CONDITIONAL SET DECODER FOR LEARNING-AUGMENTED COLUMN-AND-CONSTRAINT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Column-and-constraint generation (C&CG) solves two-stage robust optimization (2RO) problems with a nested min–max–min structure by decomposing the problem and iteratively generating worst-case scenarios through a max–min adversarial problem (AP). Learning-augmented C&CG replaces the AP with a neural network that identifies an adversarial scenario for the current first-stage decision. This prediction depends jointly on the first-stage decision and uncertainty realization. Because the two inputs have different roles and feature descriptions, existing methods encode and aggregate them separately before passing the resulting representations to the prediction network. However, separate aggregation makes their componentwise correspondence implicit, so the network must learn both adversarial-scenario identification and correspondence recovery, which increases the difficulty of the learning task. To address this issue, we propose the Conditional Set Decoder (CSD), which conditions each uncertainty-component representation on its corresponding first-stage decision component before aggregation. CSD preserves their componentwise correspondence before low-dimensional compression and avoids reconstructing it afterward. The architecture remains invariant to joint permutations of corresponding elements, supports value learning, and admits a mixed-integer linear representation. Our analysis shows that, for functions defined by interactions between corresponding elements, conditional aggregation requires a lower representation dimension than separate aggregation. Across three 2RO benchmarks—capital budgeting, robust assignment, and facility location with disruptions—CSD reduces the benchmark-specific median evaluation gap by 52.3%, 62.9%, and nearly 100%, respectively. In a controlled scenario-ranking experiment for facility location with disruptions, CSD recovers 91.1% of decision-conditioned ranking reversals, compared with 0.1% for separate aggregation.

1 INTRODUCTION

In capital budgeting and facility operations, investment choices or facility configurations must be determined before costs, returns, or disruptions are known, whereas additional investments or service assignments can be decided after this information is revealed (Subramanyam et al., 2020; Lefebvre et al., 2024). Two-stage robust optimization (2RO) models such problems as

$$\min_{x\in X} \max_{\xi\in\Xi} Q_I(x,\xi), \qquad Q_I(x,\xi) = \min_{y\in Y_I(x,\xi)} c_I(x,\xi,y), \tag{1}$$

where x and y denote decisions made before and after uncertainty is revealed, respectively; ξ is the uncertainty realization; and I specifies the problem instance. The problem has a min–max–min structure and is difficult to handle through dualization when y contains discrete variables (Yanıkoğlu et al., 2019; Dumouchelle et al., 2024).

Column-and-constraint generation (C&CG) decomposes the preceding min–max–min structure into a master problem (MP) and an adversarial subproblem (AP), and alternates between them (Zeng & Zhao, 2013; Tsang et al., 2023). The MP optimizes the first-stage decision over the finite set of generated scenarios; given its solution x_t, the AP solves

$$\max_{\xi\in\Xi} Q_I(x_t,\xi) = \max_{\xi\in\Xi} \min_{y\in Y_I(x_t,\xi)} c_I(x_t,\xi,y) \tag{2}$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

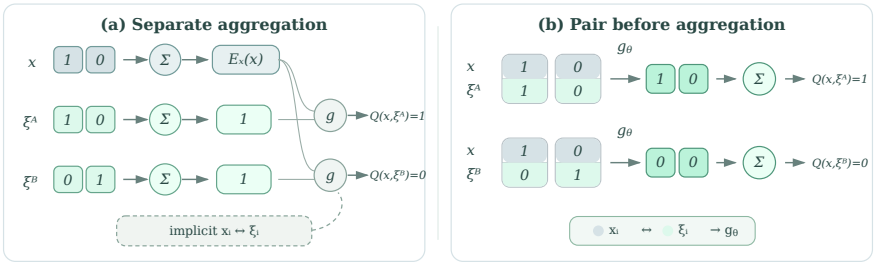


Figure 1: Aggregation order determines whether the x_i–ξ_i correspondence required by $Q(x, \xi) = x^\top \xi$ is retained.

and adds the resulting worst-case scenario to the MP. Both may become computational bottlenecks: every new scenario introduces scenario-specific copies of the second-stage decision variables and constraints, causing the MP to grow across iterations, while the AP's inherent nested max–min structure is particularly difficult to solve with discrete recourse.

Learning-augmented C&CG uses a neural approximation of the second-stage value function to simplify the MP and AP (Dumouchelle et al., 2024). In the MP, the network selects the largest-valued generated scenario, and the MP introduces second-stage decision variables and constraints only for that scenario; in the AP, the network replaces the inner minimization and is optimized over the uncertainty set to generate a new scenario. Both uses require the network to learn how the second-stage value depends jointly on the first-stage decision and uncertainty realization.

Because the neural network is optimized within the MP and AP, its architecture must remain compact. Existing methods use Deep Sets to accommodate different problem sizes and variable orderings. Because first-stage decisions and uncertainty realizations have different roles and feature descriptions, they are encoded and symmetrically aggregated separately before their low-dimensional representations are combined (Dumouchelle et al., 2024; Zaheer et al., 2017). However, separate aggregation prevents their componentwise correspondence from being modeled during aggregation. To learn the joint value function, the two aggregated representations must retain sufficient component information, and the subsequent prediction network must recover their correspondence, introducing an additional learning task.

Figure 1 illustrates this issue using $Q(x, \xi) = x^\top \xi$. For a fixed x, one-dimensional separate aggregation can map two scenarios with different function values to the same representation, preventing the subsequent prediction network from distinguishing them; forming the x_i–ξ_i interactions before aggregation preserves this distinction. Section 4 further formalizes and tests this observation.

We address this issue with the Conditional Set Decoder (CSD). Before aggregation, CSD conditions each uncertainty-component representation on the corresponding first-stage decision, keeping their componentwise correspondence explicit rather than recovering it from two independently aggregated representations. CSD supports variable-size instances and invariance to joint permutations of corresponding components, and admits a compact rectified linear unit (ReLU) mixed-integer linear representation for embedding in the MP and AP. To reduce the solution cost of the embedded network, we exploit the inputs fixed at each C&CG step through exact binary endpoint interpolation in the MP and uncertainty-space branch-and-bound in the learned AP.

Our contributions are:

- We identify a structural limitation of separate aggregation: decision–uncertainty correspondence must be recovered from independently compressed representations.
- We introduce CSD, which models aligned decision–uncertainty interactions before aggregation, supports variable-size inputs and joint permutation invariance, and admits a mixed-integer linear representation.
- We establish lower bounds on representation dimension and uniform value error for separate aggregation.

2 Related Work

Learning-augmented C&CG for two-stage robust optimization. Machine learning has been used to learn uncertainty-set representations and to accelerate optimization under uncertainty. AGRO learns a generative representation of the uncertainty set and searches its latent space for adverse realizations (Brenner et al., 2025). For solution acceleration, scenario selection reduces the number of scenarios that must be considered. NeurPRISE learns to rank scenario marginal gains under discrete uncertainty, selecting scenarios with greater impact on the optimization outcome (Lin et al., 2026). Learning also guides scenario-related search, including branching for K-adaptability (Julien et al., 2025) and adversarial search using a recourse-value surrogate in L2O-CCG (Zhou et al., 2026); Bertsimas & Kim (2024) instead predict solution strategies directly.

Beyond guiding scenario selection and search, learning can directly replace repeated value evaluations during optimization. Neur2SP learns expected recourse under a fixed distribution and embeds the neural network into an optimization model to obtain first-stage decisions (Patel et al., 2022). Robust optimization over continuous uncertainty sets also requires searching for adverse scenarios as the first-stage decision changes. Neur2RO learns the second-stage value as a function of the decision and uncertainty realization and embeds it into C&CG: predictions select retained scenarios in the MP, while optimization of the surrogate over the continuous uncertainty set generates new scenarios in the AP (Dumouchelle et al., 2024). Neur2RO aggregates first-stage decision and uncertainty features separately, then models their relationship through a value network. We retain its C&CG procedure and value supervision, while introducing interactions between corresponding decision and uncertainty components before aggregation.

Set representations and aggregation. Representing this dependence requires a surrogate that accommodates variable-size instances and supports direct embedding in the C&CG loop. Neur2RO uses separate Deep Sets encoders for the first-stage and uncertainty features (Dumouchelle et al., 2024). Shared element maps and pooling provide permutation invariance (Zaheer et al., 2017), while affine–ReLU maps and sum aggregation permit exact MILP representations. However, continuous set representations face limits imposed by their pooled dimension, making the information retained before pooling central to their expressive power (Wagstaff et al., 2019; 2022; Wang et al., 2024; Tabaghi & Wang, 2024). Set Transformers and Multi-Set Transformers introduce interactions through attention (Lee et al., 2019; Selby et al., 2022), while other architectures model exchangeable interaction matrices (Hartford et al., 2018) or compatibility between heterogeneous sets (Saito et al., 2020). Yet the softmax normalization and input-dependent products in attention generally preclude direct exact MILP embedding over continuous inputs. CSD uses the known component correspondence to learn decision–uncertainty interactions before pooling, retaining the affine–ReLU structure needed for MILP embedding.

Neural surrogates in mathematical optimization. C&CG has been accelerated through nested column generation and bundle stabilization (Lefebvre et al., 2026; Ribeiro et al., 2026). When neural value functions are embedded in C&CG, the cost of optimizing these networks also affects the efficiency of the MP and AP. In the MP, the network formulation contributes to the size and relaxation strength of the mixed-integer model. ReLU networks admit mixed-integer linear representations (Fischetti & Jo, 2018), and stronger formulations tighten their continuous relaxations (Anderson et al., 2020). Bound tightening and network size further affect computational performance (Grimstad & Andersson, 2019; Plate et al., 2026). Our MP formulation exploits binary encoder inputs, using exact endpoint interpolation to simplify the embedded network. In the AP, the trained network is optimized over the uncertainty set. Branch-and-bound methods for piecewise-linear networks combine valid bounds with domain partitioning, often by branching on activation states (Bunel et al., 2020). Our AP applies branch-and-bound directly to the uncertainty set, partitioning the domain of the uncertainty inputs to search for an adverse scenario.

3 Preliminaries

3.1 Two-stage robust optimization

We consider two-stage robust optimization problems with discrete first-stage or recourse decisions (Ben-Tal et al., 2004; Yanıkoğlu et al., 2019). For an instance I, let $x \in X_I$ be a first-stage decision, $\xi \in \Xi_I$ an uncertainty realization in a compact set Ξ_I, and $y \in Y_I$ a recourse decision. For a fixed

first-stage decision x and uncertainty realization ξ, the optimal objective value is

$$Q_I(x,\xi) = \min_{y \in Y_I} \left\{ c_I(\xi)^\top x + d_I(\xi)^\top y : T_I(\xi)x + W_I(\xi)y \le h_I(\xi) \right\}. \tag{3}$$

We set $Q_I(x,\xi) = +\infty$ when the fixed-(x,ξ) recourse problem is infeasible. The robust value of x and the 2RO optimum are

$$\mathcal{Q}_I(x) = \max_{\xi \in \Xi_I} Q_I(x,\xi), \qquad V_I^\star = \min_{x \in X_I} \mathcal{Q}_I(x). \tag{4}$$

A first-stage decision is robustly feasible if its recourse problem is feasible for every $\xi \in \Xi_I$. Following Neur2RO (Dumouchelle et al., 2024), we assume that, for any $x \in X_I$, a procedure can identify a realization $\xi \in \Xi_I$ for which the recourse problem is infeasible or verify that no such realization exists.

3.2 Column-and-constraint generation

C&CG maintains a finite subset of the uncertainty set, $\Xi_t = \{\xi^1, \dots, \xi^t\} \subseteq \Xi_I$, and solves the MP

$$\begin{aligned} L_t = \min_{x,\mu,\{y^j\}_{j=1}^t} \quad & \mu & & \\ \text{s.t.} \quad & x \in X_I, \quad y^j \in Y_I, & & j \in [t], \\ & \mu \ge c_I(\xi^j)^\top x + d_I(\xi^j)^\top y^j, & & j \in [t], \\ & T_I(\xi^j)x + W_I(\xi^j)y^j \le h_I(\xi^j), & & j \in [t]. \end{aligned} \tag{5}$$

We write $[m] = \{1, \dots, m\}$ for a positive integer m. Let x_t be the first-stage component of an optimal MP solution. When relatively complete recourse does not hold, we first apply the feasibility-checking procedure described above to x_t. If a scenario is found for which the recourse problem is infeasible, it is added to the scenario set, and the corresponding recourse variables and constraints are introduced into the MP before the next iteration. Once robust feasibility of x_t has been verified, the exact objective AP returns

$$\xi_t^\star \in \arg\max_{\xi \in \Xi_I} Q_I(x_t,\xi), \qquad \widetilde{U}_t = Q_I(x_t,\xi_t^\star). \tag{6}$$

The incumbent upper bound is $U_t = \min\{U_{t-1}, \widetilde{U}_t\}$, with $U_0 = +\infty$, and the incumbent first-stage decision is updated whenever this minimum improves. Thus $L_t \le V_I^\star \le U_t$. If $U_t - L_t > \varepsilon$, the scenario $\xi_t^\star$ is added to the scenario set, and its recourse variables and constraints are introduced into the MP; otherwise the incumbent decision is certified within tolerance. With integer recourse, the AP is a difficult max–min problem (Zeng & Zhao, 2013; Dumouchelle et al., 2024).

3.3 Separately aggregated neural surrogates

Neur2RO represents the first-stage and uncertainty inputs using separate set encoders. Let $r_{I,i}^x(x)$, $i \in [m_x]$, and $r_{I,j}^\xi(\xi)$, $j \in [m_\xi]$, denote their respective feature rows. A separately aggregated surrogate has the form

$$\begin{gathered} E_x(x) = \rho_x\Big(\mathrm{Agg}_{i\in[m_x]}\, \phi_x(r_{I,i}^x(x))\Big), \quad E_\xi(\xi) = \rho_\xi\Big(\mathrm{Agg}_{j\in[m_\xi]}\, \phi_\xi(r_{I,j}^\xi(\xi))\Big), \\ f_\theta(x,\xi) = V_\theta([E_x(x), E_\xi(\xi)]). \end{gathered} \tag{7}$$

Here Agg is permutation invariant, and the element maps share parameters across rows. Neur2RO trains this structure with value supervision and ReLU maps that admit a mixed-integer linear representation (Dumouchelle et al., 2024). Our experiments follow its sum-aggregation configuration and set $f_\theta = \widehat{Q}_\theta$.

In the benchmarks studied here, the two feature collections can be arranged as m_I aligned row pairs $\{(r_{I,i}^x(x), r_{I,i}^\xi(\xi))\}_{i=1}^{m_I}$. The two rows in a pair may occupy different feature spaces, but they describe the same optimization component. The separate architecture does not use this alignment before aggregation: the two inputs first interact in V_θ through their pooled representations.

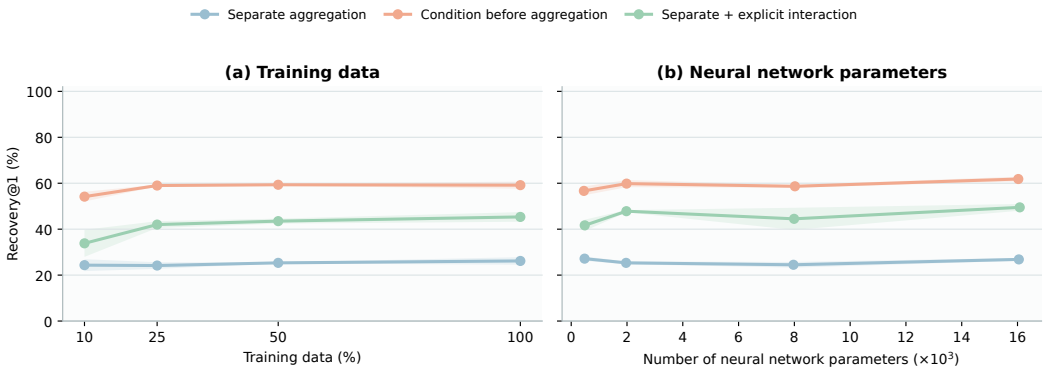


Figure 2: RAP Recovery@1 under varying training data and matched parameter counts (mean ± standard deviation over three seeds).

4 Motivation

Separate aggregation makes the componentwise correspondence between first-stage and uncertainty features implicit. The network must therefore learn the fixed-(x, ξ) value while preserving enough information to recover this correspondence after aggregation.

Figure 2 uses a controlled robust assignment (RAP) experiment to examine whether this additional learning burden can be reduced by increasing the training data or neural network size. Panel (a) varies the amount of training data, while Panel (b) compares models with matched parameter counts; Appendix C provides the complete protocol.

Worst-scenario recovery under separate aggregation improves little as the training data or parameter count increases. Adding an explicit interaction statistic improves recovery, while establishing the correspondence before aggregation performs best throughout. This result links the performance gap to the interaction information available before aggregation.

The following result explains why retaining this interaction information increases the representational burden of separate aggregation. Given a finite set of first-stage decisions $A = \{x^1, \dots, x^m\}$, define the value-response map of a scenario ξ as

$$\mathcal{R}_A(\xi) = \big(Q(x^1, \xi), \dots, Q(x^m, \xi)\big). \tag{8}$$

This map describes how the second-stage value of the same scenario varies across first-stage decisions.

Theorem 1 (Value-response dimension)**.** *Suppose there are a nonempty open set $\Omega \subseteq \mathbb{R}^r$ and a homeomorphism $h : \Omega \to U \subseteq \Xi$ such that $\mathcal{R}_A \circ h$ is injective. If Q has an exact separate factorization on $A \times U$,*

$$Q(x, \xi) = G(E_x(x), E_\xi(\xi)), \qquad E_\xi : U \to \mathbb{R}^{d_\xi},$$

with continuous E_ξ, then $d_\xi \geq r$. Moreover, for any $a_0 \in \Omega$ and $\rho > 0$ with $\overline{B}_\rho(a_0) \subseteq \Omega$, define

$$\gamma_\rho = \inf_{\|s\|_2 = \rho} \|\mathcal{R}_A(h(a_0 + s)) - \mathcal{R}_A(h(a_0 - s))\|_\infty.$$

If $\gamma_\rho > 0$, every continuous separately encoded surrogate with $d_\xi < r$ has uniform value error at least $\gamma_\rho/2$ on $A \times h(\partial B_\rho(a_0))$.

Theorem 1 shows that separate aggregation must retain the scenario information needed to distinguish values across first-stage decisions. Appendix A gives the proof and constructions with uncertainty in the objective, the constraints, or both. In these constructions, the required scenario dimension grows with the number of independently distinguishable components, while aggregation after componentwise interaction needs only a scalar state. Adding a pooled product to Separate can improve recovery, as Figure 2 shows, but leaves the representation requirement unchanged in some recourse families. Proposition 3 in Appendix A.4 gives such a family: the augmented separate encoder still needs at least r scenario dimensions, whereas conditional aggregation needs one. This motivates learning the componentwise interactions before aggregation.

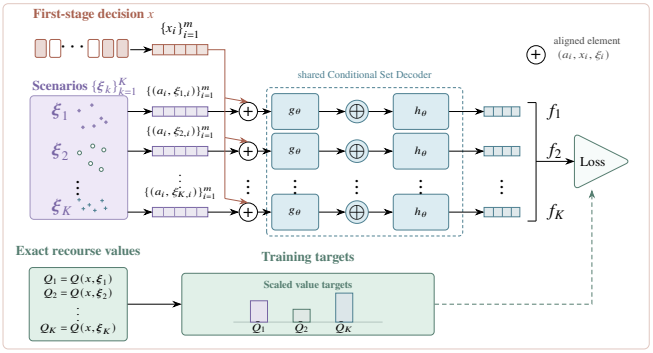


Figure 3: Conditional Set Decoder. For a candidate scenario ξ^k, each aligned first-stage–uncertainty feature pair is mapped before masked sum aggregation to produce its fixed-decision value $\widehat{Q}_\theta(x, \xi^k)$.

5 Conditional Set Decoder

5.1 Conditioning before aggregation

For the aligned features defined in Section 3, Conditional Set Decoder (CSD) jointly maps each aligned feature pair before aggregation:

$$\begin{aligned} e_i(x,\xi) &= g_\theta([r^x_{I,i}(x), r^\xi_{I,i}(\xi)]), \\ z(x,\xi) = \bigoplus_{i\in[m_I]} e_i(x,\xi), &\quad \widehat{Q}_\theta(x,\xi) = h_\theta(z(x,\xi)). \end{aligned} \tag{9}$$

Here $\bigoplus$ denotes masked sum aggregation. The architecture is invariant to simultaneous permutations of the aligned feature pairs and allows each element representation to depend on the corresponding first-stage and uncertainty features before aggregation. Both g_θ and h_θ are ReLU networks, supporting variable-size inputs and mixed-integer linear representations.

To isolate the effect of aggregation structure, CSD and Separate use the same input features, masked sum operator, affine–ReLU components, and scalar output, with comparable trainable parameter counts. Their structural difference is whether corresponding first-stage and uncertainty features interact before aggregation or only afterward.

5.2 Learning and use in C&CG

A training group fixes an instance I and a first-stage decision x, and contains candidate scenarios labeled by their exact second-stage values, $G = (I, x, \{(\xi^k, Q_I(x, \xi^k))\}_{k=1}^{K_G})$. The surrogate $\widehat{Q}_\theta$ minimizes mean squared error on scaled value labels; Appendix B gives the complete objective.

At iteration t, the learned model replaces the exact AP in identifying a candidate scenario:

$$\widehat{\xi}_t \in \arg\max_{\xi\in\Xi_I} \widehat{Q}_\theta(x_t, \xi). \tag{10}$$

Following the Neur2RO solution procedure, $\widehat{Q}_\theta$ is also embedded in the MP to select one retained scenario, which is linked to the MP's single exact recourse problem (Dumouchelle et al., 2024). Feasibility separation remains exact when required. Appendix B specifies the scenario update and stopping rules.

Since all trainable maps consist of affine layers and ReLU activations and the aggregation operator is linear, $\widehat{Q}_\theta$ admits an exact mixed-integer linear representation (Fischetti & Jo, 2018; Anderson et al., 2020). In the learned AP, x_t is fixed and ξ is optimized; in the MP, the retained scenarios are fixed and x is optimized. Appendix B gives the complete network embedding.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Benchmarks and protocol. We evaluate CSD on capital budgeting (CB) (Dumouchelle et al., 2024), robust assignment (RAP) (Bodur et al., 2026), and facility location with disruptions (DFLP) (Lefebvre et al., 2024), using 250, 250, and 480 test instances, respectively. Both architectures are embedded in the same Neur2RO argmax MP and learned AP loop and share its initialization, feasibility separation, scenario updates, stopping rules, and solver settings. Separate uses the Neur2RO set representation; CSD changes only when aligned first-stage and uncertainty features interact. Complete formulations, loop details, and test-set provenance are given in Appendices F, B, and K, respectively.

CB uses both solver-side components in Appendix B; DFLP and RAP use exact binary endpoint interpolation in the MP.

Training. Labels are obtained by solving the deterministic second-stage problem with the first-stage decision and uncertainty realization fixed. The architectures use the same features, labels, groups, splits, and 500-epoch mean-squared-error budget, with comparable parameter counts. Across all three benchmarks, features and value labels are min–max scaled using statistics from the training split. Sum aggregation is used throughout. Selected checkpoints are deployed unchanged on disjoint test instances. Full configurations are given in Appendices H and G.

Evaluation. Returned first-stage decisions are evaluated independently of the learned model. For the Multi NF and C&CG baselines, we use the reference values and solution times reported in Bodur et al. (2026) and Lefebvre et al. (2024), respectively. CB uses the common-pool evaluation. In the main tables, each observation is one test case evaluated with one training seed. Medians are computed over the included runs within each problem-size group; the overall row pools these runs across sizes. The Instances column counts distinct test cases, without multiplying by the number of training seeds. Runtime includes both MP and learned AP solves. Appendix I defines all metrics.

6.2 END-TO-END PERFORMANCE

Capital budgeting. We assess progress toward the robust optimum using the common-pool gap. For each instance and training seed, both returned decisions are evaluated on the union of the scenarios generated by the two architectures. Each architecture's lower bound is obtained from a network-free MP over its own generated scenarios. The complete definition is given in Appendix I.1. Table 1 reports the completed runs over training seeds 11, 17, and 23.

Table 1: Median CB results by problem size.

		CSD			Separate		
n	**Instances**	**Pool gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Pool gap (%)** ↓	**Time (s)** ↓	**Iterations**
10	50	**1.684**	**38.18**	6	2.766	93.24	3
20	50	**0.638**	**600.00**	5	0.749	614.07	5
30	50	**0.234**	**600.00**	4	0.332	660.03	4
40	50	**0.071**	**479.71**	4	0.243	660.02	4
50	50	**0.014**	**224.80**	3	0.063	660.03	4
All	250	**0.188**	**458.53**	4	0.394	614.29	4

CSD obtains a lower median pool gap at every problem size. Overall, it reduces the gap from 0.394% to 0.188% (52.3%) and the runtime from 614.29 to 458.53 seconds (25.4%). Both architectures take a median of four iterations. The relative gap reduction exceeds 70% for $n \geq 40$.

Robust assignment. Table 2 reports median results over the complete RAP test grid and training seeds 11, 17, and 23. Reference gaps use the published Multi NF reference values. Appendix I.2 defines the metric, and Appendix K records the test-set provenance.

CSD reduces the overall median reference gap from 11.12% to 4.12% (62.9%). The difference grows with problem difficulty: for $(n, K) = (25, 5)$ and $(25, 8)$, the median gaps decrease by 77.8%

Table 2: Median RAP results by problem size.

			CSD			Separate			Multi NF
n	K	**Instances**	**Gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Time (s)** ↓
20	2	50	**1.68**	**1.05**	4	1.69	1.32	2	0.03
20	3	50	**2.73**	**1.81**	5	5.55	1.96	2	0.35
20	4	50	**4.83**	3.23	6	10.38	**2.78**	2	1.76
25	5	50	**4.22**	25.32	18	19.01	**4.96**	2	80.50
25	8	50	**6.78**	71.48	38.5	26.59	**10.40**	2	–
All		250	**4.12**	3.91	7	11.12	**2.78**	2	–

A dash indicates that the published Multi NF runs did not all finish within the 30-minute limit.

Table 3: Median DFLP results by problem size.

			CSD			Separate			Published C&CG
$\|F\|$	$\|C\|$	**Instances**	**Gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Time (s)** ↓
10	20	60	**0.00**	467.63	9	100.00	**0.19**	1.5	3.61
10	30	60	**0.00**	103.73	10	100.00	**0.34**	1.5	13.40
10	40	60	**0.00**	139.65	9	100.00	**0.31**	1.5	11.53
10	50	60	**0.00**	254.64	8	100.00	**0.51**	1.5	134.20
15	20	60	**0.78**	185.71	13	71.10	**0.32**	1.5	86.98
15	30	60	**0.00**	359.45	11	54.85	**0.69**	1.5	2205.61
15	40	60	**2.67**	229.63	8	61.26	**0.68**	1.5	–
15	50	60	**1.76**	449.11	8	56.76	**0.81**	1.5	–
All		480	**0.00**	241.14	9	84.40	**0.44**	1.5	84.75

A dash indicates that at least half of the published C&CG runs reached the 3,600-second time limit. Gaps are computed per instance before aggregation.

and 74.5%, respectively. Separate stops after two iterations at every size, while CSD continues for 18 and 38.5 median iterations on these two largest settings. The additional iterations increase runtime relative to Separate but coincide with substantially better decisions. At $(25, 5)$, CSD also remains faster than the published Multi NF run; at $(25, 8)$, the published runs do not all finish within their time limit.

Facility location with disruptions. For DFLP, the published reference is an optimum for 433 instances and a C&CG lower bound for the remaining 47. The reference-gap definition is given in Appendix I.2.

CSD reduces the overall median reference gap from 84.40% to 0.00%. Its median gap is zero for five of the eight size groups and remains below 3% for the other three, whereas Separate has a median gap above 50% in every group. Separate stops after an overall median of only 1.5 iterations, compared with 8–13 across size groups for CSD. The correspondence ablation in Appendix J.1 shows a similar pattern: shuffling component correspondence in CSD reduces the median iteration count from 9 to 2 and raises the median reference gap to 82.32%. This evidence shows that Separate fails to capture the correspondence between decision and uncertainty components, causing premature termination and poor solution quality. CSD has a lower median runtime than the published C&CG baseline on instances with $|F| = 15$ and $|C| > 30$.

6.3 Effect of aggregation structure

Controlled experiment. We vary the number of interacting components in the recourse family from Theorem 1 and compare Separate, CSD, and CSD with shuffled correspondence under matched data and parameter counts. The complete protocol is given in Appendix D.

With a single interacting component ($r = 1$), all three models accurately identify the worst scenario. As the number of interactions grows, Separate loses worst-scenario recovery while CSD remains stable. At $r = 32$, their value-prediction R^2 values are 79.74% and 99.77%, but their Top-1 recoveries are 33.87% and 94.67%. The larger recovery gap shows that an aggregate value fit can

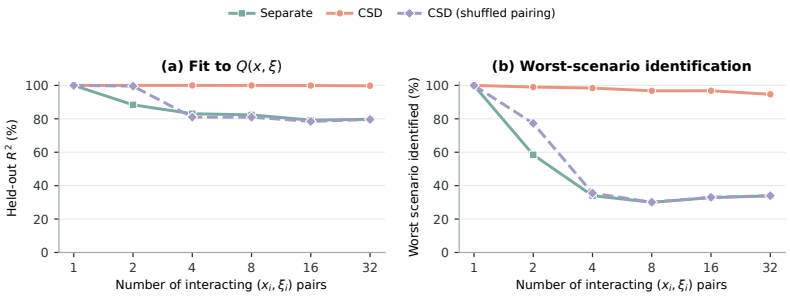


Figure 4: Pairwise interaction experiment (mean $\pm$ standard deviation over three seeds).

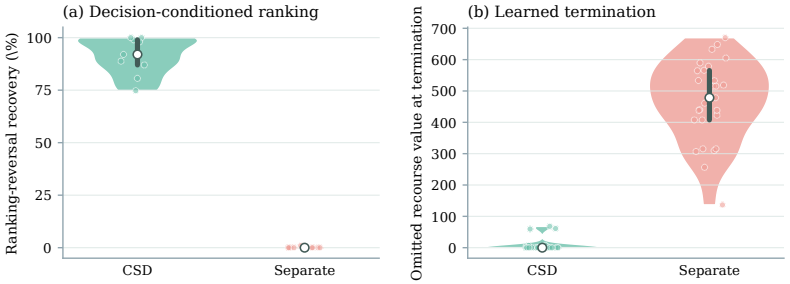


Figure 5: DFLP ranking-reversal recovery and recourse value omitted at learned termination.

conceal errors at the top of the scenario ordering, precisely where the learned AP operates. Shuffling preserves the marginal inputs, data, and model size but reduces CSD to the Separate recovery level from $r = 4$ onward. This isolates the aligned correspondence as the relevant information and matches Theorem 1: a one-dimensional separate scenario representation must preserve a growing value response, whereas CSD forms the required interactions before scalar aggregation.

DFLP case study. We next test the same mechanism with the exact DFLP recourse function. The controlled set contains 297 held-out pairs for which changing one component of the first-stage decision reverses the exact ordering of two scenarios. The complete protocol is given in Appendix E.

CSD recovers 91.1% of the exact ranking reversals, compared with 0.1% for Separate. At learned termination, the exact AP finds a scenario whose recourse value exceeds every accepted scenario in 7 of 27 CSD runs and all 27 Separate runs; the corresponding median omitted deteriorations are zero and 478.46.

Figure 2 shows that adding an explicit interaction feature improves Separate's worst-scenario recovery. Table 12 evaluates this modification in the full DFLP loop. The median reference gaps are 84.40% for Separate, 12.55% after adding the opening–availability product, and 0.00% for CSD. CSD improves the median gap by 12.55 percentage points over the feature-augmented model. Proposition 3 identifies the underlying representational distinction: the augmented separate encoder can still require r scenario dimensions for a recourse family that conditional aggregation represents with a single pooled dimension.

7 Conclusion

We studied how aggregation order affects neural surrogates embedded in C&CG. CSD forms aligned decision–uncertainty interactions before aggregation, reducing the representation dimension required for the analyzed recourse families. We also give a sufficient condition for value approximation to preserve worst-scenario identification. Across CB, RAP, and DFLP, CSD reduces median evaluation gaps from 0.394% to 0.188%, from 11.12% to 4.12%, and from 84.40% to 0.00%, respectively. The advantage persists across training-data and network sizes. Controlled experiments and DFLP ablations link these gains to decision-conditioned scenario ranking.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript organization, translation and language editing, code debugging, and interpretation of results. We also used these tools to check consistency in notation and presentation, verify and format references, and compile the manuscript in LaTeX. We have reviewed all AI-assisted work. The authors take responsibility for the final content of this work, including all text, claims, and materials prepared with the assistance of generative AI tools.

REFERENCES

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183:3–39, 2020.

A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

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.

Merve Bodur, Timothy C. Y. Chan, and Ian Yihang Zhu. Network flow models for robust binary optimization with selective adaptability. *INFORMS Journal on Computing*, 2026.

Aron Brenner, Rahman Khorramfar, Jennifer Sun, and Saurabh Amin. A deep generative learning approach for two-stage adaptive robust optimization. In *International Conference on Learning Representations*, 2025.

Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020.

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

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

Bjarne Grimstad and Henrik Andersson. ReLU networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, 131:106580, 2019.

Jason Hartford, Devon Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1909–1918, 2018.

Esther Julien, Krzysztof Postek, and Ş. İlker Birbil. Machine learning for K-adaptability in two-stage robust optimization. *INFORMS Journal on Computing*, 37(3):644–665, 2025.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, 2019.

Henri Lefebvre, Enrico Malaguti, and Michele Monaci. Adjustable robust optimization with discrete uncertainty. *INFORMS Journal on Computing*, 36(1):78–96, 2024.

Henri Lefebvre, Martin Schmidt, and Johannes Thürauf. Column generation in column-and-constraint generation for adjustable robust optimization with interdiction-type linking constraints. *Mathematical Programming Computation*, 18:259–296, 2026.

Tianjue Lin, Jianan Zhou, Jieyi Bi, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Learning scenario reduction for two-stage robust optimization with discrete uncertainty. *arXiv preprint arXiv:2605.14494*, 2026.

Rahul Mihir Patel, Justin Dumouchelle, Elias Khalil, and Merve Bodur. Neur2SP: Neural two-stage stochastic programming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23992–24005, 2022.

Christoph Plate, Mirko Hahn, Alexander Klimek, Caroline Ganzer, Kai Sundmacher, and Sebastian Sager. An analysis of optimization problems involving ReLU neural networks. *Optimization and Engineering*, 2026.

Débora Alves Ribeiro, Ricardo Saraiva de Camargo, Gilberto de Miranda Jr., and Fátima Machado de Souza Lima. Bundle-enhanced column-and-constraint generation method for solving two-stage robust optimization problems. *Computers & Operations Research*, 193:107525, 2026.

Yuki Saito, Takuma Nakamura, Hirotaka Hachiya, and Kenji Fukumizu. Exchangeable deep neural networks for set-to-set matching and learning. In *Computer Vision – ECCV 2020*, pp. 626–646, 2020.

Kira A. Selby, Ahmad Rashid, Ivan Kobyzev, Mehdi Rezagholizadeh, and Pascal Poupart. Learning functions on multiple sets using Multi-Set Transformers. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 1760–1770, 2022.

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.

Puoya Tabaghi and Yusu Wang. Universal representation of permutation-invariant functions on vectors and tensors. In *Proceedings of The 35th International Conference on Algorithmic Learning Theory*, volume 237 of *Proceedings of Machine Learning Research*, pp. 1134–1187, 2024.

Man Yiu Tsang, Karmel S. Shehadeh, and Frank E. Curtis. An inexact column-and-constraint generation method to solve two-stage robust optimization problems. *Operations Research Letters*, 51 (1):92–98, 2023.

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6487–6494, 2019.

Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Michael A. Osborne, and Ingmar Posner. Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151): 1–56, 2022.

Peihao Wang, Shenghao Yang, Shu Li, Zhangyang Wang, and Pan Li. Polynomial width is sufficient for set representation with high-dimensional features. In *International Conference on Learning Representations*, 2024.

İhsan Yanıkoğlu, Bram L. Gorissen, and Dick den Hertog. A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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.

Zhiyi Zhou, Ján Drgoňa, and Yury Dvorkin. L2O-CCG: Adversarial learning with set generalization for adaptive robust optimization. *arXiv preprint arXiv:2603.22548*, 2026.

A PROOFS

A.1 PROOF OF THEOREM 1

Let $\Omega \subseteq \mathbb{R}^r$ be nonempty and open, and let $h : \Omega \to U \subseteq \Xi$ be a homeomorphism such that $\mathcal{R}_A \circ h$ is injective. Suppose that $E_\xi(h(a)) = E_\xi(h(b))$ for $a, b \in \Omega$. Exact factorization gives

$$Q(x^j, h(a)) = Q(x^j, h(b)), \qquad j = 1, \dots, m,$$

and hence $\mathcal{R}_A(h(a)) = \mathcal{R}_A(h(b))$. Injectivity implies $a = b$. Thus $E_\xi \circ h : \Omega \to \mathbb{R}^{d_\xi}$ is a continuous injection.

If $d_\xi < r$, append $r - d_\xi$ zero coordinates to its output. This produces a continuous injection from the open set $\Omega \subseteq \mathbb{R}^r$ into $\mathbb{R}^r$. Invariance of domain requires its image to be open in $\mathbb{R}^r$, but that image lies in $\mathbb{R}^{d_\xi} \times \{0\}^{r-d_\xi}$, which has empty interior. This contradiction proves $d_\xi \geq r$. Notice that continuity of G and E_x is not needed; continuity of the scenario encoding is essential.

For the quantitative statement, take $a_0 \in \Omega$ and $\rho > 0$ such that $\overline{B}_\rho(a_0) \subseteq \Omega$, and define

$$\gamma_\rho = \inf_{\|s\|_2 = \rho} \|\mathcal{R}_A(h(a_0 + s)) - \mathcal{R}_A(h(a_0 - s))\|_\infty. \tag{11}$$

Assume $\gamma_\rho > 0$. Let $\widehat{Q}(x, \xi) = \widehat{G}(\widehat{E}_x(x), \widehat{E}_\xi(\xi))$ have a continuous scenario encoding of dimension $d_\xi < r$. By the Borsuk–Ulam theorem, the map $s \mapsto \widehat{E}_\xi(h(a_0 + s))$ on the sphere $\{s : \|s\|_2 = \rho\}$ takes the same value at some antipodal pair $s, -s$. Consequently $\widehat{Q}(x^j, h(a_0 + s)) = \widehat{Q}(x^j, h(a_0 - s))$ for every j. If the uniform error on $A \times h(\partial B_\rho(a_0))$ is ϵ, the triangle inequality gives

$$\|\mathcal{R}_A(h(a_0 + s)) - \mathcal{R}_A(h(a_0 - s))\|_\infty \leq 2\epsilon.$$

Therefore $\epsilon \geq \gamma_\rho / 2$.

A.2 DISCRETE SCENARIO SETS

For a finite scenario set, the relevant implication is collision rather than a topological dimension bound.

Proposition 2 (Discrete aggregation collision)**.** *Let ξ^i and ξ^j be two candidate scenarios. If a separate representation satisfies $E_\xi(\xi^i) = E_\xi(\xi^j)$ while $Q(x, \xi^i) \neq Q(x, \xi^j)$ for some $x \in A$, then no output map G can represent Q exactly on $A \times \{\xi^i, \xi^j\}$.*

Proof. The shared scenario representation gives $G(E_x(x), E_\xi(\xi^i)) = G(E_x(x), E_\xi(\xi^j))$, whereas the two target values differ. □

Thus separate aggregation must distinguish scenarios with different value-response vectors. A finite set alone, however, does not imply a latent dimension lower bound: finitely many distinct vectors can be assigned distinct scalar codes. Proposition 2 is the only claim needed for the discrete candidate sets used in our experiments.

A.3 UNCERTAINTY IN THE OBJECTIVE, CONSTRAINTS, OR BOTH

The following constructions show that the separation in Theorem 1 can occur for each location of uncertainty. They are existence results, not statements about every member of the corresponding problem class.

Uncertainty only in the second-stage objective. For any $r \geq 1$, let $X = \{0, 1\}^r$, $\Xi = [0, 1]^r$, and

$$Q_{\text{obj}}(x, \xi) = \min_{y \in \mathbb{R}^r_+} \{\xi^\top y : y = x\} = x^\top \xi. \tag{12}$$

The constraints are deterministic, the uncertainty appears only in the objective, and the unique feasible recourse decision is $y = x$. For $A = \{e_1, \dots, e_r\}$, $\mathcal{R}_A(\xi) = \xi$ on $(0, 1)^r$, so every continuous exact separate factorization has $d_\xi \geq r$.

Uncertainty only in the second-stage constraints. With the same X and Ξ, define

$$Q_{\mathrm{con}}(x,\xi) = \min_{y\in\mathbb{R}^r_+}\{\mathbf{1}^\top y : y_i \geq x_i\xi_i,\ i\in[r]\} = x^\top\xi. \tag{13}$$

The objective is deterministic and uncertainty enters only the recourse constraints. The problem has relatively complete recourse, and $y = x\odot\xi$ is optimal. Again $\mathcal{R}_A(\xi) = \xi$ for $A = \{e_1,\dots,e_r\}$, implying $d_\xi \geq r$.

Uncertainty in both. Let $x = (x^o, x^c) \in \{0,1\}^{r_o+r_c}$ and $\xi = (u,v) \in [0,1]^{r_o+r_c}$. Define

$$Q_{\mathrm{both}}(x,u,v) = \min_{y^o,y^c\geq 0} \quad u^\top y^o + \mathbf{1}^\top y^c \tag{14}$$

$$\text{s.t.} \quad y^o = x^o, \qquad y^c_j \geq x^c_j v_j, \quad j\in[r_c]. \tag{15}$$

Here u enters the objective and v enters the constraints, and

$$Q_{\mathrm{both}}(x,u,v) = (x^o)^\top u + (x^c)^\top v.$$

If the decision set contains $(e_i,0)$ for $i\in[r_o]$ and $(0,e_j)$ for $j\in[r_c]$, its selected value responses recover every coordinate of (u,v). Hence they are injective on $(0,1)^{r_o+r_c}$ and $d_\xi \geq r_o + r_c$. The sum follows from these independently distinguishable directions, not merely from uncertainty appearing in two places. If the same uncertainty coordinates enter both places, the effective response dimension need not double. For example,

$$\min_{y^o,y^c\geq 0}\{\xi^\top y^o + \mathbf{1}^\top y^c : y^o = x,\ y^c_i \geq x_i\xi_i\} = 2x^\top\xi$$

has response dimension r, not $2r$.

The same constructions also make the approximation bound explicit. For the first two, center a radius-$\rho < 1/2$ sphere at $\frac{1}{2}\mathbf{1}\in(0,1)^r$. Since the selected response is the identity, $\gamma_\rho = 2\rho/\sqrt{r}$; every continuous separate surrogate with $d_\xi < r$ therefore has uniform error at least $\rho/\sqrt{r}$. The combined construction gives the analogous bound with $r = r_o + r_c$.

One-dimensional conditional construction. For binary $b\in\{0,1\}$ and $t\in[0,1]$,

$$\mathrm{ReLU}(b + t - 1) = bt. \tag{16}$$

Consequently, Equations (12) and (13) are represented exactly by the one-dimensional pooled state

$$z = \sum_{i=1}^{r} \mathrm{ReLU}(x_i + \xi_i - 1).$$

Equation (15) uses

$$z = \sum_{i=1}^{r_o} \mathrm{ReLU}(x^o_i + u_i - 1) + \sum_{j=1}^{r_c} \mathrm{ReLU}(x^c_j + v_j - 1).$$

The output is $Q = z$ in all three cases. These are finite ReLU element maps with sum aggregation and a scalar pooled state. Exactness relies on binary first-stage components and uncertainty in $[0,1]$; a finite ReLU network cannot represent multiplication exactly on an open set where both factors are continuous. Mean aggregation recovers the construction only when the number of components is fixed or supplied to the output map.

A.4 Separate aggregation with an explicit interaction statistic

A small change to Separate is to append $s(x,\xi) = \sum_i x_i\xi_i$ to the value head while keeping both encoders. We examine whether this added feature removes the dimension requirement in Theorem 1.

Proposition 3 (Dimension separation with an explicit interaction statistic)**.** *For every $r\geq 1$, there is a recourse family with $2r$ components, binary first-stage decisions, and uncertainty in $[0,1]^{2r}$ such that every exact representation*

$$Q(x,\xi) = G\big(E_x(x), E_\xi(\xi), s(x,\xi)\big), \qquad s(x,\xi) = \sum_{i=1}^{2r} x_i\xi_i,$$

with continuous $E_\xi : \Xi \to \mathbb{R}^{d_\xi}$ requires $d_\xi \geq r$. The same family admits an exact conditional representation with a finite ReLU component map and a one-dimensional pooled state. Both representations have access to the same static component attributes.

Proof. Arrange the components in r pairs with static weights $(a_{2j-1}, a_{2j}) = (1, 0)$. Define the feasible decisions $x(b)$ and scenarios $\xi(t)$ by

$$(x_{2j-1}, x_{2j}) = (b_j, b_j), \qquad (\xi_{2j-1}, \xi_{2j}) = (t_j, 1 - t_j), \quad b \in \{0, 1\}^r, \quad t \in [0, 1]^r.$$

The recourse value is

$$Q(x, \xi) = \min_{y \geq 0} \left\{ \sum_{i=1}^{2r} a_i \xi_i y_i : y = x \right\} = \sum_{j=1}^{r} b_j t_j. \tag{17}$$

On this family, the appended statistic satisfies

$$s(x(b), \xi(t)) = \sum_{j=1}^{r} b_j \big[t_j + (1 - t_j) \big] = \sum_{j=1}^{r} b_j.$$

It therefore remains fixed as t varies for each decision b. If $E_\xi(\xi(t)) = E_\xi(\xi(t'))$, the augmented representation gives equal values for every b. Taking $b = e_j$ in Equation (17) yields $t_j = t'_j$ for each j. Hence $t \mapsto E_\xi(\xi(t))$ is a continuous injection on $(0, 1)^r$, and the invariance-of-domain argument in Theorem 1 gives $d_\xi \geq r$.

For the conditional representation, use the shared component map $g(a_i, x_i, \xi_i) = \mathrm{ReLU}(x_i + \xi_i + a_i - 2)$. Since $a_i, x_i \in \{0, 1\}$ and $\xi_i \in [0, 1]$, this map equals $a_i x_i \xi_i$. Thus $z = \sum_i g(a_i, x_i, \xi_i) = Q(x, \xi)$, and the identity output map completes the scalar representation. □

In this example, the added product sum is unchanged across scenarios, although the recourse value varies with the component weights. The separate scenario encoder must therefore retain the information needed to distinguish those values. The conditional component map uses the weights before aggregation and directly produces the required scalar value. Table 12 compares Separate with this added feature against CSD on DFLP.

A.5 Worst-scenario recovery

Proposition 4 (Worst-scenario recovery)**.** *Fix x, and let $\xi^\star$ be the unique maximizer of $Q(x, \cdot)$ over a finite candidate set. If*

$$\sup_{\xi} |\widehat{Q}(x, \xi) - Q(x, \xi)| \leq \epsilon$$

and the exact top-two value margin is $\Delta(x) > 2\epsilon$, then $\arg\max_\xi \widehat{Q}(x, \xi) = \xi^\star$.

Proof. For every $\xi \neq \xi^\star$, $\widehat{Q}(x, \xi^\star) - \widehat{Q}(x, \xi) \geq \Delta(x) - 2\epsilon > 0$. □

B Learning and C&CG Details

B.1 Learning objectives

A training group is $G = (I, x, \{(\xi^k, Q^k)\}_{k=1}^{K_G})$, where $Q^k = Q_I(x, \xi^k)$. Let $\bar{Q} = \mathcal{S}(Q)$ denote the min–max-scaled value label. The value-supervision objective is

$$\mathcal{L}_{\text{value}}(\theta) = N^{-1} \sum_{n=1}^{N} (f_\theta(x_n, \xi_n) - \bar{Q}_n)^2. \tag{18}$$

B.2 Neur2RO loop and stopping

Predictions over the retained scenarios select

$$j_\theta(x) \in \arg\max_{j \in [t]} \widehat{Q}_\theta(x, \xi^j).$$

The selected scenario is linked to a true recourse block as in the published argmax master. Scenario addition compares the predicted adversarial maximum with the predicted historical maximum (Dumouchelle et al., 2024).

The released configurations allow at most 40 C&CG iterations with a relative stopping tolerance of 10^{-4}. The total, MP, and AP time budgets are 3,600, 1,800, and 300 seconds for CB and DFLP, and 600, 300, and 60 seconds for RAP, respectively. Each MP and AP call is also capped by the remaining total time budget.

B.3 Mixed-integer linear programming representation

Finally, given valid pre-activation bounds $\ell \le a \le u$, a scalar $v = \max\{0, a\}$ is represented with $b \in \{0, 1\}$ through

$$v \ge a, \qquad v \ge 0, \qquad v \le a - \ell(1 - b), \qquad v \le u\, b.$$

Composing these constraints with affine layers and linear aggregation gives the mixed-integer linear programming (MILP) representation used in the master and adversarial problems.

B.4 Solver-side acceleration

The neural argmax MP and learned AP repeatedly optimize the same trained surrogate. Two implementation components exploit the input fixed in each problem: exact interpolation in the MP and uncertainty-space branch-and-bound in the learned AP. Algorithms 1 and 2 summarize the corresponding procedures.

MP: exact binary interpolation and output bounds. Consider a retained scenario ξ^j. In the separately aggregated model, its fixed uncertainty representation is evaluated once outside the MP, while the first-stage representation is embedded once and shared by all retained scenarios. For CSD, let s_i^j collect the fixed features of component i under ξ^j, and suppose that the remaining encoder input is a binary first-stage variable x_i. Define the two endpoint outputs

$$h_{i,0}^j = g_\theta([s_i^j, 0]), \qquad h_{i,1}^j = g_\theta([s_i^j, 1]).$$

Since $x_i \in \{0, 1\}$, the element representation has the exact identity

$$g_\theta([s_i^j, x_i]) = h_{i,0}^j + (h_{i,1}^j - h_{i,0}^j)x_i. \tag{19}$$

Consequently, the pooled state is affine in the MP variables,

$$z^j(x) = c^j + \sum_i D_i^j x_i, \qquad c^j = \sum_i h_{i,0}^j, \quad D_i^j = h_{i,1}^j - h_{i,0}^j, \tag{20}$$

with the right-hand side divided by the number of valid components when mean aggregation is used. Equation (19) replaces the element-encoder ReLU formulation by an affine equality in the MP.

Valid componentwise bounds for Equation (20) are

$$\ell_k^j = c_k^j + \sum_i \min\{0, D_{ik}^j\}, \qquad u_k^j = c_k^j + \sum_i \max\{0, D_{ik}^j\}. \tag{21}$$

For an affine layer $a = Wv + b$, these bounds are propagated as

$$\ell_a = W^+\ell_v + W^-u_v + b, \qquad u_a = W^+u_v + W^-\ell_v + b,$$

where $W^+ = \max\{W, 0\}$ and $W^- = \min\{W, 0\}$; a ReLU maps them to $(\max\{0, \ell_a\}, \max\{0, u_a\})$. This gives finite valid bounds for every retained prediction $f_\theta(x, \xi^j)$ and for their maximum. Supplying these bounds to the argmax indicator formulation strengthens its continuous relaxation.

Algorithm 1 Exact MP interpolation for binary first-stage inputs

Require: Trained element map g_θ; retained scenarios $S = \{\xi^j\}_{j=1}^K$; binary first-stage variables x; aggregation divisor ν (1 for sum and the number of valid components for mean)

for $j = 1, \dots, K$ **do**
  for each aligned component i **do**
    $h^j_{i,0} \leftarrow g_\theta([s^j_i, 0])$ and $h^j_{i,1} \leftarrow g_\theta([s^j_i, 1])$
    $D^j_i \leftarrow (h^j_{i,1} - h^j_{i,0})/\nu$
  end for
  $c^j \leftarrow \sum_i h^j_{i,0}/\nu$ and $z^j(x) \leftarrow c^j + \sum_i D^j_i x_i$
  Compute (ℓ^j, u^j) using equation 21
  Propagate (ℓ^j, u^j) through the post-aggregation network
  Embed the post-aggregation ReLUs using the propagated bounds
end for
Build the argmax constraints using the resulting prediction bounds
return the exact neural argmax MP formulation

AP: branch-and-bound over the uncertainty set. For a fixed first-stage decision x, the learned AP is

$$\max_{\xi \in \Xi_I} f_\theta(x, \xi). \tag{22}$$

When $\Xi_I = [\underline{\xi}, \overline{\xi}]$ is low-dimensional and the fixed-decision network features are affine in ξ, those features and adjacent linear maps can be folded into an equivalent scalar ReLU network of ξ. We solve Equation (22) by spatial branch-and-bound directly over Ξ_I. Each node is a sub-box $B = [\ell_B, u_B]$. Affine ReLU relaxations provide a valid neural-output upper bound $U(B)$, while direct evaluation at feasible points supplies an incumbent $L^\star$. A node is pruned whenever

$$U(B) \le L^\star + \epsilon_{\text{rel}} \max\{|L^\star|, 10^{-9}\},$$

and an unpruned node is bisected along its widest uncertainty coordinate. The learned AP is certified to the requested relative tolerance when all remaining sub-boxes satisfy this test.

The uncertainty-space search evaluates and bounds the folded network directly. It receives a fraction α of the total learned-AP time budget T_{AP}, with $\alpha = 0.5$ by default. If this stage ends with open sub-boxes, the incumbent ξ warm starts the Gurobi formulation of the learned AP using the remaining total budget. If no time remains, the current incumbent is returned. Gurobi also solves the MP and the recourse models.

Algorithm 2 Uncertainty-space branch-and-bound for the learned AP

Require: Fixed decision x; box $\Xi_I = [\underline{\xi}, \bar{\xi}]$; folded network $f_\theta(x, \cdot)$; relative tolerance ϵ_{rel}; total AP time budget T_{AP}; BaB allocation fraction $\alpha \in (0, 1]$

Start the AP clock and set $T_{\mathrm{BaB}} \leftarrow \alpha T_{\mathrm{AP}}$
Evaluate the center and, when the dimension permits, the corners of Ξ_I; store the best point $(\xi^\star, L^\star)$
Compute the valid root bound $U(\Xi_I)$ and initialize the priority queue $\mathcal{N} \leftarrow \{\Xi_I\}$
while $\mathcal{N} \neq \varnothing$ and elapsed AP time is below T_{BaB} **do**
    Remove a box $B = [\ell_B, u_B]$ with largest upper bound $U(B)$
    if $U(B) \leq L^\star + \epsilon_{\mathrm{rel}} \max\{|L^\star|, 10^{-9}\}$ **then**
        Discard B and continue
    end if
    Bisect B along its widest uncertainty coordinate
    for each child box B' **do**
        Evaluate its center and update $(\xi^\star, L^\star)$
        Compute the valid affine–ReLU bound $U(B')$
        if $U(B') > L^\star + \epsilon_{\mathrm{rel}} \max\{|L^\star|, 10^{-9}\}$ **then**
            Insert B' into $\mathcal{N}$
        end if
    end for
end while
if $\mathcal{N} = \varnothing$ **then**
    return $(\xi^\star, L^\star)$ ▷ Optimal within tolerance
else
    $T_{\mathrm{rem}} \leftarrow \max\{0, T_{\mathrm{AP}} - \text{elapsed AP time}\}$
    if $T_{\mathrm{rem}} > 0$ **then**
        Warm-start the learned-AP MILP with $\xi^\star$ and solve with time limit T_{rem}
        Update $(\xi^\star, L^\star)$ if the MILP finds a better candidate
    end if
    return incumbent $(\xi^\star, L^\star)$
end if

Implementation conditions. The MP interpolation applies to binary encoder inputs with the other row features fixed for a retained scenario. The AP component uses a low-dimensional box uncertainty set and valid bounds on each sub-box. For CB, the uncertainty features are affine and the surrogate is a ReLU network, so the fixed-decision feature map and adjacent linear layers yield the required folded network. Benchmarks satisfying the binary feature contract share the MP component; APs satisfying the box and bound conditions use the uncertainty-space component.

Recorded acceleration checks. Table 4 reports the controlled comparisons used to validate the two components. The DFLP MP comparisons fix the trained network, first-stage model, and retained scenario set. On the nine-scenario instance, the original formulation retained an 855% mixed-integer programming (MIP) gap after 30 seconds, whereas finite prediction bounds proved optimality in 3.05 seconds and reduced the search from 176,488 to 2,625 nodes. On the 11-scenario instance, solve time decreased from 4.23 to 3.32 seconds. A three-instance end-to-end check kept the first-stage decisions, scenario sequences, iteration counts, stopping conditions, and objectives unchanged; total time decreased from 69.53 to 5.09 seconds, with a maximum objective difference of 1.14×10^{-13}.

The CB comparison fixes the seed-17 checkpoint and the C&CG limits and changes only the learned-AP backend. Across five instances, total time decreased from 2,593.25 to 145.30 seconds. We first compare the independently reconstructed, network-free MP lower bounds. Adding scenarios strengthens this minimization MP, so a higher bound reflects progress toward the robust optimum. The recorded lower-bound improvement is accompanied by a decrease in mean common-pool diagnostic from 1.941% to 1.271%. This measures the gap between each run's lower bound and its returned decision's value on the deduplicated union of scenarios from both runs.

The archive also contains 198 instance–seed pairs with both a standard-backend result completed before the accelerator switch and a result from the separate accelerated run. Both runs terminate before their respective time limits in 96 pairs. On this subset, median elapsed time is 1,429.80

Table 4: Controlled checks of the solver-side acceleration components.

Benchmark	Component	Cases	Original (s)	Accelerated (s)	Speedup	Consistency check
DFLP	MP bounds, 9 scenarios	1	> 30.00	3.05	$> 9.8\times$	Same objective
DFLP	MP bounds, 11 scenarios	1	4.23	3.32	$1.27\times$	Same objective
DFLP	End-to-end MP acceleration	3	69.53	5.09	$13.66\times$	Same C&CG trajectory
CB	AP quality calibration	5	2,593.25	145.30	$17.85\times$	Higher clean-MP lower bound
CB	Completed paired runs	96	1,429.80	31.99	$40.14\times$	Same decision in 91 pairs
CB	Both complete within 600 s	24	297.12	10.34	$20.86\times$	Same decision in all pairs

Table 5: CB solution-quality comparison over all stored standard–accelerated pairs.

Items	Pairs	Same x	Mean lower-bound change (%)	Mean diagnostic (%)	
				Standard	Accelerated
10	100	89	-0.108	**1.779**	1.896
20	93	4	0.551	1.356	**0.941**
30	5	0	3.131	3.281	**0.183**
All	**198**	**93**	**0.284**	1.618	**1.404**

versus 31.99 seconds, and the median paired speedup is $40.14\times$. The returned first-stage decision is identical in 91 of the 96 pairs. Median accumulated AP time decreases from 1,326.23 to 5.46 seconds, whereas median MP time is 26.48 and 27.51 seconds, respectively. Among the 24 pairs in which both runs finish within 600 seconds, all 24 return the same first-stage decision and the median paired speedup is $20.86\times$. At the same 600-second horizon, the number of completed runs increases from 24 to 114 among all 198 pairs. The historical standard runs used a longer overall time limit, so time-limited all-pair ratios are excluded from the speedup statistic.

Table 5 applies the same lower-side and common-pool evaluation to all 198 stored pairs. For each run, a network-free MP over its own generated scenarios gives the lower bound, and both returned decisions are evaluated on the deduplicated union of the pair's scenarios. Because the MP is a minimization problem, a positive lower-bound change is an improvement. The diagnostic is the common-pool decision value minus the run's own-pool lower bound, normalized by the absolute lower bound. Of the 198 accelerated clean MPs, all solve to optimality; 196 standard clean MPs solve to optimality and two use valid solver bounds.

The lower bound and common-pool gap together assess solution progress. A higher clean-MP lower bound reflects stronger scenario constraints, while a smaller common-pool gap measures a smaller distance between the run's lower bound and its returned decision's value on the shared scenario pool.

Across all pairs, the mean clean-MP lower bound increases by 0.284%, while the mean diagnostic decreases from 1.618% to 1.404%. The accelerated runs therefore produce a stronger lower-side bound on average and reduce the average residual measured on the shared scenario pool. The aggregate improvement is driven by the larger gains at $n = 20$ and $n = 30$. The separate five-instance calibration isolates the AP backend under identical limits; the 198-pair table uses every available pre-switch match and records the broader deployed behavior.

The same implementations were then used beyond the controlled subset. The stored experiment archive contains 820 DFLP CSD runs produced after enabling the finite-bound MP formulation and 363 CB CSD runs produced with the uncertainty-space branch-and-bound backend. These counts document deployment coverage; they are not treated as additional paired speedup observations.

C Motivation Experiment Protocol

Each group in Figure 2 fixes one first-stage decision and evaluates 50 candidate uncertainty realizations using their exact fixed-(x, ξ) values. Recovery@1 records whether the largest prediction identifies a candidate attaining the maximum value in its group. Panel (a) varies the fraction of an 800-group training pool at approximately 4K matched parameters; panel (b) varies the matched parameter count using the full pool. The explicit-interaction baseline adds the pooled product of the first-stage decision and uncertainty-dependent features to separate aggregation. Curves report three seeds, with disjoint 200-group sets for checkpoint selection by Recovery@1 and final reporting.

D Controlled Representation Experiment Protocol

The controlled experiment in Figure 4 uses

$$Q(x,\xi) = r^{-1}\sum_{i=1}^{r} x_i\xi_i, \qquad x \in \{0,1\}^r, \quad \xi \in [0,1]^r,$$

for $r \in \{1,2,4,8,16,32\}$. The factor r^{-1} keeps the target scale comparable across dimensions. First-stage components are sampled independently from $\mathrm{Bernoulli}(0.5)$ and uncertainty components independently from $\mathrm{Uniform}[0,1]$. Each feature row also contains a fixed component identifier.

Separate uses a one-dimensional scenario representation, while CSD uses a one-dimensional pooled state. The shuffled variant cyclically shifts the first-stage components before pairing them with the uncertainty components. This preserves both marginal inputs and changes only their correspondence. All three architectures have 2,402 trainable parameters.

Training uses batches of 256, learning rate 2×10^{-3}, weight decay 10^{-5}, and training seeds 11, 17, and 23. Checkpoint selection maximizes Recovery@1 on 4,096 sampled validation pairs evaluated every 100 updates. Evaluation uses 1,000 held-out first-stage decisions and 32 candidate scenarios per decision. Figure 4 reports the metrics defined in Appendix I.3; curves are means and bands are standard deviations across the three training seeds.

E DFLP Pairing Experiment Protocol

The controlled study in Figure 5 uses the first three held-out DFLP instances with $(|F|,|C|) = (10,20)$. We first collect all 10-component decision and scenario patterns occurring in the training data. Using random seed 2027, we select 100 pairs (x^a, x^b) with equal cardinality per instance, where x^b replaces exactly one nonzero component of x^a. For a component shared by the two decisions, the associated scenario pair (ξ^a, ξ^b) contains that shared component and the component specific to x^a or x^b, respectively. Both scenario patterns must also occur in the training data.

For each candidate pair, exact fixed-scenario recourse solves compute

$$\Delta_a = Q(x^a,\xi^a) - Q(x^a,\xi^b), \qquad \Delta_b = Q(x^b,\xi^a) - Q(x^b,\xi^b).$$

The controlled set contains the 297 candidates satisfying $\Delta_a\Delta_b < 0$ and $\min\{|\Delta_a|,|\Delta_b|\} > 10^{-6}$. A model recovers a reversal when its two predicted differences have the signs of Δ_a and Δ_b. Pair construction and filtering use no learned scores. Both architectures have 705 trainable parameters, and the reported points correspond to training seeds 11, 17, and 23.

For the termination audit, let S_t be the scenarios accepted when the learned loop stops at decision x_t. We solve the benchmark exact AP and compute

$$\delta_t^{\mathrm{exact}} = \max_{\xi\in\Xi} Q(x_t,\xi) - \max_{\xi\in S_t} Q(x_t,\xi).$$

We record an omitted adverse scenario when $\delta_t^{\mathrm{exact}}$ exceeds the exact AP tolerance. The audit covers both architectures, all three training seeds, the same three held-out instances, and $\Gamma \in \{2,3,4\}$, giving 27 runs per architecture.

Correspondence ablation. The DFLP correspondence ablation uses the same 16,479 labeled samples, training–validation split, 705-parameter conditional architecture, optimizer, 500-epoch budget, and checkpoint-selection rule as the reported DFLP model. For each decision group, a fixed-point-free permutation is applied to the first-stage component blocks before they are paired with uncertainty rows. The same permutation is shared by every candidate scenario in that group. This preserves the multiset of first-stage entries, all uncertainty entries, all static instance features, group sizes, and labels; only component correspondence is changed. Results are aggregated over training seeds 11, 17, and 23.

F Complete Problem Formulations

The three benchmarks instantiate the min–max–min model in Equation (4). For a finite retained scenario set $S = \{\xi^1, \dots, \xi^K\}$, their MP has the common form

$$\min_{x \in X, \mu, \{y^k\}_{k=1}^K} \mu \quad \text{s.t.} \quad y^k \in Y(x, \xi^k), \quad \mu \geq q(x, y^k, \xi^k), \quad k \in [K]. \tag{23}$$

The exact adversarial problem at a fixed x is

$$\max_{\xi \in \Xi} Q(x, \xi), \qquad Q(x, \xi) = \min_{y \in Y(x,\xi)} q(x, y, \xi). \tag{24}$$

Reward-maximization benchmarks are multiplied by -1 when put in this canonical minimization form. Thus, each scenario block below specifies both the columns and constraints added to Equation (23), while its uncertainty set and recourse value specify Equation (24).

Capital budgeting (CB). Project i has nominal cost $\bar{c}_i$, nominal revenue $\bar{r}_i$, and uncertainty loadings $\Phi_i, \Psi_i \in \mathbb{R}^4$. Define $c_i(\xi) = (1 + \Phi_i^\top \xi/2)\bar{c}_i$ and $r_i(\xi) = (1 + \Psi_i^\top \xi/2)\bar{r}_i$, with $\Xi_{\text{CB}} = [-1, 1]^4$. The reward form of the problem is

$$\max_{x \in \{0,1\}^n} \min_{\xi \in \Xi_{\text{CB}}} \max_{y \in \{0,1\}^n} r(\xi)^\top (x + \alpha y) \tag{25}$$

$$\text{s.t.} \quad x + y \leq \mathbf{1}, \qquad c(\xi)^\top (x + y) \leq B, \tag{26}$$

where $0 < \alpha < 1$ discounts projects postponed to the second stage. A first-stage decision is admissible only if $\max_{\xi \in \Xi_{\text{CB}}} c(\xi)^\top x \leq B$; this feasibility adversary is separated before the objective AP. We use the loan-free variant of Subramanyam et al. (2020).

Robust assignment (RAP). Let $S \subseteq L \times M$ be the admissible assignment pairs. Pair $i = (\ell, m)$ has nominal reward ξ_i^0, agent weight a_ℓ, and task capacity b_m. The first stage makes at most a fraction β of the pairs available, and the second stage chooses a feasible assignment among them:

$$\max_{x \in \{0,1\}^{|S|}} \min_{\zeta \in \Xi_{\text{RAP}}} \max_{y \in \{0,1\}^{|S|}} \zeta^\top y \tag{27}$$

$$\text{s.t.} \quad \sum_i x_i \leq \lfloor \beta |S| \rfloor, \qquad y_i \leq x_i, \tag{28}$$

$$\sum_{\ell:(\ell,m) \in S} a_\ell y_{\ell m} \leq b_m \qquad \forall m \in M, \tag{29}$$

$$\sum_{m:(\ell,m) \in S} y_{\ell m} \leq 1 \qquad \forall \ell \in L, \tag{30}$$

where

$$\Xi_{\text{RAP}} = \left\{ \zeta : \sum_i \left| \zeta_i / \xi_i^0 - 1 \right| \leq 0.1|S|, \ \left| \zeta_i / \xi_i^0 - 1 \right| \leq 0.5 \ \forall i \right\}. \tag{31}$$

The implementation uses the equivalent degradation coordinates $\zeta_i = \xi_i^0(1 - 0.5\xi_i)$, $0 \leq \xi_i \leq 1$, and $\sum_i \xi_i \leq 0.2|S|$ (Bodur et al., 2026).

Facility location with disruptions (DFLP). Let x_i indicate whether facility $i \in F$ is opened, with fixed cost f_i, capacity u_i, customer demand d_j, transportation cost t_{ij}, and service profit p_j. A binary disruption vector removes the capacity of at most Γ facilities:

$$\Xi_{\text{DFLP}} = \left\{ \xi \in \{0,1\}^{|F|} : \sum_{i \in F} \xi_i \leq \Gamma \right\}. \tag{32}$$

Writing $a_{ij} = t_{ij} - p_j$, the fixed-scenario value is

$$Q_{\text{DFLP}}(x, \xi) = \min_{y \in \{0,1\}^{|F| \times |C|}} \sum_{i \in F} f_i x_i + \sum_{i \in F} \sum_{j \in C} a_{ij} y_{ij} \tag{33}$$

$$\text{s.t.} \quad \sum_{i \in F} y_{ij} \leq 1 \qquad \forall j \in C, \tag{34}$$

$$\sum_{j \in C} d_j y_{ij} \leq u_i x_i (1 - \xi_i) \qquad \forall i \in F. \tag{35}$$

The robust objective is $\min_{x \in \{0,1\}^{|F|}} \max_{\xi \in \Xi_{\text{DFLP}}} Q_{\text{DFLP}}(x, \xi)$ (Lefebvre et al., 2024).

Table 6: Per-component feature contracts. Dimensions exclude the set axis.

Benchmark	d_x	d_ξ	d_{cond}	Conditional row
CB	3	4	5	$[c_i(\xi)/\bar{c}_i, r_i(\xi)/\bar{r}_i, \bar{c}_i, \bar{r}_i, x_i]$
RAP	4	4	5	$[1-0.5\xi_i, \xi_i^0, a_\ell, b_m, x_i]$
DFLP	9	9	10	$[x_i, 1-\xi_i, f_i, u_i, d_j, t_{ij}, p_j, a_{ij}, \bar{i}, \bar{j}]$

Table 7: Training and validation data used by the reported checkpoints.

Benchmark	**Train groups**	**Train samples**	**Val. groups**	**Val. samples**	**Training seeds**	**Checkpoint selection**
CB	3,676	174,399	919	43,512	11, 17, 23	Recovery@1
RAP	2,336	116,800	585	29,250	11, 17, 23	Recovery@1
DFLP	640	13,203	160	3,276	11, 17, 23	Recovery@1

G Feature Contracts and Model Architectures

Each feature row corresponds to one aligned component in the benchmark's first-stage and uncertainty descriptions. Static instance attributes are repeated on both sides of the separate model. The conditional model receives the same attributes once, together with the aligned first-stage and uncertainty entries. Table 6 gives the exact row contracts before normalization.

Here $\bar{i}$ and $\bar{j}$ are normalized indices. Rows are placed in a canonical order and padded only for batching; a mask ensures that padded rows contribute zero under sum pooling.

The separate architecture computes

$$h_x = \rho_x(\text{pool}_i\, \phi_x(r_i^x)), \quad h_\xi = \rho_\xi\Big(\text{pool}_i\, \phi_\xi(r_i^\xi)\Big), \quad f_\theta = V([h_x, h_\xi]), \tag{36}$$

whereas CSD computes

$$f_\theta = V\Big(\rho\Big(\text{pool}_i\, \phi([r_i^x, r_i^\xi])\Big)\Big). \tag{37}$$

All maps are ReLU multilayer perceptrons and the scalar output predicts the fixed-(x, ξ) value.

H Implementation and Training

Computational setup. Experiments use an AMD EPYC 7763 central processing unit (CPU) and NVIDIA A100 graphics processing units (GPUs). Optimization uses Gurobi 13.0.1 and `gurobi-machinelearning` 1.5.6; neural training uses PyTorch 2.6.0, CUDA 12.4, and Python 3.10. Within each benchmark, both architectures use the same optimization code, solver settings, and run limits.

Training data. Each group fixes an instance and a first-stage decision and varies the uncertainty realization. Labels are exact fixed-(x, ξ) recourse values computed by the benchmark-owned second-stage solver. Table 7 reports the splits used by the current main experiments. CB and RAP use the same group-disjoint split for both architectures. DFLP is additionally source-instance-disjoint between training and validation.

All reported models minimize mean squared error with Adam for 500 epochs. Sum aggregation and zero dropout are used throughout. RAP uses Adam weight decay 10^{-2}; the other reported models use no weight decay. All reported checkpoints are selected by validation Recovery@1.

Selected architectures. Table 8 reports the layer widths and parameter counts stored in the deployed checkpoints. A sequence such as $5 \to 24 \to 6$ gives the input, hidden, and output widths of an element network. The post-aggregation and output networks are listed separately.

Leakage controls. Every stored dataset carries group identifiers and a declared feature schema. The release audit compares instance-parameter fingerprints between training and test assets. It finds zero overlaps for CB (4,595 training groups and 250 test instances), RAP (2,921 and 250), and DFLP (800 groups, 160 training instances, and 160 test instances). For DFLP, the same audit also verifies

Table 8: Architectures used in the main experiments.

Benchmark	Architecture	Row input	Element network	Post-aggregation	Output network	Parameters
CB	CSD	5	$5 \to 24 \to 6$	$6 \to 48 \to 10$	$10 \to 10 \to 1$	**1,153**
	Separate	$3/4$	$3/4 \to 16 \to 4$	$4 \to 32 \to 8$	$16 \to 8 \to 1$	**1,153**
RAP	CSD	5	$5 \to 24 \to 6$	$6 \to 48 \to 12$	$12 \to 12 \to 1$	1,297
	Separate	$4/4$	$4 \to 16 \to 4$	$4 \to 32 \to 8$	$16 \to 8 \to 1$	1,169
DFLP	CSD	10	$10 \to 18 \to 6$	$6 \to 24 \to 8$	$8 \to 8 \to 1$	**705**
	Separate	$9/9$	$9 \to 10 \to 4$	$4 \to 14 \to 8$	$16 \to 6 \to 1$	**705**

zero source and group overlap between training and validation and that every group belongs to one source instance.

I Evaluation Protocols

Statistical unit and aggregation. A downstream observation consists of a test case, a training seed, and an architecture. Each gap is computed for that observation before aggregation. The main tables pool observations over the reported training seeds and take the median within each size group. The overall row takes the median over all included observations across sizes. For DFLP, a test case is an instance paired with a disruption budget Γ.

Run inclusion. Runs that reach the time limit and return a valid first-stage decision are included. Runtime and iteration statistics use runs with a returned decision; gap statistics additionally require a completed independent evaluation. For DFLP, reference-gap statistics use certified fixed-decision evaluations. Runs without a returned decision and unavailable gap evaluations do not contribute numerical observations to the corresponding medians.

I.1 Capital budgeting common-pool gap

The common-pool gap measures progress toward the robust optimum using the scenarios generated by both architectures. For an instance I and training seed, let Ξ_I^{CSD} and Ξ_I^{sep} be the distinct scenarios generated by CSD and Separate, respectively. Their common pool is

$$\Xi_I^{\mathrm{pool}} = \Xi_I^{\mathrm{CSD}} \cup \Xi_I^{\mathrm{sep}}.$$

For architecture $a \in \{\mathrm{CSD}, \mathrm{sep}\}$, let x_I^a be its returned first-stage decision. We remove the neural network and solve the standard MP over Ξ_I^a to obtain the lower bound L_I^a. The decision is evaluated on the common pool as

$$\widehat{Q}_I^{\mathrm{pool}}(x_I^a) = \max_{\xi \in \Xi_I^{\mathrm{pool}}} Q_I(x_I^a, \xi).$$

The reported CB gap is

$$\mathrm{Gap}_{I,\mathrm{pool}}^a = 100\, \frac{\widehat{Q}_I^{\mathrm{pool}}(x_I^a) - L_I^a}{|L_I^a|}. \tag{38}$$

Adding scenarios strengthens the standard MP lower bound and expands the scenario set used to evaluate a fixed decision. The common pool supplies a shared set of evaluation scenarios, and the gap measures the distance between each run's lower bound and its returned decision's value over this pool.

I.2 Reference gaps

Let z_I^a denote the independently evaluated objective of the decision returned by architecture a, and let z_I^{ref} denote the published reference value. RAP is a reward-maximization problem, for which we report

$$\mathrm{Gap}_{I,\mathrm{RAP}}^a = 100\, \frac{z_I^{\mathrm{ref}} - z_I^a}{\max\{|z_I^{\mathrm{ref}}|, 10^{-12}\}}. \tag{39}$$

Table 9: Validation ranking performance (mean ± standard deviation across training seeds 11, 17, and 23).

Benchmark	Architecture	Groups	Recovery@1 ↑	Recovery@5 ↑	MRR ↑
CB	CSD	919	$\mathbf{0.561 \pm 0.005}$	$\mathbf{0.933 \pm 0.000}$	$\mathbf{0.719 \pm 0.003}$
	Separate	919	0.472 ± 0.004	0.818 ± 0.002	0.622 ± 0.004
RAP	CSD	585	$\mathbf{0.411 \pm 0.001}$	$\mathbf{0.881 \pm 0.001}$	$\mathbf{0.604 \pm 0.000}$
	Separate	585	0.281 ± 0.007	0.750 ± 0.008	0.482 ± 0.007
DFLP	CSD	160	$\mathbf{0.715 \pm 0.011}$	$\mathbf{0.894 \pm 0.033}$	$\mathbf{0.795 \pm 0.017}$
	Separate	160	0.292 ± 0.013	0.688 ± 0.018	0.467 ± 0.018

Here z_I^{ref} is the published Multi NF reference value (Bodur et al., 2026). DFLP is a minimization problem, for which we report

$$\mathrm{Gap}^a_{I,\mathrm{DFLP}} = 100\,\frac{z_I^a - z_I^{\mathrm{ref}}}{\max\{|z_I^a|, |z_I^{\mathrm{ref}}|, 10^{-12}\}}. \tag{40}$$

Here z_I^{ref} is the published optimum when available and the published C&CG lower bound otherwise. Gaps are computed per run before taking medians over the reported instances and training seeds.

I.3 Prediction metrics

For validation group g, let q_{gk} and s_{gk} be the exact value and predicted score of candidate k, respectively. Let $T_g = \arg\max_k q_{gk}$ and let

$$r_g = \min_{k \in T_g} \mathrm{rank}_g(k),$$

where $\mathrm{rank}_g(k)$ is the position of candidate k after sorting s_{gk} in descending order. With G validation groups, mean reciprocal rank (MRR) is defined together with Recovery@K as

$$\mathrm{Recovery@K} = \frac{1}{G}\sum_{g=1}^{G} \mathbf{1}[r_g \le K], \qquad \mathrm{MRR} = \frac{1}{G}\sum_{g=1}^{G} \frac{1}{r_g}. \tag{41}$$

Thus all exact maximizers are accepted when the true maximum is tied. For the controlled experiment, value-prediction quality is

$$R^2 = 1 - \frac{\sum_n (s_n - q_n)^2}{\sum_n (q_n - \bar{q})^2}, \tag{42}$$

computed over all held-out candidate values. Validation tables first average each metric over groups for each training seed and then report the mean and standard deviation across seeds.

J Validation Diagnostics

Prediction diagnostics are computed on validation groups using the metrics in Appendix I.3. Each group fixes an instance and first-stage decision while varying candidate scenarios.

J.1 DFLP Correspondence Ablation

Table 10 applies the controlled correspondence change defined in Appendix E to the complete DFLP validation split. All three rows use 705 trainable parameters, the same 160 validation groups, and the same three training seeds.

Shuffling correspondence reduces Recovery@1 from 0.715 to 0.260 and produces validation performance in the same range as separate aggregation. Because the change preserves model size, data split, marginal inputs, labels, and training budget, the result isolates the aligned component information used before aggregation.

Table 11 reports the corresponding downstream results. It follows the problem-size grouping and per-run median aggregation used in Table 3.

Table 10: DFLP validation results after ablating component correspondence.

Architecture	Recovery@1 ↑	Recovery@5 ↑	MRR ↑
CSD	$\mathbf{0.715 \pm 0.011}$	$\mathbf{0.894 \pm 0.033}$	$\mathbf{0.795 \pm 0.017}$
CSD (shuffled correspondence)	0.260 ± 0.016	0.660 ± 0.041	0.433 ± 0.015
Separate	0.292 ± 0.013	0.688 ± 0.018	0.467 ± 0.018

Table 11: Median DFLP downstream results after ablating component correspondence.

			CSD (shuffled correspondence)			CSD	Separate
$\lvert F\rvert$	$\lvert C\rvert$	**Instances**	**Gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Gap (%)** ↓	**Gap (%)** ↓
10	20	60	100.00	0.70	2	**0.00**	100.00
10	30	60	100.00	0.97	2	**0.00**	100.00
10	40	60	100.00	1.12	2	**0.00**	100.00
10	50	60	84.23	12.68	3	**0.00**	100.00
15	20	60	72.56	0.99	2	**0.78**	71.10
15	30	60	54.86	1.58	2	**0.00**	54.85
15	40	60	63.06	4.48	2	**2.67**	61.26
15	50	60	50.34	13.02	3	**1.76**	56.76
All		480	82.32	1.36	2	**0.00**	84.40

Results pool training seeds 11, 17, and 23. The CSD and Separate gap columns reproduce Table 3. Each entry is the median over case–seed observations; Instances counts distinct test cases.

The validation and downstream results show the same pattern. Table 10 shows that shuffling reduces Recovery@1 from 0.715 to 0.260, close to Separate at 0.292. Table 11 reports all 1,440 certified evaluations of the shuffled model over three training seeds. Its median iteration count is 2, compared with 9 for CSD and 1.5 for Separate. Its median gap is 82.32%, compared with 0.00% for CSD and 84.40% for Separate.

The result is consistent across the eight problem-size groups. For $|F| = 10$, shuffled correspondence has median iteration counts of two to three and median gaps between 84.23% and 100.00%, whereas normal CSD has zero median gap. For $|F| = 15$, the shuffled gaps range from 50.34% to 72.56% and differ from Separate by at most 6.42 percentage points. Because shuffling changes only the component correspondence, the observed changes in adversarial-scenario identification, learned termination, and downstream decisions are attributable to the correspondence available before aggregation.

J.2 DFLP Explicit-Interaction Ablation

We augment Separate with a scalar interaction feature: the sum of the products of the normalized opening decision and availability over aligned facility–customer rows. This feature is concatenated with the two separately aggregated representations before the value head. The resulting model has 711 parameters, compared with 705 for Separate and CSD. Training uses the same 16,479 labeled samples, split, optimizer, 500-epoch budget, and checkpoint-selection rule. Across the three training seeds, mean validation Recovery@1 is 0.600, compared with 0.292 for Separate and 0.715 for CSD.

Table 12 follows the layout and median aggregation of Table 11. The explicit-interaction model is evaluated on all 480 test cases for each of three training seeds, giving 1,440 independently certified decision evaluations. The learned C&CG iteration limit, solver budgets, stopping rules, and reference-gap definition follow the DFLP protocol. Recorded solve times include both the generic network embedding and its accelerated formulation using exact binary interpolation and finite-bound propagation in the MP and learned AP.

With the explicit interaction, the overall median reference gap is 12.55%, and the median iteration count is 8. Median gaps range from 5.31% to 22.57% across the eight size groups. The validation results show that supplying an aligned interaction feature improves adversarial scenario identification within the Separate architecture; CSD achieves higher Recovery@1 by learning interactions at the component level.

Table 12: Median DFLP downstream results with an explicit interaction feature added to Separate.

			Separate + explicit interaction			CSD	Separate
$\|F\|$	$\|C\|$	**Instances**	**Gap (%)** ↓	**Time (s)** ↓	**Iterations**	**Gap (%)** ↓	**Gap (%)** ↓
10	20	60	10.81	195.61	9	**0.00**	100.00
10	30	60	13.82	28.26	10	**0.00**	100.00
10	40	60	5.31	18.69	10	**0.00**	100.00
10	50	60	17.82	14.26	7	**0.00**	100.00
15	20	60	6.92	29.84	12	**0.78**	71.10
15	30	60	8.49	41.11	10	**0.00**	54.85
15	40	60	22.57	14.99	6	**2.67**	61.26
15	50	60	14.44	46.55	7	**1.76**	56.76
All		480	12.55	29.36	8	**0.00**	84.40

Results pool training seeds 11, 17, and 23. The CSD and Separate gap columns reproduce Table 3. Each entry is the median over case–seed observations; Instances counts distinct test cases.

Table 13: Test-set provenance and reference-value semantics.

Benchmark	**Evaluation grid**	**Cases**	**Published reference**
CB	$n \in \{10, 20, 30, 40, 50\}$; 50 instances per size	250	Comparison solutions; common-pool diagnostic in this work
RAP	$(n, K) \in \{(20, 2), (20, 3), (20, 4), (25, 5), (25, 8)\}$	250	Published Multi NF reference values
DFLP	$\|F\| \in \{10, 15\}$, $\|C\| \in \{20, 30, 40, 50\}$; $\Gamma \in \{2, 3, 4\}$	480	433 optima and 47 C&CG lower bounds

K Test-Set Provenance

Table 13 records the complete grids and the semantics of their published reference columns. Reference values are consumed as provided; they are not recomputed during learned C&CG.

CB instances are from the Neur2RO release (Dumouchelle et al., 2024); RAP instances and Multi NF reference values are from Bodur et al. (2026); and DFLP instances and reference tables are from Lefebvre et al. (2024).

L Variation in Downstream Gaps

Figure 6 reports the downstream gap by problem-size group. The curves reproduce the medians in the main tables, and the bands have the following definitions. For CB and DFLP, the lower and upper boundaries are the minimum and maximum of the seed-specific median gaps within each size group. For RAP, the boundaries are the 16th and 84th percentiles of bootstrap medians: each of 10,000 bootstrap samples draws 50 test cases with replacement from the corresponding size group of the plotted run. Bootstrap sampling uses random seed 2027.

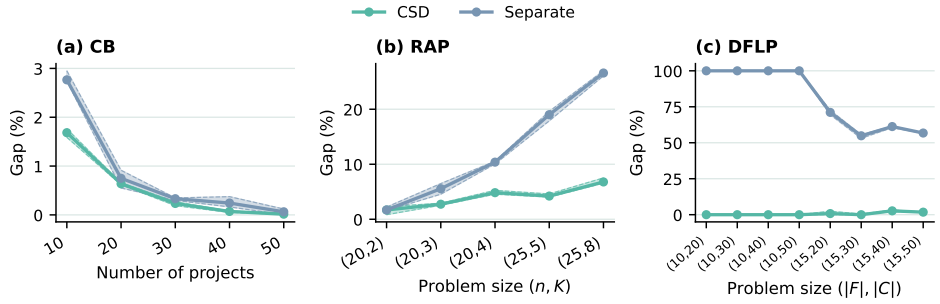


Figure 6: Median downstream gaps across problem sizes. CB and DFLP bands span the seed-specific medians; RAP bands show the 16th–84th percentiles of medians from bootstrap resampling of test cases.

M ADDITIONAL SOLVE RESULTS

Table 14 expands the DFLP termination analysis from Section 6. Each entry uses the exact AP at the returned first-stage decision and the accepted scenario set, according to the protocol in Appendix E.

Table 14: DFLP learned-termination audit on the controlled subset.

Architecture	Runs	Omitted scenario	Rate (%)	Iterations	Omitted deterioration
CSD	27	**7**	**25.9**	9	**0.00**
Separate	27	27	100.0	2	478.46

The exact AP finds an omitted adverse scenario when its worst recourse value exceeds the largest exact value among accepted scenarios by more than the AP tolerance. The corresponding median normalized reference gaps are 0.0002% for CSD and 100.0% for Separate.