Learning to Branch In Context: Adaptive Variable Selection for MILP

Anonymous authors
Paper under double-blind review

Abstract

A branch-and-bound solve produces observations of how individual branching variables affect the search. Pseudocost rules use these observations to update variable-specific estimates, whereas learned branchers usually consume them only through solver-maintained features or a history of states and actions. We study in-context branching (ICB), which exposes observed directional bound gains as an explicit, identity-linked memory. A learned prior and an evidence-weighted estimator predict directional gains; a fixed scoring rule selects the branching variable. The model is trained by strong-branching distillation and uses new observations without updating its parameters. On canonical combinatorial-auction instances, removing memory from the same checkpoint increases the medium-size node aggregate from 1,205 to 3,204, with all runs solved. On a reconstructed class-disjoint MILP-Evolve cohort, context raises the solved fraction from 55.6% to 69.1% and reduces the paired primal–dual integral by approximately 15%. Comparisons with reproduced branching policies are family-dependent: some transfer results are competitive, but reliability pseudocost branching remains stronger overall. The experiments distinguish improvements due to event memory from those retained when the same policy is evaluated without memory.

1 Introduction

Branching variable selection determines how a branch-and-bound solver partitions a mixed-integer linear program (MILP). Different choices can lead to markedly different search trees (Achterberg et al., 2005). Strong branching (SB) evaluates candidate variables by temporarily solving their child linear programs. Its computational cost motivates learned approximations, including feature-based rankers (Khalil et al., 2016), graph neural networks (Gasse et al., 2019), and hybrid graph–feature models (Gupta et al., 2020). These policies reduce the cost of selecting a variable, but approximating a local expert does not guarantee a small complete search tree (Cheng & Basu, 2026).

Classical branching rules also learn from the current solve. Pseudocost branching maintains directional gain estimates for each variable. Reliability pseudocost branching (RPB) supplements these estimates with SB when the available observations are insufficient (Achterberg et al., 2005; Achterberg & Berthold, 2009). A learned policy may receive the resulting statistics as state features. It may also condition on previous candidate sets and selected actions, as in Mamba-Branching (Zhang et al., 2026). Neither representation makes the raw numerical outcomes and their variable attribution explicit. We ask: does an identity-linked memory of observed branch outcomes improve a learned policy during a single solve?

We address this question with *in-context branching* (ICB). Each usable outcome is recorded only after the corresponding child LP bound becomes available. The event identifies the originating variable, the branch direction, and the gain per unit of variable movement. A candidate queries this memory through a given identity relation. Its prediction combines a feature-based prior, weighted evidence from matching events, and a learned residual. An optional similarity channel uses events from other variables. Training distills SB decisions on exploration-diversified trajectories. Deployment changes the memory but not the network parameters.

This formulation separates the prior learned across instances from the information acquired within an instance. It is related to in-context learning of decision rules (Laskin et al., 2023; Lee et al., 2023), with supervised branching decisions as the training target. Identity matching is supplied by the solver.

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.

The learned components determine how outcomes affect branching estimates; they are not required to rediscover equality between variable identifiers.

We evaluate the complete policy against reproduced branchers and separately intervene on the memory of a fixed checkpoint. On combinatorial auctions, context substantially improves performance under size transfer. On a reconstructed MILP-Evolve benchmark (Li et al., 2025), the improvement persists for held-out problem classes. Independent set provides a different case: its favorable transfer results largely remain when memory is removed. These results establish where explicit outcome memory is useful without attributing every improvement of the complete policy to in-context adaptation.

Our contributions are threefold.

- *An explicit feedback interface for learned branching.* Observed directional branch outcomes form an identity-linked memory that is queried by the current candidates, without parameter updates during deployment.
- *A separation of context utility from policy quality.* Fixed-checkpoint interventions on the memory are combined with comparisons against reproduced external branchers, distinguishing event-driven improvements from those of the learned prior and training recipe.
- *Evaluation under size and class shift.* Canonical Gasse cells and a class-disjoint MILP-Evolve reconstruction show substantial but family-dependent benefits of event memory, together with negative controls and limitations of the complete policy.

2 Background and related work

Branching in MILP. Consider a minimization problem

$$\min_{x\in\mathbb{R}^n} c^\top x \quad \text{subject to} \quad Ax \le b, \quad l \le x \le u, \quad x_j \in \mathbb{Z}\ (j \in \mathcal{I}), \tag{1}$$

where $\mathcal{I} \subseteq \{1,\dots,n\}$ indexes integer variables. At a search node u, let x^u solve its LP relaxation and let L_u be the corresponding lower bound. A fractional candidate j gives two children with constraints $x_j \le \lfloor x_j^u \rfloor$ and $x_j \ge \lceil x_j^u \rceil$. SB estimates the quality of this split from both child LPs. Pseudocost rules replace repeated lookahead with observed per-unit gains. RPB combines these estimates with selective lookahead (Achterberg et al., 2005; Achterberg & Berthold, 2009). We change variable selection; node selection, bounding, and pruning remain solver operations.

Learning to branch. Existing approaches differ in where adaptation to the current instance resides: in solver-maintained statistics, as in pseudocost rules and RPB; in models fitted within the instance; in encoders learned across instances; or in sequence models conditioned on the search history. Khalil et al. (2016) fit an instance-specific ranker from early SB decisions. Gasse et al. (2019) instead learn a policy across instances using a bipartite variable–constraint graph. Subsequent work studies efficient hybrid encoders (Gupta et al., 2020), search-tree features (Zarpellon et al., 2021), parent–child regularization (Gupta et al., 2022), and symbolic policies (Kuang et al., 2024). Reinforcement-learning methods address the longer-horizon cost of branching (Etheve et al., 2020; Scavuzzo et al., 2022; Parsonson et al., 2023), and ranking methods use richer supervision than a single expert action (Huang et al., 2022). Mamba-Branching models sequences of candidate sets and actions (Zhang et al., 2026). These methods differ in their inputs and training objectives; many already consume historical solver statistics. Our distinction is explicit access to attributed numerical outcomes and the evaluation of that information channel.

In-context decision making. A model trained across tasks can change its predictions from examples in its input without updating its weights (Garg et al., 2022). Algorithm distillation learns from the interaction histories of an improving learner (Laskin et al., 2023); decision pretraining predicts expert actions from a query and preceding interactions (Lee et al., 2023). In ICB, a task is one MILP solve, an observation is a completed directional branch outcome, and a query is the current candidate set. MILP-Evolve provides a complementary test of this construction across problem classes (Li et al., 2025).

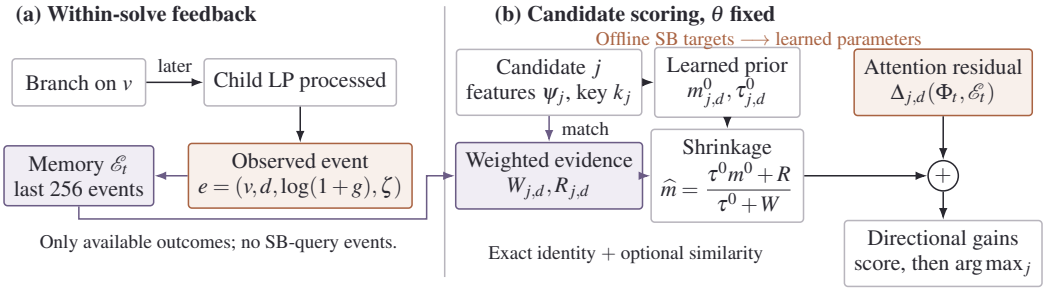

Figure 1: **In-context branching.** An executed branch produces a usable event only after the corresponding child bound becomes available. At a later decision, given variable identities route this evidence to the current candidates. A learned prior and weighted evidence form directional shrinkage estimates, followed by an attention residual and a fixed branching readout. Strong-branching labels supervise training only. Deployment updates the memory, not the parameters.

3 In-context branching

3.1 A causal event interface

Decisions and observations have separate indices. At decision t, the solver exposes a fractional candidate set C_t and feature vectors $\Phi_t = (\phi_{j,t})_{j \in C_t}$, with $\phi_{j,t} \in \mathbb{R}^{40}$. These include LP quantities, constraint statistics, and historical solver features. Let $\psi_{j,t} = \mathcal{N}(M \odot \phi_{j,t})$, where M is a fixed binary mask and $\mathcal{N}$ is the training-set normalization. Our main models mask the pseudocost score and directional count features, columns 8–14, in both candidate vectors and saved event descriptors. These statistics are policy-dependent: during collection they arise from the exploration trajectory, whereas at deployment they would arise from the learned policy's own decisions. Masking removes this direct channel and improved deployment robustness in our comparisons (Section 5); other historical features remain available (Appendix A.2).

An event is indexed by its availability order ℓ. Suppose child u was created by branching on variable v_ℓ in direction $d_\ell \in \{\downarrow, \uparrow\}$ at parent $p(u)$. Define

$$g_\ell = \frac{[L_u - L_{p(u)}]_+}{\max(\delta_{v_\ell, d_\ell}, \varepsilon)}, \qquad \delta_{j,\downarrow} = x_j^{p(u)} - \lfloor x_j^{p(u)} \rfloor, \quad \delta_{j,\uparrow} = 1 - \delta_{j,\downarrow}, \tag{2}$$

where $[z]_+ = \max(z, 0)$ and $\varepsilon = 10^{-9}$. Bounds are written in minimization convention. A usable event is $e_\ell = (v_\ell, d_\ell, \gamma_\ell, \zeta_\ell)$, where $\gamma_\ell = \log(1 + g_\ell)$ and ζ_ℓ is the normalized, masked feature vector saved when the branch was executed. It is published only when the child bound is known. Non-finite bounds and ambiguous directions are omitted; a branch need not produce two usable events.

Let a_ℓ^{obs} and a_t^{dec} be positions in the ordered solver callback stream, not physical times. The memory at a decision is

$$\mathcal{E}_t = \text{tail}_B\big((e_\ell)_{a_\ell^{\text{obs}} < a_t^{\text{dec}}}\big), \qquad B = 256. \tag{3}$$

This definition excludes future outcomes and offline SB queries. The buffer is reset for every solve. Figure 1 separates the feedback path from the training-only supervision path.

Our experimental questions are stated relative to interventions on this interface.

Definition 1. *A frozen policy is feedback-dependent if, for a fixed query (C, Φ), changing the available memory $\mathcal{E}$ changes its output. Feedback is useful with respect to a prediction or solver metric if the correctly attributed memory improves that metric relative to controlled interventions on the memory—its removal, or permutations of its attributions or values—with all other inputs and the parameters held fixed.*

Feedback dependence alone does not show that a beneficial online estimation rule has been learned; Sections 4.3 and 4.4 therefore measure usefulness directly.

3.2 Identity-conditioned gain estimation

For readability, we suppress t below and index only the events in $\mathcal{E}_t$. Three notions of identity must be kept apart. First, the solver's persistent variable index establishes which events belong to which candidate: it supplies $m_{j\ell} = \mathbf{1}\{j = v_\ell\}$, and this exact-match relation is given to the architecture. What the network must learn is how to use matched outcomes—how much to trust them and how to combine them with the prior—not how to equate identifiers. Second, the input representation of identity is a random key $k_j \sim \mathcal{N}(0, I_{16}/16)$, shared by a variable's candidate and event representations, fixed within a solve, and resampled for each training presentation. The network receives keys rather than global variable indices; this removes a fixed identifier vocabulary, not the possibility of learning from recurring structural patterns. Third, saved structural descriptors ζ_ℓ carry information about a variable without unique identity; they feed only the similarity channel below. Conditioning on the per-solve key map is implicit in the policy notation.

Prior and evidence. A feature network predicts a directional prior and positive pseudo-counts: $(m_j^0, \tau_j^0) = f_\theta^{\text{prior}}(f_j, \psi_j)$, where f_j is the current LP fractionality, $m_j^0 \in \mathbb{R}^2_{\geq 0}$, and $\tau_j^0 \in \mathbb{R}^2_{>0}$. An event network maps (d_ℓ, γ_ℓ) to a nonnegative weight w_ℓ and a transformed gain r_ℓ. The prior does not receive the random key.

The similarity channel assigns attention only to nonmatching variables. For $b_{j\ell} = q_\theta(f_j, \psi_j)^\top k_\theta^{\text{str}}(\zeta_\ell)/\sqrt{d_s}$, set

$$\alpha_{j\ell} = \frac{(1 - m_{j\ell}) \exp b_{j\ell}}{\sum_{r:\, m_{jr}=0} \exp b_{jr}}, \qquad a_{j\ell} = m_{j\ell} + \rho \alpha_{j\ell}, \quad \rho \geq 0. \tag{4}$$

If the denominator is empty, all $\alpha_{j\ell}$ are zero. The normalization is over nonmatching events of both directions; aggregation then separates the directions. With $\rho = 0$, only the exact-identity channel remains. The evidence statistics are

$$W_{j,d} = \sum_\ell \mathbf{1}\{d_\ell = d\} a_{j\ell} w_\ell, \qquad R_{j,d} = \sum_\ell \mathbf{1}\{d_\ell = d\} a_{j\ell} w_\ell r_\ell. \tag{5}$$

Matching and summation are deterministic. The event transformation, weights, similarity projections, and trust parameter ρ are learned.

Shrinkage and residual. The intermediate log-gain estimate is

$$\widehat{m}_{j,d} = \frac{\tau_{j,d}^0 m_{j,d}^0 + R_{j,d}}{\tau_{j,d}^0 + W_{j,d}}. \tag{6}$$

When $W_{j,d} > 0$, this is a convex combination of $m_{j,d}^0$ and $R_{j,d}/W_{j,d}$ with prior weight $\tau_{j,d}^0/(\tau_{j,d}^0 + W_{j,d})$. Empty evidence gives $\widehat{m}_{j,d} = m_{j,d}^0$. We use shrinkage as a parameterization, not as a calibrated Bayesian posterior. It operates in transformed gain space and need not equal an arithmetic pseudocost average.

A residual network uses four pre-norm cross-attention blocks of width 256. Candidates attend to events; each head adds a learned multiple of $m_{j\ell}$ to its attention logits. The zero-initialized output head produces $\Delta_{j,d}$, giving

$$\widetilde{m}_{j,d} = [\widehat{m}_{j,d} + \Delta_{j,d}]_+, \qquad \widehat{g}_{j,d} = \exp(\widetilde{m}_{j,d}) - 1. \tag{7}$$

The prior, event weights, and residual are trained jointly. Equation (6) reduces to its prior for empty memory; the complete network additionally contains the residual path. Consequently, context-off denotes the state-only restriction of the trained network, not necessarily the prior subnetwork alone. Appendix A.4 states the exact algebraic properties and their scope.

Readout. For a candidate j, the predicted child improvements are $u_{j,\downarrow} = f_j \widehat{g}_{j,\downarrow}$ and $u_{j,\uparrow} = (1 - f_j)\widehat{g}_{j,\uparrow}$. Our implementation uses

$$S_j = (1 - \kappa) \min(u_{j,\downarrow}, u_{j,\uparrow}) + \kappa \max(u_{j,\downarrow}, u_{j,\uparrow}), \qquad \kappa = 0.167, \tag{8}$$

$$\ell_{\theta,j} = \beta \log(1 + S_j), \qquad p_\theta(j \mid \Phi_t, \mathcal{E}_t) = \frac{\exp \ell_{\theta,j}}{\sum_{i \in C_t} \exp \ell_{\theta,i}}, \qquad a_t \in \operatorname*{arg\,max}_{j \in C_t} \ell_{\theta,j}. \tag{9}$$

Algorithm 1 ICB within one solve (parameters θ frozen)

Require: frozen θ; capacity $B = 256$; mask M and normalizer $\mathcal{N}$; solver callbacks arrive in chronological order

1: $\mathcal{E} \leftarrow \emptyset$; a key $k_v \sim \mathcal{N}(0, I_{16}/16)$ is drawn for each variable v on first use and kept for the solve
2: **while** the solver is running **do**
3: **on** child bound L_u known for branching (v, d) at $p(u)$: $g \leftarrow [L_u - L_{p(u)}]_+ / \max(\delta_{v,d}, \varepsilon)$ (2); if usable, $\mathcal{E} \leftarrow \mathrm{tail}_B\big(\mathcal{E} \,\|\, (v, d, \log(1{+}g), \zeta_v)\big)$
4: **on** branching request t with (C_t, Φ_t):
5: $\psi_j \leftarrow \mathcal{N}(M \odot \phi_{j,t})$ and $m_{j\ell} \leftarrow \mathbf{1}\{j = v_\ell\}$ for $j \in C_t$, $e_\ell \in \mathcal{E}$ ▷ given identity
6: $(m_j^0, \tau_j^0) \leftarrow f_\theta^{\text{prior}}(f_j, \psi_j)$; $(w_\ell, r_\ell) \leftarrow f_\theta^{\text{ev}}(d_\ell, \gamma_\ell)$; (W_j, R_j) by (4)–(5)
7: $\widetilde{m}_j \leftarrow \big[(\tau_j^0 m_j^0 + R_j)/(\tau_j^0 + W_j) + \Delta_j(\Phi_t, \mathcal{E})\big]_+$ (6)–(7) ▷ one batched forward pass
8: **return** $a_t \in \arg\max_{j \in C_t} \log(1 + S_j)$ to the solver, with S_j from (8)
9: **end while**

Here β is the implementation's logit scale. The weighted min–max rule specifies our readout; native SCIP score settings are configured separately. There is no independent action head. For a positive logit scale, monotonicity of $\log(1 + \cdot)$ makes logit and score maximization equivalent. We write $\pi_\theta(\Phi_t, \mathcal{E}_t) = a_t$ for the deployed decision rule, with ties resolved by the implementation.

3.3 Strong-branching distillation

Collection alternates pseudocost branching and RPB in segments of 5–25 decisions. At each recorded decision, with probability 0.15, a side-LP procedure evaluates all candidates without changing the executed trajectory. These evaluations provide SB scores $q_{t,j}^{\text{SB}}$ and a selected target $a_t^{\text{SB}} \in \arg\max_j q_{t,j}^{\text{SB}}$. They supply training labels, not events. Only fully labelled decisions enter the supervised query set.

For $z_t = (\Phi_t, \mathcal{E}_t, a_t^{\text{SB}}, Y_t)$, Y_t contains the stored log-transformed directional targets from replaying the event-derived pseudocost table. The training objective is

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{z_t \in \mathcal{D}} \big[-\log p_\theta(a_t^{\text{SB}} \mid \Phi_t, \mathcal{E}_t) + \lambda \, \text{Huber}(\widetilde{m}_t, Y_t) \big], \qquad \lambda = 1. \tag{10}$$

The classification term distills lookahead decisions, while the auxiliary term anchors directional predictions to replayed outcomes. These are different targets. In particular, $\log(1 + \bar{g})$ is not generally equal to the average of $\log(1{+}g)$. The reported checkpoints do not use the optional listwise-KL term. Model selection and recipe exceptions are specified in Appendix C. This objective is a surrogate; it does not directly minimize search-tree size.

3.4 Deployment and context interventions

Algorithm 1 summarizes one solve. Line 3 is the observation side: an outcome is attributed to the executed branching, converted to a unit gain, and appended to the rolling buffer; ineligible outcomes are dropped. Lines 4–8 are the decision side: masked features, the given match relation, prior and event networks, deterministic evidence aggregation, the residual, and the fixed readout are evaluated for all candidates in one forward pass, and control returns to the solver. Three invariants hold. *Causality*: the decision at t depends only on events with $a_\ell^{\text{obs}} < a_t^{\text{dec}}$, Eq. (3). *Frozen parameters*: θ and the normalization statistics are never updated; adaptation flows through $\mathcal{E}_t$ alone. *Cold start*: with $\mathcal{E}_t = \emptyset$ the evidence sums vanish and the shrinkage estimate equals the prior (Proposition 1), while the residual path still acts on the state, so context-off is the state-only restriction of the trained network. Both candidate sets and event histories depend on the policy's earlier decisions.

For an intervention T on memory, define $\pi_\theta^T(\Phi, \mathcal{E}) = \pi_\theta(\Phi, T(\mathcal{E}))$ and, for a solver cost C,

$$J_C(T) = \mathbb{E}_{I,\xi}\big[C\big(\text{Solve}(I, \xi; \pi_\theta^T)\big)\big], \tag{11}$$

where ξ collects solver and key randomness. Comparing $J_C(T)$ to $J_C(\text{id})$ measures usefulness in the sense of Definition 1 at the level of the entire solve. It does not hold later search states fixed. In particular, $T_\varnothing(\mathcal{E}) = \varnothing$ tests a checkpoint's dependence on memory, not its advantage over a separately trained state-only model. Offline value and attribution permutations instead compare predictions at fixed queries. The two protocols answer different questions.

Table 1: **Canonical Gasse benchmark.** Each cell has 20 instances × five solver seeds in SCIP 6.0.1, with a 3,600s CPU-clock limit. Time (s) and processed nodes are one-shifted geometric means over all runs. A timed-out run does not yield a complete-tree cost. RPB, GCNN and Mamba are reruns; dashes denote missing evaluations. Paired complete-tree comparisons and context-off rows appear in Appendices D.2 and D.3.

		Easy			Medium			Big		
Family	Model	solved	time	nodes	solved	time	nodes	solved	time	nodes
Set cover	RPB	100%	11.6	54	100%	77.2	2,326	65%	2,035	113,767
	GCNN	–	–	–	–	–	–	67%	1,857	74,270
	Mamba	100%	10.7	488	100%	147.4	9,889	28%	3,342	178,142
	ICB	100%	10.5	298	100%	114.0	4,153	35%	4,981	83,349
Comb. auctions	RPB	100%	5.0	11	100%	29.3	696	100%	207.3	9,173
	GCNN	100%	3.4	71	100%	35.7	673	100%	343.6	7,973
	Mamba	100%	3.1	133	100%	34.2	2,946	100%	462.5	31,817
	ICB	100%	3.5	152	100%	31.1	1,205	95%	424.5	14,371
Facility loc.	RPB	100%	25.0	22	100%	152.8	155	93%	955.6	147
	GCNN	100%	27.1	106	100%	147.2	333	95%	539.1	387
	Mamba	100%	21.2	61	100%	115.1	346	99%	439.4	325
	ICB	100%	24.5	188	100%	123.9	412	96%	560.5	504
Indep. set	RPB	100%	8.4	22	100%	124.6	2,618	45%	2,346	55,336
	GCNN	–	–	–	47%	1,235	86,194	1%	3,491	116,901
	Mamba	100%	10.8	98	79%	635.7	13,034	7%	3,481	42,350
	ICB	100%	40.0	72	95%	508.1	1,383	26%	6,370	14,107

4 Experiments

4.1 Protocol

Benchmarks and training. The canonical Gasse benchmark (Gasse et al., 2019) contains set covering, combinatorial auctions, capacitated facility location, and maximum independent set at easy, medium, and big sizes. Each cell evaluates 20 MILPs with five solver seeds. We train one model per family on easy instances; medium and big instances test size transfer. We also evaluate a single generalist on held-out classes of a MILP-Evolve reconstruction (Li et al., 2025). Multiple knapsack and extended transfer sets appear in Appendix D.

Comparators. We rerun RPB, the GCNN of Gasse et al. (2019), and Mamba-Branching (Zhang et al., 2026). The learned baselines use the authors' implementations under a common solver configuration. Their features, training collections, and teachers are not identical to ours. These are system comparisons, not matched-data architecture ablations. GCNN inference uses a GPU; ICB and Mamba inference use CPU in these runs. ICB context-off uses the corresponding full-context checkpoint without retraining. Main tables use the checkpoints recorded in the experiment registry. Development included live canonical-cell comparisons of alternative recipes; Appendix C identifies this selection exposure.

Solver and measurements. Our comparisons use SCIP 6.0.1, root-only cuts, no restarts, default primal heuristics, no objective cutoff, and permutation seeds 0–4. Gasse runs have a 3,600-second internal CPU-clock limit. Reported time is external wall time, including Python inference and contention on a shared host; it can exceed the CPU limit. We therefore do not interpret these timings as isolated latency measurements.

We report completion rates and one-shifted geometric means. A timed-out run contributes its *processed* node count, not the size of a complete proof. For complete trees we use pairwise jointly solved runs, preserving (instance, seed) pairs and bootstrapping over instances (Appendix D.2). These conditional comparisons must be read with completion rates. MILP-Evolve additionally uses SCIP's relative primal–dual integral (PDI) under the same 200-second CPU-clock budget for all policies. Aggregate definitions are in Appendix D.1.

4.2 Comparison with reproduced branchers

Table 1 shows substantial variation across families. On set-covering and auction transfer instances, ICB builds smaller trees than our Mamba rerun on jointly solved runs, but generally does not match RPB or GCNN. For auction medium, all methods solve all runs: ICB uses 1,205 nodes compared with 2,946 for Mamba, 673 for GCNN, and 696 for RPB. On set-cover medium, the corresponding ICB and Mamba aggregates are 4,153 and 9,889; the paired ratios against Mamba range from 0.41 to 0.62 on set covering and from 0.41 to 0.45 on the auction transfer cells (Appendix D.2). Set-cover big nevertheless favors the comparators: ICB completes 35% of runs against 65% for RPB and 67% for GCNN.

On independent-set medium, ICB solves 95 of 100 runs, compared with 79 for Mamba, 47 for GCNN, and 100 for RPB. On the 95 runs solved by both ICB and RPB, the paired tree-size ratio is 0.50, with reported 95% bootstrap interval $[0.35, 0.75]$. This is a conditional advantage, not overall solver dominance: on independent-set big, ICB completes 26% of runs and RPB completes 45%. Facility location remains a weak regime for ICB: all learned methods build larger trees than RPB there, and ICB additionally trails GCNN and Mamba. We retain all four families to expose these differences rather than selecting favorable cells. Appendix D.2 provides the pair counts and available intervals.

4.3 Effect of event memory

The context-off intervention leaves weights, candidate features, and readout fixed. Table 2 traces its effect across four combinatorial-auction scales for the model trained at the easy size. At easy and medium size, where all runs finish, memory reduces the node aggregate from 228 to 152 and from 3,204 to 1,205. At big size, context improves completion from 87% to 95%; at the 350-item/1,750-bid stress size, from 49% to 65% (comparators at the stress sizes are in Appendix D.4). These observations support useful feedback under size transfer. The descriptive node ratio is not monotone across all four scales and does not estimate complete-tree cost at timeouts.

Table 2: Context effect across scales on combinatorial auctions (model trained at the easy size). The node ratio divides the two processed-node aggregates and is descriptive; at scales with timeouts it is not a paired complete-tree ratio. The effect is large under size transfer but not monotone across every scale.

	ICB		ICB context-off		
Scale	solved	nodes	solved	nodes	node ratio
Easy (100/500)	100%	152	100%	228	1.50
Medium (200/1,000)	100%	1,205	100%	3,204	2.66
Big (300/1,500)	95%	14,371	87%	53,062	3.69
Bigger (350/1,750)	65%	36,205	49%	85,372	2.36

Table 3 separates the two evidence routes. On auction big, the exact-only intervention completes 98% of runs, compared with 95% for full context and 87% for context-off, and breaking the exact matches while retaining similarity degrades both completion and nodes. The results support exact attribution as a useful component. They do not establish an additional benefit from structural similarity; since an ineffective permutation does not rule out all computation through a channel, we conclude only that exact attribution suffices for most of the observed benefit in this cell. On independent-set medium, completion varies from 95% to 98% across interventions and node aggregates remain close. Thus the favorable independent-set system results do not, in these experiments, establish a benefit from memory. The state-only restriction retains most of the performance.

4.4 Generalization across problem classes

We reconstructed the MILP-Evolve generation and class-splitting pipeline (Li et al., 2025). It yielded 571 generated training classes, 16 seed directories, 81 validation classes, and 165 test classes. A single ICB generalist—one model, not per-class models—was trained on 30,861 SB-labelled queries from the training classes. Evaluation selects the first three instances per test class and applies the released root-solve filter plus one solver-crash exclusion to every policy. The resulting cohort contains

Table 3: **Context interventions on fixed checkpoints.** Each cell uses 20×5 runs. Exact-only disables structural-similarity input; shuffled-sim permutes event descriptors; sim-only breaks exact matches; context-off removes memory. Entries report completion and processed-node SGMs. Full definitions are in Appendix A.7.

	Comb. auct. big		Indep. set medium	
Context mode	solved	nodes	solved	nodes
full context	95%	14,371	95%	1,383
exact only	98%	12,471	97%	1,350
shuffled sim	90%	13,457	97%	1,353
sim only	90%	21,576	98%	1,312
context off	87%	53,062	98%	1,402

Table 4: **Class-disjoint MILP-Evolve evaluation.** A single generalist is tested on the same filtered cohort of 340 MILPs for every policy, using SCIP 6.0.1 and a 200s CPU-clock limit. Context-off changes only the event memory. Time (s), processed nodes and relative PDI are one-shifted geometric means. PDI measures anytime progress; processed nodes do not give complete-tree costs for unfinished runs. Cohort construction is in Appendix B.3; cross-stack GCN results are reported separately in Appendix F.

Policy (SCIP 6.0.1)	solved	time	nodes proc.	PDI
RPB	79.1%	75.6	322	309.5
ICB	69.1%	95.5	746	340.7
ICB shuffled-sim	70.0%	93.7	714	337.2
ICB context-off	55.6%	126.8	1,430	402.0

340 instances drawn from the held-out-class pool. It is a reconstruction, not an asserted reproduction of the archived benchmark population (Appendix B.3).

With the same checkpoint, event memory increases completion from 55.6% to 69.1%, a difference of 13.5 percentage points (Table 4). The paired ratio $\mathrm{GM}[(1+\mathrm{PDI}_{\mathrm{ICB}})/(1+\mathrm{PDI}_{\mathrm{off}})]$ is 0.848. This indicates improved anytime progress as well as more completed solves. RPB remains stronger, with 79.1% completion and lower PDI. These are descriptive effect estimates; class-clustered confidence intervals have not been established for this cohort.

Offline validation queries provide a complementary check. In the bucket with 3–9 prior events for the oracle-selected variable, accuracy is 0.430 with real memory, 0.407 with permuted values, 0.388 with permuted attributions, and 0.277 with context removed. The growth of the real-memory advantage with accumulated evidence is consistent with evidence-driven adaptation on unseen classes. Buckets contain different queries and are not controlled additions of evidence to a fixed state. The 10+ bucket has only 42 queries (Appendix E). Our reproduction of the released GCN is reported separately because it runs in SCIP 8.0.4 and does not recover the published completion rate (Appendix F).

5 Discussion and conclusion

Explicit event memory improves the evaluated policy on combinatorial auctions and on held-out MILP-Evolve classes. It is not uniformly useful: independent-set performance largely survives context removal, and multiple knapsack gives almost identical context-on and context-off results (Appendix D.6). The latter is consistent with the high predictability of the expert label from state features alone on that family (98.7% held-out accuracy without context), although this does not establish that its history is intrinsically uninformative. Training also varies: independent-set medium node aggregates are 1,383, 3,759, and 1,401 across three seeds (Appendix D.5). These results distinguish feedback use from the quality of the prior and the training recipe.

The current comparisons do not establish that memory improves a stronger, separately trained state-only encoder. Our model uses candidate summaries rather than the full MILP graph. Masking pseudocost features removes one policy-dependent input, but other solver features and the event stream

still depend on the executed trajectory. In a reported 40-run independent-set screen, the unmasked variant solved only four runs; this supports a robustness concern, not a proof that masking eliminates distribution shift. Timings on a shared host, conditional tree comparisons, and development-time exposure to some evaluation cells further limit performance claims.

ICB provides an explicit interface between outcomes observed during a solve and subsequent variable selection. Its parameters remain fixed, while identity-linked evidence can change the policy's predictions. The present results support this interface as a useful component of learned branching, not as a uniformly superior replacement for RPB. Matched-data comparisons with stronger graph or hybrid priors and more efficient memory processing are the next tests of its broader value.

Reproducibility statement

Section 3 specifies event availability, the estimator, readout, and training objective. Appendix A details the feature set and masking, observation timing, identity keys, algebraic properties of the estimator, and the definitions of all context interventions. Appendix B documents instance generation for each benchmark, solver configuration and clock semantics, and the MILP-Evolve reconstruction with its filters and exclusions. Appendix C records training recipes and model selection, including the exploratory exceptions. Appendix D defines the aggregates and paired statistics and reports the complete result tables; Appendices E and F contain the offline interventions and the cross-stack reproduction.

Use of generative AI

Generative AI tools assisted with implementation coding, literature search, and manuscript editing. The authors reviewed the code, the numerical records, the references, and the final text, and are responsible for the content.

References

Tobias Achterberg and Timo Berthold. Hybrid branching. *Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*, 2009.

Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

Hongyu Cheng and Amitabh Basu. Theoretical challenges in learning for branch-and-cut. In *International Conference on Machine Learning (ICML)*, 2026.

Marc Etheve, Zacharie Alès, Côme Bissuel, Olivier Juan, and Safia Kedad-Sidhoum. Reinforcement learning for variable selection in a branch and bound algorithm. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, 2020.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Prateek Gupta, Maxime Gasse, Elias B. Khalil, M. Pawan Kumar, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In *Advances in Neural Information Processing Systems*, 2020.

Prateek Gupta, Elias B. Khalil, Didier Chételat, Maxime Gasse, Yoshua Bengio, Andrea Lodi, and M. Pawan Kumar. Lookback for learning to branch. *Transactions on Machine Learning Research*, 2022. URL https://arxiv.org/abs/2206.14987.

Zeren Huang, Wenhao Chen, Weinan Zhang, Chuhan Shi, Furui Liu, Hui-Ling Zhen, Mingxuan Yuan, Jianye Hao, Yong Yu, and Jun Wang. Branch ranking for efficient mixed-integer programming via offline ranking-based policy learning. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2022.

Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

Yufei Kuang, Jie Wang, Yuyan Zhou, Xijun Li, Fangzhou Zhu, Jianye Hao, and Feng Wu. Towards general algorithm discovery for combinatorial optimization: Learning symbolic branching policy from bipartite graph. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 25623–25641, 2024. URL `https://proceedings.mlr.press/v235/kuang24a.html`.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations*, 2023.

Jonathan N. Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Sirui Li, Janardhan Kulkarni, Ishai Menache, Cathy Wu, and Beibin Li. Towards foundation models for mixed integer linear programming. In *International Conference on Learning Representations (ICLR)*, 2025.

Christopher W. F. Parsonson, Alexandre Laterre, and Thomas D. Barrett. Reinforcement learning for branch-and-bound optimisation using retrospective trajectories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Lara Scavuzzo, Feng Yang Chen, Didier Chételat, Maxime Gasse, Andrea Lodi, Neil Yorke-Smith, and Karen Aardal. Learning to branch with tree MDPs. In *Advances in Neural Information Processing Systems*, 2022.

Giulia Zarpellon, Jason Jo, Andrea Lodi, and Yoshua Bengio. Parameterizing branch-and-bound search trees to learn branching policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Ce Zhang, Bin Zhang, and Guoliang Fan. Towards better branching policies: Leveraging the sequential nature of branch-and-bound tree. In *International Conference on Learning Representations (ICLR)*, 2026.

A Method details and algebraic properties

A.1 Notation

The event index is distinct from the decision index. The same variable may appear repeatedly in both sequences. Table 5 summarizes quantities used in the estimator.

Table 5: Notation for the ICB interface and estimator.

Symbol	Meaning
t, ℓ	Branching-decision index and event-availability index
C_t, Φ_t	Current candidates and their raw feature vectors
$M, \mathcal{N}$	Pseudocost-feature mask and training-set normalization
$\psi_{j,t}, \zeta_\ell$	Normalized candidate features and saved event descriptor
v_ℓ, d_ℓ, g_ℓ	Event variable, direction, and nonnegative unit gain
γ_ℓ	$\log(1 + g_\ell)$
$\mathcal{E}_t$	Last $B = 256$ events available before decision t
$k_j, m_{j\ell}$	Random identity key and given exact-match indicator
$m^0_{j,d}, \tau^0_{j,d}$	Prior log-gain estimate and positive pseudo-count
w_ℓ, r_ℓ	Learned event weight and transformed gain
$\alpha_{j\ell}, \rho$	Similarity attention and its nonnegative trust parameter
$W_{j,d}, R_{j,d}$	Effective evidence weight and weighted gain sum
$\widehat{m}_{j,d}, \widetilde{m}_{j,d}$	Shrinkage estimate and residual-corrected estimate
$\widehat{g}_{j,d}, S_j, \ell_{\theta,j}$	Unit-gain prediction, branch score, and classification logit

A.2 Candidate features and normalization

The 40-dimensional representation extends the original 25 candidate features in Table 6 with 15 quantities describing the objective coefficient, nonzero count, coefficient extrema and mean, reduced cost, basis status, LP value, distance to the root LP value, variable locks, and incumbent geometry. These are feature summaries, not a full bipartite graph representation. Some are dynamic or history-dependent, even when described as structural features.

Columns 8–14, using zero-based inclusive indices, are masked in both the candidate and event paths. The stored normalization applies per-column training means and standard deviations, followed by clipping to $[-10, 10]$. Writing this transformation as $\psi = \mathcal{N}(M \odot \phi)$ separates information removal from numerical normalization. In our implementation the normalization statistics are estimated on already-masked training features, so a masked coordinate has mean zero and unit fallback scale and remains exactly zero after normalization. Normalization statistics are not updated on evaluation instances.

Table 6: Baseline candidate feature columns. "Masked" marks the columns zeroed in the main ICB configuration. History-derived columns other than the masked pseudocost statistics remain available to the model.

Col	Name	Source (SCIP API)	Class	Masked
0	lp_frac	lpcandsfrac	LP state	
1	avg_primal_sol	SCIPvarGetAvgSol	primal history	
2–3	branch_depth up/down	AvgBranchdepthCurrentRun	branch history	
4	conflict_score	VarConflictScore/avg	conflict history	
5	conflictlen_score	VarConflictlengthScore/avg	conflict history	
6	inference_score	VarAvgInferenceScore/avg	inference history	
7	cutoff_score	VarAvgCutoffScore/avg	outcome history	
8	pscost_score	VarPseudocostScore/avg	pseudocost	×
9–10	pscost count up/down (global)	PseudocostCount	pseudocost	×
11–12	pscost count up/down (per branch)	PseudocostCount	pseudocost	×
13–14	pscost count up/down (per call)	PseudocostCount	pseudocost	×
15–16	n_impl 0/1	SCIPvarGetNImpls	static structure	
17–18	clique_ratio 0/1	VarGetNCliques/global	static structure	
19–20	avg_cutoffs up/down	VarAvgCutoffsCurrentRun	outcome history	
21–22	avg_conflen up/down	VarAvgConflictlengthCurrentRun	conflict history	
23–24	avg_infer up/down	VarAvgInferencesCurrentRun	inference history	

A.3 Observation timing and identity

The event handler associates a processed node with its pending parent branch. It records the outcome before requesting a branching decision at that node, provided the relevant LP bound is known. Branch direction is inferred from local variable bounds; ambiguous directions and non-finite parent or child bounds are omitted. Outcomes are delayed observations of executed branches, not counterfactual outcomes for every candidate. The policy does not require two observed children per branch. Nor does the passive log reconstruct every statistic maintained internally by SCIP.

Saved descriptors refer to the variable at branch execution. Fallback branches without an available descriptor use the implementation's missing-descriptor path, recorded as zero contribution to similarity in the implementation audit. This convention must not be confused with observing a structural vector of zeros. Strong-branching side queries used in collection are excluded from the event buffer.

Keys have distribution $\mathcal{N}(0, I_{16}/16)$. Deployment caches a key on the variable's first use and reuses it until the solve ends. Training resamples the key map at each sample presentation. Exact matching is computed from the solver's persistent identity map, not by approximate comparison of random vectors. Changing the random-key draw can still change a finite network's output, since the numerical keys are input features. Resampling prevents lookup through a fixed identifier vocabulary; it does not alone prove permutation invariance or rule out structural memorization.

A.4 Properties of the evidence estimator

The following are algebraic consequences of the parameterization, not performance guarantees. They concern the intermediate estimator before the residual correction and do not assume that gains are independent, stationary, or unbiased.

Proposition 1 (Shrinkage and cold start)**.** *Fix a candidate and direction. Assume $\tau^0 > 0$ and nonnegative effective weights $a_\ell w_\ell$. For $W > 0$, the estimate in Eq.* (6) *satisfies*

$$\widehat{m} = \omega m^0 + (1-\omega)\bar{r}, \qquad \omega = \frac{\tau^0}{\tau^0 + W} \in (0,1), \quad \bar{r} = \frac{R}{W}.$$

If there is no effective evidence, then $W = R = 0$ and $\widehat{m} = m^0$.

Proof. Substitute $R = W\bar{r}$ into Eq. (6) and divide the two numerator terms by $\tau^0 + W$. With nonnegative weights, $W = 0$ implies that every effective weight is zero and hence $R = 0$. □

The word "posterior" in implementation names refers to this shrinkage form. No likelihood, calibrated posterior variance, or posterior-sampling guarantee is established by the architecture. Its confidence τ^0 is a learned pseudo-count. In addition, the empty-memory identity applies to $\widehat{m}$, not automatically to $\widetilde{m}$: zero initialization of a residual output head does not force its output to remain zero after training. The context-off experiments evaluate the complete network's empty-memory path.

Proposition 2 (A fixed-input update)**.** *Consider the exact-only estimator for a fixed candidate and direction, with the prior and existing event transformations fixed. Suppose a new matching event with weight $w \geq 0$ and transformed gain r is appended, without evicting an old event. Then*

$$\widehat{m}^+ - \widehat{m} = \frac{w}{\tau^0 + W + w}\,(r - \widehat{m}). \tag{12}$$

Thus the intermediate estimate moves toward r.

Proof. Let $D = \tau^0 + W$ and $U = \tau^0 m^0 + R = D\widehat{m}$. The new estimate is $(U + wr)/(D + w)$. Subtracting U/D gives $w(r - \widehat{m})/(D + w)$. □

With eviction of an event (w_o, r_o) from the same candidate and direction, the corresponding identity is

$$\widehat{m}^+ - \widehat{m} = \frac{w(r - \widehat{m}) - w_o(r_o - \widehat{m})}{\tau^0 + W + w - w_o}. \tag{13}$$

In the full model, changing a raw gain can change its learned weight, attention, and residual. Adding an event can renormalize similarity attention. Neither Eq. (12) nor the convex-combination form therefore implies unconditional monotonicity of the complete policy.

Log-space estimation. Even unweighted averaging of event inputs estimates $|E|^{-1}\sum_\ell \log(1+g_\ell)$, not $\log(1+|E|^{-1}\sum_\ell g_\ell)$. By concavity of the logarithm,

$$\exp\left(\frac{1}{|E|}\sum_\ell \log(1+g_\ell)\right) - 1 \;\le\; \frac{1}{|E|}\sum_\ell g_\ell.$$

Consequently, ICB is a learned estimator inspired by pseudocosts, not an exact reimplementation of their arithmetic averages. The auxiliary replay regression and the SB classification loss constrain different aspects of its prediction.

Readout and monotonicity. For fixed $f_j \in (0,1)$ and $\kappa \in [0,1]$, S_j is nondecreasing in each of the two directional gain predictions. Both minimum and maximum are coordinatewise nondecreasing, and their coefficients are nonnegative. The maps $m \mapsto \exp(m) - 1$ and $s \mapsto \log(1+s)$ preserve order. For a positive logit scale, the candidate ranking respects this order. This proves monotonicity of the readout conditional on its inputs, not monotonicity of those inputs with respect to raw historical observations.

Order and identity. For a fixed retained event set, the exact sums do not depend on its presentation order. A consistent relabeling of candidate and event identities leaves the matching relation unchanged. These statements do not remove the chronological restriction in Eq. (3): changing which events occupy the most recent B positions changes the model input.

A.5 Attention and supervised targets

The residual's attention contains a head-specific additive match bias. For a query embedding h_j, event embedding z_ℓ, and head h, its attention weights have the form

$$A^h_{j\ell} = \mathrm{softmax}_\ell\left(\frac{(Q_h h_j)^\top (K_h z_\ell)}{\sqrt{d_h}} + b_h m_{j\ell}\right), \tag{14}$$

with padding excluded. Four pre-norm blocks produce the residual in Eq. (7). The exact-match bias and the deterministic sums are distinct uses of the same given relation. Learned projections and feedforward layers are shared across candidates.

The replay target is $Y_{t,j,d} = \log(1+\bar{g}_{t,j,d})$ for the directional means stored by the data builder. For a scalar residual z, the Huber penalty with threshold $\eta > 0$ is

$$h_\eta(z) = \begin{cases} \frac{1}{2}z^2, & |z| \le \eta, \\ \eta(|z| - \frac{1}{2}\eta), & |z| > \eta. \end{cases}$$

The vector Huber term in Eq. (10) applies this penalty to the stored directional targets. The loss documented for the final runs is SB-argmax cross-entropy plus the auxiliary Huber term with weight one, not SB-score regression or listwise KL alone.

A.6 Deployment procedure

Algorithm 1 in Section 3.4 summarizes the callback loop. The procedure does not terminate after returning the first branching choice: control is handed back to the solver, and the loop continues until the solve ends, after which the event buffer and the key map are discarded. It changes only branching decisions, not the validity of bounds or the solver's pruning rules. Fallback branchings without an available candidate matrix produce events without descriptors (Appendix A.3).

A.7 Interventions and their scope

All reported context interventions keep model weights fixed. Context-off empties the event memory. Value shuffling permutes event gains within each direction. Attribution shuffling moves event keys and their matching structure together, leaving numerical values unchanged. Similarity shuffling permutes saved structural descriptors while retaining keys, directions, and values. The reported exact-only intervention removes event structural descriptors; the reported similarity-only intervention breaks candidate–event exact matches while preserving descriptors.

These names refer to the implemented input interventions. In particular, breaking exact matches does not remove every possible computation in the residual attention path. Removing a descriptor is also not equivalent to zero similarity weight unless the implementation handles it that way. These controls identify useful information paths in specific settings; they do not uniquely identify every internal computation.

B Datasets and evaluation protocols

B.1 Canonical Gasse benchmark

The data follow the official generators of Gasse et al. (2019). Set covering has 1,000 columns and 500/1,000/2,000 rows. Auctions use 100/500, 200/1,000, and 300/1,500 items/bids. Facility location has 100 facilities with 100/200/400 customers. Independent set has 500/1,000/1,500 vertices. Each evaluation cell contains 20 instances with solver seeds 0–4. The generator and seed sequence determine the files; identical dimensions alone do not determine an identical benchmark.

The stress evaluation has 100 additional independent-set instances at the same 1,500-vertex size, with one seed. It increases coverage, not size. Auction stress instances have 350 items and 1,750 bids: $7/6$ of the canonical big dimensions and 3.5 times the easy dimensions.

B.2 Solver configuration and clocks

The common runner uses SCIP 6.0.1 with `separating/maxrounds=0`, `presolving/maxrestarts=0`, and `timing/clocktype=1`. The limits are 3,600 CPU seconds for the canonical Gasse cells and 200 for the MILP-Evolve evaluation. Wall time is the elapsed external process time. Parallel evaluation on a shared host can affect methods differently; no equal-overhead assumption is made. Per-run internal CPU times were not retained, so the wall times cannot be converted retrospectively to solver-clock measurements.

The numerical readout of ICB is Eq. (8). It must not be confused with the SB-label combiner or with the solver's configurable `branching/scorefunc`. SCIP exposes separate sum and product score choices; the existence of a `scorefac=0.167` parameter does not establish that the default uses the sum rule. The paper fixes the ICB readout rather than asserting equality of all three scoring procedures.

B.3 MILP-Evolve reconstruction

The reconstruction uses the MILP-Evolve release at commit `419cf7b`. Applying the released class-splitting procedure with its specified seed produced 571 generated training classes, 81 validation classes, and 165 test classes. The training set also contains 16 seed directories. We retain this reconstructed split rather than identifying it with the archived official benchmark population.

ICB collection on five instances per training directory completed 2,912 episodes. Of these, 2,096 instances produced branching queries, giving 30,861 fully SB-labelled decisions. The generalist was trained once for all classes. Evaluation follows the released script's first-three-instance selection and its default-solver root-closure filter. One instance was excluded for a reproducible SCIP 6.0.1 crash. The same filter and exclusion apply to every policy, leaving the 340-instance comparison cohort. The filtered cohort is drawn from the held-out-class pool; it is not assumed to retain instances from every class.

B.4 Multiple knapsack

The test and transfer sizes are 100 items with 6 and 12 knapsacks, respectively (Scavuzzo et al., 2022). Each has 40 instances with five seeds. The authors' pretrained policies run in their Ecole/SCIP 7.0.1 environment; ICB runs in SCIP 6.0.1. These results are reported in separate blocks.

C Training and model selection

The final architecture is a posterior-style estimator with similarity and an exact-match attention bias, four blocks of width 256, 16-dimensional keys, candidate features enabled, and columns 8–14 masked. Final commands use the loss in Eq. (10) with auxiliary weight one and listwise weight zero. Collection alternates pscost and RPB in segments of 5–25 decisions, with per-decision SB sampling probability 0.15. The MILP-Evolve collection uses a 300-second limit per episode.

Checkpoint selection uses held-out SB-argmax accuracy. The documented instance-level holdout is 15% of the collection. Default recipe values are learning rate 10^{-3} and 20 epochs; a lower-rate recipe uses $3 \cdot 10^{-4}$ and 30 epochs. Independent-set low-rate choices and rejection of a set-cover low-rate variant were informed by live canonical-cell results. These are exploratory decisions, not an untouched final-test protocol.

Reported tables preserve the archived numerical snapshot, including the individually reported training seeds. Recipe selection and epoch selection are distinct: the former included the exploratory cell-level comparisons above, whereas the latter used held-out query accuracy.

D Complete results and statistical definitions

D.1 Aggregation and censoring

For nonnegative run values $x_1, \ldots, x_n$, the one-shifted geometric mean is

$$\mathrm{SGM}_1(x) = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log(1+x_i)\right) - 1. \tag{15}$$

This is a summary of recorded values. For timed-out runs, recorded nodes are a lower bound on the eventual proof effort, not a completed tree. These node aggregates are not a substitute for solved fraction or for a time-budgeted progress measure.

For methods A, B, let $\mathcal{P}_{AB}$ be the set of (instance, seed) runs solved by both. The reported complete-tree ratio is

$$R_N(A,B) = \exp\left(\frac{1}{|\mathcal{P}_{AB}|}\sum_{r\in\mathcal{P}_{AB}}\log\frac{N_A(r)}{N_B(r)}\right). \tag{16}$$

All supporting node counts must be positive. This unshifted paired ratio need not equal the quotient of two shifted aggregates, and different method pairs can use different subsets. It therefore does not define a transitive leaderboard. Bootstrap resampling is clustered by MILP instance, retaining the matched solver seeds within each sampled instance. Counts below refer to paired runs, not independent MILPs.

For the MILP-Evolve comparison, the archived PDI statistic is

$$R_{\mathrm{PDI}}(A,B) = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log\frac{1+\mathrm{PDI}_A(i)}{1+\mathrm{PDI}_B(i)}\right). \tag{17}$$

Writing $P(u)$ and $D(u)$ for the incumbent and dual bounds at solver clock time u, the recorded PDI has the form

$$\mathrm{PDI} = \int_0^{T_{\mathrm{run}}} \mathrm{gap}_{\mathrm{SCIP}}(P(u), D(u))\, du,$$

where $\mathrm{gap}_{\mathrm{SCIP}}$ includes the solver's conventions for missing bounds and objective signs. The terminal time is the recorded solver stopping time. Exact reconstruction requires the bound-change trace and clock conventions; final bounds alone are insufficient. We use the recorded statistic and do not combine it with Ecole's unnormalized absolute-bound integral. The retained snapshot lacks class-clustered intervals for the PDI ratio and solved-rate difference. No such intervals are claimed in this version.

D.2 Pairwise complete-tree comparisons

Table 7: **Conditional complete-tree comparisons.** Each block reports the geometric mean of ICB/baseline node ratios, paired-run count n, and a 95% instance-clustered bootstrap interval on jointly solved runs. Stars mark intervals excluding one. Auction intervals are unavailable and no significance is asserted for those entries. Independent-set big versus GCNN has only one pair. Different comparator columns can use different subsets.

	vs. RPB			vs. GCNN			vs. Mamba		
Cell	ratio	n	CI	ratio	n	CI	ratio	n	CI
Set cover easy	5.76*	100	[4.22, 7.77]	–	–		0.62*	100	[0.53, 0.76]
Set cover medium	1.79*	100	[1.54, 2.13]	–	–		0.42*	100	[0.38, 0.47]
Set cover big	1.37*	35	[1.29, 1.47]	2.07*	35	[1.94, 2.22]	0.41*	27	[0.36, 0.46]
Comb. auct. easy	15.85	100	–	2.16	100	–	1.21	100	–
Comb. auct. medium	1.74	100	–	1.79	100	–	0.41	100	–
Comb. auct. big	1.56	95	–	1.79	95	–	0.45	95	–
Facility loc. easy	9.69*	100	[6.13, 15.08]	1.77*	100	[1.52, 2.08]	3.47*	100	[2.16, 5.98]
Facility loc. medium	2.71*	100	[2.04, 3.69]	1.24*	100	[1.10, 1.39]	1.19*	100	[1.08, 1.35]
Facility loc. big	3.50*	93	[2.65, 5.14]	1.30*	95	[1.19, 1.44]	1.60*	96	[1.25, 2.40]
Indep. set easy	3.58*	100	[2.23, 5.69]	–	–		0.80	100	[0.48, 1.47]
Indep. set medium	0.50*	95	[0.35, 0.75]	0.018*	47	[0.009, 0.041]	0.10*	79	[0.07, 0.15]
Indep. set big	0.28*	26	[0.23, 0.33]	(0.19)	1	–	0.11*	7	[0.06, 0.21]

D.3 Context-off results

Table 8: Context-off (ICB context-off) rows for all canonical cells, same protocol and reporting as Table 1.

	Easy			Medium			Big		
Family	solved	time	nodes	solved	time	nodes	solved	time	nodes
Set cover	100%	10.8	400	100%	104.8	5,076	35%	2,817	115,740
Comb. auctions	100%	4.0	228	100%	57.1	3,204	87%	1,131	53,062
Facility loc.	100%	25.4	195	100%	134.8	481	96%	570.5	534
Indep. set	100%	37.9	74	98%	421.2	1,402	30%	3,587	17,364

D.4 Extended transfer evaluations

Table 9: Stress tests. Left: 100 *additional* independent-set instances at the canonical big size (1,500 nodes; coverage extension, one seed). Right: combinatorial auctions at 350/1,750 ($\approx 1.17\times$ the canonical big size, $3.5\times$ the training size), 20 instances $\times$ 5 seeds. Nodes are processed nodes.

	Indep. set 1,500 (n=100)			Comb. auct. 350/1,750 (n=100)		
Model	solved	time	nodes	solved	time	nodes
RPB	53%	2,124	59,322	97%	591	25,264
Mamba	4%	3,532	38,778	70%	1,898	71,889
ICB	25%	2,793	15,175	65%	1,783	36,205
ICB context-off	30%	2,672	18,459	49%	2,979	85,372

D.5 Training-seed variation

Table 10: Independent set medium across three training seeds of the same recipe (20×5 runs each). Three seeds do not support a distributional claim; we report them individually.

Training seed	solved	nodes proc.
seed 0	95%	1,383
seed 1	91%	3,759
seed 2	97%	1,401

D.6 Multiple knapsack results

Table 11: Multiple knapsack (Scavuzzo et al., 2022), 40 instances $\times$ 5 seeds (test $=$ 100 items/6 knapsacks, transfer $= 100/12$). Left: our environment (SCIP 6.0.1). Right: the authors' pretrained policies run with their code (Ecole, SCIP 7.0.1). The solver versions differ on this family; comparisons are within-block. Entries are processed-node SGMs. Context brings ICB no measurable benefit here; no learned policy outperforms RPB on transfer in either block. This reproduction does not refute the published tree-MDP results, which used a different protocol.

SCIP 6.0.1 (ours)			Ecole + SCIP 7.0.1 (their code)		
Model	test	transfer	Model	test	transfer
RPB	78	391	relpscost	111	579
ICB	125	667	il	145	2,057
ICB context-off	125	668	mdp	111	933
			tmdp+DFS	107	665
			tmdp+ObjLim	89	696

E Offline context interventions

Table 12 groups validation queries by the amount of evidence for the SB-selected variable. This variable is known for analysis, not at deployment. The buckets compare different query populations; their trend is not a controlled learning curve. The listed counts sum to 4,761 queries, and the 10+ group contains only 42.

Table 12: Offline validation accuracy by evidence bucket (MILP-Evolve generalist). Buckets are cross-sectional: they compare different queries and do not show the effect of adding events to a fixed query. The 10+ bucket is small.

Events for label variable	real	shuffled values	shuffled keys	context off	n
0	0.172	0.159	0.165	0.148	3,191
1–2	0.365	0.327	0.334	0.279	982
3–9	0.430	0.407	0.388	0.277	546
10+	0.357	0.357	0.286	0.119	42

F MILP-Evolve cross-stack reproduction

The released GCN pipeline was reconstructed in SCIP 8.0.4 with Ecole 0.8.1. Its checkpoint reached validation top-5 accuracy 0.758. It did not reproduce the published solved fraction. Table 13 retains these measurements as an implementation report, not as evidence of a state-of-the-art comparison. Identical MILP files do not control for solver-version and feature-distribution differences. The Ecole-side recorded primal–dual integral uses an absolute-bound scale dominated by large objective magnitudes. It is not pooled with SCIP's relative PDI. The reconstruction log documents fixes to resume handling and process/file-descriptor management; those engineering notes are separate from the comparative results.

Table 13: Cross-stack block on identical instances. The two blocks run in different solver stacks with different baselines and feature distributions; identical MILP files do not make the learned policies directly comparable, and no cross-stack ranking is claimed. The released paper reports a higher solved fraction for its GCN than our reproduction attains; the cause of this reproduction gap is unresolved. Processed-node aggregates under different completion rates are not complete-tree comparisons.

Solver	Policy	solved	time	nodes proc.
SCIP 8.0.4	default	72.9%	37.8	287
SCIP 8.0.4	GCN (released pipeline)	47.6%	57.3	199
SCIP 6.0.1	RPB	79.1%	75.6	322
SCIP 6.0.1	ICB	69.1%	95.5	746