Learning Scenario Reduction for Two-Stage Robust Optimization with Discrete Uncertainty

Tianjue Lin[1] **Jianan Zhou**[1*] **Jieyi Bi**[1]
Yaoxin Wu[2*] **Wen Song**[3] **Zhiguang Cao**[4] **Jie Zhang**[1]

[1]Nanyang Technological University [2]Eindhoven University of Technology
[3]Shandong University [4]Singapore Management University

tianjue002@ntu.edu.sg, jianan004@ntu.edu.sg, jieyi001@ntu.edu.sg,
y.wu2@tue.nl, wensong@email.sdu.edu.cn, zgcao@smu.edu.sg, zhangj@ntu.edu.sg

Abstract

Two-Stage Robust Optimization (2RO) with discrete uncertainty is challenging, often rendering exact solutions prohibitive. Scenario reduction alleviates this issue by selecting a small, representative subset of scenarios to enable tractable computation. However, existing methods are largely *problem-agnostic*, operating solely on the uncertainty set without consulting the feasible region or recourse structure. In this paper, we introduce PRISE, a *problem-driven* sequential lookahead heuristic that constructs reduced scenario sets by evaluating the marginal impact of each scenario. While PRISE yields high-quality scenario subsets, each selection step requires solving multiple subproblems, making it computationally expensive at scale. To address this, we propose NeurPRISE, a neural surrogate model built on a GNN-Transformer backbone that encodes the per-scenario structure via graph convolution and captures cross-scenario interactions through attention. NeurPRISE is trained via imitation learning with a gain-aware ranking objective, which distills marginal gain information from PRISE into a learned scoring function for scenario ranking and selection. Extensive results on three 2RO problems show that NeurPRISE consistently achieves competitive regret relative to comprehensive methods, maintains strong scalability with varying numbers of scenarios, and delivers 17–200$\times$ speedup over PRISE. NeurPRISE also exhibits strong zero-shot generalization, effectively handling instances with larger problem scales (up to $5\times$), more scenarios (up to $4\times$), and distribution shifts.

1 Introduction

Deterministic optimization paradigms typically assume that all problem parameters are precisely known at the time of decision-making. In practice, however, this assumption rarely holds. Parameters are inherently uncertain, often subject to fluctuating customer demand, measurement inaccuracies, or estimation errors. Neglecting these uncertainties can have severe consequences: solutions deemed "optimal" under estimated parameters are often fragile, leading to constraint violations or catastrophic performance degradation when deployed in real-world environments. Robust Optimization (RO) mitigates this risk by seeking solutions that remain feasible and performant under the worst-case parameter realizations within a predefined uncertainty set [1, 3, 9]. Two-Stage Robust Optimization (2RO) extends this paradigm by allowing recourse decisions to be deferred until after the uncertainty is observed [2, 35, 36], effectively balancing proactive planning with adaptive flexibility.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

While most literature focuses on continuous uncertainty models (e.g., box [30], polyhedral [10], or ellipsoidal [1] sets), discrete uncertainty sets are often more representative of real-world settings where decision-makers rely on finite collections of historical scenarios or expert-defined scenarios [19]. However, 2RO under discrete uncertainty, especially when coupled with discrete decision variables, is notoriously difficult. The interplay between first-stage commitments and second-stage recourse renders exact solution methods (e.g., Column-and-Constraint Generation (CCG) [35, 36]) prohibitive as the scenario space scales [26], creating a critical need for efficient solution strategies.

Traditional heuristic approaches, including k-adaptability [4, 21], decision rules [6], and uncertainty set splitting [29], typically gain tractability by imposing structural restrictions on the solution space, often at the expense of optimality. Alternatively, scenario reduction simplifies the problem at the input level: it compresses the uncertainty set into a smaller, representative subset to enable tractable exact solving. However, most existing reduction methods remain *problem-agnostic* [17, 18], operating exclusively on the uncertainty set while ignoring the feasible region and recourse dynamics. While [14] introduce a problem-driven approach for single-stage RO by formulating scenario reduction as a Mixed-Integer Linear Programming (MILP), this formulation does not readily extend to 2RO, as it may break into a much harder nested (e.g., tri-level) optimization problem.

In this paper, we study 2RO under discrete uncertainty. We introduce PRoblem-drIven SEquential lookahead (PRISE), a heuristic that frames scenario reduction as an objective-aware sequential decision process. This framework constructs the reduced scenario set incrementally by evaluating the *marginal gain* associated with each candidate scenario. Specifically, PRISE identifies the scenario that, when integrated into the current partial set, yields the most significant improvement in the approximation of the robust objective. By the *min-max-min* structure of 2RO, the reduced-scenario objective value is *monotonically non-decreasing* as scenarios are added. Empirically, PRISE exhibits strong *diminishing returns*: the most critical worst-case scenarios are identified in the first few iterations, so this incremental construction yields a nested sequence of subsets that is near-optimal across all cardinalities. Our results show that PRISE converges to within 1% of the full-scenario objective using 2–8% of the scenario set, whereas problem-agnostic baselines need at least 2–4$\times$ as many scenarios for equivalent coverage (Appendix B.3).

While PRISE yields high-quality scenario subsets, it becomes computationally prohibitive as the selection horizon expands, as each selection step requires solving the reduced problem for every candidate scenario subset. To address this, we propose NeurPRISE, a neural surrogate model designed to accelerate the selection process. NeurPRISE leverages a GNN-Transformer backbone: a GNN encodes each scenario structure (e.g., MILP) onto a bipartite constraint-variable graph via message passing, while Transformer blocks capture cross-scenario interactions through attention. The model is trained via Imitation Learning (IL) using a gain-aware ranking objective. This objective distills the marginal gain information from PRISE into a learned scoring function, enabling NeurPRISE to rank scenarios by their expected contribution to the robust objective. Empirical results show that this learned policy dramatically accelerates scenario reduction while maintaining solution quality comparable to the original PRISE heuristic. Our contributions are summarized as follows:

- We investigate 2RO within the challenging yet practical regime of discrete uncertainty. To address this, we introduce PRISE, a problem-driven sequential lookahead heuristic for effective scenario reduction. We theoretically prove its monotonicity and provide empirical evidence of its strong diminishing returns.
- To mitigate the computational overhead of PRISE, we propose NeurPRISE, a neural surrogate model that leverages a GNN-Transformer backbone to capture both scenario-level structure and set-level interactions. NeurPRISE is trained via IL using a gain-aware ranking objective, allowing it to rank scenarios by their expected contribution to the robust objective.
- Experiments across three 2RO problem classes (Selection, Vertex Cover, and Capacitated Facility Location) show that NeurPRISE is reliably competitive with classical and ML-based baselines, approaching PRISE-oracle quality at a fraction of its cost. Furthermore, NeurPRISE exhibits strong generalization to larger problem scales, higher scenario counts, and distribution shifts.

2 Related Work

Two-Stage Robust Optimization. Robust optimization [3] seeks solutions that remain feasible and performant under the worst-case realizations of uncertainty. Two-stage robust optimization

generalizes this framework by allowing for recourse after uncertainty is revealed [2], albeit at a significantly higher computational cost. To achieve exact 2RO solutions, CCG [35, 36] provides a framework that avoids the full extensive-form reformulation, which otherwise necessitates a block of recourse variables and constraints for every possible scenario. Instead, CCG iteratively identifies the current worst-case scenario and appends its associated recourse block to the master problem, terminating when the robust objective can no longer be improved. To sidestep the complexity of exact methods, various heuristics have been developed. Decision-rule approaches [5, 6] are widely used for continuous uncertainty, where restricting recourse to a structured function of the uncertainty often yields a tractable reformulation. K-adaptability [4, 21, 31] pre-commits to K recourse plans before uncertainty is revealed, effectively diversifying recourse policies to hedge against both continuous and discrete uncertainty. Recently, ML approaches have emerged to further accelerate these processes. For instance, [7] train a model to predict solution strategies, namely the first-stage decision, the worst-case scenario, and the active constraint set. However, their approach requires continuous recourse and polyhedral uncertainty. [11] propose a data-driven framework that leverages a deep generative model to produce less conservative uncertainty realizations, but their method requires continuous uncertainty for their gradient-based approach. [19] use ML to identify critical initial scenarios for CCG, targeting solver warm-start rather than scenario set reduction. [12] learn a surrogate for the full two-stage objective as a function of the first-stage decision and uncertainty realization, embedding it within an MILP solver to accelerate worst-case search.

Scenario Reduction. Scenario reduction seeks to construct a compact yet representative subset of scenarios to alleviate computational burdens. In stochastic programming, classical approaches focus on minimizing probability distance metrics [13, 22], while more recent methods integrate objective sensitivity and constraint structure to enhance decision quality [8, 23, 24, 27]. In Two-Stage Stochastic Programming (2SP), [33] learn latent representations to facilitate scenario reduction, while [32] select representative scenarios using a first-stage solution consistency reward. However, this reward is tailored to the expected-value objective in 2SP and cannot extend to the worst-case setting of 2RO. Within the realm of robust optimization, the most relevant prior work is [18], who investigates scenario reduction for 2RO by solving a combinatorial selection problem with provable approximation guarantees. However, their formulation is problem-agnostic, relying solely on the geometry of the uncertainty set without exploiting the feasible region or recourse structure. While [14] propose a problem-driven criterion for single-stage RO with objective uncertainty, their criterion is designed for the single-stage setting and does not extend to 2RO, where the objective depends on the recourse decision. Our work bridges this gap by extending the problem-driven paradigm to 2RO through a learning-based approach that directly evaluates the marginal impact of each scenario.

3 Preliminaries

In this paper, we focus on linear two-stage robust optimization over a finite discrete uncertainty set $\Xi = \{\xi_1, \ldots, \xi_S\}$ of size S, supplied by the decision-maker (e.g., derived from historical scenarios). Following the notation of [35], the problem is formulated as follows:

$$\begin{aligned} \min_{x \in X} \quad & c^\top x + \max_{\xi \in \Xi} \min_{y \in F(x,\xi)} b_\xi^\top y, \\ \text{s.t.} \quad & Ax \geq d, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ denote the first- and second-stage decisions; $c \in \mathbb{R}^n$ and $b_\xi \in \mathbb{R}^m$ are cost vectors; and $F(x, \xi) = \{y \in Y : Gy \geq h - Ex - M\xi\}$ is the recourse feasible set. We consider integer first-stage decisions ($X \subseteq \mathbb{Z}_+^n$) with integer or continuous recourse ($Y \subseteq \mathbb{Z}_+^m$ or $\mathbb{R}_+^m$), and assume *relatively complete recourse*: $F(x, \xi) \neq \emptyset$ for all feasible x and $\xi \in \Xi$. The remaining parameters are $A \in \mathbb{R}^{p \times n}$, $d \in \mathbb{R}^p$, $G \in \mathbb{R}^{r \times m}$, $h \in \mathbb{R}^r$, $E \in \mathbb{R}^{r \times n}$, and $M \in \mathbb{R}^{r \times q}$.

Core Notation. Let $Q(x, \xi) := \min_{y \in F(x,\xi)} b_\xi^\top y$ denote the *recourse cost* of first-stage decision x under scenario ξ. For any first-stage decision x, we denote its *full-scenario cost* as

$$Z(x) := c^\top x + \max_{\xi \in \Xi} Q(x, \xi).$$

For any scenario subset $R \subseteq \Xi$, we define the *restricted-scenario objective value* $V(R)$ as follows:

$$V(R) \;:=\; \min_{x \in X} \left[c^\top x \;+\; \max_{\xi \in R} Q(x, \xi) \right], \tag{2}$$

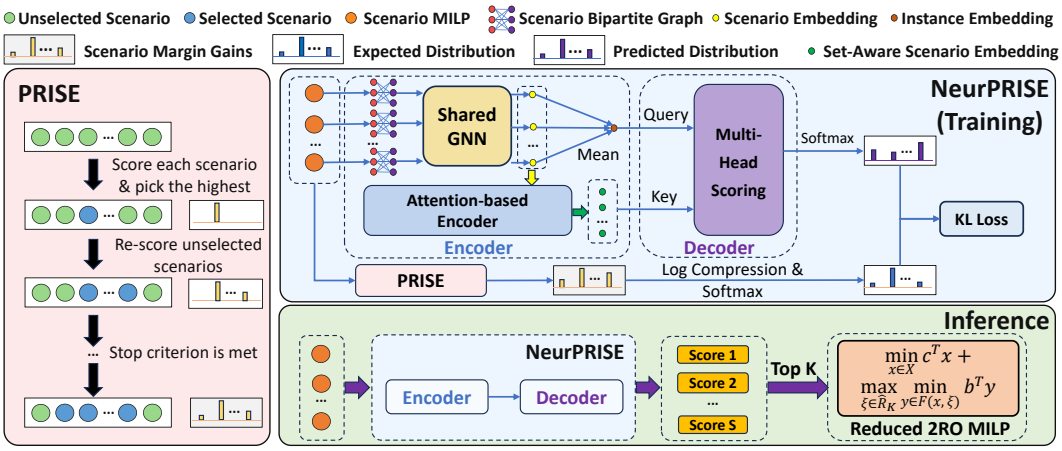


Figure 1: An illustrative overview of our approach. *Left:* PRISE; *Upper Right:* NeurPRISE Framework; *Lower Right:* NeurPRISE Inference.

with the convention $V(\emptyset) := 0$. The value $V(\Xi)$ corresponds to the *full-scenario objective value*. When $R = R^{(k)}$ with $|R^{(k)}| = k$, $V(R^{(k)})$ is the reduced-scenario objective value used to evaluate a k-scenario reduction. To evaluate the quality of a reduced scenario set $R^{(k)}$, let $x^{(k)\star}$ denote an optimal first-stage solution to the reduced problem in Eq. (2). The regret metric measures the suboptimality of $x^{(k)\star}$ relative to the true full-scenario optimum [12]:

$$\text{Regret}(R^{(k)}) \;=\; \frac{Z(x^{(k)\star}) \;-\; V(\Xi)}{V(\Xi)} \times 100. \tag{3}$$

Since $V(\Xi) = \min_{x \in X} Z(x)$, we have $\text{Regret}(R^{(k)}) \geq 0$, with equality if and only if $x^{(k)\star}$ is optimal for the full-scenario problem. In this paper, we treat Ξ as the ground-truth discrete uncertainty set, rather than as samples drawn from an underlying continuous distribution. For finite Ξ, Problem (1) can be reformulated as a deterministic-equivalent MILP whose size scales linearly with $|\Xi|$ (see Appendix C.1), but this formulation becomes computationally expensive when $|\Xi|$ is large.

4 Methodology

In this section, we first introduce PRoblem-drIven SEquential lookahead (PRISE), a heuristic that formulates scenario reduction as an objective-aware sequential decision-making process. We then propose NeurPRISE, a neural surrogate model that alleviates computational overhead while preserving decision quality by learning from PRISE.

4.1 PRISE

PRISE constructs a reduced scenario set $R \subseteq \Xi$ sequentially, scoring each candidate scenario by its marginal impact on the reduced-scenario objective value $V(R)$ (Eq. (2)). At step t, given the selected set R_t (initialized as $R_0 = \emptyset$), PRISE evaluates each candidate $\xi_j \in \Xi \setminus R_t$ by solving the reduced problem on $R_t \cup \{\xi_j\}$ and selects:

$$\xi_t^{\text{PRISE}} \in \arg \max_{\xi_j \in \Xi \setminus R_t} V(R_t \cup \{\xi_j\}). \tag{4}$$

Equivalently, PRISE selects the scenario with the largest marginal gain $V(R_t \cup \{\xi_j\}) - V(R_t)$: the scenario that most increases the reduced robust objective after the first-stage decision is re-optimized. Since evaluating $V(R_t \cup \{\xi_j\})$ entails solving the reduced MILP, the selection is inherently *problem-driven*—it depends on the full problem structure (objective, constraints, recourse), not on the scenario alone. PRISE terminates after selecting a small fraction of $|\Xi|$ scenarios (Appendix B.3). The full procedure is detailed in Algorithm 1. PRISE is characterized by two key properties: 1) *Monotonicity*: $V(R)$ is monotonically non-decreasing as scenarios are added (Proposition 1). 2) *Empirical diminishing returns*: Although $V(R)$ is not submodular in general (Remark 1), marginal

Algorithm 1 PRISE

Input: scenario set $\Xi=\{\xi_1,\ldots,\xi_S\}$ with $S=|\Xi|$; maximum budget K; tolerance $\epsilon\geq 0$
Output: compression budget $\hat{k}$; reduced set R; supervision data $\mathcal{D}$
Initialize $R \leftarrow \emptyset$, $\mathcal{D} \leftarrow \emptyset$, and $v_{\text{prev}} \leftarrow 0$
while $|R| < K$ **do**
  for each unselected scenario $\xi_j \in \Xi \setminus R$ **do**
    $\text{score}(\xi_j) \leftarrow V(R \cup \{\xi_j\})$
  end for
  $\xi^\star \leftarrow$ scenario with highest score
  $\Delta \leftarrow \text{score}(\xi^\star) - v_{\text{prev}}$
  if $\Delta \leq \epsilon$ **then**
    break
  end if
  Add $(R, \xi^\star, \Delta)$ to supervision data $\mathcal{D}$
  Add $\xi^\star$ to the reduced set: $R \leftarrow R \cup \{\xi^\star\}$
  $v_{\text{prev}} \leftarrow \text{score}(\xi^\star)$
end while
$\hat{k} \leftarrow |R|$
return $\hat{k}$, R, $\mathcal{D}$

gains decrease rapidly in practice. This diminishing-returns behavior motivates the loss design in Section 4.2.2. PRISE's iterative structure is similar to the column-and-constraint generation [35, 36]. Appendix B.2 presents the theoretical guarantees for PRISE, while Appendix B.4 clarifies its relationship to column-and-constraint generation.

4.2 NeurPRISE

While PRISE produces high-quality reduced sets, it requires solving $\mathcal{O}(K\cdot|\Xi|)$ MILPs per instance. We amortize this cost through imitation learning: NeurPRISE is a neural surrogate that learns to replicate PRISE's selection in a single forward pass. Formally, we frame scenario reduction as a set-selection task, where our goal is to learn a scoring function that approximates PRISE's marginal-gain ranking. NeurPRISE is envisioned to capture problem-driven selection logic and efficiently predict the scenario set, substantially reducing computational overhead.

4.2.1 Model Architecture

Given a 2RO instance with scenario set Ξ, the encoder represents each scenario subproblem independently and captures cross-scenario dependencies; the decoder then scores each scenario against an instance context. The model outputs logits for all $|\Xi|$ scenarios, enabling flexible top-k selection without retraining. An overview is provided in Figure 1.

Encoder. We model each scenario $\xi_j \in \Xi$ as a bipartite graph G_j. We use the standard constraint–variable bipartite representation [16]. Details on graph construction, GNN layers and feature engineering are given in Appendix D.1 (Figure 3). Each scenario graph is encoded into a d_s-dimensional embedding via GNN followed by global mean pooling over the node embeddings within G_j:

$$\mathbf{h}_j = \text{Pool}\left(\text{GNN}_\phi(G_j)\right) \in \mathbb{R}^{d_s}. \tag{5}$$

To capture inter-scenario dependencies, we refine these embeddings through a Transformer encoder, yielding set-aware scenario embeddings:

$$\mathbf{Z}_{\text{sce}} = \text{TransformerEnc}(\mathbf{h}_{1:|\Xi|}) \in \mathbb{R}^{|\Xi|\times d_s}. \tag{6}$$

Decoder. The decoder produces per-scenario importance scores by relating an instance context to individual scenarios via multi-head scoring. The instance context embedding is obtained by averaging the pre-Transformer GNN embeddings:

$$\mathbf{z}_{\text{inst}} = \frac{1}{|\Xi|}\sum_{j=1}^{|\Xi|}\mathbf{h}_j \in \mathbb{R}^{d_s}. \tag{7}$$

The decoder receives two inputs: 1) *Instance Context* ($\mathbf{z}_{\text{inst}}$)*:* A vector $[1 \times d_s]$ representing the aggregated instance-level structure, which serves as the *Query* (Q). 2) *Set-Aware Scenario Embeddings* ($\mathbf{Z}_{\text{sce}}$)*:* A matrix $[|\Xi| \times d_s]$ representing the encoded features of all $|\Xi|$ scenarios, which serves as the *Key* (denoted $\mathbf{K}_{\text{attn}}$ to distinguish from budget k).

The inputs are projected into H parallel heads with head dimension d_k. For each head $h \in \{1, \ldots, H\}$ and scenario j, we compute a scaled dot-product score:

$$s_{h,j} = \frac{(\mathbf{z}_{\text{inst}} \mathbf{W}_h^Q) \cdot (\mathbf{Z}_{\text{sce},j} \mathbf{W}_h^K)}{\sqrt{d_k}} \in \mathbb{R}, \tag{8}$$

where $\mathbf{W}_h^Q, \mathbf{W}_h^K$ are learnable projection matrices. Each head captures a different aspect of scenario importance relative to the first-stage structure. The H per-head scores for each scenario are combined via a mixing multilayer perceptron (MLP) to produce the final logit:

$$z_j = \text{MLP}([s_{1,j}, \ldots, s_{H,j}]), \quad j \in \{1, \ldots, |\Xi|\}, \tag{9}$$

where the MLP maps $\mathbb{R}^H \to \mathbb{R}$ via a single hidden layer with ReLU activation. In parallel, it produces a vector of logits $\{z_j\}_{j=1}^{|\Xi|}$ for all scenarios, representing the predicted importance of each scenario. We refer to the complete learned pipeline—GNN-Transformer encoder, multi-head scoring decoder, and top-k selection—as **NeurPRISE**. NeurPRISE scores all $|\Xi|$ candidates in a single forward pass.

4.2.2 Loss Function

PRISE assigns each scenario a one-shot importance score: for each selected scenario ξ_{j_t}, the marginal gain is scored by $\Delta_t = V(R_t \cup \{\xi_{j_t}\}) - V(R_t)$ (Algorithm 1, line 8), and unselected scenarios receive a score of zero. We collect these into a target gain vector $g \in \mathbb{R}^{|\Xi|}$ with $g_j = \Delta_t$ if scenario j was selected at step t, and $g_j = 0$ otherwise. Due to the diminishing-returns property (Section 4.1), the gains g_j exhibit a steep decay: the most critical scenario may contribute a gain orders of magnitude larger than later-selected ones (Fig. 2a). Without compression, the softmax target would concentrate nearly all mass on a single scenario. We apply log-compression to flatten this decay:

$$y_j = \log(1 + g_j). \tag{10}$$

We convert these compressed gains into a soft target distribution P over all $|\Xi|$ scenarios via temperature-scaled softmax, and define the predicted distribution $\hat{P}$ from its one-shot logits z_j:

$$P_j = \frac{\exp(y_j/\tau)}{\sum_{\ell=1}^{|\Xi|} \exp(y_\ell/\tau)}, \qquad \hat{P}_j = \frac{\exp(z_j)}{\sum_{\ell=1}^{|\Xi|} \exp(z_\ell)}, \tag{11}$$

where τ is a temperature that flattens the target, ensuring unselected scenarios receive a non-trivial gradient signal. The training objective is:

$$\mathcal{L}(\theta) = D_{\text{KL}}\left(P \,\|\, \hat{P}\right). \tag{12}$$

NeurPRISE produces $\hat{P}$ in a single forward pass over all $|\Xi|$ scenarios. At inference, the user selects the top-k scenarios for any desired budget $k \leq K$, with no step iteration required.

We refer to this objective as the *Score-Weighted KL* (SWKL) loss. The forward direction $D_{\text{KL}}(P\|\hat{P})$ weights the loss by the target P, ensuring every high-gain scenario receives gradient signal rather than only the single best one. SWKL performs consistently across all evaluated problems and instance sizes; a full ablation against BCE and R2R is in Appendix F.2.

5 Experiments

All experiments are conducted on a server with an AMD EPYC 7543 32-Core CPU (2.80 GHz) and NVIDIA A6000 Ada GPUs with 48 GB of VRAM. Optimization problems are solved using Gurobi 12.0.0 [20]. The neural architecture is implemented using PyTorch 2.8.0 [28] and PyG 2.7.0 [15]. All Gurobi calls use 40 threads and MIP gap tolerance 10^{-4} unless otherwise noted. Our implementation is publicly available at `https://github.com/lintianjue/NeurPRISE`.

Table 1: Performance comparison on small-scale instances (250 test instances). A dash (-) denotes a k-agnostic method with identical results across budgets.

Prob.	Method	$k=1$		$k=2$		$k=4$		$k=6$	
		Reg.↓	Time(s)	Reg.↓	Time(s)	Reg.↓	Time(s)	Reg.↓	Time(s)
SEL	Exact	0.00	6.1	-	-	-	-	-	-
	PRISE	5.93	49.8	2.14	91.7	0.92	126.0	0.82	131.7
	Random	22.11	3.8	16.13	3.8	12.29	3.9	9.72	4.1
	SOR	5.45	25.9	4.57	52.9	3.44	63.4	2.83	64.5
	K-means	10.89	25.8	9.02	60.8	7.35	90.5	7.44	110.1
	MaxSum	6.93	4.7	4.50	3.8	2.47	3.9	1.96	4.2
	GK-RF	**4.00**	26.6	2.73	27.8	1.74	28.2	1.16	30.1
	Neur2RO	5.88	2.2	-	-	-	-	-	-
	NeurPRISE	5.59	**2.0**	**2.18**	2.3	**0.79**	2.8	**0.42**	3.4
VC	Exact	0.00	450.8	-	-	-	-	-	-
	PRISE	13.85	82.1	7.91	583.6	3.00	3,008.5	1.04	5,680.2
	Random	16.11	3.8	12.49	13.0	10.58	26.9	9.65	40.6
	SOR	11.15	27.4	8.52	67.9	6.71	92.0	5.63	111.6
	K-means	12.13	25.6	10.88	72.4	9.33	114.5	7.98	149.1
	MaxSum	11.75	4.5	**8.25**	14.2	5.76	31.2	4.40	46.9
	GK-RF	14.43	126.3	9.45	156.0	5.00	199.5	3.29	244.3
	Neur2RO	**11.13**	5.8	-	-	-	-	-	-
	NeurPRISE	14.75	**2.9**	9.36	14.8	**4.20**	38.5	**2.34**	57.8

Problems. Since [18] is the most relevant prior study, we evaluate on its 2RO problem classes: **Selection (SEL)**, which selects a minimum-cost subset of items with uncertain costs; **Vertex Cover (VC)**, which finds a minimum-cost node cover with uncertain node costs. Detailed formulations are in Appendix C. Instances are denoted SEL-n-s and VC-n-s, where n is the number of items/nodes and s the number of scenarios. Moreover, we also evaluate the **Capacitated Facility Location Problem (CFLP)** where uncertainty appears in the constraints. Given space limitations, we defer its formulation, experimental setup, and results to Appendix G.

Instance Generation. We generate 2,500 *small* instances (SEL-20-50, VC-20-50), 500 for *medium* instances (SEL-50-50, VC-50-50), and 500 for *large* instances (SEL-50-100, VC-50-100). Per-node scenario costs are drawn i.i.d. from $\mathrm{Uniform}[1, 100]$, following [18].

Training. For each instance size, we split instances into training, validation, and test sets (80%/10%/10%), yielding 250 test instances for small and 50 each for medium and large. We train with AdamW for at most 200 epochs with early stopping (patience 10). The evaluation pipeline is fully deterministic (top-k scoring). The validation set is used for model selection; we report the checkpoint results with the lowest validation loss. In the main paper, we report results with seed 42. We further examine variance across five training seeds in Section E.2.

Baselines. We compare against three categories of methods. *Reference methods:* **Exact** (Gurobi, full scenario set) provides the exact full-scenario objective value $V(\Xi)$; **PRISE** (our expert heuristic) provides the labeling oracle for NeurPRISE training. *Scenario reduction baselines:* **Random** (k scenarios sampled uniformly); **K-means** [18] (cluster S scenarios into k groups, select the nearest original scenario to each centroid); **MaxSum** [19], which ranks scenarios by the sum of their cost coefficients and retains the top-k; **SOR** [18], which optimally selects k representative scenarios with a provable worst-case approximation guarantee. *Learning-based 2RO:* **GK-RF** [19], a random-forest classifier that relies on comprehensive feature engineering and is trained on scenario-relevance labels from CCG; it retains the top-k scenarios by predicted relevance; **Neur2RO** [12], an end-to-end ML approach that trains a neural surrogate of the two-stage objective and embeds it into a CCG loop.

5.1 Comparison Analysis

We evaluate methods across reduction budgets $k \in \{1, 2, 4, 6\}$, where k denotes the number of scenarios retained, and report mean regret and runtime summed over the test instances. For scenario

Table 2: Medium- and large-scale comparison (50 test instances each). All notation follows Table 1.

Prob.	Scale	Method	$k=1$		$k=2$		$k=4$		$k=6$	
			Reg.↓	Time(s)	Reg.↓	Time(s)	Reg.↓	Time(s)	Reg.↓	Time(s)
SEL	Med.	Exact	0.00	4.8	-	-	-	-	-	-
		PRISE	6.64	12.0	2.11	24.3	0.90	41.2	0.59	45.3
		Random	13.56	0.7	9.74	0.8	8.98	1.2	7.77	1.5
		SOR	8.01	5.3	6.00	16.6	5.37	21.8	4.91	20.2
		MaxSum	6.66	1.5	6.27	0.8	4.24	1.2	3.33	1.5
		GK-RF	4.62	9.2	**3.91**	9.6	2.36	9.8	2.09	10.2
		Neur2RO	5.56	**0.6**	-	**-**	-	**-**	-	**-**
		NeurPRISE	**4.38**	0.7	4.12	0.8	**2.19**	1.2	**1.76**	1.5
	Large	Exact	0.00	9.6	-	-	-	-	-	-
		PRISE	5.80	21.9	2.55	47.7	0.87	95.2	0.49	112.9
		Random	13.54	0.8	10.51	0.8	8.44	1.2	8.88	1.4
		SOR	8.30	20.3	6.32	107.9	6.05	144.3	5.00	169.2
		MaxSum	7.33	0.8	5.60	0.9	4.03	1.3	3.26	1.5
		GK-RF	5.00	13.1	4.03	13.5	**2.53**	13.8	1.75	14.2
		Neur2RO	6.58	**0.7**	-	**-**	-	**-**	-	**-**
		NeurPRISE	**4.18**	0.8	**3.61**	1.0	**2.53**	1.3	**1.62**	1.8
VC	Med.	Exact	0.00	1,690.5	-	-	-	-	-	-
		PRISE	16.35	33.5	9.17	526.5	3.83	2,965.7	2.28	7,594.9
		Random	17.10	2.1	12.51	10.5	10.15	26.9	9.39	48.1
		SOR	13.77	5.9	11.39	29.5	9.06	49.6	8.40	75.7
		MaxSum	13.65	**1.3**	**9.56**	11.3	6.90	31.9	5.20	71.4
		GK-RF	16.57	79.3	9.79	94.1	**6.16**	131.8	4.50	203.7
		Neur2RO	**12.87**	3.1	-	**-**	-	**-**	-	**-**
		NeurPRISE	15.26	**1.3**	9.69	12.6	6.50	40.1	**3.65**	100.5
	Large	Exact	0.00	4,781.7	-	-	-	-	-	-
		PRISE	16.07	63.6	9.20	1,016.5	3.98	6,166.4	2.88	18,713.4
		Random	15.61	**1.3**	12.71	9.3	9.47	26.9	8.69	49.5
		SOR	13.09	19.6	9.52	112.3	7.65	165.6	6.65	218.4
		MaxSum	13.41	**1.3**	**7.99**	9.8	6.91	28.1	5.37	55.6
		GK-RF	18.36	104.2	10.26	118.5	6.91	148.8	**4.82**	207.8
		Neur2RO	**12.85**	2.3	-	**-**	-	**-**	-	**-**
		NeurPRISE	14.92	1.5	9.06	11.9	**6.85**	36.5	4.89	87.1

reduction methods, runtime includes only scenario selection and solving the reduced MILP; loading instances and models is excluded so that it does not affect the comparison. A dash (-) denotes a k-agnostic method with identical results across scenario-reduction budgets. Bold marks the lowest regret and runtime among non-oracle methods in each column (an exact tie is bolded for both). Tables 1 and 2 present results on small, medium and large instances, respectively, where consistent trends hold across all three scales. PRISE attains the lowest regret in most columns with $k\geq 2$, at a high wall-clock cost, as expected for a labeling oracle. NeurPRISE amortizes this cost by learning from PRISE labels, replacing repeated lookahead MILP evaluations with a single scoring pass. Overall, NeurPRISE achieves strong performance relative to non-oracle baselines: it attains the lowest regret in 14 of the 24 (problem, scale, k) columns. On SEL, NeurPRISE achieves the lowest regret in most of the columns. On VC, NeurPRISE is competitive with the leading baselines at $k{=}2$ and is the best, or within a small margin of the best, among all scenario-reduction and learning-based methods when $k\geq 4$. NeurPRISE exhibits runtime comparable to scenario-based methods, while being substantially faster than PRISE, Exact, and GK-RF, whose features require solving a MILP for each scenario online, making it 2–69$\times$ slower than NeurPRISE on VC at medium and large scale.

5.2 Flexibility and Scalability

Flexible scenario budgets. NeurPRISE produces a full scenario ranking, from which reduced sets of any size k can be extracted without retraining. This allows practitioners to flexibly adjust the scenario budget at deployment time without regenerating labels or re-running the training pipeline. Given the specified scenario budget k, the downstream MILP runtime can also be traded by relaxing solver gap tolerances. When $k = 4$ on large VC instances, a 7% gap tolerance yields a $2\times$ speedup while regret remains nearly unchanged. With a more aggressive 25% tolerance, runtime improves by $7\times$ at the cost of only a 3.4% increase in regret, providing a practical quality–speed knob when tight runtime constraints are present (Appendix E.4).

Sub-linear runtime scaling with scenario count. NeurPRISE substantially limits the downstream optimization cost with much fewer scenarios, saving considerable time relative to Exact method. As downstream MILP is solved only on the selected k scenarios, its complexity is governed by k, not the original scenario count S. An ablation on the same instances (Appendix E.5) confirms the speedup effect: on large instances, increasing S by $5\times$ raises total runtime by only $1.2\times$. This sub-linear scaling allows NeurPRISE to remain practical when many candidate scenarios are available at deployment.

5.3 Generalization

Since PRISE labeling cost grows super-linearly with problem size, generalization without retraining is essential. NeurPRISE is particularly suitable for this setting because its learned scoring rule is amortized across instances: once trained, the same model can be reused to rank scenarios under different scenario budgets, larger scenario sets, and shifted deployment distributions without generating new PRISE labels. We evaluate NeurPRISE along three axes:

Problem size (up to $5\times$). NeurPRISE generalizes to problem sizes up to $5\times$ larger than its training scale without retraining, a critical property since PRISE labeling cost at target scale can exceed the training cost. The large test instances scale problem size and scenario count relative to training; the small-scale model still matches or outperforms target-scale models trained directly on those sizes (Table 10, Appendix E.1.2). On ultra-large instances at $5\times$ problem size (Table 3), NeurPRISE achieves the lowest regret on SEL across all budgets and on VC at $k\geq 4$.

Scenario count (up to $4\times$). We demonstrate that the superior regret performance of our method effectively generalizes across substantially different numbers of scenarios. NeurPRISE scales naturally to larger scenario sets because the attention-based encoder processes scenario embeddings independently. Models trained on s=50 transfer to s=100, 200 without retraining (Table 4). NeurPRISE maintains its advantage across scenario counts.

Table 3: Generalization to $5\times$ problem size.

Prob.	Method	Regret (%) ↓			
		k=1	k=2	k=4	k=6
SEL	MaxSum	6.45	5.66	3.47	2.96
	Neur2RO	6.85	-	-	-
	NeurPRISE	**5.21**	**3.26**	**2.12**	**0.81**
VC	MaxSum	14.88	**9.75**	7.06	5.76
	Neur2RO	**14.79**	-	-	-
	NeurPRISE	15.98	10.09	**5.89**	**4.20**

Table 4: Scenario generalization: S=50 to 100, 200.

Pr.	Method	S=100 ↓			S=200 ↓		
		k=2	k=4	k=8	k=4	k=8	k=16
SEL	MaxSum	6.00	3.57	2.44	2.84	1.99	1.23
	Neur2RO	6.35	-	-	5.09	-	-
	NeurPRISE	**3.35**	**1.25**	**0.32**	**1.24**	**0.55**	**0.07**
VC	MaxSum	9.49	5.55	3.44	5.59	3.56	1.88
	Neur2RO	12.29	-	-	10.52	-	-
	NeurPRISE	**8.96**	**5.05**	**1.75**	**4.78**	**2.48**	**0.87**

Distribution shift. We test whether NeurPRISE, trained on uniform scenarios, generalizes to Normal and Multimodal (MM) distributions without retraining (Tables 5–6; setup details in Appendix E.1.1). NeurPRISE$_{\text{uni}}$ outperforms MaxSum and Neur2RO across most settings under both shifted distributions, and performs comparably to in-distribution models (NeurPRISE$_{\text{norm}}$, NeurPRISE$_{\text{mm}}$), confirming that a single uniform-trained model generalizes across distribution families.

Table 5: Distribution shift (Normal).

Prob.	Method	k=1 ↓	k=2 ↓	k=4 ↓	k=6 ↓
SEL	MaxSum	3.09	2.91	2.13	1.62
	Neur2RO	2.96	-	-	-
	NeurPRISE$_{\text{uni}}$	**2.27**	2.18	1.89	1.56
	NeurPRISE$_{\text{norm}}$	**2.27**	**2.10**	**1.70**	**1.35**
VC	MaxSum	4.69	3.93	2.35	0.70
	Neur2RO	4.69	-	-	-
	NeurPRISE$_{\text{uni}}$	**4.56**	**3.71**	2.39	0.89
	NeurPRISE$_{\text{norm}}$	4.67	4.07	**2.23**	**0.62**

Table 6: Distribution shift (MM).

Prob.	Method	k=1 ↓	k=2 ↓	k=4 ↓	k=6 ↓
SEL	MaxSum	1.91	1.50	1.08	0.77
	Neur2RO	1.55	-	-	-
	NeurPRISE$_{\text{uni}}$	1.19	**0.82**	**0.46**	**0.28**
	NeurPRISE$_{\text{mm}}$	**1.17**	0.94	0.58	0.36
VC	MaxSum	7.53	**6.22**	5.19	4.18
	Neur2RO	**7.08**	-	-	-
	NeurPRISE$_{\text{uni}}$	7.78	6.71	**4.66**	3.84
	NeurPRISE$_{\text{mm}}$	7.62	6.56	4.92	**3.82**

6 Conclusion and Limitations

We propose PRISE, a problem-driven sequential lookahead heuristic for scenario reduction in two-stage robust optimization (2RO), and NeurPRISE, a GNN–Transformer trained to imitate PRISE using a gain-aware KL objective. Across three problem classes, NeurPRISE achieves competitive regret and runs substantially faster than PRISE. It also generalizes zero-shot to larger instances, larger scenario pools, and distribution shifts.

We acknowledge certain limitations of our approach, such as the cost of generating PRISE labels (Appendix B.3) and the downstream MILP cost inherent to 2RO with discrete uncertainty, which grows with the size of the reduced scenario set (Appendix E.5). We also propose ways to alleviate both: cross-size generalization (Section 5.3), cross-problem transfer (Appendix E.1.3), and cheaper labels from approximate PRISE solves (Appendix E.6) reduce the offline labeling expense, while relaxed solver tolerances trade a small regret increase for faster solves (Appendix E.4). Future work will continue to investigate cheaper labeling oracles and further reductions of the downstream optimization cost.

Acknowledgments

This research is supported by the National Research Foundation, Singapore under its AI Singapore AI Research Fundamental Research Collaborative (US-NSF Researcher Call) (AISG Award No: AISG3-RP-2025-036-USNSF) and AI Singapore Programme (AISG Award No: AISG3-RP-2022-031). This work is also supported by the National Natural Science Foundation of China (Grant No. 52472461) from the Second Research Institute of the Civil Aviation Administration of China. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, or the other funding agencies. We would like to thank the anonymous reviewers and (S)ACs of NeurIPS 2026 for their constructive comments and service to the community.

References

[1] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88(3):411–424, 2000.

[2] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical programming*, 99(2):351–376, 2004.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[4] Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[5] Dimitris Bertsimas and Angelos Georghiou. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, 63(3):610–627, 2015.

[6] Dimitris Bertsimas and Angelos Georghiou. Binary decision rules for multistage adaptive mixed-integer optimization. *Mathematical Programming*, 167(2):395–433, 2018.

[7] Dimitris Bertsimas and Cheol Woo Kim. A machine learning approach to two-stage adaptive robust optimization. *European Journal of Operational Research*, 319(1):16–30, 2024.

[8] Dimitris Bertsimas and Nishanth Mundru. Optimization-based scenario reduction for data-driven two-stage stochastic optimization. *Operations Research*, 71(4):1343–1361, 2023.

[9] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[10] Dimitris Bertsimas and Aurélie Thiele. Robust and data-driven optimization: modern decision making under uncertainty. In *Models, methods, and applications for innovative decision making*, pages 95–122. INFORMS, 2006.

[11] Aron Brenner, Rahman Khorramfar, Jennifer Z Sun, and Saurabh Amin. A deep generative learning approach for two-stage adaptive robust optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CKXul9iX77.

[12] Justin Dumouchelle, Esther Julien, Jannis Kurtz, and Elias Boutros Khalil. Neur2RO: Neural two-stage robust optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

[13] Jitka Dupačová, Nicole Gröwe-Kuska, and Werner Römisch. Scenario reduction in stochastic programming. *Mathematical programming*, 95(3):493–511, 2003.

[14] Jamie Fairbrother, Marc Goerigk, and Mohammad Khosravi. Problem-driven scenario reduction and scenario approximation for robust optimization. *arXiv preprint arXiv:2410.08863*, 2024.

[15] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/abs/1903.02428. arXiv:1903.02428.

[16] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[17] Marc Goerigk and Martin Hughes. Representative scenario construction and preprocessing for robust combinatorial optimization problems. *Optimization Letters*, 13(6):1417–1431, 2019.

[18] Marc Goerigk and Mohammad Khosravi. Optimal scenario reduction for one-and two-stage robust optimization with discrete uncertainty in the objective. *European Journal of Operational Research*, 310(2): 529–551, 2023.

[19] Marc Goerigk and Jannis Kurtz. Data-driven prediction of relevant scenarios for robust combinatorial optimization. *Computers & Operations Research*, 174:106886, 2025.

[20] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2025. URL https://docs.gurobi.com/projects/optimizer/en/12.0/. Version 12.0.0.

[21] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[22] Holger Heitsch and Werner Römisch. Scenario reduction algorithms in stochastic programming. *Computational optimization and applications*, 24(2):187–206, 2003.

[23] Réne Henrion and Werner Römisch. Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming*, 191(1):183–205, 2022.

[24] Mike Hewitt, Janosch Ortmann, and Walter Rei. Decision-based scenario clustering for decision-making under uncertainty. *Annals of Operations Research*, 315(2):747–771, 2022.

[25] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

[26] Adam Kasperski and Paweł Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness analysis in decision aiding, optimization, and analytics*, pages 113–143. Springer, 2016.

[27] Julien Keutchayan, Janosch Ortmann, and Walter Rei. Problem-driven scenario clustering in stochastic optimization. *Computational Management Science*, 20(1):13, 2023.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[29] Krzysztof Postek and Dick den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574, 2016.

[30] Allen L. Soyster. Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

[31] Anirudh Subramanyam, Chrysanthos E Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, 12(2):193–224, 2020.

[32] Yang Wu, Yifan Zhang, Zhenxing Liang, and Jian Cheng. HGCN2SP: Hierarchical graph convolutional network for two-stage stochastic programming. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 53999–54014. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wu24ag.html.

[33] Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Learning scenario representation for solving two-stage stochastic integer programs. In *International Conference on Learning Representations*, 2022.

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[35] Bo Zeng and Long Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, 2013.

[36] Long Zhao and Bo Zeng. An exact algorithm for two-stage robust optimization with mixed integer recourse problems, 2012.

Appendix Contents

A Notation **14**

B PRISE: Method Details **14**

B.1 Monotonicity of the reduced-scenario objective value 14

B.2 Theoretical Guarantees 15

B.3 PRISE Compression Budget and Convergence Comparison 16

B.4 Connection to Column-and-Constraint Generation 17

C Optimization Problems **17**

C.1 Deterministic-Equivalent MILP Reformulation 17

C.2 Problem Descriptions 18

D NeurPRISE Architecture and Training **18**

D.1 Encoding 18

D.1.1 Bipartite Graph Representation 18

D.1.2 GINE Architecture 18

D.1.3 Problem-Specific Feature Engineering 19

D.2 Training Details 19

E Additional Experimental Results **20**

E.1 Generalization 20

E.1.1 Distribution Shift 20

E.1.2 Do We Need Target-Scale Training? 20

E.1.3 Cross-Problem Transfer 21

E.2 Seed Variance Analysis 21

E.3 Time-Budgeted Exact Solver Comparison 21

E.4 Speed–Quality Tradeoff via Solver Tolerance 22

E.5 Scenario-Count Growth Control 23

E.6 Offline cost reduction: Cheaper Labels 23

E.7 Sensitivity to PRISE Supervision 23

F Ablation Studies **24**

F.1 Encoding and Fusion Strategy 24

F.2 Loss Comparison 24

G Extended Analysis for CFLP **24**

G.1 Problem Formulation 24

G.2 Encoding and Training 25

G.3 Performance and Generalization 26

G.4 Feasibility Analysis 28

A Notation

For quick reference, we summarize the main symbols used throughout the paper. Core notation (Q, V, Regret) is defined in Section 3; the explicit MILP reformulations are in Appendix C.1.

Symbol	Meaning
$x \in X$	first-stage decision (here-and-now)
$y \in F(x, \xi)$	second-stage decision (recourse)
$\xi \in \Xi$	uncertainty realization
$\Xi = \{\xi_1, \ldots, \xi_S\}$	discrete uncertainty set, $\lvert\Xi\rvert = S$
$R \subseteq \Xi$	reduced scenario set
$c,\ b_\xi$	first- / second-stage cost vectors
$Q(x, \xi)$	recourse value, $\min_{y \in F(x,\xi)} b_\xi^\top y$ (Section 3)
$V(R)$	reduced-scenario objective value, $\min_{x \in X} \left[c^\top x + \max_{\xi \in R} Q(x, \xi)\right]$ (Eq. 2)
$V(\Xi)$	full-scenario objective value (ground truth)
$Z(x)$	full-scenario cost, $c^\top x + \max_{\xi \in \Xi} Q(x, \xi)$; $V(\Xi) = \min_{x \in X} Z(x)$
$x^{(k)\star}$	optimal first-stage solution of the reduced problem
$\mathrm{Regret}(R^{(k)})$	$\left(Z(x^{(k)\star}) - V(\Xi)\right) / V(\Xi) \times 100$ (Eq. 3)
K	user-specified scenario budget (max $\lvert R\rvert$)
$\hat{k}$	compression budget, i.e. smallest k with $(V(\Xi) - V(R^{(k)}))/V(\Xi) \leq 1\%$ (Table 7)
R_t	PRISE selected set at step t
Δ_t	PRISE marginal gain at step t
$\mathcal{D}$	PRISE supervision dataset

B PRISE: Method Details

B.1 Monotonicity of the reduced-scenario objective value

We formalize the monotonic behavior of the reduced-scenario objective value $V(R)$, defined in Eq. (2), with respect to the growth of the selected scenario set R.

Proposition 1 (Monotonicity under set inclusion)**.** *For any $R \subseteq R' \subseteq \Xi$, the objective values satisfy*

$$V(R) \ \leq \ V(R') \ \leq \ V(\Xi). \tag{13}$$

Consequently, if $\{R^{(k)}\}_{k \geq 1}$ is a nested sequence with $R^{(k)} \subseteq R^{(k+1)} \subseteq \Xi$ (e.g., the PRISE construction), then $\{V(R^{(k)})\}_{k \geq 1}$ is non-decreasing, and the underestimation gap $V(\Xi) - V(R^{(k)})$ is non-increasing in k.

Proof. Fix any $x \in X$. Since $R \subseteq R'$, the pointwise monotonicity of the maximum gives

$$\max_{\xi \in R} Q(x, \xi) \ \leq \ \max_{\xi \in R'} Q(x, \xi). \tag{14}$$

Adding $c^\top x$ to both sides and taking the minimum over $x \in X$ preserves the inequality, so

$$V(R) \ = \ \min_{x \in X} \left[c^\top x + \max_{\xi \in R} Q(x, \xi)\right] \ \leq \ \min_{x \in X} \left[c^\top x + \max_{\xi \in R'} Q(x, \xi)\right] \ = \ V(R'). \tag{15}$$

Applying the same argument with $R' \subseteq \Xi$ gives $V(R') \leq V(\Xi)$, proving (13). For a nested sequence $\{R^{(k)}\}$, $R^{(k)} \subseteq R^{(k+1)}$ implies $V(R^{(k)}) \leq V(R^{(k+1)})$ by the first inequality in (13). Finally, since $V(\Xi)$ is constant, $V(\Xi) - V(R^{(k)})$ is non-increasing. □

Remark 1 (Non-submodularity of V)**.** *Although PRISE empirically shows diminishing returns (Section 4.1), the set function $V(R)$ in Eq. (2) is* not *submodular in general. Recall that a set function g is submodular if $g(S \cup \{e\}) - g(S) \geq g(T \cup \{e\}) - g(T)$ for all $S \subseteq T$ and $e \notin T$; informally, adding an element to a smaller set yields at least as much marginal gain as adding it to a larger set.*

Counterexample. *Consider a 2RO instance with three feasible first-stage decisions $X = \{a, b, c\}$ and three scenarios $\{s_1, s_2, s_3\}$. Let $f(x, \xi) := c^\top x + Q(x, \xi)$ denote the total cost (first-stage plus recourse) of decision x under scenario ξ:*

$f(x,s)$	s_1	s_2	s_3
a	*9*	*1*	*5*
b	*1*	*9*	*6*
c	*4*	*4*	*8*

Decisions a and b each hedge against one of s_1, s_2 but are exposed to the other. Decision c is balanced across s_1 and s_2 (cost 4 each) but pays the highest cost under s_3.

We can compute $V(R) = \min_{x\in\{a,b,c\}} \max_{s\in R} f(x, s)$ for all subsets:

R	$\max_s f(a,s)$	$\max_s f(b,s)$	$\max_s f(c,s)$	$V(R)$	
$\{s_1\}$	*9*	*1*	*4*	*1*	*(b)*
$\{s_2\}$	*1*	*9*	*4*	*1*	*(a)*
$\{s_3\}$	*5*	*6*	*8*	*5*	*(a)*
$\{s_1, s_2\}$	*9*	*9*	*4*	*4*	*(c)*
$\{s_1, s_3\}$	*9*	*6*	*8*	*6*	*(b)*
$\{s_2, s_3\}$	*5*	*9*	*8*	*5*	*(a)*
$\{s_1, s_2, s_3\}$	*9*	*9*	*8*	*8*	*(c)*

PRISE trace on this instance. *PRISE (Algorithm 1) greedily selects the candidate maximising $V(R_t \cup \{\xi_j\})$:*

- ***Step 0*** *($R_0 = \varnothing$): Evaluate $V(\{s_1\})$=1, $V(\{s_2\})$=1, $V(\{s_3\})$=5. Select s_3 (highest). Gain $\Delta_0 = 5$.*
- ***Step 1*** *($R_1 = \{s_3\}$): Evaluate $V(\{s_3, s_1\})$=6, $V(\{s_3, s_2\})$=5. Select s_1. Gain $\Delta_1 = 6 - 5 = 1$.*
- ***Step 2*** *($R_2 = \{s_3, s_1\}$): Evaluate $V(\{s_3, s_1, s_2\})$=8. Select s_2. Gain $\Delta_2 = 8 - 6 = 2$.*

Note that all marginal gains are strictly positive, yet the gains are non-monotone*: $\Delta_1 = 1 < 2 = \Delta_2$. The greedy trace recovers the full-scenario objective value $V(\{s_1, s_2, s_3\}) = 8$.*

Submodularity violation. *The marginal gain of s_2* increases *from $\Delta(\{s_3\}, s_2) = 0$ to $\Delta(\{s_3, s_1\}, s_2) = 2$ as the base set grows from $\{s_3\} \subset \{s_3, s_1\}$, violating the submodularity condition. Intuitively, when only s_3 is present, decision a achieves $V(\{s_3\}) = 5$ and $V(\{s_3, s_2\}) = 5$—adding s_2 is "free" because a already hedges against it (cost 1). But once s_1 is included, a's high cost under s_1 (cost 9) forces a switch to b, and s_2 now imposes an unavoidable cost increase (since b has cost 9 under s_2, forcing a further switch to c). This interaction between scenarios through the first-stage decision makes V non-submodular.*

B.2 Theoretical Guarantees

PRISE is one-step optimal by construction: at each iteration, it selects a candidate attaining the largest marginal increase in the reduced-scenario objective. However, because V is monotone but not necessarily submodular, this local optimality does not imply the classical global approximation ratio for greedy maximization. Nevertheless, the quality of the resulting decision can be certified a posteriori without assuming submodularity or solving the full 2RO problem to optimality.

For any feasible first-stage decision $x \in X$, define its regret as

$$\mathrm{Regret}(x) := \frac{Z(x) - V(\Xi)}{V(\Xi)} \times 100. \tag{16}$$

Proposition 2 (A posteriori regret bound)**.** *Let $R \subseteq \Xi$, B_R be a valid lower bound on the reduced optimum $V(R)$, and $\bar{x}_R \in X$ be a feasible first-stage decision returned by the reduced solver. Define*

$$U_R := Z(\bar{x}_R) = c^\top \bar{x}_R + \max_{\xi\in\Xi} Q(\bar{x}_R, \xi). \tag{17}$$

Then

$$B_R \le V(\Xi) \le U_R. \tag{18}$$

Table 7: Compression ratio $\hat{k}/|\Xi|$ with budget $K{=}8$; N/C denotes the percentage of instances that did not converge within K steps.

		$\hat{k}/\|\Xi\|$ (%)↓		% instances with $\hat{k}/\|\Xi\| \leq \uparrow$				N/C↓
Problem	Method	Mean	Med.	2%	4%	8%	12%	(%)
SEL-20-50	Random	15.3	16.0	1.2	1.6	5.2	8.0	89.2
	MaxSum	10.9	12.0	10.0	21.6	38.4	51.2	39.6
	PRISE	**4.2**	**4.0**	**34.4**	**79.6**	**95.2**	**96.4**	**3.6**
VC-20-50	Random	15.8	16.0	0.0	0.4	1.2	3.2	94.8
	MaxSum	13.9	16.0	0.4	4.4	13.6	25.2	59.2
	PRISE	**7.8**	**8.0**	**1.2**	**18.8**	**71.6**	**91.6**	**4.4**

If $B_R > 0$, the regret of $\bar{x}_R$ satisfies

$$\text{Regret}(\bar{x}_R) \;\leq\; \frac{U_R - B_R}{B_R} \times 100. \tag{19}$$

Proof. Removing scenarios relaxes the full problem, and therefore $B_R \leq V(R) \leq V(\Xi)$. Moreover, evaluating the feasible decision $\bar{x}_R$ over all scenarios yields $V(\Xi) = \min_{x \in X} Z(x) \leq Z(\bar{x}_R) = U_R$, proving Eq. (18). For $B_R > 0$,

$$\text{Regret}(\bar{x}_R) = \left(\frac{U_R}{V(\Xi)} - 1\right) \times 100 \;\leq\; \left(\frac{U_R}{B_R} - 1\right) \times 100 = \frac{U_R - B_R}{B_R} \times 100, \tag{20}$$

which establishes Eq. (19). □

The certificate remains available when the reduced MILP is terminated early: the solver's certified best bound provides B_R, while any feasible incumbent provides $\bar{x}_R$. Computing U_R requires only fixing $\bar{x}_R$ and evaluating its recourse cost over all scenarios.

This posteriori bound complements the pre-solve, data-dependent bound of Goerigk and Khosravi [18, Corollary 11]. SEL and VC satisfy the required assumptions of nonnegative objective uncertainty and scenario-independent recourse. Hence, once R has been selected, their domination factor $\alpha(R)$ can be computed from the scenario coefficients before solving either the reduced or full 2RO problem, and it yields

$$Z(x^\star_R) \leq \alpha(R) V(\Xi), \qquad \text{Regret}(R) \leq 100 \times [\alpha(R) - 1], \tag{21}$$

where $x^\star_R$ denotes an optimal first-stage solution of the reduced problem $V(R)$ in Eq. (2).

B.3 PRISE Compression Budget and Convergence Comparison

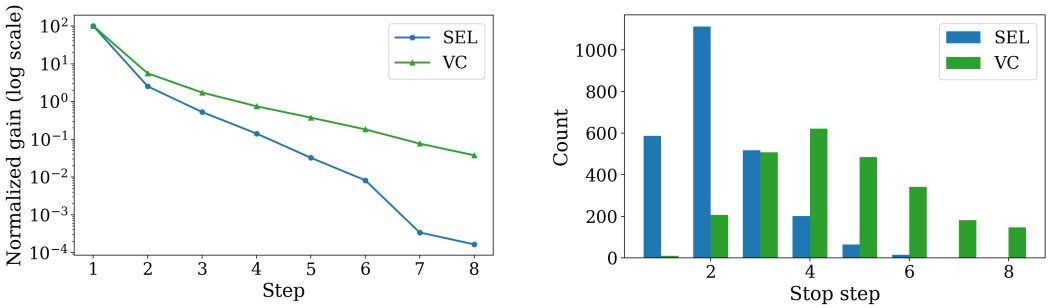
We define the *compression budget* $\hat{k}$ as the minimum number of scenarios needed for a method's selected subset $R^{(k)}$ to reach within 1% of the full-scenario objective value:

$$\hat{k} \;=\; \min\{k : (V(\Xi) - V(R^{(k)}))/V(\Xi) \leq 1\%\}.$$

This is an analysis metric for Table 7, distinct from PRISE's internal stopping criterion (Algorithm 1), which uses marginal gain tolerance ϵ. Reported as the compression ratio $\hat{k}/|\Xi|$, it measures how aggressively a method can shrink the scenario set without sacrificing solution quality.

PRISE compresses scenario sets much more aggressively than baselines. MaxSum and Random require substantially larger scenario budgets to reach comparable worst-case coverage. Gains are empirically non-increasing in 93–99% of consecutive PRISE steps, validating the diminishing-returns assumption underlying the SWKL loss design (Section 4.2.2).

Labeling cost. PRISE is not designed as a fast solver—it is a labeling oracle. At each step, PRISE evaluates the marginal gain of every remaining candidate scenario, each requiring a full MILP solve; this yields $O(\hat{k} \cdot S)$ solves per instance, where $\hat{k}$ is the number of selection steps. Take Table 1 as an example, it reports measured PRISE wall-clock times on 250 test instances using 40-thread Gurobi (same hardware as Section 5). Scaling to the 2,000 training and 250 validation instances gives



(a) Mean marginal gain across iterations.

(b) Distribution of PRISE stop steps.

Figure 2: Empirical analysis of PRISE on small-scale instances. (a) Marginal gains empirically exhibit diminishing returns on average; (b) Distribution of PRISE's stop step.

an estimated labeling cost of approximately 14.5 wall-clock hours for both small-scale problems combined (SEL ≈0.33 h, VC ≈14.2 h). This is a one-time offline cost; once labels are generated, NeurPRISE training and inference require no further PRISE calls. At larger scales, PRISE labeling cost grows substantially, but the generalization results in Section 5.3 show that small-scale labels transfer effectively, avoiding the need for expensive large-scale labeling across the tested problems.

B.4 Connection to Column-and-Constraint Generation

PRISE's iterative scenario-addition structure is motivated by column-and-constraint generation (CCG) [35, 36], a standard exact method for 2RO. Both methods incrementally expand a scenario subset, but they differ fundamentally in their scenario-selection rule and algorithmic purpose:

- **CCG** solves the restricted master problem $V(\widehat{R})$ to obtain a first-stage solution $x^\star(\widehat{R})$, then adds the scenario that is worst for this *fixed* solution: $\xi^{\mathrm{CCG}} \in \arg\max_{\xi\in\Xi} Q(x^\star(\widehat{R}),\xi)$. This adversarial separation step is designed to close the optimality gap and certify convergence to the exact full-scenario value $V(\Xi)$.
- **PRISE** instead *re-solves* the reduced problem on $R_t \cup \{\xi_j\}$ and selects the scenario with the largest marginal gain $V(R_t \cup \{\xi_j\}) - V(R_t)$. This criterion measures the scenario's post-reoptimization impact, rather than its adversariality against a fixed incumbent solution.

The key distinction is therefore in the next-scenario selection logic: CCG asks "which scenario is worst for the current solution?" whereas PRISE asks "which scenario pushes the reduced objective furthest after re-optimization?" CCG is an exact procedure that converges to $V(\Xi)$, while PRISE is a heuristic that builds a compact reduced set for downstream use.

C Optimization Problems

C.1 Deterministic-Equivalent MILP Reformulation

The deterministic-equivalent MILP reformulation follows [35]. For finite discrete Ξ, the problem (1) can be reformulated by adding a copy of the recourse variables and constraints for each scenario:

$$\begin{aligned} \min_{x,\,\eta,\,\{y^{(s)}\}} \quad & c^\top x + \eta \\ \text{s.t.} \quad & Ax \geq d, \\ & \eta \geq b_{\xi_s}^\top y^{(s)}, \quad \forall \xi_s \in \Xi, \\ & Gy^{(s)} \geq h - Ex - M\xi_s, \quad \forall \xi_s \in \Xi, \\ & x \in X,\ y^{(s)} \in Y,\ \eta \in \mathbb{R}, \end{aligned} \tag{22}$$

where $y^{(s)}$ is the recourse decision for scenario ξ_s. Given a reduced scenario set $R^{(k)} \subseteq \Xi$ with $|R^{(k)}| = k \ll S$, the *reduced MILP* is obtained by replacing Ξ with $R^{(k)}$ in (22).

C.2 Problem Descriptions

Selection Problem (SEL). Each of n items has an uncertain cost; the goal is to select a fixed-size subset minimizing total cost. First-stage decisions commit to a partial selection, while the remaining choices are deferred to the second stage after costs are realized. Both stages involve binary decisions. Following [18], we fix the selection size to $\lfloor n/2 \rfloor$ and sample costs uniformly from $\{1, \ldots, 100\}$; the selection size enters as the RHS feature of the corresponding constraint node (Table 8), so the architecture accommodates other values without modification. Instances are denoted SEL-n-s.

Given n items and scenario set $\Xi = \{\xi_1, \ldots, \xi_S\}$, let c_i be the first-stage cost and $d_i^{(s)}$ the second-stage cost of item i under scenario s. The MILP reformulation is:

$$\begin{aligned} \min_{x,\, \eta,\, \{y^{(s)}\}} \quad & \textstyle\sum_{i=1}^{n} c_i\, x_i + \eta \\ \text{s.t.} \quad & \eta \geq \textstyle\sum_{i=1}^{n} d_i^{(s)}\, y_i^{(s)}, && \forall s \in [S], \\ & \textstyle\sum_{i=1}^{n} \big(x_i + y_i^{(s)}\big) = \lfloor n/2 \rfloor, && \forall s \in [S], \\ & x_i + y_i^{(s)} \leq 1, && \forall i \in [n],\ \forall s \in [S], \\ & x_i,\, y_i^{(s)} \in \{0, 1\}. \end{aligned} \tag{23}$$

Vertex Cover (VC). Given a graph $G = (V, E)$ with uncertain node costs, the goal is to select a minimum-cost subset of nodes covering every edge. In the 2RO setting, first-stage decisions commit to a partial cover, strategically reserving the remainder for the second stage; this allows the decision-maker to avoid committing to currently expensive nodes whose costs may be less critical under the realized scenario. Following [18], we generate random graphs with edge probability $10/n$ (average degree ≈ 10) and uniform costs from $\{1, \ldots, 100\}$. Instances are denoted VC-n-s.

Let c_i and $d_i^{(s)}$ be the first- and second-stage costs of node i:

$$\begin{aligned} \min_{x,\, \eta,\, \{y^{(s)}\}} \quad & \textstyle\sum_{i \in V} c_i\, x_i + \eta \\ \text{s.t.} \quad & \eta \geq \textstyle\sum_{i \in V} d_i^{(s)}\, y_i^{(s)}, && \forall s \in [S], \\ & (x_i + y_i^{(s)}) + (x_j + y_j^{(s)}) \geq 1, && \forall (i,j) \in E,\ \forall s \in [S], \\ & y_i^{(s)} \leq 1 - x_i, && \forall i \in V,\ \forall s \in [S], \\ & x_i,\, y_i^{(s)} \in \{0, 1\}. \end{aligned} \tag{24}$$

D NeurPRISE Architecture and Training

D.1 Encoding

D.1.1 Bipartite Graph Representation

Following [16], we represent each scenario-level MILP subproblem as a bipartite graph $G_j = (V_c \cup V_v, E)$, where V_c denotes constraint nodes, V_v denotes variable nodes, and edges connect variables to the constraints in which they appear (Fig. 3). While the graph topology is fixed across scenarios of the same instance, node features vary with scenario-dependent coefficients (e.g., costs, demands). We use this standard constraint–variable bipartite design directly.

D.1.2 GINE Architecture

We employ a Graph Isomorphism Network with edge features (GINE) [25, 34] as the GNN backbone. The update rule for node i at layer ℓ is:

$$\mathbf{g}_i^{(\ell+1)} = \mathrm{MLP}^{(\ell)}\left((1 + \epsilon^{(\ell)}) \cdot \mathbf{g}_i^{(\ell)} + \sum_{j \in \mathcal{N}(i)} \mathrm{ReLU}\Big(\mathbf{g}_j^{(\ell)} + \mathbf{e}_{ji} \Big) \right), \tag{25}$$

where $\mathbf{e}_{ji}$ is the edge feature (constraint coefficient) and $\epsilon^{(\ell)}$ is a learnable scalar. We use two GINE layers followed by global mean pooling over the node index within each scenario graph to obtain per-scenario embeddings $\mathbf{h}_j \in \mathbb{R}^{d_s}$.

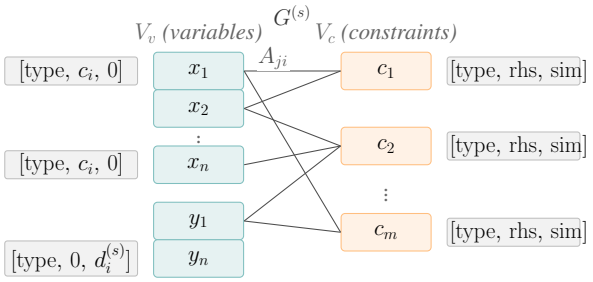


Figure 3: Constraint–variable bipartite graph for a scenario MILP (SEL/VC), where variables form V_v and constraints form V_c. Edges carry coefficient features A_{ji}.

D.1.3 Problem-Specific Feature Engineering

Table 8: Node and edge features used in the GNN encoder by problem class. "Type ind." denotes one-hot node-kind or variable-role indicators.

PROB.	NODE FEATURES	d_{node}	EDGE FEATURES	d_{edge}
SEL	type ind. ($\times 4$), obj. coeff, scenario cost, RHS, cosine sim.	8	constraint coeff.	1
VC	type ind. ($\times 4$), obj. coeff, scenario cost, RHS, cosine sim., degree	9	constraint coeff.	1

Feature Engineering. Variable node features include a type indicator (distinguishing x, y, and η variables), first-stage cost c_i, and scenario-specific second-stage cost $d_i^{(s)}$. Constraint node features include the right-hand side (RHS) value and a *cosine similarity* between the constraint's coefficient row and the objective vector, measuring how aligned each constraint is with the optimization direction. Edge features are coefficient matrix entries. For VC, variable nodes also include the *graph degree* (incident edge count), capturing structural importance.

D.2 Training Details

Table 9 lists the default architecture, training, loss, and evaluation settings.

Input normalization. Before feeding data to the GNN encoder, we apply per-instance scale normalization to floating-point features. The normalization is organized by semantic feature groups: features with the same unit and decision meaning are normalized jointly, while heterogeneous features are normalized separately by their own scale. This rule removes dependence on the magnitude of each instance while preserving relative magnitudes that are meaningful for decision making.
The deterministic first-stage cost vector c and the scenario-dependent second-stage cost matrix $D = [d^{(1)}, \ldots, d^{(S)}]$ represent costs of the same items or nodes across two decision stages. They therefore form one semantic cost group and are normalized with a shared denominator:

Table 9: Default NeurPRISE hyperparameters.

COMPONENT	PARAMETER	VALUE
GINE encoder	Hidden dimension	128
	Output dimension d_s	64
	Number of layers	2
Transformer encoder	Embedding dimension d	64
	Feed-forward dimension	128
	Number of heads	8
	Number of layers	2
Scoring decoder	Number of heads H	4
	Head dimension d_k	32
	Scoring MLP	$H \to 4H \to 1$
Training	Optimizer	AdamW
	Learning rate	6×10^{-4}
	Weight decay	10^{-2}
	Batch size	32
	Max epochs	200
	Early stopping patience	10
Loss (SWKL)	Temperature τ	5
	Log-compression	$\log(1+\Delta)$
	Precision	bfloat16 (AMP)
Evaluation	MIP gap tolerance	10^{-4}
	Gurobi threads	40

$$\tilde{c} = \frac{c}{\|[c;\ \mathrm{vec}(D)]\|_\infty + \epsilon}, \qquad \tilde{D} = \frac{D}{\|[c;\ \mathrm{vec}(D)]\|_\infty + \epsilon}, \tag{26}$$

where $\epsilon = 10^{-8}$ avoids division by zero. The shared denominator preserves the relative magnitude between first- and second-stage costs, which is important for the model to assess scenario severity.

E Additional Experimental Results

E.1 Generalization

This section provides supplementary details for the generalization experiments in the main paper. Distribution shift setup and instance generation protocols are described below (results in Section 5.3); training-set scale analysis follows in Section E.1.2.

E.1.1 Distribution Shift

Motivation and setup. In practice, the scenario distribution at deployment may differ from the training distribution. A model that overfits to the training distribution would require retraining for each new regime. We test whether NeurPRISE, trained on the uniform distributions described in Section 5, generalizes to structurally different scenario families without retraining. We also train an in-distribution model on each target distribution (NeurPRISE$_{\text{norm}}$, NeurPRISE$_{\text{mm}}$) to quantify the gap from not having target-distribution training data. We test robustness to two non-uniform families:

- **Normal.** Per-node means $\mu_v \sim \text{Uniform}[25, 75]$ and standard deviations $\sigma_v = \text{clip}(0.15\,\mu_v,\ 3,\ 15)$ are sampled once per problem size. Scenario costs are drawn as $\mathcal{N}(\mu_v, \sigma_v)$, clipped to $[1, 100]$. This creates a unimodal distribution shift from uniform training data.
- **Multimodal (MM).** Following the instance generation protocol of [19], we sample an instance-specific number of modes $K \sim \text{Uniform}\{3, \ldots, 8\}$. For each mode $k \in [K]$, per-node midpoints $\mu_k \in [25, 75]$ and relative deviations $\delta_k \in [0.1, 0.5]$ are drawn uniformly. Each scenario picks a mode k at random and then samples node costs uniformly in $[(1-\delta_k)\mu_k,\ (1+\delta_k)\mu_k]$. This creates clustered demand regimes and tests whether the learned policy can identify important scenarios across distinct cost patterns.

Results are reported in the main paper (Section 5.3, Tables 5–6).

E.1.2 Do We Need Target-Scale Training?

Models trained on small instances (2,000 out of 2,500) are compared against models trained directly on the target scale (400 out of 500) in Table 10. The small-scale model performs comparably to the target-scale model across both target scales, confirming that the architecture captures transferable structural features rather than scale-specific patterns.

A natural follow-up is whether the small-scale model's advantage comes from architectural transfer or the $5\times$ data advantage (2,000 vs. 400 instances). Table 11 isolates this factor by varying the training-set size at fixed small scale ($N \in \{500, 1500, 2500\}$, 80/10 train/val split). For SEL, regret decreases with both training-set size and scenario budget; at $k \geq 2$, N=1,500 already approaches full-model quality. For VC, performance is similar across all three training-set sizes. Overall, the model is sample-efficient: N=1,500 instances yield competitive results for both problems, and even N=500 achieves strong generalization regret.

Table 10: Small-scale versus target-scale training comparison.

		Medium ↓				Large ↓			
Pr.	Model	k=1	k=2	k=4	k=6	k=1	k=2	k=4	k=6
SEL	Sm.	5.08	**2.72**	**1.51**	**0.88**	4.86	**3.37**	**1.93**	**1.39**
	Tgt.	**4.38**	4.12	2.19	1.76	**4.18**	3.61	2.53	1.62
VC	Sm.	**15.17**	10.06	**5.81**	3.98	15.05	9.09	**5.70**	5.09
	Tgt.	15.26	**9.69**	6.50	**3.65**	**14.92**	**9.06**	6.85	**4.89**

Table 11: Effect of training-set size on in-distribution and generalization performance.

		In-Distribution ↓				Generalization ↓			
Pr.	N	k=1	k=2	k=4	k=6	k=1	k=2	k=4	k=6
SEL	500	**3.54**	3.08	2.05	1.41	**4.49**	3.85	1.94	1.68
	1500	5.54	2.22	0.87	0.57	4.77	3.27	1.72	1.01
	2500	5.59	**2.18**	**0.79**	**0.42**	5.08	**2.72**	**1.51**	**0.88**
VC	500	14.44	9.26	4.51	2.97	**14.20**	**9.06**	6.20	**3.65**
	1500	**14.42**	**8.92**	4.41	2.38	15.55	10.04	**5.59**	3.95
	2500	14.75	9.36	**4.20**	**2.34**	15.17	10.06	5.81	3.98

E.1.3 Cross-Problem Transfer

We next investigate if a model trained on one problem class can provide a useful initialization for another. Specifically, we consider transfer in both directions between SEL and VC small size problem.

We consider target-problem label budgets of 1%, 2.5%, 5%, and 10% for fine-tuning the source-trained model. We report the performance of the resulting source-initialized model. For a fair comparison, we also measure the regret of a model trained from scratch using the same label budget. This comparison allows us to quantify the transfer gain at each label budget. The results demonstrate effective cross-problem transfer in both directions under reduced label budgets.

SEL→VC. A VC model is initialised from a trained SEL model by copying all weights and zero-filling the additional input column. As shown in Table 12, at the 5% label budget, SEL initialisation reduces regret by 18.5% relative to training from scratch with the same budget. Because generating VC labels is substantially more expensive than generating SEL labels, pretraining on SEL provides a practical means of reducing target-task labeling costs.

VC→SEL. To align the input dimensions, an SEL model is initialised from a trained VC model by dropping the VC-specific degree column and copying all remaining weights. As shown in the lower block of Table 12, VC initialisation reduces regret at every reduced label budget, with gains ranging from 18.1% at 20 labels to 43.8% at 100 labels.

Table 12: Cross-problem transfer. The target-only model is trained from scratch using the same target labels. Gain is $(\text{target-only} - \text{transferred})/\text{target-only}$.

TRANSFER	Label budget	Regret (%) ↓		Gain ↑
		Transferred model	Target-only model	
SEL→VC	1%	**4.65**	5.17	10.1%
	2.5%	**4.48**	5.35	16.3%
	5%	**4.21**	5.17	18.5%
	10%	**4.18**	4.43	5.6%
	100%	**4.07**	4.35	6.3%
VC→SEL	1%	**1.48**	1.80	18.1%
	2.5%	**1.34**	1.67	19.7%
	5%	**1.06**	1.88	43.8%
	10%	**1.02**	1.74	41.7%
	100%	0.85	**0.82**	–3.6%

E.2 Seed Variance Analysis

To assess sensitivity to training stochasticity, we train the default model five times with seeds $\{1, 2, 3, 4, 42\}$, keeping data and PRISE labels fixed (Table 13). Standard deviations are consistently small relative to mean regret and far smaller than the gaps between NeurPRISE and the strongest baselines, confirming that the reported advantages are robust to training randomness.

E.3 Time-Budgeted Exact Solver Comparison

Both NeurPRISE and a time-budgeted MILP are approximate solvers; we compare them under equal wall-clock budgets at medium and large scale.

For each scenario budget $k \in \{4, 6\}$, the time-budgeted MILP on the full scenario problem receives a fixed per-instance Gurobi time limit (1 s for k=4, 2 s for k=6), comparable to NeurPRISE's per-instance time. We omit SEL because NeurPRISE's inference time is too short to yield a meaningful solver budget. Table 14 shows that NeurPRISE outperforms the time-budgeted MILP in VC in medium and large scale. This confirms that NeurPRISE is most valuable when the time budget is tight relative to the full-scenario MILP.

Table 13: Seed variance analysis for NeurPRISE across five training seeds.

Prob.	Seed	k=1 ↓	k=2 ↓	k=4 ↓	k=6 ↓
SEL	1	5.77	2.21	0.80	0.56
	2	5.89	2.36	0.77	0.54
	3	6.24	2.35	0.81	0.52
	4	5.80	2.39	0.97	0.58
	42	5.59	2.18	0.79	0.42
	Avg	**5.86**±0.22	**2.30**±0.09	**0.83**±0.07	**0.52**±0.06
VC	1	14.42	9.28	4.50	2.83
	2	14.03	8.55	4.40	2.69
	3	13.89	8.86	4.11	2.79
	4	14.11	8.88	4.31	2.77
	42	14.75	9.36	4.20	2.34
	Avg	**14.24**±0.31	**8.99**±0.30	**4.31**±0.14	**2.68**±0.18

Table 14: Time-budgeted comparison. Both methods receive the same per-instance wall-clock budget, set to NeurPRISE's end-to-end time. Regret is defined in Eq. 3.

Prob.	Scale	k	Budget	Regret (%) ↓	
				NeurPRISE	Time-budgeted MILP
VC	Medium	4	1 s	**6.50**	16.65
	Medium	6	2 s	**3.65**	15.08
	Large	4	1 s	**6.85**	50.76
	Large	6	2 s	**4.89**	41.48

E.4 Speed–Quality Tradeoff via Solver Tolerance

NeurPRISE's runtime is dominated by the downstream exact MILP solver, not the neural inference (Table 2). Since NeurPRISE already selects high-quality worst-case scenarios, the downstream solver may not need to solve the reduced MILP to full precision to obtain a good first-stage decision; we investigate whether relaxing the MIP gap tolerance yields a favorable speed–quality trade-off. We include Neur2RO as a strong learning-based runtime baseline for reference, since it is among the fastest baselines. Figure 4 sweeps tolerance levels on VC at medium and large scale. NeurPRISE provides a tunable speed–quality frontier: relaxing tolerance from exact to 25% reduces solve time while trading regret. Beyond moderate tolerance, additional solver effort yields diminishing regret improvement: since $V(R^{(k)})$ is itself an approximation of $V(\Xi)$, solving the reduced problem to full precision does not guarantee the best first-stage decision.

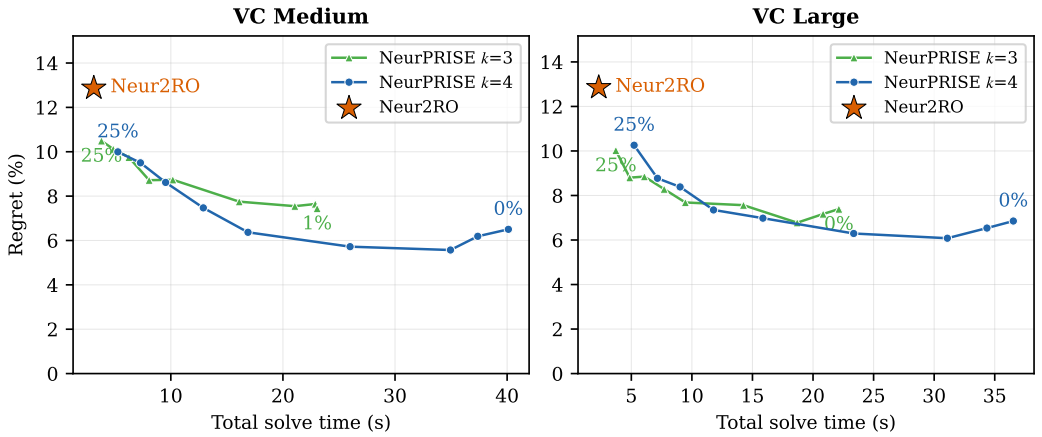


Figure 4: Regret (%) vs. total solve time (50 instances) at different MIP gap tolerances.

E.5 Scenario-Count Growth Control

NeurPRISE controls runtime growth as the original scenario set expands. Although neural inference scores all S scenarios and therefore scales approximately linearly with S, the downstream MILP is solved only on the selected k scenarios.

Medium and large instances in Table 2 differ in both instance hardness and scenario count S. To separate the two, we take the 50 medium test instances (native S=50) and the 50 large test instances (native S=100), vary S by truncating each scenario set, and fix the reduction budget at k=6. Tables 15 and 16 report runtime ratios relative to the smallest S in each population.

NeurPRISE's end-to-end runtime has two components: (i) neural inference, which scores all S scenarios via a forward pass; and (ii) the MILP solve, performed on the k-scenario reduced problem. The first component grows with S, while the second is governed by k. Since the MILP solve dominates at k=6, total runtime grows much more slowly than the original scenario count.

Table 15: Medium instances.

S	Infer.	Solve	Total
10	1.0×	1.0×	1.0×
20	1.1×	1.2×	1.2×
50	1.2×	1.5×	1.5×

Table 16: Large instances.

S	Infer.	Solve	Total
20	1.0×	1.0×	1.0×
50	1.2×	1.1×	1.1×
100	2.3×	1.2×	1.2×

The pattern is consistent across both instance populations. On GPU, inference grows sub-linearly with S, and the MILP solve grows far more slowly than the original scenario count. Consequently, total runtime stays well below linear growth in S. This confirms that NeurPRISE decouples the dominant downstream optimization cost from the original scenario count, providing practical runtime growth control as the uncertainty set expands.

E.6 Offline cost reduction: Cheaper Labels

A natural question is whether PRISE's labeling cost can be further reduced by using approximate solutions. We evaluate the following three relaxed PRISE configurations on medium-scale VC instances; VC is the most computationally demanding problem in our experiments. Three relaxed configurations differ from the exact labeling procedure *only* in the forward-pass inner solve: **tl2s** caps each candidate solve at 2 seconds and **gap002** and **gap005** relax the MIP gap tolerance to 0.02 and 0.05, respectively. Each configuration produces a complete label set for VC medium; we train NeurPRISE on each with three seeds $\{1, 2, 42\}$ and evaluate on the same 50 held-out instances at $k \in \{1, 2, 4, 6\}$.

Table 17 shows that relaxed labeling incurs only a modest increase in regret while yielding substantial time savings, particularly under **gap005**.

Table 17: Mean regret (%, averaged over three training seeds) and labeling time savings (%) on VC medium for NeurPRISE with exact and relaxed PRISE labels.

Method	Label time saving ↑	k=1 ↓	k=2 ↓	k=4 ↓	k=6 ↓
Exact	—	**15.06**	9.46	6.12	**3.49**
tl2s	15.6	15.46	9.57	6.36	3.62
gap002	19.7	15.52	9.63	**6.10**	3.64
gap005	**63.7**	15.25	**9.41**	6.57	3.73

E.7 Sensitivity to PRISE Supervision

To examine whether the PRISE labels remain a useful training signal when they are perturbed, we test how sensitive NeurPRISE is to such perturbations. We take the existing PRISE label file of the VC small instances and apply one perturbation to it, which creates a new perturbed label file: (i) multiplying each marginal gain by $e^{\sigma z}$, where z is standard Gaussian noise; (ii) shuffling the selection order of the selected scenarios; or (iii) remapping the selected scenarios among all

scenarios. We then retrain NeurPRISE from each perturbed label file with the same five training seeds as Table 13. Table 18 reports the results; the unperturbed row is taken from Table 13. Gain noise changes mean regret only slightly, within the variation across training seeds. Shuffling the selection order raises mean regret at $k{=}1$, while at $k{\geq}2$ it stays within the range of the unperturbed and gain-noise rows. Remapping increases mean regret at every budget, with a much larger variation across training seeds. NeurPRISE is thus robust to the gain values in PRISE labels but sensitive to which scenarios PRISE selects.

Table 18: NeurPRISE regret (%) on VC small instances when trained on perturbed PRISE supervision.

Training supervision	$k{=}1\downarrow$	$k{=}2\downarrow$	$k{=}4\downarrow$	$k{=}6\downarrow$
Unperturbed	14.24±0.31	8.99±0.30	4.31±0.14	2.68±0.18
Gain noise, $\sigma{=}0.1$	14.16±0.20	8.96±0.15	4.35±0.08	2.70±0.08
Gain noise, $\sigma{=}0.7$	14.31±0.23	8.90±0.12	4.42±0.15	2.64±0.10
Selection order shuffled	15.40±0.39	9.05±0.18	4.42±0.22	2.48±0.10
Selected scenarios remapped among all scenarios	15.02±3.43	11.58±3.67	9.26±3.83	7.73±3.73

F Ablation Studies

F.1 Encoding and Fusion Strategy

NeurPRISE first encodes each scenario graph with a shared GNN and then refines the resulting per-scenario embeddings through cross-scenario attention to obtain set-aware scenario embeddings (Section 4.2.1). We ablate how instance context and scenario embeddings are fused in the decoder, comparing three strategies: (i) **LF (Late Fusion):** instance context and scenario embeddings are produced by separate GNNs; fusion occurs *after* cross-scenario attention, via context-to-scenario attention in the decoder; (ii) **EF (Early Fusion):** instance context and scenario embeddings are produced by separate GNNs; the instance-context embedding is concatenated as a global token and jointly refined with scenario embeddings *during* cross-scenario attention; (iii) **Ours:** no separate instance-context encoder; the instance context is derived by mean pooling over pre-attention GNN embeddings, and scenarios are scored via multi-head scoring between this pooled context and the attention-refined scenario embeddings.

Results are in Table 19 (mean over 5 seeds). We adopt Ours as the default: it avoids a separate encoder for instance context, reducing model complexity without sacrificing performance.

F.2 Loss Comparison

Since this task can be viewed as scenario selection, a natural baseline is per-scenario binary classification (**BCE**): multi-hot targets from the PRISE trace, treating every selected scenario equally. Our approach, **SWKL** (Score-Weighted KL), converts log-compressed marginal gains into a probability distribution via temperature-scaled softmax and trains with KL divergence, so that ordinal misrankings are penalised in proportion to gain differences.

Table 20 compares the two losses at small and medium scale. At small scale SWKL is moderately better, winning more cells on VC; at medium the advantage becomes clear on SEL and VC. We adopt SWKL as the default for its broader advantage across problems.

G Extended Analysis for CFLP

G.1 Problem Formulation

Among the three problem classes studied, CFLP is structurally distinct: scenario uncertainty enters through customer demands in the constraint right-hand side rather than through objective coefficients. This distinction has two important consequences. First, the SOR baseline [18], which requires objective-coefficient uncertainty, is inapplicable to CFLP. Second, because demand extremes can exceed the total opened capacity, reduced-set recourse may be infeasible—a failure mode absent from SEL and VC, where recourse is always feasible regardless of the selected scenarios. Additionally,

Table 19: Encoder-fusion ablation on small instances, averaged over 5 seeds.

Prob.	Model	Regret (%) ↓			
		k=1	k=2	k=4	k=6
SEL	LF	**5.31**	2.32	1.05	0.68
	EF	5.39	**2.27**	0.98	0.58
	Ours	5.86	2.30	**0.83**	**0.52**
VC	LF	**14.01**	**8.73**	4.38	2.68
	EF	14.07	8.99	**4.31**	**2.66**
	Ours	14.24	8.99	**4.31**	2.68

Table 20: Loss-function ablation.

Prob.	Scale	Loss	k=1	k=2	k=4	k=6
SEL	Small	BCE	6.34	2.44	**0.70**	**0.16**
		SWKL	**5.59**	**2.18**	0.79	0.42
	Medium	BCE	7.20	6.51	4.34	3.08
		SWKL	**4.38**	**4.12**	**2.19**	**1.76**
VC	Small	BCE	15.23	9.46	4.47	2.38
		SWKL	**14.75**	**9.36**	**4.20**	**2.34**
	Medium	BCE	16.02	9.87	**6.01**	3.77
		SWKL	**15.26**	**9.69**	6.50	**3.65**

CFLP employs a compact problem-specific graph encoding that reduces the scenario graph from $O(mn)$ variable nodes (under the standard constraint–variable representation) to $O(m+n)$ facility and customer nodes, yielding significant memory and throughput gains. These characteristics motivate a dedicated analysis separate from the SEL/VC results in the main text.

We now describe the CFLP problem details. The first stage decides which facilities to open (high-stakes infrastructure investment); the second stage assigns customers to open facilities after uncertain demand is realized. The 2RO structure is especially valuable here: since only the realized demand scenario must be served, fewer facilities may suffice compared to a single-stage robust model that hedges against all scenarios simultaneously. We adopt the instance parameter ranges of [35] (fixed costs $f_j \in [100, 1000]$, capacity costs $a_j \in [10, 100]$, max capacity $K_j \in [200, 700]$, transportation costs $c_{ij} \in [1, 1000]$, demands $d_i \in [10, 500]$), using discrete scenario uncertainty in place of their budgeted polyhedral set. Let f_j be the fixed cost of opening facility j, a_j the unit capacity cost, K_j the maximum capacity, c_{ij} the unit transportation cost from facility j to customer i, and $d_i^{(s)}$ the demand of customer i under scenario s. Instances are denoted CFLP-m-n-s (m facilities, n customers, S scenarios). We use Small(CFLP-30-30-50), Medium(CFLP-70-70-50), and Large(CFLP-70-70-100) configurations:

$$
\begin{aligned}
\min_{x,\, z,\, \eta,\, \{y^{(s)}\}} \quad & \textstyle\sum_{j=1}^{m} f_j\, x_j + \sum_{j=1}^{m} a_j\, z_j + \eta \\
\text{s.t.} \quad & \eta \geq \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\, y_{ij}^{(s)}, && \forall s \in [S], \\
& z_j \leq K_j\, x_j, && \forall j \in [m], \\
& \textstyle\sum_{i=1}^{n} y_{ij}^{(s)} \leq z_j, && \forall j \in [m],\ \forall s \in [S], \\
& \textstyle\sum_{j=1}^{m} y_{ij}^{(s)} \geq d_i^{(s)}, && \forall i \in [n],\ \forall s \in [S], \\
& y_{ij}^{(s)} \geq 0,\ z_j \geq 0,\ x_j \in \{0,1\}.
\end{aligned}
\tag{27}
$$

Note that the scenario-dependent term $d_i^{(s)}$ appears in the demand constraint RHS (line 3), not in the objective coefficients—this places CFLP outside the scope of [18], whose framework requires uncertainty to enter through the objective coefficients (see Section 5).

G.2 Encoding and Training

Following [33], we use a problem-specific bipartite graph with facility nodes and customer nodes. Facility features include fixed cost f_j, capacity cost a_j, and maximum capacity K_j; customer features include scenario-specific demand $d_i^{(s)}$; and edges encode transportation costs c_{ij} (Table 21). Importantly, NeurPRISE is agnostic to the choice of scenario graph encoding: different encodings can be used as long as the encoder outputs one fixed-dimensional embedding per scenario. We evaluate the CFLP-specific encoding against the standard constraint–variable encoding below.

For CFLP, the standard constraint–variable (CV) encoding represents assignment variables $y_{ij}^{(s)}$, producing $O(mn)$ variable nodes per scenario. In contrast, the problem-specific (PS) encoding in Section D.1 uses facility and customer nodes and reduces the graph size to $O(m+n)$.

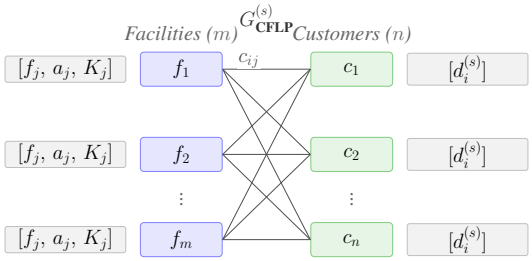


Figure 5: Problem-specific graph for CFLP.

Table 21: CFLP graph encoding features.

PROB.	NODE FEATURES	d_{node}	EDGE FEATURES	d_{edge}
CFLP	facility/customer ind. ($\times 2$), f_j, a_j, K_j, $d_i^{(s)}$	6	transport cost c_{ij}	1

To justify this compact encoding, we compare CV and PS on small-scale CFLP instances. Table 22 reports regret, peak GPU memory, and training throughput. PS achieves lower regret at every budget level (k=1, 2, 4, 6) while training $4.3\times$ faster per epoch with $3.6\times$ less GPU memory at the same batch size. Despite explicitly representing the full assignment-variable structure, CV does not improve over PS at this scale. These results suggest that PS preserves the information needed for scenario selection while avoiding the memory cost of the full CV representation.

Table 22: Encoding ablation on CFLP at small scale. Result is evaluated at k=6. GPU is the peak allocated memory; s/epoch is the median epoch time.

Encoding	**Regret% (↓)**	**GPU (MB)**	**s/epoch**
CV	0.16	10,303	23.1
PS	**0.10**	2,858	5.4

Training configuration. For CFLP instances, we reduce the batch size to fit GPU memory and scale the learning rate linearly as $\text{lr} = 6\times10^{-4}(B/32)$, where B is the effective batch size. Small and medium scale problems use B=24 and lr=4.5×10^{-4}; Large scale problem uses B=16 and lr=3×10^{-4}. For the encoding ablation (Table 22), CV and PS use the same B=24 and learning rate.

Input normalization. All remaining floating-point features are normalized by semantic group. In CFLP, each heterogeneous feature type forms its own group, e.g., capacities, demands, fixed costs and transportation costs are normalized separately because their scales are not directly comparable. For a generic group g with tensor F_g, we use

$$\tilde{F}_g = \frac{F_g}{\|F_g\|_\infty + \epsilon}. \tag{28}$$

Since these features are non-negative, this maps each group to $[0, 1]$ without shifting the origin.

G.3 Performance and Generalization

On CFLP, MaxSum is naturally competitive in-distribution because demand-only uncertainty makes aggregate demand a strong proxy for worst-case recourse cost. At $k = 6$, NeurPRISE achieves the best regret at small scale and remains comparable to MaxSum at medium and large scales, while being much faster than PRISE (Table 23). Random/K-means are mostly infeasible because they often miss demand-extreme scenarios. Neur2RO is primarily a surrogate-based objective prediction method, not a demand-critical scenario-selection method; we adapt its auxiliary feasibility check to our settings to avoid infeasibility, and all Neur2RO CFLP results include this check.

For generalization, we evaluate three deployment changes: larger problem size, larger scenario count, and shifted demand distributions. The size-scaling experiment increases CFLP instances by

$5\times$, and the scenario-count experiment evaluates larger scenario sets with $S = 100$ and $S = 200$ (Table 24). The distribution-shift experiment considers two non-uniform demand families, both unseen during training (Table 25). In the normal setting, per-customer demand means satisfy $\mu_j \sim \text{Uniform}[180, 260]$, with $\sigma_j = \text{clip}(0.08\,\mu_j,\ 12,\ 22)$ and demands clipped to $[10, 500]$. In the multimodal setting, each instance sample 3-6 demand regimes, each with per-customer centers drawn from $[80, 380]$, and per-customer deviations from $[0.05, 0.20]$. Each scenario selects one regime randomly and draws demands within the corresponding deviation band, with a shared per-scenario scaling factor (std deviation 8%) to simulate global demand shifts (e.g., peak-season surges).

NeurPRISE performs consistently well under these generalization tests. Under $5\times$ size scaling and larger scenario sets, NeurPRISE obtains the lowest regret, outperforming both MaxSum and Neur2RO (Table 24). This suggests that the learned scoring rule captures transferable structural signals of scenario importance, rather than merely fitting demand. Training on small instances is also sufficient: the small-scale model performs comparably to, target-scale training. Under both normal and multimodal shifts, NeurPRISE trained on uniform data remains the best among methods that do not use target-distribution training data, outperforming MaxSum and Neur2RO (Table 25). Its regret is also close to the in-distribution NeurPRISE model, indicating that uniform-trained NeurPRISE loses little from not being retrained on the target demand distribution.

Table 23: CFLP performance across three scales at k=6. —: more than 75% infeasible.

	SMALL		MEDIUM		LARGE	
METHOD	REG.↓	TIME	REG.↓	TIME	REG.↓	TIME
Exact	0.00	151.2	0.0	395.3	0.0	836.2
PRISE	0.00	1,623.7	0.0	1,820.0	0.0	6,157.0
K-means	—	—	—	—	—	—
Random	—	—	—	—	—	—
MaxSum	0.15	28.2	**0.1**	37.9	**0.1**	40.3
Neur2RO	1.25	**23.1**	0.9	**16.0**	0.9	**34.3**
NeurPRISE	**0.10**	28.7	**0.1**	40.7	**0.1**	42.2

Table 24: CFLP generalization.

METHOD	$5\times$ size	S=100	S=200
	k=6 ↓	k=8 ↓	k=16 ↓
MaxSum	0.03	0.16	0.08
Neur2RO	0.36	1.25	1.00
NeurPRISE	**0.02**	**0.13**	**0.01**

Table 25: CFLP distribution shift at k=6.

METHOD	NORM. ↓	MM ↓
MaxSum	0.19	0.15
Neur2RO	0.94	1.88
NeurPRISE$_{\text{uni}}$	0.14	0.13
NeurPRISE$_{\text{in}}$	**0.11**	**0.11**

G.4 Feasibility Analysis

Unlike SEL and VC, where uncertainty enters only objective coefficients and recourse is always feasible, CFLP has demand uncertainty in the constraint right-hand side (Eq. 27). A first-stage decision $x^{(k)\star}$ optimized for a reduced set $R^{(k)}$ may open insufficient facility capacity to serve full demand under scenarios outside $R^{(k)}$, causing the recourse infeasible. Table 26 reports infeasibility rates (% of test instances where at least one scenario in Ξ causes infeasible recourse) for each method and scenario budget across three problem scales. K-Means and Random exhibit high infeasibility because they miss demand extremes: K-Means selects near-centroid scenarios, while Random hits max-demand scenarios with probability $\approx k/S$. MaxSum is feasible in this case: with a dense assignment, the recourse feasibility of Eq. 27 reduces to total demand not exceeding total opened capacity, and MaxSum always selects the scenario with the largest total demand. As mentioned above, Neur2RO applies its feasibility check during CCG, which maintains zero infeasibility here.

Table 26: CFLP infeasibility rate (%).

Scale	Method	k=1 ↓	k=2 ↓	k=4 ↓	k=6 ↓
Small	K-Means	99.2	99.2	97.2	93.6
	Random	95.2	94.0	91.6	88.8
	MaxSum	**0.0**	-	-	-
	Neur2RO	**0.0**	-	-	-
	NeurPRISE	21.6	5.6	0.4	**0.0**
Medium	K-Means	100.0	98.0	98.0	86.0
	Random	98.0	92.0	86.0	86.0
	MaxSum	**0.0**	-	-	-
	Neur2RO	**0.0**	-	-	-
	NeurPRISE	10.0	**0.0**	**0.0**	**0.0**
Large	K-Means	100.0	100.0	100.0	100.0
	Random	98.0	98.0	94.0	92.0
	MaxSum	**0.0**	-	-	-
	Neur2RO	**0.0**	-	-	-
	NeurPRISE	10.0	2.0	**0.0**	**0.0**

NeurPRISE recovers feasibility through scenario selection alone without any explicit feasibility check: its infeasibility rate drops monotonically with k and reaches 0% at k=6 across all three scales. This is because demand-extreme scenarios carry large marginal gains in the PRISE labeling, so they rank high in the learned scoring and are included in the selected set once k is sufficient.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim that NeurPRISE is competitive with scenario-reduction and learning-based baselines, approaching the labeling oracle quality at a fraction of its cost, with generalization across scales, scenarios, and distributions. All claims are supported by the experimental results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the offline PRISE labeling cost and how the paper reduces it, and the downstream MILP cost. Remark 1 shows that the robust objective is not submodular. The scope (linear 2RO over finite discrete uncertainty sets) is stated in Section 3.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Proposition 1 states its assumption ($R \subseteq R' \subseteq \Xi$) and provides a complete proof in Appendix B.1. The non-submodularity counterexample (Remark 1) is fully worked out with explicit numerical values.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Hardware, software versions, hyperparameters (Table 9), instance generation procedures (Appendix C), feature engineering details (Appendix D.1), and training protocols (Appendix D.2) are fully specified. The evaluation pipeline is deterministic (top-k scoring).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is available at https://github.com/lintianjue/NeurPRISE, as stated in Section 5.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training details (optimizer, learning rate, batch size, epochs, early stopping) are in Section 5 and Appendix D.2. Instance generation, train/val/test splits, and evaluation protocols are fully specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: A dedicated seed variance analysis (Appendix E.2, Table 13) trains 5 independent seeds and reports mean $\pm$ std, showing max std ≤ 0.31 pp across all configurations — far smaller than the performance gaps between methods. Ablation tables (Tables 19, 13) report mean $\pm$ std over 5 seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware is specified in Section 5 (AMD EPYC 7543 CPU, NVIDIA A6000 Ada GPU). Wall-clock evaluation times are reported in Tables 1–2. The oracle labeling cost is estimated in Appendix B.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work develops a neural surrogate model for scenario reduction in robust optimization. It does not involve human subjects, sensitive data, or dual-use concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is a fundamental methods paper that combines machine learning with robust optimization. The techniques are general-purpose and do not raise specific societal concerns beyond those of optimization research broadly.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release models or data with misuse potential. The neural surrogate and scenario reduction methods are standard academic tools.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All software (Gurobi, PyTorch, PyG) and prior methods (Neur2RO, SOR, MaxSum) are properly cited. Gurobi is used under an academic license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper documents the instance generation procedures, model architecture, and training pipeline. The code is available at `https://github.com/lintianjue/NeurPRISE`.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No crowdsourcing or human subjects are involved.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects research is conducted; IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used for writing assistance and code development only. Neither constitutes a core methodological component, so declaration is not required per the checklist guidelines.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.