SD, Instance name: PGP2, Feasibility tolerance: 10^{-6}, Relative mean value difference tolerance: 10^{-8}, CI tolerance: 10^{-4}.

count of cuts in the consolidated master problem, the training size, and the distribution of data points across cells is depicted in Figure 3 (also see Figure 5 for CEP in Appendix F). The policy is unstable in the initial rounds, as evident from the oscillating behavior of the fraction of feasible instances in Figure 3a. However, the optimal instance's behavior exhibits fewer oscillations and stabilizes after 20 rounds because the policy-prescribed solution is acceptable for most of the feasible instances (see Table 6).

Instance Name	Size	# of cells	# of cuts	Feasible instances	Null hypothesis accepted	Feasibility gap (Maximum)	Relative mean value difference (Maximum)
PGP2	358	74	180	100%	100%	10^{-6}	6.8×10^{-9}
	1000	–	–	98.2% (±0.4%)	98.3% (±0.4%)	1.11	0.01
CEP	428	38	50	100%	100%	10^{-6}	0.00032
	1000	–	–	99.4% (±0.2%)	99.5% (±0.2%)	39.79	0.018

Table 6: 2-SLPs are solved with SD, Right-hand sides generated with Latin Hypercube Sampling, Feasibility tolerance: 10^{-6}, Null hypothesis: Mean value difference is 10^{-8}.

5 Conclusion

In this paper, we developed data-driven PLDC policies to solve 2-SLPs with varying right-hand sides for the first-stage constraints. The policies are trained using information collected from the execution of stage decomposition-based solution algorithms on a set of 2-SLP instances. Specifically, these policies characterize the optimal bases derived from previous solves as vector-valued piecewise-linear functions, with the number of pieces equal to the number of unique optimal bases identified. We proposed a training problem that can be solved using off-the-shelf solvers and whose optimal solution determines the parameters of such piecewise-linear functions.

We presented static procedures for cases in which the training data is collected during execution of the L-shaped and SD methods. These procedures account for differences in solution methods when creating the cells necessary for policy design. We also presented a sequential policy design process that monitors policy quality and allows the addition of new bases information to the policy if needed. The finite-round convergence of this process confirms that our PLDC policy recovers the pieces of the optimal policy over the support set of right-hand-side vectors after solving a finite number of 2-SLP instances. After this round, the policy is guaranteed to deliver a high-quality solution for new instances and, in fact, recovers the optimal solution for right-hand sides that lie within the convex hull of observed cells. The empirical study demonstrates that the policy-prescribed solutions are feasible for a significant number of new instances, and in fact, optimal for these instances. The computational results from the sequential procedure indicate that it efficiently uses the information revealed in new solves to continuously improve the policy's performance.

It is important to note that the effort required to train the policy is one-time, and repeatedly identifying a high-quality solution from the trained policy requires minimal computational effort, as it only involves evaluating a function with sparse parameters. In contrast, solving 2-SLP problems from scratch every time the parameter values change is computationally expensive. Therefore, utilizing the policy will enable us to prescribe solutions within tight time bounds associated with decision-making settings in practice (recall the ED problem in power systems).

While the focus of this paper was on the first-stage parameter, a 2-SLP problem is also parametrized by the second-stage uncertainty parameters. Many practical settings that can be modeled as 2-SLP require adapting their solutions to parameter values that drift over time, and it is natural to ask how to design policies in this context. The policy presented in this paper also provides the necessary foundation for extending it to multi-stage SLP, where the right-hand side at each stage is random and depends on the previous-stage decision. We will undertake these intriguing extensions in our future research.

Acknowledgment

We would like to express our sincere gratitude to the U.S Department of Energy-Office of Science, DE-SC0023361, for the financial support to this research. We are also grateful to

Southern Methodist University for its support and assistance in providing the necessary computational and working facilities throughout the course of this work.

References

Bayraksan, G. and Morton, D. P. (2006). Assessing solution quality in stochastic programs. *Mathematical Programming*, 108(2):495–514.

Bayraksan, G. and Morton, D. P. (2011). A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913.

Bertsimas, D. and Bidkhori, H. (2015). On the performance of affine policies for two-stage adaptive optimization: a geometric perspective. *Mathematical Programming*, 153(2):577–594.

Bertsimas, D. and Goyal, V. (2012). On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical programming*, 134(2):491–531.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena scientific Belmont, MA.

California Independent System Operator (CAISO) (2022). Technical report 2210. `https://www.caiso.com/documents/2210.pdf`. Accessed: February 26, 2026.

Dupačová, J. (1990). Stability and sensitivity-analysis for stochastic programming. *Annals of operations research*, 27(1):115–142.

Dyer, M. and Stougie, L. (2006). Computational complexity of stochastic programming problems. *mathematical programming*, 106:423–432.

Gangammanavar, H., Sen, S., and Zavala, V. M. (2015). Stochastic optimization of sub-hourly economic dispatch with wind energy. *IEEE Transactions on Power Systems*, 31(2):949–959.

Georghiou, A., Kuhn, D., and Wiesemann, W. (2019). The decision rule approach to optimization under uncertainty: methodology and applications. *Computational Management Science*, 16(4):545–576.

Georghiou, A., Wiesemann, W., and Kuhn, D. (2015). Generalized decision rule approximations for stochastic programming via liftings. *Mathematical Programming*, 152:301–338.

Higle, J. L. and Sen, S. (1991). Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research*, 16(3):650–669.

Higle, J. L. and Sen, S. (1994). Finite master programs in regularized stochastic decomposition. *Mathematical Programming*, 67(1):143–168.

Higle, J. L. and Sen, S. (1996). *Stochastic decomposition: a statistical method for large scale stochastic linear programming*, volume 8. Springer Science & Business Media.

Kripfganz, A. and Schulze, R. (1987). Piecewise affine functions as a difference of two convex functions. *Optimization*, 18(1):23–29.

Kuhn, D., Wiesemann, W., and Georghiou, A. (2011). Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130:177–209.

Le Thi, H. A. and Pham Dinh, T. (2018). Dc programming and dca: thirty years of developments. *Mathematical Programming*, 169(1):5–68.

McKay, M. D. (1992). Latin hypercube sampling as a tool in uncertainty analysis of computer models. In *Proceedings of the 24th conference on Winter simulation*, pages 557–564.

Robinson, S. M. and Wets, R. J.-B. (1987). Stability in two-stage stochastic programming. *SIAM Journal on Control and Optimization*, 25(6):1409–1416.

Royset, J. O., Chen, L. L., and Eckstrand, E. (2025). Rockafellian relaxation and stochastic optimization under perturbations. *Mathematics of Operations Research*, 50(3):1585–1610.

Royset, J. O. and Lejeune, M. A. (2024). Risk-adaptive local decision rules. *Operations Research*.

Ruszczyński, A. (1986). A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical programming*, 35:309–333.

Ruszczynski, A. and Shapiro, A., editors (2003). *Stochastic Programming*. Handbooks in Operations Research and Management Science, Volume 10. Elsevier, Amsterdam.

Scholtes, S. (2012). *Introduction to piecewise differentiable equations*. Springer.

Sen, S. and Liu, Y. (2016). Mitigating uncertainty via compromise decisions in two-stage stochastic linear programming: Variance reduction. *Operations Research*, 64(6):1422–1437.

Shapiro, A., Dentcheva, D., and Ruszczynski, A. (2021). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shapiro, A. and Nemirovski, A. (2005). On complexity of stochastic programming problems. *Continuous optimization: Current trends and modern applications*, pages 111–146.

Siahkamari, A., Gangrade, A., Kulis, B., and Saligrama, V. (2020). Piecewise linear regression via a difference of convex functions. In *International conference on machine learning*, pages 8895–8904. PMLR.

Van Slyke, R. M. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM journal on applied mathematics*, 17(4):638–663.

Walkup, D. and Wets, R. (1969). Lifting projections of convex polyhedra. *Pacific Journal of Mathematics*, 28(2):465–475.

Wallace, S. W. and Ziemba, W. T. (2005). *Applications of stochastic programming*. SIAM.

Zhang, Y., Liu, J., and Zhao, X. (2023). Data-driven piecewise affine decision rules for stochastic programming with covariate information. *arXiv preprint arXiv:2304.13646*.

Appendix: Omitted Proofs and Numerical Results

A Proof of Lemma 2.1

Proof. Proof. To prove this lemma, it suffices to show the existence of u_k^ℓ and v_k^ℓ that are valid subgradients of $(\phi_1(\cdot))_k$ and $(\phi_2(\cdot))_k$ at all points belonging to the convex hull of $\mathcal{C}^\ell$. We establish this as follows. Using Kripfganz and Schulze (1987), $(\phi_1(b))_k$ can be chosen as a sum of finitely many once-folded PL convex functions. Each once-folded PL convex function has a kink at the intersection of two neighboring cells (see Figure 1 in Kripfganz and Schulze (1987)). It implies that $(\phi_1(b))_k$ restricted to any cell is linear. The second function $(\phi_2(b))_k$ can be set to $(\phi_1(b))_k - (\phi(b))_k$. Clearly, $(\phi_2(b))_k$ restricted to any cell is also linear. Therefore, there exists convex functions $(\phi_1(\cdot))_k$ and $(\phi_2(\cdot))_k$ such that restricting them to $\text{Conv}(\mathcal{C}^\ell)$ are linear functions. Hence, there exists $u_k^\ell \in \partial(\phi_1(b))_k$ and $v_k^\ell \in \partial(\phi_2(b))_k$ for all $b \in \text{Conv}(\mathcal{C}^\ell)$. □

B Proof of Theorem 3.1

Proof. Proof. Consider the collection of $m_1 + 1$ vectors from a cell $\bar{\mathcal{C}}^\ell$, denoted as $\bar{\mathcal{D}}^\ell = \{b^{\ell,i_0}, b^{\ell,i_1}, \ldots, b^{\ell,i_{m_1}}\}$. By assumption, the vectors $b^{\ell,i_1} - b^{\ell,i_0}, \ldots, b^{\ell,i_{m_1}} - b^{\ell,0}$ are linearly independent. The probability of observing each element of $\mathcal{B}$ is almost surely positive, and consequently, the probability of observing all elements of $\bar{\mathcal{D}}^\ell$ is one. Furthermore, the number of pieces in the optimal policy is finite. Therefore, there exists a finite round $t^\star$ with probability one, such that the procedure discovers $\bar{L}$ cells, $\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^{\bar{L}}$, and each contains the set $\bar{\mathcal{D}}^\ell$. It means that, at round $t^\star$, the procedure discovers $\bar{L}$ cells, each containing at least $m_1 + 1$ vectors.

At round t, we have a policy ϕ^t and a batch of right-hand side vectors $\widehat{\mathcal{B}}_t$ of size n_t. For notational simplicity, we first consider the policy structure described in (4) for LPs, i.e, $\phi^t(b) = x^\star(b)$ and prove the optimal piece recovery using $m_1 + 1$ right-hand side vectors. We will exploit it for 2-SLPs, as they have an additional variable η, that is, $\phi^t(b) = (\hat{x}(b), \hat{\eta}(b))$.

The support set $\mathcal{B}$ is decomposed into $\bar{L}$ cells i.e., $\mathcal{B} = \cup_{\ell=1}^{\bar{L}} \bar{\mathcal{C}}^\ell$, where $\mathcal{C}^\ell$ is associated with an optimal basis $\mathbf{B}^\ell$. The ℓ-th piece of the optimal policy is an affine function given by $((\mathbf{B}^\ell)^{-1}b)_k + h_k^{\star,\ell}$ for all $k \in B^\ell$ and is a constant zero function for all $k \in N^\ell$. Our first and most crucial step is to prove that there exists a finite round $t^\star$ such that for all $t \geqslant t^\star$, $(\phi^t(b))_k = ((\mathbf{B}^\ell)^{-1}b)_k + h_k^{\star,\ell}$ for all $k \in B^\ell$ and $(\phi^t(b))_k = 0$ for all $k \in N^\ell$, for all $\ell \in [\bar{L}]$. This result indicates that the PLDC policy ϕ^t has recovered the optimal piece for every cell in $t^\star$ rounds. We prove it next.

According to Lemma 2.1, the policy $\phi^{t^\star}(b)$, when restricted to cell $\mathcal{C}^\ell$, is an affine function. In other words, for all $b' \in \mathcal{C}^\ell$, we have

$$(\phi_1(b'))_k = \langle u_k^\ell, b' - b^\ell \rangle + x^\star(b^\ell)_k + z_k^\ell,$$
$$(\phi_2(b'))_k = \langle v_k^\ell, b' - b^\ell \rangle + z_k^\ell.$$

Using this, we can compute $\phi^{t^\star}(b')$ as follows:

$$\begin{aligned}(\phi^{t^\star}(b'))_k &= (\phi_1(b'))_k - (\phi_2(b'))_k \\ &= \langle u_k^\ell - v_k^\ell, b' \rangle \\ &\quad + \langle u_k^\ell - v_k^\ell, -b^\ell \rangle + x^\star(b^\ell)_k \qquad (26) \\ &=: \langle g_k^\ell, b' \rangle + h_k^\ell. \qquad (27)\end{aligned}$$

After $t^\star$ rounds, we have $\bar{\mathcal{D}}^\ell \subseteq \mathcal{C}^\ell$. Therefore, $\phi^{t^\star}(b^{\ell,i_j}) = x^\star(b^{\ell,i_j})$ for all $j \in \{0\} \cup [m_1]$ from Theorem 2.2. We also have $\phi^\star(b^{\ell,i_j}) = x^\star(b^{\ell,i_j})$ because $\bar{\mathcal{C}}^\ell$ is associated with optimal basis $\mathbf{B}^\ell$. It implies that $\phi^{t^\star}(b^{\ell,i_j}) - \phi^\star(b^{\ell,i_j}) = 0$ for all $b^{\ell,i_j} \in \bar{\mathcal{D}}^\ell$. This yields

$$\langle g_k^\ell - g_k^{\star,\ell}, b^{\ell,i_j} \rangle = h_k^{\star,\ell} - h_k^\ell, \qquad (28)$$
$$\langle g_k^\ell - g_k^{\star,\ell}, b^{\ell,i_0} \rangle = h_k^{\star,\ell} - h_k^\ell. \qquad (29)$$

By utilizing (28) and (29), we obtain $\langle g_k^\ell - g_k^{\star,\ell}, b^{\ell,i_j} - b^{\ell,i_0} \rangle = 0$ for all $j \in [m_1]$. This can be expressed in the matrix form as $(g_k^\ell - g_k^{\star,\ell})^\top Y = \mathbf{0}$, where $Y \in \mathbb{R}^{m_1 \times m_1}$ is an invertible matrix with its j-th column being $b^{\ell,i_j} - b^{\ell,i_0}$. Thus, $(g_k^\ell - g_k^{\star,\ell})^\top = \mathbf{0}Y^{-1} = \mathbf{0}$, indicating that $g_k^\ell = g_k^{\star,\ell}$. Plugging this into (28), we get $h_k^{\star,\ell} = h_k^\ell$, which means the offsets of $\phi^{t^\star}$ and $\phi^\star$ are identical. Therefore, at round $t^\star$, the PLDC policy recovers the optimal piece in each cell, meaning for each ℓ-th cell,

$$(\phi^{t^\star}(b'))_k = ((\mathbf{B}^\ell)^{-1} b')_k + (h^{\star,\ell})_k. \qquad (30)$$

Hence, if our procedure observes right-hand side vectors after $t^\star$ rounds, for which $\mathbf{B}^\ell$ is an optimal basis, the policy prescribes an optimal solution.

Proof of part 1. The cells for the PLDC policy (13) are derived from the (CM) problem. At a given round t, the consolidated set $\mathcal{A}$ comprises all pieces of the polyhedral function that were active at the L-Shaped solutions obtained in previous rounds. Since $\mathcal{D}^\ell \subseteq \mathcal{C}^\ell$ at round $t^\star$, it follows that $\mathcal{A}$ includes all pieces that, when incorporated into (CM), generate optimal bases for all $b \in \mathcal{D}^\ell$. Define $\bar{t} := t^\star + 1$. The policy at this round takes the following form for all $b \in \mathcal{C}^\ell$:

$$\begin{aligned}\phi^{\bar{t}}(b) &= u^\ell \left(\begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} b^\ell \\ \boldsymbol{\alpha} \end{pmatrix} \right) + \begin{pmatrix} x^\star(b^\ell) \\ \eta^\star(b^\ell) \end{pmatrix} + z^\ell \\ &\quad - v^\ell \left(\begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} b^\ell \\ \boldsymbol{\alpha} \end{pmatrix} \right) - z^\ell \\ &= (\hat{u}^\ell - \hat{v}^\ell)(b - b^\ell) + \begin{pmatrix} x^\star(b^\ell) \\ \eta^\star(b^\ell) \end{pmatrix} \\ &= (\hat{u}^\ell - \hat{v}^\ell) b - (\hat{u}^\ell - \hat{v}^\ell) b^\ell + \begin{pmatrix} x^\star(b^\ell) \\ \eta^\star(b^\ell) \end{pmatrix}, \\ &=: \hat{g}^\ell b + \hat{h}^\ell. \qquad (31)\end{aligned}$$

Here, $\hat{u}^\ell$ and $\hat{v}^\ell$ are the matrices of dimension $(d_x+1)\times m_1$, obtained by extracting first m_1 columns of matrices u^ℓ and v^ℓ respectively. Notice that $\phi^{\bar{t}}(b)$ computes the first-stage decision x and epigraph variable η. However, $\phi^\star(b)$ yields values of slack variables as well corresponding to inequality constraints in (CM). Since, we are interested only in (x,η), we focus on the indices of these variables only. From now onward, B^ℓ and N^ℓ denote the basic and nonbasic variables indices in (x,η). The optimal policy $\phi^\star$ over a cell $\mathcal{C}^\ell$ can be expressed as

$$\begin{aligned}(\phi^\star(b))_k &= ((\mathbf{B}^\ell)^{-1}[b;\alpha])_k + \tilde{h}^\ell_k \\ &=: g^{\star,\ell}_k b + h^{\star,\ell}_k,\end{aligned} \tag{32}$$

where $\alpha\in\mathbb{R}^{|\mathcal{A}|}$ and $g^{\star,\ell}_k$ is a row vector of size m_1. From Theorem 2.2 and using the fact that $\bar{t}\geqslant t^\star+1$, we get

$$(\phi^{\bar{t}}(b^{\ell,i_j}))_k - (\phi^\star(b^{\ell,i_j}))_k = 0, \tag{33}$$

for all $b^{\ell,i_j}\in\bar{\mathcal{D}}^\ell, k\in B^\ell\cup N^\ell$. Upon substituting (31) and (32) in (33), we get

$$(\hat{g}^\ell_k - g^{\star,\ell}_k)b^{\ell,i_j} = h^{\star,\ell}_k - \hat{h}^\ell_k,\ \forall\ j\in[m_1], \tag{34}$$

$$(\hat{g}^\ell_k - g^{\star,\ell}_k)b^{\ell,i_0} = h^{\star,\ell}_k - \hat{h}^\ell_k. \tag{35}$$

Combining (34)-(35), we get

$$(\hat{g}^\ell_k - g^{\star,\ell}_k)(b^{\ell,i_j} - b^{\ell,i_0}) = 0\ \forall\ j\in[m_1]. \tag{36}$$

Using previous arguments, row vector $(\hat{g}^\ell_k - g^{\star,\ell}_k)$ must be equal to zero as it is orthogonal to a basis of m_1 dimensional space. Therefore, $\hat{g}^\ell_k = g^{\star,\ell}_k$. Substituting it (34), we obtain $h^{\star,\ell}_k = \hat{h}^\ell_k$. Hence, after $\bar{t}$ rounds, the PLDC policy recovers the optimal piece of every cell.

Therefore, $R^{n_t}_{\mathcal{X}}(\phi^t)=0$ and $p_{m_t}(\phi^t)=0$ almost surely, for all $t\geqslant\bar{t}$. We have a feasibility stopping time $T_{\mathcal{X}}(\epsilon)=\inf_{t\geqslant1}\{t\mid R^{n_t}_{\mathcal{X}}(\phi^t)+\frac{z_{\mu/2}}{\sqrt{4n_t}}\leqslant\epsilon\}$. Then,

$$\begin{aligned}&\mathbb{P}(T_{\mathcal{X}}(\epsilon)=\infty)\\ &\leqslant \lim_{t\to\infty}\mathbb{P}\Big(R^{n_t}_{\mathcal{X}}(\phi^t)+\frac{z_{\mu/2}}{\sqrt{4n_t}}>\epsilon\Big)\\ &= 0.\end{aligned}$$

The last equality holds because $R^{n_t}_{\mathcal{X}}(\phi^t)=0$ for all $t\geqslant\bar{t}$ and $\frac{z_{\mu/2}}{\sqrt{4n_t}}\to0$ as $t\to\infty$. Hence, $\mathbb{P}(T_{\mathcal{X}}(\epsilon)<\infty)=1$. Similarly, consider an optimality stopping time

$$T_f(\epsilon)=\inf_{t\geqslant1}\{t\mid p_{m_t}(\phi^t)+\frac{z_{\mu/2}}{\sqrt{4m_t}}\leqslant\epsilon\}. \tag{37}$$

Using the fact that $p_{m_t}(\phi^t)=0$ almost surely for all $t\geqslant\bar{t}$, we get $\mathbb{P}(T_f(\epsilon)<\infty)=1$. This completes the proof of part 1.

Proof of part 2. Recall that $\widehat{\mathcal{B}}$ is the training data of right-hand side vectors at round t of Algorithm 2. The optimal value of 2-SLP (1) at $b=b^i$, denoted by $f^\star(b^i)$, satisfies $f^\star(b^i)\leqslant c^\top x^\star(b^i)+\mathbb{E}[Q(x^\star(b^i),\tilde{\xi})]$ for all $b^i\in\widehat{\mathcal{B}}$. Since $x^\star(b^i)$ is optimal almost surely, the upper bound is within an ϵ-neighborhood of $f^\star(b^i)$ (see Theorem 2 in (Higle

and Sen, 1991) and Theorem 5 in (Higle and Sen, 1994)). For a given $\epsilon'(<\epsilon)$ and any $b^i \in \widehat{\mathcal{B}}$, there exists a sample size N_i such that the sample variance of $\{Q(x^\star(b^i), \xi^j)\}_{j\in[N_i]}$ is below a threshold ϵ'. Therefore, at a given round t, we can choose $N \geqslant \max_{i\in|\widehat{\mathcal{B}}|}\{N_i\}$ to compute out-of-sample approximation (16). Using the sample size N, the variance of $\{Q(x^\star(b^i), \xi^j)\}_{j\in[N]}$ is less than a threshold ϵ' for all $b^i \in \widehat{\mathcal{B}}$. As a result, the out-of-sample function $c^\top x^\star(b^i) + \frac{1}{N}\sum_{j=1}^{N} Q(x^\star(b^i), \xi^j)$ is a high quality estimate for all $b^i \in \widehat{\mathcal{B}}$, in the sense that out-of-sample evaluation lies in the ϵ-neighborhood of $f^\star(b^i)$. This implies that the affine pieces in $\bar{\mathcal{A}}$ are also of high quality for all right-hand sides belonging to set $\widehat{\mathcal{B}}$. Now using the arguments made earlier, we have $\mathcal{D}^\ell \subseteq \mathcal{C}^\ell$ for all $t \geqslant t^\star$. Therefore, the consolidated problem (CM) constructed with $\bar{\mathcal{A}}$ yields bases for all $b \in \mathcal{D}^\ell$ which are optimal with probability one, i.e, $\phi^t(b) = (x^\star(b), \eta^\star(b))$.

By repeating the steps discussed in part 1 to recover the optimal piece, the PLDC policy $\phi^t(b)$ equals the statistically optimal piece in every cell for all $t \geqslant t^\star + 1 =: \bar{t}_{sd}$. It means after a finite number of rounds $\bar{t}_{sd}$, the policy-prescribed solution $\hat{x}^t(b)$ equals the SD solution $x^\star(b)$ for all $b \in \mathcal{B}$. Hence, the function values $c^\top \hat{x}^t(b) + Q(\hat{x}^t(b), \xi^s)$ and $c^\top x^\star(b) + Q(x^\star(b), \xi^s)$ are equal almost surely for all samples $s \in [M]$ used to calculate the mean estimator of gap $f(x^\star(b)) - f(\hat{x}^t(b))$. As a result, the null hypothesis is accepted with high probability and hence $r_{m_t}(\phi^t) = 0$ for all $t \geqslant \bar{t}_{sd}$ with probability one. Thus, $\mathbb{P}(T_f(\epsilon) = \infty) = 0$. □

C Test Instances

We use instances of five problems, denoted PGP2, CEP, 4NODE, 20TERM, and STORM, for our experiments. The first is a power generation problem, while the second addresses a capacity expansion planning problem. Both instances are small-scale in terms of the first-stage variables and constraints, as well as the number of second-stage scenarios. However, they exhibit enough variability in the optimal bases for different right-hand sides of the first-stage constraints. The instance 4NODE is a freight fleet scheduling problem. Due to its large scenario size, we solve its Sample Average Approximation (SAA) using the L-Shaped method. 20TERM is a freight transportation problem, and STORM is an air freight scheduling problem. We solve the SAA using the L-Shaped for 20TERM and STORM because they have an exponential number of scenarios. 20TERM and STORM are challenging problems because of their larger sets of dual vertices of the scenario subproblems, as noted in Sen and Liu (2016). This results in numerous, vastly different active cuts across the multiple solves, leading to millions of constraints and variables in the training problem. Thus, we determine the policies for 20TERM and STORM on high-performance computing resources. Table 7 lists the relevant characteristics of the test instances. We refer the reader to Higle and Sen (1996) for more details about these problems.

Instance name	1st stage variables/constraints	2nd stage scenario size	# of right-hand side perturbed	Solve SAA (size)
PGP2*	4/2	576	2	✗(–)
CEP*	8/5	216	5	✗(–)
4NODE*	52/14	32,768	3	✓(1000)
20TERM†	63/3	10^{12}	3	✓(500)
STORM†	121/185	10^{81}	5	✓(500)

Table 7: Test instances used in the experiments. $*$: policy computed on ordinary laptop machine, †: policy computed on high-performance computing machine.

D Advantages over standard prediction models

A piecewise linear model can be trained using the training set $\{b^i, x^\star(b^i)\}_{i=1}^n$. For example, Siahkamari et al. (2020) learns the parameters of the PL function by solving a convex problem with linear constraints. If we find these parameters using our training problem (10), we get the policy

$$\begin{aligned}\phi(b)_k = &\max_{j\in[n]}(u_k^j)^\top(b-b^j) + x^\star(b^j)_k + z_k^j \\ &- \max_{j\in[n]}(v_k^j)^\top(b-b^j) + z_k^j,\end{aligned} \tag{38}$$

which exactly resembles the policy developed in Siahkamari et al. (2020). However, policy (38) suffers from overfitting issues as demonstrated in the Table 8. For the mean value problem of pgp2, we determine our PLDC policy (4) and the policy described in (38). It is evident from Table 8 that the PLDC policy performs well and improves as we increase the training size. However, all the validation instances are infeasible at the solution predicted by the policy (38). This behavior is permissible because when the number of data points increases, the parameter size escalates and the training problem has a broad flexibility to choose the parameters, i.e., it can choose one piece per data point, making it prone to overfitting. Therefore, in contrast to machine learning predictions, our goal is to devise the policies that retain the linear pieces associated with each distinct optimal basis found in previous solves. We utilize information from previous solves of decomposition algorithms to enable the policy for prescribing feasible or near-optimal solutions. Our aim is not merely to learn the parameters of a piecewise linear function, but to store previous solve information in a compact form that can be efficiently accessed for future changes to the right-hand side vectors. Neural networks are more sophisticated black-box models widely used for prediction. Likewise, in the policy (38), the training process using a neural network is uninformed of the large number of constraints and the structural properties of the stochastic programs.

Policy	Size	# of cells	Feasible instances	Optimal instances	Feasibility gap (Maximum)	Optimality gap (Maximum)
This work	160	8	100%	100%	10^{-6}	0.0
	40	–	100%	97.5%	10^{-6}	0.002
Eq. 38	160	–	100%	100%	10^{-6}	0.0
	40	–	0.0%	0.0%	1676.48	–
This work	320	7	100%	100%	10^{-6}	0.0
	80	–	98.75%	100%	3.92	0.0
Eq. 38	320	–	100%	100%	10^{-6}	0.0
	80	–	0.0%	0.0%	758.35	–
This work	480	9	100%	100%	10^{-6}	0.0
	120	–	99.17%	100%	0.63	0.0
Eq. 38	480	–	100%	100%	10^{-6}	0.0
	120	–	0.0%	0.0%	1548.42	–

Table 8: Right-hand sides generated uniformly at random. Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.001.

E Results of Sequential Procedure 2 with L-Shaped Method

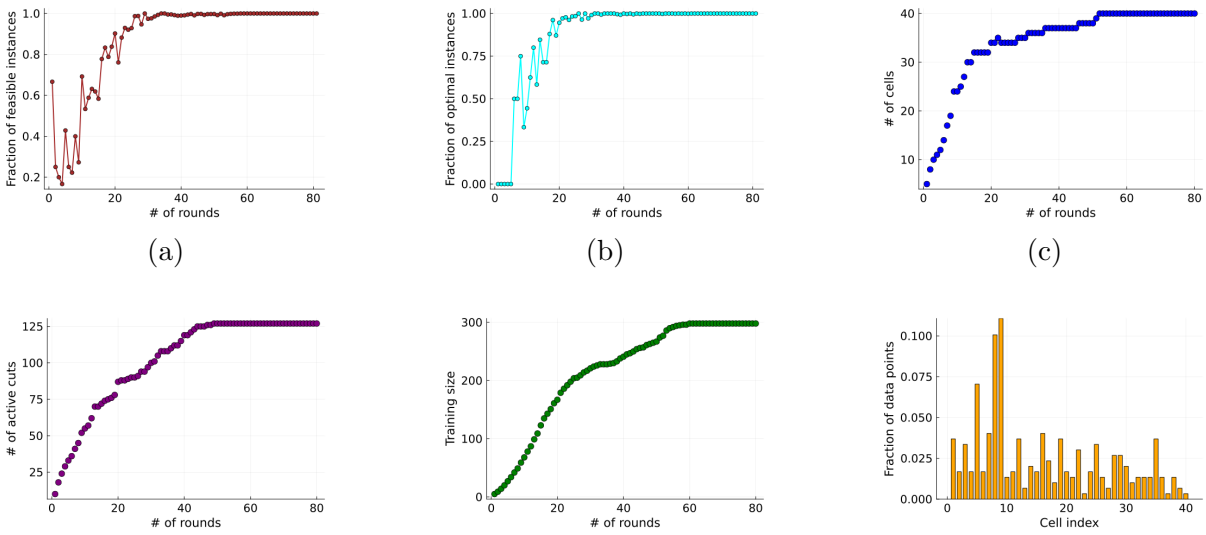

Figure 4: Stage decomposition method: L-Shaped, Instance name: CEP, Feasibility tolerance: 10^{-6}, Optimality tolerance: 0.001, CI tolerance: 10^{-4}.

F Results of Sequential Procedure 2 with Stochastic Decomposition Method

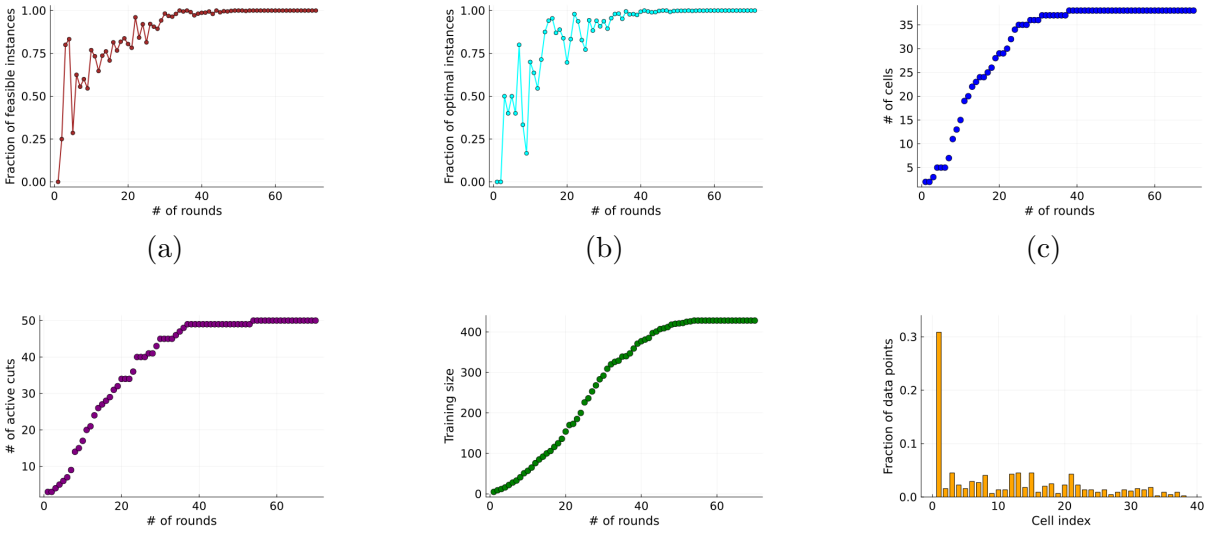

Figure 5: Stage Decomposition Method: SD, Instance name: CEP, Feasibility tolerance: 10^{-6}, Relative mean value difference tolerance: 10^{-8}, CI tolerance: 10^{-4}.