SHARED ORDER MARGINALIZATION FOR TRAINING GRAPH TRANSFORMERS ON INDEPENDENT SETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Transformers can learn to select tasks with additive values and pairwise conflicts by constructing independent sets. The reward depends on the selected set, yet policy-gradient training gives different signals to different orders that construct it. Averaging these signals reduces order-dependent noise but requires accounting for the graph deletions caused by every alternative. We propose SOM, Shared Order Marginalization, to train from a broad family of same-solution orders while sharing their graph computation. For policies with fixed vertex selection rates, SOM represents when each vertex remains available across alternative starting positions. This representation supplies the order family's probability and all vertex derivatives without rebuilding the graph for each start. For a solution with $2K$ vertices, it exactly averages $K2^K$ orders in $O(n + m + K^2)$ arithmetic operations, where n and m are the graph's vertex and edge counts. The resulting estimator preserves the expected-reward gradient and reduces covariance under matched sampling. On a unit-rate star, one cyclic average removes a score component that requires quadratically many layers of the specified exact adjacent-pair averaging. On weighted vehicle-routing conflict graphs, a GraphGPS–SOM initializer followed by CHILS has a mean normalized-value difference of -0.02 percentage points from native CHILS under the same wall limit, with a paired interval crossing zero. Against fixed-configuration RLOO, SOM reaches the common target 1.22× faster; separate tuning reduces the speed ratio to 1.110. Graphormer supplies a second-backbone comparison, and the MIS and sparse panels delimit the benefit. Our code is avaliable at `https://anonymous.4open.science/status/SOM-DC11`.

1 INTRODUCTION

Learning for combinatorial optimization trains neural policies to reuse experience across related instances (Sakaue, 2021; Acikalin et al., 2025). A common task is to select compatible actions with maximum total value, such as route modifications in vehicle routing (Dong et al., 2021). With additive values and pairwise conflicts, this is maximum weight independent set (MWIS). A Graph Transformer predicts which vertices to select, while a sequential sampler removes conflicting neighbors to maintain feasibility (Rampásek et al., 2022; Ying et al., 2021). This creates a mismatch: the task rewards a set, but policy-gradient training depends on the order that produced it.

Policy-gradient methods weight sampled log-probability gradients by reward (Williams, 1992; Kool et al., 2020). Equivalent-solution methods reuse an outcome through alternative trajectories, symmetry-aware training, or conditional averaging (Kim et al., 2022; 2024; Tong et al., 2025). Although their objectives differ, each exploits multiple training signals from one solution. For independent sets, compatible vertices selected in different orders can receive the same reward but produce different gradients. Averaging with the correct conditional probabilities removes this order variation without changing the expected-reward objective.

The difficulty is computing a broad average economically. An early choice immediately removes its neighbors, whereas a late choice leaves them competing for longer. Equivalent orders therefore have different probabilities and intermediate candidate sets. Evaluating them separately repeats overlapping graph work; averaging only nearby swaps can retain sensitivity to a vertex's early or late position. Fig. 1 summarizes this trade-off between averaging range and computation.

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.

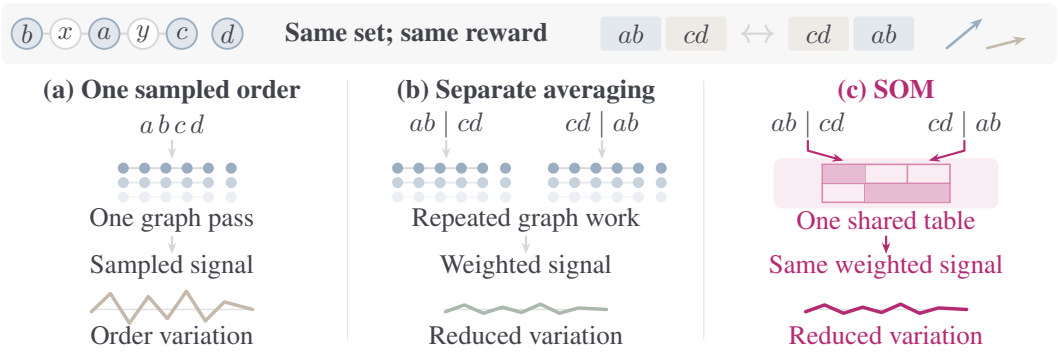


Figure 1: **Same solution, less order noise.** A single sampled order retains order variation. Separate averaging and SOM produce the same probability-weighted signal; SOM shares the graph calculation across alternatives. Signals are schematic.

We propose SOM, Shared Order Marginalization, to average a broad family of same-solution orders while sharing their graph calculations. SOM retains the selected set and reward, but averages within-pair orientations and cyclic starting positions. It applies to policies that predict positive vertex rates once per graph and keep their relative values fixed during construction. The key observation is that each vertex contributes to the available mass only until its first deleting block. Representing these intervals in shared tables supplies the order probabilities and every original vertex derivative without rebuilding the graph for each start.

Our analysis establishes a computation bound linear in graph size and quadratic in solution size, together with gradient preservation and covariance reduction under matched sampling. A unit-rate star separates the averaging mechanisms: the specified local layers need quadratically many iterations to suppress a position-dependent score component, while one cyclic average removes it. Under matched wall limits, the SOM-initialized CHILS pipeline differs in mean normalized value by -0.02 percentage points from native CHILS; its interval crosses zero. It reaches common quality 1.22× faster than fixed-configuration RLOO, while separate tuning narrows the difference. Appendix E separates initialization, uncertainty, tuning and resource accounting.

2 Preliminary and Challenge

2.1 Learning a feasible selection policy

Let $G = (V, E)$ be a finite simple graph with $n = |V| > 0$ vertices, $m = |E|$ edges, and positive values w_v. An independent set contains no adjacent vertices. The MWIS objective is

$$S^* \in \arg\max_{S \subseteq V} \left\{ w(S) := \sum_{v \in S} w_v \ : \ \{u, v\} \notin E \text{ for all distinct } u, v \in S \right\}. \tag{1}$$

A Graph Transformer with parameters θ predicts log rates $z_{\theta,v}(G)$ and positive selection rates $\lambda_v = \exp z_{\theta,v}$. Starting from $A_1 = V$, the sampler chooses $v_t \in A_t$ with probability λ_{v_t}/Z_t, where $Z_t = \sum_{u \in A_t} \lambda_u$, and updates $A_{t+1} = A_t \setminus N[v_t]$. The closed neighborhood $N[v]$ includes v. Rates stay fixed; only the available set changes. When it is empty, the accepted order $Y = (v_1, \ldots, v_L)$ defines a maximal independent set $S(Y)$, which need not be optimal.

The order probability, normalized terminal reward, and training objective are

$$p_\lambda(Y) = \prod_{t=1}^{L} \frac{\lambda_{v_t}}{Z_t(Y)}, \qquad R_G(S) = \frac{w(S)}{w(V)}, \qquad J(\theta) = \mathbb{E}_G \mathbb{E}_{Y \sim p_{\lambda_\theta(G)}}[R_G(S(Y))]. \tag{2}$$

Let $s(Y) = \nabla_z \log p_\lambda(Y)$ and $B_\theta = \partial z_\theta / \partial \theta$. A policy-gradient estimate is

$$\widehat{g}(Y) = (R_G(S(Y)) - \beta) B_\theta^\top s(Y). \tag{3}$$

An independent baseline β preserves unbiasedness (Williams, 1992). Whole-batch RLOO uses $N > 1$ independent solutions per graph and centers each reward against the other $N - 1$ rewards (Kool et al., 2020). It compares solution rewards but retains each sampled order's score.

2.2 SAME SOLUTION, DIFFERENT TRAINING SIGNALS

Consider the path 0–1–2 with rates $(1, 1, 2)$. The two endpoint orders yield the same set and reward, yet

$$p_\lambda(0,2) = \tfrac{1}{4}, \quad p_\lambda(2,0) = \tfrac{1}{2}, \qquad s_0(0,2) = \tfrac{3}{4}, \quad s_0(2,0) = -\tfrac{1}{4}. \tag{4}$$

The same vertex's score changes sign. Conditional on the endpoint set, the correct averaging weights are $1/3$ and $2/3$, not equal weights. More generally, a parameter-independent record $\mathcal{A}(Y)$ groups orders with the same terminal set and defines

$$\bar{s}_{\mathcal{A}}(Y) = \frac{\sum_{Y' :\, \mathcal{A}(Y')=\mathcal{A}(Y)} p_\lambda(Y') s(Y')}{\sum_{Y' :\, \mathcal{A}(Y')=\mathcal{A}(Y)} p_\lambda(Y')} = \mathbb{E}[s(Y) \mid \mathcal{A}(Y)]. \tag{5}$$

Conditional averaging preserves the reward gradient (Kool et al., 2020; Tong et al., 2025), but a terminal set of size L has $L!$ valid orders. Their differing deletion times change the normalizers, including contributions from unselected vertices. A useful estimator must therefore average a sufficiently broad family while sharing its graph calculations and retaining all rate derivatives.

3 METHOD

SOM combines a broader order average with shared graph computation. Completing a pair deletes the same vertices in either internal order, so its alternatives can be summed locally. Across cyclic starts, each original vertex contributes until its first deleting block, allowing these sums to share a mass table. Fig. 2 illustrates the construction. We retain the fixed-rate policy from Section 2.

3.1 A CYCLIC FAMILY OF SAME-SOLUTION ORDERS

For an accepted order $Y = (y_1, \ldots, y_L)$ with $L = 2K > 0$, form consecutive pairs $C_b = \{y_{2b+1}, y_{2b+2}\}$ for $b = 0, \ldots, K-1$, and arrange them on a directed cycle. The record $\mathcal{C}(Y)$ forgets the starting block and each pair's orientation. Its complete preimage contains $K2^K$ orders with the same set and reward. The record is independent of numerical rates, so Eq. 5 applies even though different samples can induce different pairings. Pairing alternatives and their audits are given in Appendix D.10.

Fix a starting block $r \in \{0, \ldots, K-1\}$, put $S = S(Y)$, and name the endpoints $C_b = \{a_b, c_b\}$, with periodic block indices. Write $\lambda(A) = \sum_{v \in A} \lambda_v$ and $N[A] = \bigcup_{v \in A} N[v]$. Before block b, the common active set is

$$A_{r,b} = V \setminus N\left[\bigcup_{h=r}^{b-1} C_h\right]. \tag{6}$$

We need its mass and the masses remaining after either endpoint:

$$\begin{aligned} &Z_{r,b} = \lambda(A_{r,b}), \qquad M^a_{r,b} = \lambda(A_{r,b} \cap N[a_b]), \quad M^c_{r,b} = \lambda(A_{r,b} \cap N[c_b]), \\ &U_{r,b} = Z_{r,b} - M^a_{r,b}, \qquad V_{r,b} = Z_{r,b} - M^c_{r,b}. \end{aligned} \tag{7}$$

Both remaining masses are positive because the other endpoint survives. Summing the two internal orders and multiplying across completed pairs gives

$$q_r = \left(\prod_{v \in S} \lambda_v\right) \prod_{b=r}^{r+K-1} \frac{U_{r,b} + V_{r,b}}{Z_{r,b} U_{r,b} V_{r,b}}, \qquad q_{\mathcal{C}} = \sum_{r=0}^{K-1} q_r. \tag{8}$$

Thus the conditional score is

$$\omega_r = \frac{q_r}{q_{\mathcal{C}}}, \qquad \bar{s}(Y) = \nabla_z \log q_{\mathcal{C}} = \sum_r \omega_r \nabla_z \log q_r = \mathbb{E}[s(Y) \mid \mathcal{C}(Y)]. \tag{9}$$

This includes unselected vertices through the mass denominators. For odd L, singleton cycles average the L rotations without pair flips. Appendix B.2 gives their factorization and edge cases.

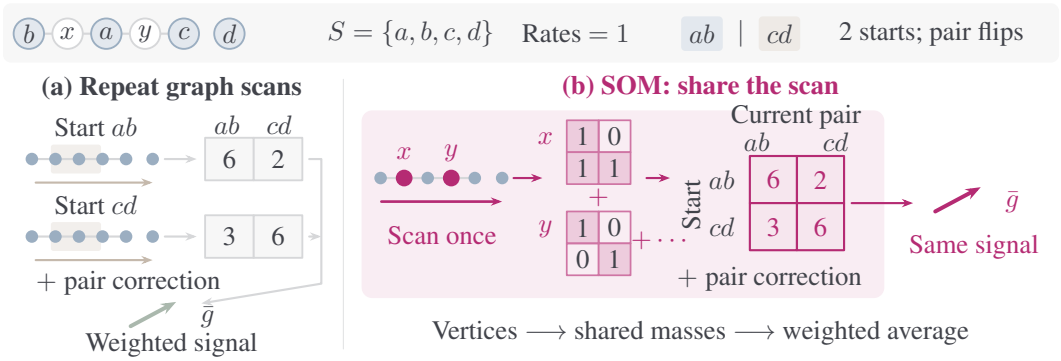


Figure 2: **Scan once, reuse across starts.** The unit-rate example groups selected vertices into $ab \mid cd$. Separate scans and SOM give the same mass table: rows are starts, columns are current blocks. The masks for x, y use the same row/column order; endpoint corrections supply within-pair masses.

3.2 Sharing masses and all vertex derivatives

Rather than rescan the graph for each start, we build each vertex's deleting-block list:

$$D_v = \{b \in \{0, \dots, K-1\} : v \in N[C_b]\}. \tag{10}$$

Each list is nonempty because the accepted set is maximal. A single neighborhood scan constructs all lists in $O(n + m)$ work. Between consecutive deleting blocks, the vertex remains available. Across starts and current blocks, these contributions form rectangles in the Z table and intervals in the endpoint-removal tables. A table row selects the starting block, and a column selects the current block. Between consecutive hits of v, the eligible start–current-block pairs form a rectangle: add λ_v at its four signed corners, then recover its contribution by two prefix sums. Endpoint-removal masses use the corresponding intervals of starts. Overlapping neighborhoods produce separate endpoint contributions but only one deletion from the active set.

In Fig. 2, vertex x is deleted only by pair ab, so it contributes to three displayed entries. Vertex y is deleted by either pair and contributes only to the two diagonal entries. Adding every vertex's footprint produces the same masses as separate scans, while retaining the dependency of each entry on each rate.

Theorem 3.1 (Shared cyclic probability and complete score)**.** For a finite simple graph with fixed positive rates and a supported complete accepted order of even length $L = 2K > 0$, both $q_{\mathcal{C}}$ and all n coordinates of $\bar{s}$ can be computed in $O(n + m + K^2)$ arithmetic operations and storage. The computation sums the full $K2^K$-order record without enumerating its orders. It excludes the original sampling, reward calculation, actor evaluation, and serialization. The singleton construction for odd L satisfies the same bound with $K = L$; optional reflection changes only a constant factor.

After differentiating the block factors, the transpose of the same rectangle and interval operations sends their weighted derivatives back to every vertex. Writing the derivative propagated through the mass arguments with respect to λ_v as d_v gives $\bar{s}_v = \mathbf{1}_{\{v \in S\}} + \lambda_v d_v$, without storing a separate length-n score for each start. Appendix B.2 supplies the indexing and transpose. Generic AD with the same shared representation has the same bound. When $K = \Theta(n)$ on sparse graphs, the quadratic table can dominate; sharing does not guarantee a speedup in that regime.

3.3 Why the averaging range matters

Local pair averaging removes neighboring-order differences but can retain broad position dependence. On a unit-rate star, a leaf selected late contributes to more normalization terms. Define a local layer as the exact conditional expectation of the current score over adjacent pairs, alternating $(1, 2), (3, 4), \dots$ with $(2, 3), (4, 5), \dots$ and retaining unpaired endpoints.

Theorem 3.2 (A score-level local–cyclic separation)**.** On the unit-rate star with even $L \geq 4$, condition on the terminal set of all leaves. There is a constant $c_* > 0$ such that, for every fixed $0 < \eta < c_*$, $\Theta(L^2)$ alternating local layers are necessary and sufficient to reduce the centered tagged-leaf score variance to at most an η fraction of its raw value as L grows through even values. Conditioning once on the cyclic pair record instead gives the complete terminal score $\mathbb{E}[s(Y) \mid S]$. The same conditional variance ratio holds in any fixed actor direction with nonconstant leaf log-rate derivatives; identical derivative rows at all leaves remove this contrast already.

Cyclic starts place each leaf at every position and remove this slow component. Appendix B.3 gives the spectral argument. The comparison concerns exact averaging layers, not the wall time of practical local samplers. General graphs can retain order variation, as quantified below.

What the cyclic average leaves. Let $s_S = \mathbb{E}[s(Y) \mid S(Y)]$ be the complete terminal-conditioned score. For fixed G, θ, define $\mathcal{V}_{\mathrm{raw}} = \mathbb{E}\|s - s_S\|_2^2$ and $\mathcal{V}_{\mathrm{SOM}} = \mathbb{E}\|\bar{s} - s_S\|_2^2$. Since the cyclic record retains S, conditional expectation gives

$$\mathcal{V}_{\mathrm{raw}} = \mathbb{E}\|s - \bar{s}\|_2^2 + \mathcal{V}_{\mathrm{SOM}}, \qquad \rho = \frac{\mathcal{V}_{\mathrm{SOM}}}{\mathcal{V}_{\mathrm{raw}}} \in [0, 1]. \tag{11}$$

The ratio is defined only when $\mathcal{V}_{\mathrm{raw}} > 0$; otherwise both variances vanish. A large order family alone does not bound ρ away from one. Thus the star's complete removal is a mechanism example, not a graph-uniform claim. The same decomposition applies after the fixed actor Jacobian B_θ^T, but does not replace a measurement of the reward-weighted RLOO gradient.

3.4 Training with the conditional score

For $N \geq 2$ independent trajectories at fixed G and θ, let $R_i = R_G(S(Y_i))$ and $c_i = R_i - (N-1)^{-1} \sum_{j \neq i} R_j$. Replace the raw scores in whole-batch RLOO by their conditional averages:

$$\widehat{g}_{\mathrm{SOM}} = B_\theta^\mathsf{T} \left(\frac{1}{N} \sum_{i=1}^{N} c_i \bar{s}(Y_i) \right). \tag{12}$$

Proposition 3.3 (Matched gradient estimators)**.** At fixed G and θ, exact SOM preserves $\nabla_\theta J_G(\theta)$, where $J_G(\theta) = \mathbb{E}[R_G(S(Y)) \mid G]$; averaging over graphs preserves $\nabla_\theta J(\theta)$. For either likelihood-ratio estimation with a common fixed or independent baseline, or whole-batch RLOO with $N \geq 2$, its covariance matrix is no greater in the positive-semidefinite order than that of the corresponding raw-score estimator with the same draws and batch size. The conclusion also holds against a comparator that conditions on fixed adjacent pairs for even L and retains the original order for odd L.

The record preserves each terminal reward, so conditioning the complete batch also preserves its reward coefficients. The covariance claim follows from total covariance; the fixed-pair comparison uses nested records (Kool et al., 2020; Shi et al., 2022; Tong et al., 2025). Appendix B.1 gives the proof. The guarantee compares matched draws, not matched compute.

Implementation. Algorithm 1 implements Eq. 12 with one encoder evaluation and one backward pass for a graph's complete RLOO group. The score computation uses detached rates; only the final proxy loss is differentiated through the actor. Centering by $P = I - \mathbf{1}\mathbf{1}^\mathsf{T}/n$ leaves the exact score unchanged because a common rate rescaling preserves the policy. GPU prefix sums build the shared tables, and the same output-loss interface supports DDP, FSDP2, and Megatron-Core. Matched baselines receive the same forward reuse. Numerical tolerances and distributed normalization are detailed in Appendix C.

Finite-precision scope. If each computed score has original-coordinate error at most ε, the projected batch gradient obeys

$$\|\widetilde{g} - \widehat{g}_{\mathrm{SOM}}\|_2 \leq \frac{\varepsilon}{N} \sum_i |c_i| \, \|B_\theta^\mathsf{T} P\|_{\infty \to 2} \leq \varepsilon \|B_\theta^\mathsf{T} P\|_{\infty \to 2}, \tag{13}$$

using $|c_i| \leq 1$ for normalized rewards. This pointwise guarantee, derived in Appendix B, is distinct from exact unbiasedness or a covariance ordering for rounded scores.

Algorithm 1 One SOM training step for a graph

Require: Graph $G = (V, E, w)$ with $n > 0$, actor θ, $N \geq 2$ draws, optimizer state o

$z \leftarrow z_\theta(G)$; $\lambda \leftarrow \exp(\mathrm{stopgrad}(z))$ ▷ encode once
for $i = 1, \ldots, N$ **do**
  $Y_i \leftarrow \mathrm{SampleGreedy}(G, \lambda)$; $S_i \leftarrow S(Y_i)$; $L_i \leftarrow |Y_i|$
  $R_i \leftarrow w(S_i)/w(V)$
  if L_i is even **then**
    $\mathcal{C}_i \leftarrow \mathrm{CyclicPairs}(Y_i)$; $K_i \leftarrow L_i/2$
  else
    $\mathcal{C}_i \leftarrow \mathrm{CyclicSingletons}(Y_i)$; $K_i \leftarrow L_i$
  end if
  $D_i \leftarrow \mathrm{DeletingBlocks}(G, \mathcal{C}_i)$
  $T_i \leftarrow \mathrm{SharedMasses}(\lambda, \mathcal{C}_i, D_i)$
  $(q_{\mathcal{C}_i}, \omega_i) \leftarrow \mathrm{BlockFactors}(T_i, \lambda, \mathcal{C}_i)$
  $\bar{s}_i \leftarrow \mathrm{SharedTranspose}(T_i, \omega_i, \lambda, S_i, D_i)$
end for
$c_i \leftarrow R_i - (\sum_j R_j - R_i)/(N-1)$ for all i
$u \leftarrow N^{-1} \sum_i c_i \bar{s}_i$; $u \leftarrow u - \mathbf{1}(\mathbf{1}^\mathsf{T} u)/n$
$\mathcal{L} \leftarrow -\langle \mathrm{stopgrad}(u), z \rangle$; $(\theta, o) \leftarrow \mathrm{OptimizerStep}(\theta, o, \nabla_\theta \mathcal{L})$

4 Experiment

We evaluate SOM on weighted MWIS and unweighted MIS, comparing solution quality and training efficiency.

4.1 Experimental Setup

Datasets and models. We use GraphGPS (Rampásek et al., 2022) as the main model, with Graphormer (Ying et al., 2021) and GatedGCN (Bresson & Laurent, 2017) as additional backbones. For weighted MWIS, we train on induced MWIS-VR subgraphs with original weights and test on complete held-out parent graphs (Dong et al., 2021). For MIS, we use RB200–300, RB800–1200, and ER700–800, following the splits of X^2GNN (Acikalin et al., 2025).

Baselines. We compare two groups of methods: training estimators and independent-set solvers. The first includes REINFORCE, RLOO (Kool et al., 2020), additional sampling, fixed and alternating pair averaging, and same-solution replay (Kim et al., 2024). These methods use the same encoder and sampler. The second includes CHILS (Großmann et al., 2025) and weighted degree-greedy for MWIS, and X^2GNN, pCQO-MIS (Alkhouri et al., 2025), KaMIS (Großmann et al., 2024; Hespe et al., 2019), and degree-greedy for MIS.

Evaluation. Each GraphGPS estimator receives 24 training GPU-hours per run, with 5 seeds paired across methods and matched test-time limits. We report reference gaps using certified optima where available and best-known feasible solutions otherwise; lower is better. Training efficiency is measured by time to a common validation target and the fraction of runs reaching it. Appendix D.7 gives the aggregation rules.

Weighted results first average variants within each held-out parent, then weight parents equally; paired uncertainty respects the parent/seed hierarchy. The validation target is fixed before test evaluation. We report target attainment alongside restricted mean GPU-hours through a common deadline, retaining every assigned seed. Early failures are reported with their actual consumed compute, not treated as successful runs at the deadline or silently excluded. Thus neither a favorable subset of graphs nor a successful-seed-only average determines the comparison.

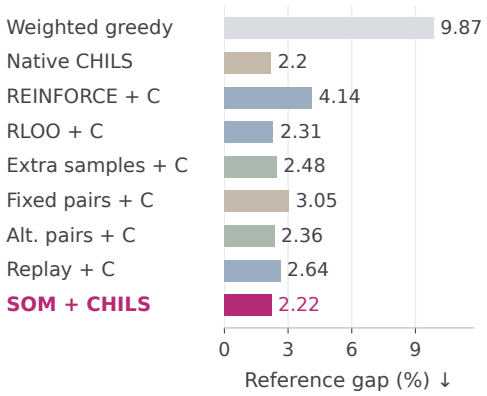
Implementation details. Experiments use eight NVIDIA H20 GPUs. Unless stated otherwise, each model predicts rates once per graph and reuses the forward pass across sampled solutions. Matched methods share the optimizer, data, and inference procedure. Training time includes sampling, averaging, communication, and updates. Model and distributed settings are given in Appendices D.2 and D.6.

4.2 Performance Evaluation

Performance comparison with baselines. Fig. 3 compares SOM with the baselines on MWIS-VR and the three MIS benchmarks under matched budgets.

Figure 3(a) compares neural initializers followed by CHILS with native CHILS under a common wall deadline, not a CHILS-free solver. The mean normalized-value difference is -0.02 percentage points; its paired interval crosses zero. MIS uses native decoders without CHILS; KaMIS remains stronger on these suites. Appendix E reports pre/post-search values, all comparison units, intervals and the separate CPU/GPU allocations.

(a) MWIS pipelines · same wall limit



(b) MIS · native decoders

	RB small	RB large	ER
Greedy	8.94	13.12	10.58
KaMIS	2.06	3.86	3.11
X2GNN	3.71	6.57	5.49
pCQO-MIS	4.85	7.34	5.79
RLOO	5.76	9.23	7.03
SOM	**3.12**	**5.36**	**4.18**

Reference gap (%) ↓

Figure 3: Pipeline reference gaps (%). Left: neural initializers + CHILS versus native weighted baselines, same wall limit. Right: native MIS decoders. Intervals and resource differences: Appendix E.

Learning progress and policy scope. Fig. 4 compares fixed-rate RLOO and SOM with per-step re-encoding RLOO, including an additional-sample control. SOM retains one encoder call per solution, while re-encoding achieves lower final gaps at greater per-solution cost. Because fixed-rate RLOO and SOM use the same policy class, their difference is not an increase in expressiveness. Appendix D.8 reports the separate equal-update comparison.

(a) Policies + CHILS · equal budget

	Gap (%) ↓	Target (GPU-h) ↓	Infer. (ms) ↓	Encoder calls ↓
Fixed RLOO	2.31	5.64	4.79	1
Fixed SOM	**2.22**	**4.61**	**4.9**	**1**
Re-encode RLOO	2.08	6.12	301.86	64
Re-encode extra	1.92	5.67	315.22	64

(b) GraphGPS learning · MWIS-VR

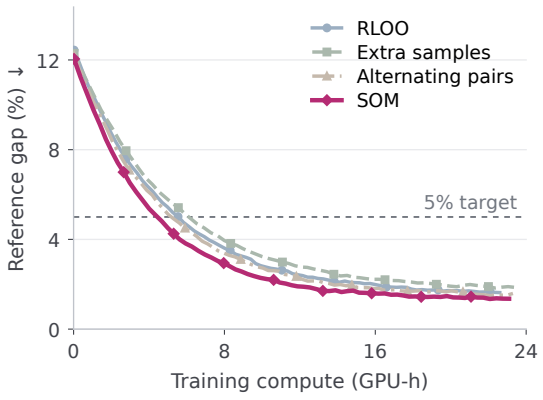


Figure 4: Policy scope and learning. Left: fixed-rate and re-encoding policies under matched budgets; columns report quality, target cost, latency, and encoder calls. Right: GraphGPS checkpoints aligned by update, plotted at mean GPU-hours.

Effectiveness of order averaging. We compare parameter-gradient covariance at frozen checkpoints using matched RLOO batches. SOM reduces the covariance trace by 32.3% relative to native RLOO. This checks whether order-noise reduction survives backpropagation, not whether it alone causes the learning-speed gain. The relative mean discrepancy is 0.006, compared with reference uncertainty 0.013; this is a numerical/statistical check, not a proof of exact machine unbiasedness. The star control and reference protocol are reported in Appendix D.4.

Residual variation and pairing. Exact enumeration on small ER, RB, and MWIS-VR instances evaluates the residual in Eq. 11. On MWIS-VR subgraphs, SOM removes 56.0% of score variation; it does not remove all order dependence. Enumeration covers 68.800% of the prespecified diagnostic draws, so this result is conditional on the eligible small-graph panel, not a terminal oracle for complete test graphs. Cyclic pairs also reduce parameter-score variance by 30.8% relative to fixed pairs on matched even-length draws. These diagnostics separate the gain from rotating the starting block from within-pair averaging. Fig. 7 gives both panels; enumeration coverage and record audits remain in Appendices D.9–D.10.

Odd-length coverage. Odd-length singleton cycles reduce parameter-score variance by 22.0%, despite averaging a smaller family. We retain both parities under the original sampling distribution and compute pooled covariance from the complete vector moments, rather than average parity-specific ratios. Fig. 8(a) reports occurrence and cost, including the singleton-on-even control in Appendix D.11.

4.3 Efficiency Evaluation

We next evaluate whether the learning benefit compensates for the additional averaging cost.

Training efficiency. We measure training time to a common validation target using GraphGPS, Graphormer, and GatedGCN. Fig. 5 compares target GPU-hours across backbones and decomposes the complete step cost.

On GraphGPS, SOM reduces target time by 18.3% relative to fixed-configuration RLOO. Appendix E.4 reconciles native/SOM step time, update count and overhead. Separately tuned RLOO/SOM have a target-cost ratio of 1.110; Appendix E.8 gives paired ratios and multiple targets.

Efficiency of shared computation. We compare separate-start reconstruction, shared generic AD, and the shared specialized implementation on identical graphs, rates, sampled orders, and accuracy requirements. Fig. 5 reports averaging time, complete step time, and peak memory.

Shared specialized computation reduces complete step time by 55.2% over separate-start reconstruction. Shared generic AD also improves, supporting the value of the shared representation. Reusing vertex contributions avoids repeated graph scans while preserving the same average.

Efficiency on sparse graphs. We vary graph size and sparsity to examine the cost of the quadratic table. Fig. 6 relates the complete-step time ratio to $K^2/(n+m)$, with table memory annotated.

Sharing reduces complete step time by 52.0% on MWIS-VR, but is slower on the sparse stress panel. Long independent sets enlarge the table and can outweigh the saving from graph scans. Fig. 6 shows this trade-off; failure and capacity results are given in Appendix D.12.

Low-cost alternatives. We compare SOM with additional independent samples and a practical adjacent heat-bath average at equal training GPU-hours and tuning budgets. All transition, score, and backward costs are charged. SOM reduces target time by 13.0% relative to the local control (Fig. 8(b)). This finite-budget comparison is distinct from Theorem 3.2: a sampled local chain is not a sequence of exact conditional-expectation layers.

Distributed training efficiency. We evaluate DDP (Li et al., 2020) on one to eight H20 GPUs and test FSDP2 (Zhao et al., 2023) and Megatron-Core (Narayanan et al., 2021) with larger models. DDP scaling fixes the global batch; framework comparisons use the same model. Within each configuration, SOM and RLOO receive identical resources.

Within Megatron-Core, SOM reduces target time by 12.5% over RLOO. Eight-GPU DDP efficiency is 0.69; the separate Megatron/DDP throughput ratio is 1.36. Gradient consistency and the distinction between infrastructure and learning gains are detailed in Appendix E.

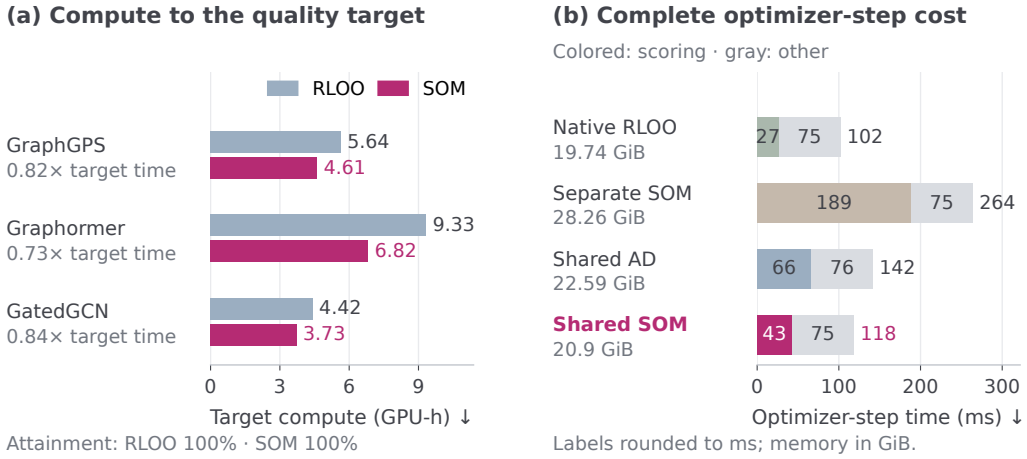


Figure 5: Training and computation cost. Left: restricted mean target GPU-hours. Right: scoring and other step stages; native RLOO is included. Full intervals and accounting: Appendix E.4.

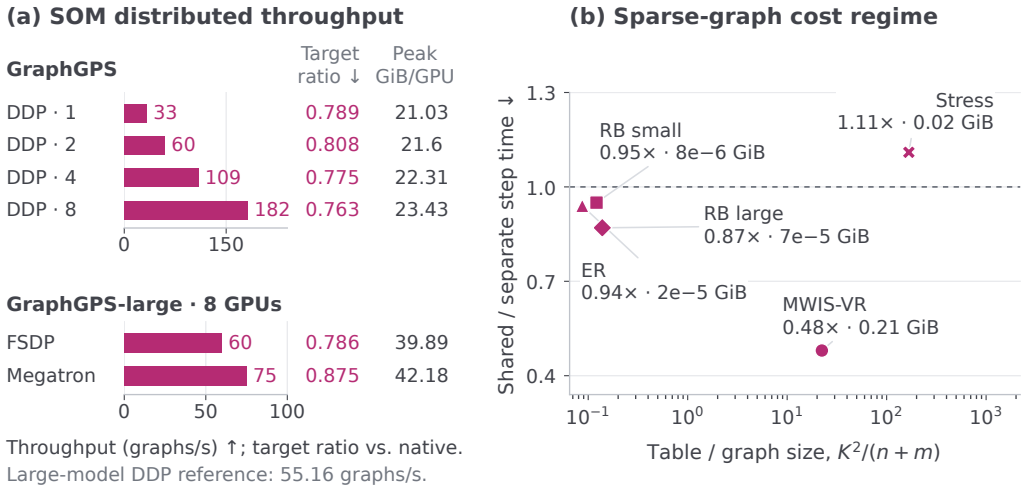


Figure 6: Scaling and computational limits. Left: DDP strong scaling and the separate larger-model configurations. Right: complete-step ratios across graph regimes; the dashed line marks equal cost. No line joins unrelated graph workloads.

5 Conclusion

SOM trains fixed-rate graph policies by probability-weighted averaging over alternative orders of the same independent set. Pairwise orientations and cyclic starts broaden the average, while shared vertex contributions supply its probability and complete score without rebuilding the graph for each start. The analysis establishes gradient preservation, covariance reduction under matched sampling, and a score-level separation from the specified exact adjacent-pair layers on stars. The comparisons concern a learned-initialization pipeline and same-policy training cost, not uniform solver superiority; appendix controls expose uncertainty and resource differences. These guarantees do not imply universal finite-budget acceleration: general graphs retain residual order variation, and long solutions can make the quadratic table costly. The method changes training, not the fixed inference decoder. Extending this reuse to decision-dependent rates, whose changing outputs prevent shared masses, remains open. A useful extension must preserve the conditional sampling law while accounting for the cost of changing network predictions; a larger family of equivalent orders alone is not sufficient.

Statement of AI Use

AI assistants, including Codex and Claude, were used to help develop and check mathematical arguments, inspect code, organize the literature, and draft and revise the manuscript.

References

Utku Umur Acikalin, Aaron M. Ferber, and Carla P. Gomes. Learning to explore and exploit with gnns for unsupervised combinatorial optimization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=vaJ4FObpXN.

Ismail Alkhouri, Cedric Le Denmat, Yingjie Li, Cunxi Yu, Jia Liu, Rongrong Wang, and Alvaro Velasquez. Differentiable quadratic optimization for the maximum independent set problem. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/alkhouri25a.html.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *CoRR*, abs/1711.07553, 2017. URL http://arxiv.org/abs/1711.07553.

Persi Diaconis and Arun Ram. Analysis of systematic scan metropolis algorithms using iwahori-hecke algebra techniques. *Michigan Mathematical Journal*, 48(1):157–190, 2000.

Yuanyuan Dong, Andrew V. Goldberg, Alexander Noe, Nikos Parotsidis, Mauricio G. C. Resende, and Quico Spaen. New instances for maximum weight independent set from a vehicle routing application. *Oper. Res. Forum*, 2(4), 2021. doi: 10.1007/S43069-021-00084-X. URL https://doi.org/10.1007/s43069-021-00084-x.

Ernestine Großmann, Sebastian Lamm, Christian Schulz, and Darren Strash. Finding near-optimal weight independent sets at scale. *J. Graph Algorithms Appl.*, 28(1):439–473, 2024. doi: 10.7155/JGAA.V28I1.2997. URL https://doi.org/10.7155/jgaa.v28i1.2997.

Ernestine Großmann, Kenneth Langedal, and Christian Schulz. Concurrent iterated local search for the maximum weight independent set problem. In Petra Mutzel and Nicola Prezza (eds.), *23rd International Symposium on Experimental Algorithms, SEA 2025, Venice, Italy, July 22-24, 2025*, volume 338 of *LIPIcs*, pp. 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi: 10.4230/LIPICS.SEA.2025.22. URL https://doi.org/10.4230/LIPIcs.SEA.2025.22.

Demian Hespe, Christian Schulz, and Darren Strash. Scalable kernelization for maximum independent sets. *ACM J. Exp. Algorithmics*, 24(1):1.16:1–1.16:22, 2019. doi: 10.1145/3355502. URL https://doi.org/10.1145/3355502.

Hyeonah Kim, Minsu Kim, Sungsoo Ahn, and Jinkyoo Park. Symmetric replay training: Enhancing sample efficiency in deep reinforcement learning for combinatorial optimization. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24110–24136. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/kim24o.html.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/0cddb777d3441326544e21b67f41bdc8-Abstract-Conference.html.

Wouter Kool, Herke van Hoof, and Max Welling. Estimating gradients for discrete random variables by sampling without replacement. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rklEj2EFvB.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f231f2107df69eab0a3862d50018a9b2-Abstract.html.

Sebastian Lamm, Christian Schulz, Darren Strash, Robert Williger, and Huashuo Zhang. Exactly solving the maximum weight independent set problem on large real-world graphs. In Stephen G. Kobourov and Henning Meyerhenke (eds.), *Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments, ALENEX 2019, San Diego, CA, USA, January 7-8, 2019*, pp. 144–158. SIAM, 2019. doi: 10.1137/1.9781611975499.12. URL https://doi.org/10.1137/1.9781611975499.12.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, 2020. doi: 10.14778/3415478.3415530. URL http://www.vldb.org/pvldb/vol13/p3005-li.pdf.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using megatron-lm. In Bronis R. de Supinski, Mary W. Hall, and Todd Gamblin (eds.), *International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2021, St. Louis, Missouri, USA, November 14-19, 2021*, pp. 58. ACM, 2021. doi: 10.1145/3458817.3476209. URL https://doi.org/10.1145/3458817.3476209.

NVIDIA. Megatron-Core: Pipeline parallel schedules, 2026. URL https://docs.nvidia.com/megatron-core/developer-guide/latest/apidocs/core/core.pipeline_parallel.schedules.html. Official documentation, accessed September 19, 2026.

Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

PyTorch Contributors. Getting started with fully sharded data parallel (FSDP2), 2026. URL https://docs.pytorch.org/tutorials/intermediate/FSDP_tutorial.html. Official documentation, accessed September 19, 2026.

Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/5d4834a159f1547b267a05a4e2b7cf5e-Abstract-Conference.html.

Shinsaku Sakaue. Differentiable greedy algorithm for monotone submodular maximization: Guarantees, gradient estimators, and applications. In Arindam Banerjee and Kenji Fukumizu (eds.), *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 28–36. PMLR, 2021. URL http://proceedings.mlr.press/v130/sakaue21a.html.

Jiaxin Shi, Yuhao Zhou, Jessica Hwang, Michalis K. Titsias, and Lester Mackey. Gradient estimation with discrete stein operators. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/a5a5b0ff87c59172a13342d428b1e033-Abstract-Conference.html.

David Stutz, Ali Taylan Cemgil, Abhijit Guha Roy, Tatiana Matejovicova, Melih Barsbey, Patricia Strachan, Mike Schaekermann, Jan Freyberg, Rajeev V. Rikhye, Beverly Freeman, Javier Perez

Matos, Umesh Telang, Dale R. Webster, Yuan Liu, Gregory S. Corrado, Yossi Matias, Pushmeet Kohli, Yun Liu, Arnaud Doucet, and Alan Karthikesalingam. Evaluating AI systems under uncertain ground truth: a case study in dermatology. *CoRR*, abs/2307.02191, 2023. doi: 10.48550/ARXIV.2307.02191. URL https://doi.org/10.48550/arXiv.2307.02191.

Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Rao-blackwell gradient estimators for equivariant denoising diffusion. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/442d8241f5ed92011e9ddaaea59c0ce2-Abstract-Conference.html.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

David Bruce Wilson. Mixing times of lozenge tiling and card shuffling markov chains. *The Annals of Applied Probability*, 14(1):274–325, 2004.

Yikai Wu, Haoyu Zhao, and Sanjeev Arora. Unrealized expectations: Comparing AI methods vs classical algorithms for maximum independent set. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=ksGoCT5zW6.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 28877–28888, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f1c1592588411002af340cbaedd6fc33-Abstract.html.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.

A Related Work

A.1 Learning independent-set solvers

Learned graph representations support several approaches to combinatorial optimization. X^2GNN combines exploration and exploitation for unsupervised optimization, while differentiable quadratic optimization provides another approach to maximum independent set (Acikalin et al., 2025; Alkhouri et al., 2025). These methods motivate task-level comparisons, but their objectives and inference procedures differ from our fixed-rate construction policy. GraphGPS and Graphormer are established graph representation architectures rather than MWIS-specific state-of-the-art solvers (Rampásek et al., 2022; Ying et al., 2021). We use them to test whether a change to the training estimator improves a common neural policy across backbones, with a residual gated graph convolutional network as a control (Bresson & Laurent, 2017).

Strong classical solvers remain essential to assessing this improvement. Exact branch-and-reduce algorithms and concurrent iterated local search exploit graph structure directly (Lamm et al., 2019; Großmann et al., 2025). Recent comparisons show that learned independent-set methods can fall short of classical algorithms under demanding evaluation (Wu et al., 2026). Accordingly, a gain over

a matched policy-gradient baseline establishes a training improvement, while comparisons with external solvers establish its task-level relevance. The two conclusions require different comparisons. Vehicle-routing conflict instances supply an application-grounded weighted task (Dong et al., 2021); unweighted benchmark families assess a related but distinct objective.

A.2 Reusing trajectories and averaging symmetries

Policy-gradient estimators weight the gradient of a discrete sample's log probability by its observed return (Williams, 1992). Baselines, control variates, and conditional expectation can reduce their variance. Kool et al. (2020) condition on an unordered collection of samples drawn without replacement and average over the order in which those samples were drawn. SOM instead averages the internal vertex-construction order of one independent set. Symmetric Replay Training reuses symmetry-related solutions through an additional training objective (Kim et al., 2024). It provides a comparison for sample efficiency, with the distinction that its imitation term changes the optimization objective.

Multi-start and symmetry-aware neural optimization already exploit nonunique solution representations. POMO uses multiple starting decisions and a shared reward baseline to train constructive solvers (Kwon et al., 2020). Sym-NCO introduces symmetry-based regularization for neural combinatorial optimization (Kim et al., 2022). Multiple starts can generate different terminal solutions; SOM instead keeps a sampled terminal set fixed and integrates an explicitly defined family of its construction orders. These works establish that exploiting multiple representations is valuable. Our question concerns the cost of the graph-dependent conditional likelihood and its complete gradient, rather than the introduction of multiple starts or symmetry itself.

Symmetry-based Rao–Blackwell estimators also occur in equivariant diffusion, including likelihood-weighted orbit constructions and estimators that exploit equivariant computation (Tong et al., 2025). These works establish the use of nonuniform orbit averages and shared neural evaluations. Discrete Stein operators provide a general framework for unbiased control variates and can express conditional-expectation corrections (Shi et al., 2022). SOM addresses the graph-dependent computation needed to apply these principles: evaluating a specified order family's probability and all original-vertex derivatives while reusing its overlapping deletion calculations. The shared tables and their transpose provide this computation for fixed-rate independent-set policies.

A.3 Partial rankings and local order dynamics

Summing permutation likelihoods is an established problem in ranking models, including the classical Plackett–Luce law (Plackett, 1975). For example, the partial-ranking calculation in Appendix B.1 of Stutz et al. (2023) uses subset dynamic programming for Plackett–Luce probabilities. Our policy differs because selecting a vertex deletes its conflicting neighbors, changing denominators in a graph-dependent way. The path example in Section 2 shows why a ranking likelihood on a static candidate set cannot be substituted for the graph policy. Our factorization applies to independent blocks whose internal order leaves the same available set at each block boundary; the computational saving comes from sharing these boundaries across cyclic starts.

The diffusion of positions under adjacent updates, including cosine eigenfunctions and slow modes, is classical in the analysis of card shuffling and systematic scans (Wilson, 2004; Diaconis & Ram, 2000). Our analysis identifies a nonvanishing projection of the sampled-order MWIS log-rate score onto such a mode, establishes the resulting conditional-variance bound, and determines when it remains visible in the parameter gradient. The star gives a tractable example with a cheap terminal-score oracle. It establishes the behavior of the specified local averaging scheme; other methods can exploit the star's structure directly.

A.4 Distributed training

Data, tensor, and pipeline parallelism address the cost of neural-network computation and model storage (Li et al., 2020; Zhao et al., 2023; Narayanan et al., 2021). They are complementary to reducing the variance of the learning signal. Our distributed implementation passes a graph-level cotangent through an ordinary scalar loss, which allows SOM and its baselines to use the same model partitioning and optimizer. The relevant system comparison is therefore time to the same

held-out quality within each framework, with estimator work and communication included. A faster framework alone would not establish a contribution from order marginalization.

B Proofs and Numerical Guarantees

All statements fix a finite simple graph $G = (V, E)$ and a vector of positive rates $\lambda \in (0, \infty)^n$ during a complete construction. The derivatives below are with respect to the independent original log rates $z_v = \log \lambda_v$, including when several rate values coincide. The actor Jacobian B_θ is applied only after computing these original-coordinate scores. We write $N[A] = \bigcup_{v \in A} N[v]$ for a union of closed neighborhoods, $\lambda(A) = \sum_{v \in A} \lambda_v$, and $J_G(\theta) = \mathbb{E}_{Y|G}[R_G(S(Y))]$. Graph averaging recovers the objective J in the main text.

B.1 Support, scores, and conditional gradient estimation

Let $Y = (y_1, \dots, y_L)$ be a complete accepted sequence, and put $A_\ell = V \setminus N[\{y_1, \dots, y_{\ell-1}\}]$. Its probability and original-coordinate score are

$$p_\lambda(Y) = \prod_{\ell=1}^{L} \frac{\lambda_{y_\ell}}{\lambda(A_\ell)}, \qquad s_v(Y) = \mathbf{1}_{\{v \in S\}} - \lambda_v \sum_{\ell=1}^{L} \frac{\mathbf{1}_{\{v \in A_\ell\}}}{\lambda(A_\ell)}. \tag{14}$$

The accepted set S is independent because selecting a vertex deletes its neighbors, and it is maximal because the process terminates only when the active set is empty. Conversely, fix a maximal independent set S and any permutation of it. No accepted vertex deletes another member of S, so each proposed choice is active and has positive probability. Once all of S has been selected, maximality implies $N[S] = V$. Thus every permutation is a supported complete sequence, and its probability is given by (14). Overlap between deletion neighborhoods does not affect this argument.

Any record used here partitions this finite support without reference to the numerical rate values. For a record value C, let $q_C = \sum_{Y : \mathcal{C}(Y) = C} p_\lambda(Y)$. The set of summands is fixed in a neighborhood of every positive rate vector. Differentiating the finite sum gives

$$\nabla_z \log q_C = \sum_{Y : \mathcal{C}(Y) = C} \frac{p_\lambda(Y)}{q_C} s(Y) = \mathbb{E}[s(Y) \mid \mathcal{C}(Y) = C]. \tag{15}$$

In particular, the derivative is of the log probability of the record. Averaging derivatives of record probability itself would be a different quantity.

Proof of Proposition 3.3. For a fixed baseline β, the terminal reward is measurable from $\mathcal{C}(Y)$, and therefore

$$(R_G(S) - \beta) B_\theta^\mathsf{T} \bar{s}(Y) = \mathbb{E}[(R_G(S) - \beta) B_\theta^\mathsf{T} s(Y) \mid \mathcal{C}(Y)]. \tag{16}$$

The expectation is $\nabla_\theta J_G(\theta)$ because $\mathbb{E}[s] = 0$ and the finite-support likelihood-ratio identity applies. An independent random baseline is handled by conditioning on that baseline first. For an average of independent draws, condition on their complete tuple of records, together with any independent pilot baseline. The law of total covariance gives the positive-semidefinite reduction.

For RLOO, define $\mathcal{F} = \sigma(\mathcal{C}(Y_1), \dots, \mathcal{C}(Y_N))$. Every reward and every coefficient $c_i = R_i - (N-1)^{-1} \sum_{j \neq i} R_j$ is $\mathcal{F}$-measurable. Independence of the original draws implies $\mathbb{E}[s(Y_i) \mid \mathcal{F}] = \bar{s}(Y_i)$. Hence the entire estimator in (12) equals $\mathbb{E}[\widehat{g}_{\mathrm{raw}} \mid \mathcal{F}]$, and

$$\mathrm{Cov}(\widehat{g}_{\mathrm{raw}}) - \mathrm{Cov}(\widehat{g}_{\mathrm{SOM}}) = \mathbb{E}[\mathrm{Cov}(\widehat{g}_{\mathrm{raw}} \mid \mathcal{F})] \succeq 0. \tag{17}$$

The raw RLOO estimator is unbiased: the cross terms $\mathbb{E}[R_j s(Y_i)]$ vanish for $j \neq i$, while $\mathbb{E}[R_i s(Y_i)] = \nabla_z J_G$. This proves both assertions relative to raw scores for the complete batch, without treating its reward coefficients as independent of their own draws.

For the fixed-pair comparison, define $\mathcal{E}(Y)$ to retain the ordered list of unordered adjacent pairs when L is positive even, and to retain the entire order otherwise. Then $\mathcal{C}(Y)$ is a function of $\mathcal{E}(Y)$: on the even branch it additionally forgets the starting pair, and on the odd branch it forgets the starting singleton. Thus the tower property gives $\mathbb{E}[\mathbb{E}[s \mid \mathcal{E}] \mid \mathcal{C}] = \bar{s}$. Repeating the single-draw and

whole-batch arguments above with $\mathbb{E}[s \mid \mathcal{E}]$ in place of s proves the additional covariance ordering. This comparison does not replace the odd branch by the different record of adjacent pairs with an unpaired tail, which need not be nested with the singleton cycle. □

The same reasoning proves covariance reduction at each alternating local layer. If $f_{j+1} = \mathbb{E}[f_j \mid \mathcal{A}_{j+1}]$, nonnesting of successive σ-fields does not obstruct the law of total covariance. For the whole batch, each local record retains S_i, so the corresponding reward coefficients remain measurable. Also, terminal conditioning satisfies $\mathbb{E}[f_j \mid S] = \mathbb{E}[s \mid S]$ at every layer. These facts do not make a composition of nonnested projections the score of a single new likelihood, nor do they order that composition against an unrelated cyclic record.

B.2 Cyclic factorization and the shared risk construction

For a fixed starting block, the family of all within-pair orientations is called a fibre.

Use positive even $L = 2K$ first. The blocks C_b, $b = 0, \dots, K-1$, have distinct labels and disjoint vertex sets. Rotations and independent within-block flips form a group acting freely on the accepted sequences in a given cyclic record. There are K choices of starting block and 2^K orientations. In particular, rotating a record does not rediscover a different partition or select a different set of blocks. Reflection normalizes the within-block flip group, so forgetting direction gives $2K$ starting-direction choices when $K \geq 3$. For $K = 1, 2$ the reflected block orders duplicate rotations and are counted only once.

For fixed r, before block b every orientation of the previous complete blocks has deleted the same union, giving (6). The endpoints a_b, c_b are nonadjacent and active. After either orientation, the new active set is $A_{r,b} \setminus (N[a_b] \cup N[c_b])$. Consequently summing the two orientation probabilities at each block commutes with multiplying across blocks. Their sum is

$$\frac{\lambda_{a_b}\lambda_{c_b}}{Z_{r,b}}\left(\frac{1}{U_{r,b}} + \frac{1}{V_{r,b}}\right), \tag{18}$$

which proves (8). All quantities are strictly positive. An overlap vertex belongs to both removal masses when appropriate, but it is removed only once from the active set after the block. This is why the two removal sets must both be formed in the pre-block active set.

Lemma B.1 (Periodic gap representation)**.** Extend D_v periodically to $\widehat{D}_v = D_v + K\mathbb{Z}$. For integers $0 \leq r < K$ and $r \leq b < r + K$, vertex v belongs to $A_{r,b}$ if and only if there are consecutive $p < q$ in $\widehat{D}_v$ such that $r, b \in \{p+1, \dots, q\}$.

Proof. By definition, v survives exactly when no block in $[r, b)$ belongs to $\widehat{D}_v$. If r, b belong to the displayed gap, there is no hit between them. Conversely, let p be the last hit strictly before r, and let q be the next hit. The absence of hits in $[r, b)$ is equivalent to $b \leq q$, proving the claim. The hit at $b = q$ has not yet been processed. Each valid cell has exactly one such gap, since the half-open intervals $(p, q]$ partition the integer line. Nonempty D_v also implies $q - p \leq K$. □

Here is an explicit finite construction. Scan selected vertices by increasing block index, and scan their closed neighborhoods. A last-seen-block marker for each original vertex appends a hit only once, while recording whether the first endpoint, second endpoint, or both caused the incidence. The lists are sorted by construction; there is no comparison sort. The total scan length is at most $L + \sum_{v \in S} \deg(v) \leq n + 2m$. Vertices not selected are included in these same lists.

For each v, consider its $|D_v|$ canonical gaps, choosing the left hit p in $\{0, \dots, K-1\}$ and the next hit q in the periodic list. Translate each gap by $-K, 0, K$ and intersect it with the grid index interval $\{0, \dots, 2K-1\}$. Each nonempty interval I contributes a rectangle $I \times I$ with value λ_v. Accumulate a rectangle $[l, h] \times [l, h]$ using four corner updates at $(l, l), (h+1, l), (l, h+1), (h+1, h+1)$, with signs $+, -, -, +$, in a difference array with an extra sentinel row and column. Two-dimensional prefix summation produces its indicator. The three translations include every gap relevant to the valid wedge $0 \leq r < K$, $r \leq b < r + K$, and no valid cell receives two copies of the same vertex. Lemma B.1 therefore gives $Z_{r,b}$. Entries outside this wedge are auxiliary values and are never interpreted as stages.

To obtain M^a and M^c, fix an endpoint incidence of v in block $b \bmod K$, and let p be the preceding hit of v, strictly before this block in the periodic list. At this block v is still active exactly for cyclic starting positions in $(p, b]$, an interval of length at most K. Represent that interval by one ordinary interval, two intervals if it wraps, or the full set of rows if its length is K. Add λ_v to the appropriate endpoint's column over those rows using one-dimensional difference updates and a sentinel row. Prefix summation along each column gives its removal mass. For column $b_0 \in \{0, \dots, K-1\}$ and starting row r, the corresponding unwrapped index is $b = r + ((b_0 - r) \bmod K)$; this is the unique lift in the valid wedge. A vertex adjacent to both endpoints receives one hit in D_v and one contribution in each endpoint column, as required.

There are $O(n+m)$ hit incidences and a constant number of updates per incidence. The tables contain $O(K^2)$ entries. Forming U, V, all block factors, all products q_r, and their sum requires $O(K^2)$ further arithmetic. Reflection repeats the construction with the opposite directed block list and only changes the constant. For odd L, use $K = L$ singleton blocks and rotate their order. This is a parameter-independent partition without requiring an odd-tail pair partition to be closed under rotations. Singleton blocks need only the gap construction for Z; their fibre probability is $\prod_{v \in S} \lambda_v \prod_b Z_{r,b}^{-1}$. The same accounting applies with $K = L$. The empty graph is handled in constant work with probability one and the empty score. A length-one accepted sequence has a one-order record and is evaluated by (14).

For completeness, the derivative pass can be written without differentiating any division through an implicit selection. Set $F_b = (U+V)/(ZUV)$. Its differential is

$$d \log F_b = -\beta_Z \, dZ - \beta_U \, dU - \beta_V \, dV, \tag{19}$$

$$(\beta_Z, \beta_U, \beta_V) = \left(\frac{1}{Z}, \frac{V}{U(U+V)}, \frac{U}{V(U+V)} \right). \tag{20}$$

Since $d \log \sum_r q_r = \sum_r \omega_r \, d \log q_r$, the cotangents of (Z, M^a, M^c) are $(-\omega_r(\beta_Z + \beta_U + \beta_V), \omega_r \beta_U, \omega_r \beta_V)$. These follow by substituting $U = Z - M^a$ and $V = Z - M^c$ in the differential. No additional derivative of ω_r appears in this first differential. Build summed-area tables of the Z cotangents, assigning zero to unused cells. Each rectangle used in the forward pass now contributes one rectangle-sum query to its vertex's cotangent. Likewise, column prefix sums of the removal cotangents allow each original cyclic interval to be transposed with one or two interval-sum queries. This is precisely the transpose of the linear mass map, evaluated in $O(n+m+K^2)$ work. Adding $\mathbf{1}_S$ from the selected numerator and multiplying the mass cotangent by λ_v gives every original log-rate coordinate. Even if $U = V$ numerically, their removal paths remain separate in this transpose.

This proves Theorem 3.1, including storage: retain the hit incidences, the $O(K^2)$ primal and cotangent tables, and $O(n)$ output accumulators. One need not retain K length-n fibre score vectors. Independently rebuilding a graph process for each rotation would repeat $O(K(n+m))$ incidence work; the theorem removes that repetition. It does not provide a lower bound for all competing algorithms. Generic reverse-mode differentiation of the same bounded-arity difference and prefix computation has the same asymptotic cost, because both its forward graph and its reverse have $O(n+m+K^2)$ operations.

B.3 A SLOW MODE OF THE ACTUAL SCORE ON A STAR

Let the center be c, let there be $L = 2K \geq 4$ leaves, and evaluate independent log-rate derivatives at $\lambda_v = 1$ for every vertex. The terminal leaf set has probability $L/(L+1)$. Each of its $L!$ orders has probability $1/[(L+1)(L-1)!]$, since the first accepted leaf deletes the center and the remaining leaves are selected without replacement. Its conditional order law is therefore uniform. For a tagged leaf at position i, (14) gives

$$s_{\mathrm{tag}}(i) = 1 - \frac{1}{L+1} - \sum_{r=1}^{i-1} \frac{1}{L-r}, \qquad \mathbb{E}[s_{\mathrm{tag}} \mid S] = \frac{1}{L(L+1)}. \tag{21}$$

Writing $H_j = \sum_{q=1}^{j} 1/q$ with $H_0 = 0$, subtracting this mean gives the centered position profile

$$g_L(i) = 1 + H_{L-i} - H_L. \tag{22}$$

These are derivatives in the full independent-rate space at the unit-rate point, not derivatives along a family that ties all rates. The center's score is the constant $-1/(L+1)$ throughout the leaf terminal.

Let P_E condition on the ordered list of unordered pairs at positions $(1,2),(3,4),\ldots$, and let P_O condition on the offset pairs $(2,3),(4,5),\ldots$, retaining unpaired endpoints. Both records retain the terminal set and sequence length. Define $f_0 = s$, $f_1 = P_E s$, $f_2 = P_O f_1$, $f_3 = P_E f_2$, and continue alternating. Each operator averages the current vector. The identities below analyze these exact conditional expectations.

Use the inner product $\langle u, v\rangle_L = L^{-1}\sum_{i=1}^{L} u_i v_i$ for position profiles. Finite summation yields

$$\sum_{j=0}^{L-1} H_j = L(H_L - 1), \qquad \sum_{j=0}^{L-1} H_j^2 = LH_L^2 - (2L+1)H_L + 2L. \tag{23}$$

For example, the first identity follows by counting the $L-q$ occurrences of $1/q$, and expanding H_j^2 as a double sum and counting the $L - \max(q,r)$ occurrences gives the second. Expanding $g_L(i)^2$ with these identities proves

$$\langle g_L, 1\rangle_L = 0, \qquad \|g_L\|_L^2 = 1 - \frac{H_L}{L}, \qquad g_L(i+1) - g_L(i) = -\frac{1}{L-i}. \tag{24}$$

Under the uniform conditional law, P_E replaces the two values in each pair by their average. Its tagged-leaf profile is consequently represented by K block values $x_h = (g_L(2h-1) + g_L(2h))/2$. Give block profiles the inner product $\langle u, v\rangle_K = K^{-1}\sum_h u_h v_h$. Their norm equals the corresponding repeated position-profile norm. On these block profiles, the operator $\mathsf{T} = P_E P_O P_E$ has the explicit form

$$\begin{aligned} (\mathsf{T}x)_1 &= (3x_1 + x_2)/4, \\ (\mathsf{T}x)_h &= (x_{h-1} + 2x_h + x_{h+1})/4 \quad (1 < h < K), \\ (\mathsf{T}x)_K &= (x_{K-1} + 3x_K)/4. \end{aligned} \tag{25}$$

Indeed, P_O averages positions $2h$ and $2h+1$, and leaves positions $1, L$ unchanged; applying P_E then averages each original pair. This calculation also covers $K = 2$, where only the endpoint equations occur.

The cosine vectors and eigenvalues of (25) are

$$\phi_r(h) = \cos\Big((h - \tfrac{1}{2})\frac{r\pi}{K}\Big), \qquad \mu_r = \frac{1 + \cos(r\pi/K)}{2} = \cos^2(r\pi/L), \quad 0 \le r < K. \tag{26}$$

The interior equation follows from the cosine addition formula, and reflection at the two endpoints verifies the endpoint equations. The vectors are orthogonal by summing finite geometric series; for $r \ge 1$, their mean is zero and their squared norm is $1/2$. They give a basis of the K block values. This is the familiar one-dimensional averaging spectrum underlying adjacent-update analyses (Diaconis & Ram, 2000; Wilson, 2004). What is needed here is that the native score has substantial overlap with its slowest nonconstant mode.

Write $a_L = \langle x, \phi_1\rangle_K$. Both x_h and $\phi_1(h)$ are strictly decreasing, and their means are zero. The identity

$$a_L = \frac{1}{2K^2}\sum_{h,j=1}^{K} (x_h - x_j)(\phi_1(h) - \phi_1(j)) > 0 \tag{27}$$

shows a nonzero slow component at every even $L \ge 4$. Expanding $x = \sum_{r=1}^{K-1} \alpha_r \phi_r$, with $\alpha_r = 2\langle x, \phi_r\rangle_K$, gives for the residual at odd depth $2k+1$

$$\|f_{2k+1,\mathrm{tag}} - \mathbb{E}[s_{\mathrm{tag}} \mid S]\|_{L^2(Y|S)}^2 = \|\mathsf{T}^k x\|_K^2 = \frac{1}{2}\sum_{r=1}^{K-1} \alpha_r^2 \mu_r^{2k}. \tag{28}$$

In particular, with $\mu_1 = \cos^2(\pi/L)$,

$$2a_L^2 \mu_1^{2k} \le \|\mathsf{T}^k x\|_K^2 \le \|x\|_K^2 \mu_1^{2k}. \tag{29}$$

Here μ_1 is an eigenvalue of $P_E P_O P_E$ on this invariant subspace. It is not being identified with the one-step norm of $P_O P_E$, whose corresponding singular factor is $\sqrt{\mu_1}$. The variance in (29) decays with μ_1^{2k}.

To establish a dimension-independent amount of initial slow-mode energy, view g_L as a step function on $[0,1]$, constant on $((i-1)/L, i/L)$. For every $x \in (0,1)$ it tends to $g(x) = 1 + \log(1-x)$. Equation (24) gives $\|g_L\|_{L^2}^2 \to 1 = \|g\|_{L^2}^2$. The sequence is bounded in L^2; together with convergence in measure this implies weak L^2 convergence. Explicitly, L^2 boundedness gives uniform integrability in L^1, so testing against bounded functions converges, and truncation extends this to all L^2 test functions. Equality of the limiting norms then gives strong L^2 convergence. Averaging adjacent cells is an L^2 contraction whose mesh tends to zero, so the repeated profile x also converges strongly to g.

The piecewise constant profile ϕ_1 converges uniformly to $\cos(\pi x)$. Hence integration by parts gives

$$a_L \longrightarrow a = \int_0^1 (1 + \log(1-x)) \cos(\pi x)\, dx = \frac{1}{\pi} \int_0^\pi \frac{\sin t}{t}\, dt > 0. \tag{30}$$

The boundary term vanishes because $(1 + \log(1-x)) \sin(\pi x) \to 0$ as $x \to 1$. Thus $c_L = 2a_L^2 / \|g_L\|_L^2$ tends to

$$c_* = 2a^2 = 2 \left[\frac{1}{\pi} \int_0^\pi \frac{\sin t}{t}\, dt \right]^2 > 0. \tag{31}$$

Strict Cauchy–Schwarz also gives $c_* < 1$, since g is not a multiple of the cosine and $\|\cos(\pi x)\|_{L^2}^2 = 1/2$.

Fix $0 < \eta < c_*$. For sufficiently large even L, the lower bound in (29) implies that reaching the fraction η of the raw tagged-leaf variance requires

$$k \geq \frac{\log(c_L/\eta)}{-2 \log \mu_1}. \tag{32}$$

Conversely, $\|x\|_K \leq \|g_L\|_L$ makes $k \geq \log(1/\eta)/(-2 \log \mu_1)$ sufficient. Since $-\log \cos^2(\pi/L) = \Theta(L^{-2})$, the required number of odd-depth cycles is $\Theta(L^2)$ for fixed η. Every even-depth norm lies between its neighboring odd-depth norms by conditional-expectation contraction, so counting individual local layers changes only constants. This proves the depth assertion for the actual score rather than an arbitrary test function.

It remains to compare cyclic conditioning. At the unit-rate point, all orders in the leaf terminal have equal probability, including all orders in each cyclic record. A tagged leaf's block visits all K block positions under rotation, and its two orientations place that leaf in either position of each block equally often. Its conditional average under the cyclic record is therefore its mean over all L positions. This equals the terminal score in (21); the center coordinate is already constant. On the other terminal, the center alone, the record is an identity. Cyclic conditioning thus gives $\mathbb{E}[s \mid S]$ for the complete score on this graph, even though the cyclic record need not contain every permutation of the leaves.

Finally, let h be any centered position profile and randomly assign its entries to the L leaf labels. For distinct leaves u, v,

$$\mathbb{E}[h_u^2] = \|h\|_L^2, \qquad \mathbb{E}[h_u h_v] = -\frac{\|h\|_L^2}{L-1}. \tag{33}$$

The second identity follows because the sum of all assigned entries is zero. Let P_{leaf} be the projector onto the zero-sum leaf coordinates, extended by zero at the center. Then the full conditional residual covariance and its actor image are

$$\mathrm{Cov}(h \mid S) = \frac{L}{L-1} \|h\|_L^2 P_{\mathrm{leaf}}, \qquad \mathrm{Cov}(B_\theta^\mathsf{T} h \mid S) = \frac{L}{L-1} \|h\|_L^2 B_\theta^\mathsf{T} P_{\mathrm{leaf}} B_\theta. \tag{34}$$

Every direction u with $P_{\mathrm{leaf}} B_\theta u \neq 0$ therefore has the same relative residual reduction as the tagged profile. If all leaf rows of B_θ coincide, this entire contrast vanishes. This proves the actor qualification in Theorem 3.2; the theorem does not assert that every network or every feature assignment realizes the slow mode.

For clarity, this conditional comparison is not a comparison of total batch variance with a zero oracle. With unit task weights, the two terminal rewards are $L/(L+1)$ and $1/(L+1)$. The center terminal contributes no order residual, but the terminal score itself remains random across the two terminal sets. More generally, if Σ_h denotes the conditional covariance in (34), the extra covariance above terminal conditioning is $\Pr(S)(R_G(S)-\beta)^2\Sigma_h/N$ for an N-draw likelihood-ratio average. For RLOO it is

$$\frac{\Pr(S)}{N}\left[(R_G(S)-J_G)^2+\frac{\mathrm{Var}(R_G(S(Y)))}{N-1}\right]\Sigma_h. \tag{35}$$

To obtain this expression, condition on all terminal sets. The order residuals are independent, centered, and multiplied by measurable RLOO coefficients. Given the ith terminal S, its squared coefficient has expectation $(R_G(S)-J_G)^2+\mathrm{Var}(R_G(S(Y)))/(N-1)$. The terminal component is orthogonal to these residuals. This proves the formula and specifies precisely which part of the covariance the local-depth result controls.

B.4 Certified score approximation and bit complexity

We now turn the exact score calculation into an approximation with a coordinate-wise error certificate. The construction first represents graph masses as integers, then bounds the error in the probability-weighted derivative coefficients. An integer transpose propagates those coefficients back to the original vertices without further accumulation error. Accounting for the final format conversion gives the score certificate below; we then bound the bit operations needed to obtain it.

The certified evaluator takes positive represented dyadic inputs. Choose a common unit $c>0$ with $\lambda_v=cM_v$, M_v positive integers, and set $W=\sum_v M_v$ and $b=\mathrm{bitlength}(W)$. The representation is only a way to evaluate the law at the supplied rate vector: differentiation holds this common unit fixed and treats all original rate coordinates as independent. It does not differentiate a greatest-common-divisor operation or identify equal masses. Constructing and checking this representation also incurs input-processing cost.

Let F be the number of distinct fibres after any reflection deduplication. In integer mass units, each pair factor is $(U+V)/(ZUV)$ and each singleton factor is $1/Z$. Remove the common selected numerator and let w_r be the product of the K factors, so $\omega_r=w_r/\sum_j w_j$. There are $T=3FK$ positive pair-risk seeds or $T=FK$ singleton-risk seeds; $T_0=3FK$ is a uniform upper bound. For a seed slot a associated with fibre r, use $\alpha_a=\omega_r\beta_a$ with (20), or $\beta_a=1/Z$ for a singleton. Every integer risk is between 1 and W, so $0<\alpha_a\le 1$.

Let R_{av} indicate that original vertex v belongs to the active subset defining risk a. In particular, the subsets for U and V are different when their deletion neighborhoods differ, even if the two masses coincide. Differentiating the full mixture gives the exact identity

$$\bar{s}_v=\mathbf{1}_{\{v\in S\}}-M_v\sum_{a=1}^{T}\mathsf{R}_{av}\alpha_a,\qquad \mathsf{R}_{av}\in\{0,1\}. \tag{36}$$

The matrix R is not materialized. The rectangle construction realizes its transpose through the signed seeds $-(\alpha_Z+\alpha_U+\alpha_V)$ for Z, α_U for M^a, and α_V for M^c. Positivity is used before this signed representation, not assumed for its internal cells.

The positive seeds can be rounded before the shared transpose. Let $\delta=2^{-t}$ for a nonnegative integer t, and approximate each seed α_a by a nonnegative integer multiple $q_a\delta$ with absolute error at most δ. Evaluate the resulting signed transpose with integer additions and subtractions, and assemble the selected indicator before converting the output format.

Proposition B.2 (Certified complete-coordinate approximation)**.** For a nonempty accepted order, keeping all fibres, this construction returns a dyadic vector $\widetilde{s}$ satisfying

$$\|\widetilde{s}-\bar{s}\|_\infty\le WT_0\delta. \tag{37}$$

For any rational $\varepsilon>0$, choosing $\delta\le\min(\varepsilon,1)/(4WT_0)$ and the interval precision specified below suffices. A subsequent output-format conversion is accepted only when its exact conversion error plus this bound is at most ε. The empty record returns its exact empty score.

Here is a sufficient finite precision, with explicit roundings. Put $\varepsilon_0 = \min(\varepsilon, 1)$ and choose the smallest nonnegative integer t satisfying

$$\delta = 2^{-t} \le \frac{\varepsilon_0}{4T_0W}, \quad e_0 = 2K-1, \quad e_{\mathrm{tot}} = 2K+F-2, \quad h_{\mathrm{rnd}} = e_0+e_{\mathrm{tot}}+3 = 4K+F. \tag{38}$$

Use $p = t + \lceil \log_2(64h_{\mathrm{rnd}}) \rceil$ significant bits and an independently stored unbounded exponent. Positive directed dyadic rounding can then be chosen to satisfy, with $u = 2^{1-p} \le \delta/(32h_{\mathrm{rnd}})$,

$$(1-u)x \le \mathrm{down}_p(x) \le x \le \mathrm{up}_p(x) \le (1+u)x, \qquad \mathrm{down}_p(x) > 0 \quad (x > 0). \tag{39}$$

These are ordinary binary rounding bounds with a conservative unit roundoff; an unbounded exponent precludes underflow of a positive enclosure. Ceiling decisions can be made by integer bit lengths and comparisons. No transcendental routine is needed.

Form each factor's integer numerator and denominator exactly and round that rational once in each direction. Thus $(U+V)/(ZUV)$ receives one rounding per bound, not separate approximate divisions and additions. There are K factor roundings and $K-1$ product roundings in each fibre, giving positive bounds

$$(1-u)^{e_0} w_r \le \underline{w}_r \le w_r \le \overline{w}_r \le (1+u)^{e_0} w_r. \tag{40}$$

Sequentially summing the F lower and upper bounds adds at most $F-1$ rounding factors to each path. For $W_f = \sum_r w_r$, the resulting total interval obeys

$$(1-u)^{e_{\mathrm{tot}}} W_f \le \underline{W}_f \le W_f \le \overline{W}_f \le (1+u)^{e_{\mathrm{tot}}} W_f. \tag{41}$$

Positive sums preserve the common multiplicative bounds, which proves this assertion by induction without a conditioning assumption on the relative fibre sizes.

Enclose each posterior by rounding $\underline{w}_r/\overline{W}_f$ down and $\overline{w}_r/\underline{W}_f$ up. Round the exact rational β_a outward once, then multiply the two intervals with one further outward rounding. The seed enclosure $[\ell_a, u_a]$ satisfies

$$(1-u)^{e_0+3}(1+u)^{-e_{\mathrm{tot}}} \alpha_a \le \ell_a \le \alpha_a \le u_a \le (1+u)^{e_0+3}(1-u)^{-e_{\mathrm{tot}}} \alpha_a. \tag{42}$$

For $u \le 1/2$, $\log(1+u) \le u$ and $\log(1-u) \ge -2u$. Hence the endpoints in (42) lie between $e^{-2h_{\mathrm{rnd}}u}\alpha_a$ and $e^{2h_{\mathrm{rnd}}u}\alpha_a$. Since $\alpha_a \le 1$ and $2h_{\mathrm{rnd}}u \le \delta/16 \le 1$, the elementary inequality $e^x - e^{-x} \le 4x$ for $0 \le x \le 1$ gives

$$u_a - \ell_a \le 8h_{\mathrm{rnd}}u \le \delta/4. \tag{43}$$

These exponential expressions prove a bound; the algorithm only uses integer and directed rational operations.

Take the exact midpoint of this dyadic interval and round it to the nearest multiple $q_a\delta$, using any fixed tie rule. Then q_a is nonnegative and

$$|q_a\delta - \alpha_a| \le \frac{u_a - \ell_a}{2} + \frac{\delta}{2} \le \frac{5\delta}{8} < \delta. \tag{44}$$

A certificate may save the endpoints and the directly checkable bound $\max(|q_a\delta - \ell_a|, |q_a\delta - u_a|)$. All fibres retain positive interval bounds throughout; $q_a = 0$ is permitted as an absolute approximation to a seed, not a claim that its exact posterior is zero.

Form integer seeds $-(q_Z + q_U + q_V), q_U, q_V$ and evaluate the entire shared transpose with exact integer additions and subtractions. By linearity its output d_v^{int} is $-\sum_a \mathsf{R}_{av} q_a$, independent of the signed internal representation. Assemble the selected indicator and correction before any format conversion:

$$\widetilde{s}_v = \frac{\mathbf{1}_{\{v \in S\}} 2^t + M_v d_v^{\mathrm{int}}}{2^t}. \tag{45}$$

Equations (36) and (44) immediately imply

$$|\widetilde{s}_v - \bar{s}_v| \le M_v \sum_a \mathsf{R}_{av} |q_a\delta - \alpha_a| \le WT_0\delta \le \varepsilon_0/4. \tag{46}$$

Thus intermediate cancellation in the rectangle transpose introduces no numerical error. Rounding the final dyadic to a grid with spacing at most $\varepsilon_0/2$ adds at most $\varepsilon_0/4$. More generally, for any

requested format compute its exact conversion error η_v and require $WT_0\delta + \max_v \eta_v \le \varepsilon$. This proves Proposition B.2. The empty sequence is handled separately; for $L = 1$, $F = K = 1$ and the singleton calculation applies without declaring its score zero.

The exact score satisfies $\mathbf{1}^\mathsf{T}\bar{s} = 0$, because multiplying every rate by the same constant leaves all sequence probabilities unchanged. For $n > 0$, let $P = I - \mathbf{1}\mathbf{1}^\mathsf{T}/n$. Then $P\bar{s} = \bar{s}$ and $\|P(\widetilde{s}-\bar{s})\|_\infty \le 2\|\widetilde{s} - \bar{s}\|_\infty$. The actor error after this projection is bounded directly by $\|B_\theta^\mathsf{T} P\|_{\infty\to 2}\|\widetilde{s} - \bar{s}\|_\infty$. For a matched batch of projected scores with raw coordinate errors at most ε, the gradient error is at most $N^{-1}\sum_i |c_i|\,\varepsilon\|B_\theta^\mathsf{T} P\|_{\infty\to 2}$. With normalized rewards in $[0, 1]$, RLOO has $|c_i| \le 1$. These are pointwise error bounds, not unbiasedness claims for rounded scores.

We conclude with a conservative bit-cost account, since the arithmetic-operation bound of Theorem 3.1 does not by itself imply a machine-time bound. Put $Q = 1 + n + m + K^2$. The mass difference and prefix tables use $O(b + \log Q)$ bits: each retained cell is a sum of polynomially many signed input masses in this explicit computation. As $q_a \le 2^t + 1$, the seed tables and vertex cotangents use $O(t + \log Q)$ bits, and the complete score numerator in (45) uses

$$B_{\mathrm{score}} = O(t + b + \log Q). \tag{47}$$

For instance, a summed-area table sums $O(Q)$ initial seeds, each rectangle query uses four entries, and each vertex accumulates at most $O(Q)$ such queries. Even this conservative accounting multiplies magnitudes only polynomially in Q.

Pair factors are between $2/W^2$ and 2, and singleton factors are between $1/W$ and 1. Products over K blocks and sums over F fibres therefore have dyadic exponent magnitudes $O(Kb + \log(F + 1))$. A conservative interval implementation may align exact endpoint mantissas before rounding, with working size

$$T_{\mathrm{bits}} = O(p + Kb + \log Q) = O(Kb + \log Q + \log(1/\varepsilon_0)). \tag{48}$$

Exact factor numerators and denominators use only $O(b)$ bits and fit this bound. If Mult and Div bound integer multiplication and division, a sufficient total bit-operation bound is

$$O(Q[\mathrm{Mult}(T_{\mathrm{bits}}) + \mathrm{Div}(T_{\mathrm{bits}})] + n\,\mathrm{Mult}(B_{\mathrm{score}}))\,, \tag{49}$$

in addition to input construction, graph-label processing, and reading the requested tolerance and output format. Retaining all tables and certificates has a conservative $O(QT_{\mathrm{bits}})$ bit bound; actual serialization remains paid work. This algorithm approximates the full conditional score with a certified error. It does not assert exact machine unbiasedness, a positive-semidefinite covariance ordering for its rounded output, or that an observation-dependent fallback to the raw score preserves unbiasedness. A generic evaluator is entitled to the same positive seeds and integer transpose.

B.5 Residual variation and observable total covariance

Section 3.3 defines the residual fraction in Eq. 11. To see the decomposition, write $s - s_S = (s - \bar{s}) + (\bar{s} - s_S)$. The second term is measurable from $\mathcal{C}$, whereas $\mathbb{E}[s - \bar{s} \mid \mathcal{C}] = 0$, so their expected inner product is zero. Applying a fixed linear map B_θ^T gives the same orthogonality in parameter-score space. The result concerns the original policy distribution; changing the terminal or parity distribution changes the estimand.

A total score-covariance measurement also has a one-sided interpretation without a terminal oracle. Put $\mathcal{I} = \mathrm{tr}\,\mathrm{Cov}(s_S)$. Total covariance gives

$$T_{\mathrm{raw}} = \mathrm{tr}\,\mathrm{Cov}(s) = \mathcal{V}_{\mathrm{raw}} + \mathcal{I}, \qquad T_{\mathrm{SOM}} = \mathrm{tr}\,\mathrm{Cov}(\bar{s}) = \mathcal{V}_{\mathrm{SOM}} + \mathcal{I}. \tag{50}$$

For $\mathcal{V}_{\mathrm{raw}} > 0$, nonnegativity of $\mathcal{I}$ implies $1 - T_{\mathrm{SOM}}/T_{\mathrm{raw}} \le 1 - \rho$. Thus total score-variance reduction lower-bounds the fraction of within-terminal variation removed, but does not identify the residual fraction because the two totals do not determine $\mathcal{I}$. The identity also holds after a fixed actor Jacobian; it does not directly apply to a reported reward-weighted RLOO covariance ratio. Empirical use additionally requires sampling uncertainty and numerical-error control. Appendix D.9 specifies the small-graph enumeration and task-scale diagnostics.

C Implementation Details

We implement the update in (12) by supplying a fixed vector at the network's log-rate output, then backpropagating an ordinary scalar loss. For $N \ge 2$ independent trajectories, let β_i be the mean

reward of the other $N-1$ draws and let $\bar{s}_{\mathcal{C}}(Y_i) = \bar{s}(Y_i)$ denote the conditional score in (9). Define

$$u = \frac{1}{N}\sum_{i=1}^{N}(R_G(S_i) - \beta_i)\bar{s}_{\mathcal{C}}(Y_i), \qquad \mathcal{L}_{\text{proxy}} = -\langle \text{stopgrad}(Pu), z_\theta(G)\rangle. \tag{51}$$

Here $P = I - \mathbf{1}\mathbf{1}^{\mathsf{T}}/n$ centers the cotangent, matching the convention used by every estimator. Exact scores sum to zero, so this projection leaves them unchanged. The gradient of this scalar loss is $-B_\theta^\top Pu$, so minimizing it performs an ascent step on the expected reward. The score computation does not create an additional neural-network backward pass. This property is shared by any estimator that supplies a cotangent at the same output, including the sampled-order and generic shared-computation baselines. SOM changes the cotangent, while the model and optimizer remain fixed.

Our implementation batches graphs by size and stores their adjacency lists in a packed representation. Prefix and interval accumulations build the shared mass tables on the GPU; reverse accumulations recover the original-vertex scores. Different graphs and sampled trajectories are processed concurrently. Within a trajectory, the factor calculations use the dependencies specified by the shared table. The arithmetic bound in Theorem 3.1 concerns total work and does not assert constant parallel depth. We separately measure table construction, score evaluation, actor forward and backward passes, communication, and peak memory to determine whether reduced training variance compensates for the added estimator work.

Probabilities are accumulated in the log domain where possible, and normalization denominators are checked before use. Higher-precision reference calculations validate the GPU score on small instances. Production runs use accumulation precision FP64 and an original-coordinate score tolerance of 1.0×10^{-6} before centering. Centering can increase the coordinate error bound by at most a factor of two. Appendix E.11 separates high-precision reference audits from the certified integer/interval evaluator. The production FP64 trace does not inherit a deterministic error certificate from agreement on a finite audit panel. The exact-arithmetic estimator is unbiased. Floating-point agreement and any certified approximation error are reported separately, rather than inferred from unbiasedness in exact arithmetic.

For distributed training, the estimator is attached to the actor's output and its scalar proxy loss. We implement this interface in PyTorch distributed data parallel, FSDP2, and Megatron-Core (Li et al., 2020; Zhao et al., 2023; PyTorch Contributors, 2026; Narayanan et al., 2021; NVIDIA, 2026). Each data-parallel replica retains complete leave-one-out groups, using independent trajectories from one graph. Tensor-parallel output partitions are gathered, or equivalently reduced under the same score definition, before constructing the graph-level cotangent. Pipeline parallelism evaluates the proxy loss at the output stage. Graph and microbatch normalization are applied once, so the distributed gradient matches the single-device objective. These changes leave the Graph Transformer's attention computation and the optimizer unchanged. The experiments compare estimators inside each identical parallel configuration, and count communication, score computation, and all participating devices in the training budget.

D Experimental Details

This appendix specifies the data, policy, estimator, and resource conventions used in the experiments. The same conventions applied to SOM and its matched baselines. Configuration identifiers and per-instance results accompany the aggregate figures, including incomplete runs and instances for which no method reaches the common quality target.

D.1 Datasets, Splits, and Evaluation Units

The weighted study used MWIS-VR route-conflict graphs with the supplied objective weights (Dong et al., 2021). We grouped each parent graph with its derived instances, alternate weightings, and duplicate topology before assigning groups to training, validation, and test. The split contained 24, 6, and 7 parents, respectively. Source identifiers, vertex maps, topology hashes, and weight hashes were retained. Parent assignment preceded subgraph extraction and method evaluation; changing a vertex labeling did not create a new statistical unit.

We extracted 4096 connected induced training patches with sizes 128–512. A fixed seeded frontier procedure selected vertices using only graph structure, starting within an eligible connected component. It did not condition on solution quality, accepted sequence length, or the behavior of any estimator. Induced edges and original vertex weights were retained. Repeated vertex sets within a split were recorded rather than resampled. Validation used 6 graphs from the disjoint validation parents. The principal weighted test used the complete held-out graphs, spanning 500–48,000 vertices and 4,000–1,200,000 edges; patch-level results were reported separately.

The unweighted evaluation used RB200–300, RB800–1200, and ER700–800 with the source protocol's generation settings and split identifiers (Acikalin et al., 2025). Each vertex had unit weight, so feasible weight equaled independent-set cardinality. Training, validation, and test instance counts were 4,096 / 256 / 512 per suite. We trained task-specific checkpoints and used validation data from the corresponding task for hyperparameter and checkpoint selection. Generator seeds and shared parent identities, where applicable, were disjoint across splits.

Graph inputs included the objective weight, degree-derived features, and the selected positional and structural encodings. Any normalization statistics shared across graphs were fitted on the training parents only. Features computed from an individual test graph's own legal input remained available at inference, with their preparation time charged. No test solution, reference value, or future checkpoint performance entered the feature pipeline. Tab. 1 records the principal data and training settings.

D.2 Graph Encoders and Training Configuration

GraphGPS combined its local graph-processing branch with global attention and the selected structural encodings (Rampásek et al., 2022). Graphormer used its structural attention biases with a vertex-level output head (Ying et al., 2021); GatedGCN used the corresponding node representations from gated message passing (Bresson & Laurent, 2017). We trained the encoders and heads jointly. In the fixed-rate estimator panels, a head produced log rates $z_\theta(G)$, and positive rates were held fixed during every sampled construction from that forward pass. Removing vertices changed the sampler's active set, not the neural hidden state. Every independent-sample and conditional-average baseline received this same forward reuse.

We used small, medium, and large GraphGPS configurations with 1.2M, 9.6M, and 76M trainable parameters. The medium configuration was the primary model. Graphormer and GatedGCN had 10M and 2M parameters. Architecture changes were analyzed separately from estimator changes: within an architecture-and-size panel, the initialization, features, positional encodings, head, optimizer schedule, and inference procedure were matched. Numerical settings for attention, rate conversion, and conditional averaging were recorded separately.

Table 1: Data and training configuration. The two panels share row heights; model settings refer to the primary GraphGPS comparison.

Data / evaluation	**Setting**	**Model / training**	**Setting**
MWIS-VR split	VR-target-v3	Depth / width	12 / 256
Patch sizes	128–512	Heads / encoding	8 / RWSE-16
MIS splits	RB-ER-target-v3	Optimizer / LR	AdamW / 1.0×10^{-4}
MWIS test time (s)	60	Decay / clipping	0.01 / 1
MIS test time (s)	30	Schedule	5% / cosine
Decode samples	64	Batch / microbatch	32 / 4
Search	MWIS: CHILS; MIS: native	Samples / multiplier	8 / 2
CPU allocation	1 CPU thread / instance	Local depth / precision	3 / BF16
Validation interval	10,000 updates	Seeds / tuning trials	5 / 8

Rates used FP64 rates, with requested score error 1.0×10^{-6}.

Paired runs began from the same encoder and optimizer state and followed the same graph order. Random streams were separated by initialization, data ordering, graph, update, sample role, and

estimator. Native and conditional estimators could share their base observations where their sampling laws agreed; additional observations used independent streams. Attribution comparisons used a fixed matched configuration. The competitive-quality comparison also gave each training method the same tuning allowance, with all choices made on validation data.

The reward was the feasible objective weight divided by the sum of positive input weights of that graph. It therefore depended only on the returned independent set and remained common to every order in a conditional record. Reward evaluation required only a weight sum. The same reward normalization was used by all matched estimators. Reported solution quality used the original, unnormalized task weights.

For a group of N solutions, RLOO used coefficients $c_i = R_G(S_i) - (N-1)^{-1} \sum_{j \neq i} R_G(S_j)$. We formed the rate cotangent $u = N^{-1} \sum_i c_i s_i$, replacing s_i by its conditional average only in the corresponding estimator. The whole group remained intact when assigned to a data-parallel rank. REINFORCE instead used a baseline estimated from an independent pilot, whose acquisition and processing costs were charged. Additional-sample controls increased the number of independent solutions and used the same leave-one-out definition for their complete group. Same-solution replay retained its auxiliary imitation objective, its own validation-selected coefficient 0.10, and its full computation cost (Kim et al., 2024).

The network backward used a detached cotangent and the proxy scalar $-\langle \mathrm{stopgrad}(Pu), z_\theta(G) \rangle$, where $P = I - \mathbf{1}\mathbf{1}^\mathsf{T}/n$ was the common centering convention. Exact scores already obeyed the common-rate scaling identity; the explicit projection also made the treatment of approximate scores identical across implementations. We used the framework's ordinary backward and minimizing optimizer step, without an additional sign reversal or differentiation through the posterior weights. Batch and microbatch factors were applied once, and accumulated gradients were compared with a single-process reference before distributed training.

The common validation-quality targets were 5.0% validation gap, selected by the fixed rule fixed on validation pilot before test evaluation. We evaluated checkpoints at the common interval in Tab. 1; the first evaluated checkpoint reaching a target defined attainment. Its elapsed time included the validation needed to observe that event. The final fixed-budget checkpoint rule was last committed update before 24 GPU-h. Targets were never chosen from the test curves or changed to remove nonattaining seeds.

D.3 Solver Comparisons and Quality Metrics

CHILS retained its native weighted search (Großmann et al., 2025). The weighted heuristic repeatedly selected an active vertex according to weight divided by one plus active degree, with a fixed tie rule. X^2GNN and pCQO-MIS retained their published unit-weight task definitions and documented decoding procedures (Acikalin et al., 2025; Alkhouri et al., 2025); KaMIS and degree-greedy supplied classical MIS references. Versions, compiler settings, threads, and hardware were fixed per comparison. Published measurements from different hardware were not inserted into the same-machine time ranking. Methods with no offline training were marked as such rather than assigned an artificial neural training budget.

All controlled neural comparisons used the same inference sampling allowance or the same inference wall limit in separately labeled panels. We reported both the policy output before local search and the feasible incumbent after the common local-search procedure. The former exposed differences learned by the policy; the latter measured the quality delivered by the complete matched workflow. External solvers retained their native search components and paid for their complete online processing.

For graph G, let W_G be the returned feasible weight and $W_G^{\mathrm{ref}} > 0$ the common reference. We reported $100(W_G^{\mathrm{ref}} - W_G)/W_G^{\mathrm{ref}}$ as a percentage reference gap. An uncertified reference was explicitly best-known rather than optimal, and a negative gap was retained if a method improved it. Every method used the same reference on the same graph. The manifest recorded its feasible vertex set, objective, provenance, version, and certification status. Independent feasibility and weight checks preceded aggregation. Relative gap reduction was $100(g_{\mathrm{baseline}} - g_{\mathrm{SOM}})/g_{\mathrm{baseline}}$ for the common aggregate gap g with a positive baseline denominator; absolute differences between percentage-valued metrics are reported in percentage points and use subtraction on the 0–100 scale.

In particular, normalized solution value is $100W_G/W_G^{\mathrm{ref}}$, so its increase equals the decrease in reference gap. Runtime and variance reductions instead divide by the baseline value.

D.4 Gradient and Mechanism Measurements

We measured gradients at checkpoints indexed by training budget, using a fixed graph panel and independent evaluation randomness. Each replicate was a complete estimator batch, including its pilot or leave-one-out coefficients. We computed the full gradient with respect to every trainable encoder and head parameter; the covariance trace was estimated from centered gradient norms. Original log-rate score covariance was a separate diagnostic. We did not replace parameter covariance by score covariance or count coordinates as independent statistical samples.

On small graphs, an independent complete-law calculation supplied the mean gradient. On the larger panel, an independent native estimator with 4096 batches supplied a reference and its sampling uncertainty. Conditional and native methods used the same objective, frozen parameters, and inference input. The mean comparison uses relative-L2 discrepancy and a reference confidence radius in the same normalized space, while the requested numerical error and final format conversion were tracked separately. The covariance evaluation used 512 independent batches per setting. Checkpoint and graph sampling rules were fixed before examining estimator differences.

The mechanism panel used unit-rate stars with the specified even leaf counts 8, 16, 32, 64, conditioning on the leaf terminal. We measured the residual original-coordinate covariance after the registered local averaging depths and after cyclic averaging. A cheap exact terminal calculation served as the reference on this family, so the experiment did not present a star-specific computational victory. The observed local decay ratio was 0.87, and the cyclic residual relative to numerical tolerance was 7.0×10^{-10}. A separate nonsymmetric rate-and-feature panel measured whether the effect survived the actual encoder Jacobian; its parameter trace ratio was 0.73.

Range ablations compared native sampling, additional samples, one fixed-pair average, alternating pairs, and cyclic averaging while retaining the encoder, reward, and training objective. Computation ablations instead fixed the same cyclic record and changed only its evaluation. This separation prevented an observed reduction in time from being explained by averaging fewer orders, and prevented a learning difference from being attributed to a faster implementation of a different estimator.

For the frozen-checkpoint comparison, the parameter-gradient trace ratio is 0.677, and the original log-rate score ratio is 0.647. The relative-L2 mean discrepancy is 0.006, with reference radius 0.013.

D.5 Shared Computation and Numerical Accuracy

The separate-start implementation rebuilt the paired calculation for each cyclic start. The shared generic implementation constructed one differentiable graph over the same shared risk masses, with all original rates represented as inputs. Its reverse preserved equal-valued but distinct paths and accumulated every coordinate jointly. The specialized implementation used the corresponding structured transpose. All three implementations included preparation, posterior evaluation, differentiation, device transfers, and required accuracy checks; none paid for another implementation's reverse before producing its own result.

The numerical audit retained positive branch contributions and checked the requested original-coordinate error 1.0×10^{-6} before centering. Applying P can increase this coordinate bound by at most a factor of two. The audit refinement and the distinct certified integer/interval path are specified in Appendix E.11; agreement on the audit panel is not a per-call certificate for ordinary FP64 training. Output-format conversion was included in that error check. The final score error against the independent reference was 2.8×10^{-7}, and the matched generic-to-specialized discrepancy was 1.6×10^{-7}. These measurements supported the implemented accuracy contract, not a claim that a finite-precision deterministic estimate was exactly unbiased. Unresolved audit records remain failures and do not trigger an unreported raw-score fallback or resampling.

Timing used synchronized end-to-end boundaries and separately recorded encoder forward, sampling, conditional averaging, parameter backward, optimizer, communication, validation, and checkpoint work. Nested stages were not added to the already measured wall time. Setup and compilation were included in cold-run totals; steady-state diagnostics used the same stated warmup 20 warmup

iterations for every implementation and did not replace the cold training comparison. The numerical audit contained 16 graphs and 8 sampled orders per graph. The separate timing microbenchmark uses batch 32 and eight draws at $n = 256, m = 2{,}048, L = 64$ (Appendix E.9); neither is a complete-parent inference measurement.

D.6 Distributed Implementation and Scaling

All GPU experiments used a server with eight NVIDIA H20 devices, host configuration 2 x 64-core x86 CPUs; 1 TiB RAM, and interconnect NVLink intra-node. The software versions were PyTorch 2.6.0, PyG 2.6.1, CUDA 12.4, NCCL 2.21.5, and Megatron-Core 0.12.0. Graph processing, encoder computation, and the conditional-average implementation were the same mathematical operations across parallel configurations; communication and placement changed with the framework.

DDP was the principal configuration (Li et al., 2020). Its backward accumulated parameter gradients through the standard framework path and synchronized them before the optimizer step. Each data-parallel replica owned complete per-graph sampling groups, preserving RLOO coefficients. Unequal graph counts in the last microbatch were weighted by their actual contribution, not by a uniform average of rank-local means. Strong scaling on one, two, four, and eight GPUs held the global graph batch, sampling groups, model, and precision fixed; gradient accumulation supplied the required local batch. Weak scaling separately held the per-device graph workload fixed. Strong-scaling efficiency was throughput on D devices divided by D times single-device throughput; weak-scaling throughput was not substituted into that definition.

FSDP2 sharded model, gradient, and optimizer state in the larger-model panel (Zhao et al., 2023). Its sharding groups and activation-checkpointing policy were full shard and per-block. We retained a common-fit comparison for throughput, and separately reported the expanded configuration that required sharding, with 150M parameters. The peak-memory reduction relative to the matching unsharded model was 38.525%.

The Megatron-Core port partitioned Transformer computation using tensor and pipeline parallelism (Narayanan et al., 2021). Tensor, pipeline, and data-parallel degrees were 2, 2, and 2, with product equal to the allocated device count; its microbatch and schedule were 2 and 1F1B. GraphGPS's local branch and graph-dependent encodings were retained and explicitly placed at their corresponding layers. Vertex rates were gathered or distributed according to whole graphs per DP group; the resulting sampling and averaging work, its waiting time, and its communication were charged. The detached proxy loss entered the final pipeline stage, with framework microbatch scaling applied exactly once.

We checked full parameter gradients across the single-process, DDP, FSDP2, and Megatron implementations on matched inputs. The maximum relative discrepancy was 2.1×10^{-6} under tolerance 1.0×10^{-5}. On the common-fit large GraphGPS configuration, Megatron improved the native estimator's throughput over DDP by 1.375 and SOM's throughput by 1.36. The within-Megatron SOM-to-RLOO target GPU-hour ratio was 0.875, whose reciprocal gave the RLOO-to-SOM time speedup of 1.14 at the same device allocation. This paired reporting distinguished infrastructure acceleration from estimator-specific learning improvement.

The graph-scale panel varied vertex count, edge count, and accepted length at a fixed encoder size; the encoder-scale panel varied model parameters on a fixed graph workload. The largest complete weighted test graph had 48000 vertices and 1200000 edges. At that scale, the single cyclic-table allocation and the averaging-time fraction of complete-step latency were 0.208 GiB and 0.360, respectively. These measurements exposed irregular graph processing and the cyclic table separately from attention and model-state memory.

D.7 Statistics, Resource Accounting, and Failure Handling

Weighted summaries first averaged graph variants within a parent and then assigned equal weight to test parents. Unweighted summaries used the source suite's independent graph instances. Training seeds were independent repetitions; checkpoints, sampled orders, vertices, and patches from the same parent were not. We report all seed-level results and paired method differences, with uncertainty computed by the specified hierarchical resampling procedure crossed parent/seed bootstrap

(2,000 draws) at confidence level 95%. No test graph or seed may be removed on the basis of its comparative result.

For fixed GPU allocations, training GPU-hours equaled allocated device count multiplied by end-to-end wall hours. Startup, compilation, data loading, CPU waiting, sampling, averaging, forward/backward, communication, optimizer steps, validation, and checkpoint publication were included. Where allocation changed, we summed device-count times duration over the recorded intervals. Validation resources were held fixed and included. Offline data preparation, hyperparameter search, and test solving were reported separately in addition to the principal per-run training cost. Classical solver CPU time, wall time, and thread count remained distinct from accelerator cost.

For target-quality comparisons with observation through the common administrative deadline B, we reported attainment and the restricted mean time $\mathbb{E}[\min(T, B)]$. Its empirical mean used every assigned seed and retained an administrative-censoring indicator for runs that reached the deadline without attaining the target. An early infrastructure or resource failure was not treated as observed target-free execution until B. When such failures occurred, we used a separately labeled failure-inclusive target score: an early failure received a deadline penalty, while its actual consumed compute and failure time were reported separately. This composite was not labeled a measured restricted mean time. A survival analysis with earlier censoring required its own explicit censoring assumptions and uncertainty; failing seeds were never silently dropped. Nonattainment did not acquire a successful time equal to the deadline, and comparisons did not select only seeds reaching both targets. The restricted-mean SOM-to-RLOO ratio was 0.82; attainment for RLOO and SOM was 100 and 100. Both methods used the same update-based validation cadence; the resulting wall-time resolution depends on the measured step cost.

Each run recorded dataset and source versions, initialization, random-stream domains, parallel layout, precision, resource allocation, actual process termination, and checkpoint identities. The counts of numerical, resource, and infrastructure failures were 0 / 0 / 0. Failures retained their completed prefix and remained in the assigned comparison; silent retries, outcome-dependent replacements, and fallback to a different estimator were disallowed. A completed checkpoint was eligible only if its update had committed within the assigned learning budget. These conventions made the reported quality and speed comparisons refer to the same actual computation, including the cost of obtaining and preserving the result.

D.8 Fixed Rates and Per-Step Re-Encoding

The policy-scope comparison used four configurations: fixed-rate GraphGPS with RLOO, the same fixed-rate GraphGPS with SOM, and per-step re-encoding GraphGPS trained with either the base RLOO group or additional independent samples. The re-encoding configurations shared the encoder depth, hidden width, parameter count, input-feature specification, positive-rate head, deletion rule, reward, and optimizer family. It received the residual adjacency and active-vertex inputs at each step. Re-encoding the residual graph allowed relative rates to change during construction; SOM's fixed-rate factorization was not used for that model. The extra-sample group size received the same validation tuning allowance, and all additional forward and backward calls entered the training budget. Both dynamic controls are reported separately; the explicit fixed-time difference below uses the additional-sample control. The comparison tested one controlled re-encoding architecture with two sampling allocations, not every autoregressive policy architecture.

The primary comparison held allocated training GPU-hours and the inference wall limit fixed. All neural forward and backward calls, residual-graph preparation, sampling, validation, and synchronization entered the cost. A separate equal-update comparison of base-sample RLOO policies held graph batches and samples per graph fixed and reported the resulting GPU-hours. The additional-sample configuration remained a separate allocation comparison: even at equal updates it consumed more sampling and re-encoding work, so its difference was not attributed solely to policy expressiveness. Neither panel substituted for the other: equal updates measured the policy restriction at the same number of learning opportunities, while equal GPU-hours measured its practical cost. One-solution quality before local search was reported alongside the final incumbent at the common inference deadline, with latency and the actual number of encoder calls per solution. A larger number of decoder samples could compensate for faster inference only in the explicitly time-matched panel.

At 250,000 updates, the re-encoding-minus-fixed-RLOO validation-gap difference was -2.505 percentage points, with a training-cost ratio of 2.359. At equal training time, the SOM-minus-additional-sample-re-encoding gap difference was 0.30. These paired comparisons locate any advantage in policy expressiveness, training efficiency, or inference cost rather than assigning all differences to the estimator.

D.9 Exact Terminal References and Larger-Graph Diagnostics

Exact terminal references used small ER and RB instances generated by their recorded parameters and induced subgraphs from held-out MWIS-VR parents. The induced subgraphs were a separate diagnostic panel, not replacements for complete task graphs. Graph selection and the native-policy terminal draws were fixed independently of estimator outcomes. For a terminal S with $L \leq 8$, we enumerated all $L!$ orders, evaluated their sequential probabilities and full original-coordinate scores, and normalized with the summed terminal probability. Thus $s_S = \sum_{Y:S(Y)=S} p_\theta(Y \mid S, G)s(Y)$ used the correct nonuniform weights. Every enumerated trajectory had to end at that same maximal independent set, and exactness referred to enumeration rather than machine arithmetic; numerical error was checked separately.

For each frozen graph and checkpoint, we computed the conditional raw and residual second moments using the same probabilities. If the complete trajectory law fit the common enumeration cap, terminal summaries were weighted by their exact probabilities. Otherwise, native-policy terminal draws supplied the outer expectation, including repetitions according to their original occurrence; the within-terminal calculation remained exact but the outer expectation was estimated. Cases exceeding $L = 8$ or the cap were retained in a coverage record rather than replaced by easier draws. The aggregate was explicitly conditional on this eligible panel when coverage was incomplete. We reported the ratio of parent-weighted variance sums, not an unweighted average of ratios; zero raw variation made the corresponding ratio undefined. The number of eligible parents and terminal draws was 47 and 344, with coverage 68.800% of the prespecified diagnostic draws.

The supplementary exact-panel comparison evaluated native scores, fixed adjacent-pair conditioning, SOM, and the complete terminal score on the same eligible draws. Fixed-pair conditioning retained the original order on odd lengths, matching the nested comparator in Proposition 3.3. For a positive raw denominator, the native and complete-terminal residual fractions were one and zero, respectively; the fixed-pair and SOM fractions measured the remaining variation between these references. This fixed-pair residual comparison was distinct from the total covariance comparison in Section 4.2.

The residual summaries were also stratified by L and parity rather than only graph family. We reported the eligible panel's full length distribution beside that of the actual task-scale native draws, including lengths excluded from enumeration. This comparison exposed whether a low aggregate residual was driven by short orders with unusually complete family coverage.

The same small-graph terminal reference evaluated the practical heat-bath estimator from Appendix D.13. Independent stationary-chain replicas supplied its mean squared distance to the exactly enumerated s_S, with the corresponding Jacobian-transformed distance, standard errors, and confidence intervals. The reference was exact within numerical tolerance; the finite-chain MSE was a Monte Carlo estimate, not an exact enumeration of all chain histories. Comparisons matched complete score-evaluation time, including initialization, transition likelihoods, incremental or fresh score calculations, and synchronization. Sweep counts, complete score evaluations, and hardware allocations were retained as additional cost descriptors rather than substituted for measured time.

The parameter-space residual applied the same frozen B_θ^T to each score and its terminal reference. It measured variation in the gradient of log probability, without introducing reward weights. This distinction matters because the full learning estimator also includes rewards and whole-batch RLOO coefficients. Its covariance was measured separately from independent complete estimator batches as in Appendix D.4. We evaluated that reward-weighted covariance on complete MWIS-VR test graphs and the original RB and ER suites, using frozen checkpoints and matched randomness. Neither this total covariance ratio nor the fraction removed by SOM was labeled a complete-terminal residual on graphs lacking an exact reference.

Appendix B.5 relates total score covariance to within-terminal order variation. We report total covariance separately from exact-terminal residuals and include sampling uncertainty; a reward-weighted RLOO ratio is not substituted into that population identity.

D.10 Record Audits and Pairing Alternatives

A conditional estimator was accepted only after its record map $T(Y)$ and full preimage had been specified. Fixed adjacent pairs retained their ordered blocks and forgot internal orientations. Shifted pairs retained the first and last singleton for even L and paired the intervening adjacent positions. The cyclic-pair record additionally forgot the starting block; the singleton-cycle record forgot only the starting vertex. These are different information-retention rules. Their estimates need not have a universal variance ordering when the records are not nested. The pairing panel measured covariance on common even-length trajectories; its full-law gradient-mean check used the same native fallback on odd lengths for every record to avoid changing the sampled population.

The exact auditor enumerated the native trajectories of each eligible small graph, grouped them by the implemented record, and reconstructed the conditional probability of every trajectory in each group. It checked that the proposed family equaled that preimage, that every native trajectory appeared in exactly one group, that conditional probabilities summed to one, and that the independently summed family probability and score agreed with the implementation. It then compared the expected reward-weighted score with the native full-law reference. For a nonzero reference gradient, mean error was the discrepancy norm divided by its norm; an absolute error was reported when that norm vanished. Numerical agreement alone did not replace the structural definition of the record.

Exploratory graph-aware and multiple-record variants were confined to small graphs where the complete preimage of the proposed map could be enumerated. Selecting a matching on the terminal graph and summing a convenient collection of rearrangements did not suffice: the collection had to be the actual preimage containing the observed trajectory. A fixed convex combination of already valid conditional estimators was evaluated as a separate estimator, with all its computation charged. Its weights were fixed before evaluating the current draw; selecting whichever record gave the most favorable score or variance estimate on that draw was excluded. These exploratory variants did not acquire the cyclic representation's complexity bound without a separate derivation.

D.11 Parity, Coverage, and Whole-Batch Effects

We used the original native samples from every frozen graph and checkpoint, recording L, the parity, the block count actually evaluated, and the retained record. Even trajectories used $K = L/2$ paired blocks; odd trajectories used $K = L$ singletons. No rejection or resampling was used to obtain a desired parity. A second evaluation used singleton cycles for both parities on the same draws, isolating the contribution of pair orientations within the even stratum. Cases with no order variation were retained, with undefined variance ratios where appropriate.

Parity-specific traces were covariances around each stratum's own mean, for both s and $B_\theta^\mathsf{T} s$. The pooled trace was computed directly on all samples, equivalently combining within-parity covariance and between-parity mean variation using observed occurrence probabilities. Averaging the even and odd trace ratios would generally give a different quantity and was not used. Alongside the all-lengths summary, the between-parity/native-total fractions are 0.0065 in score space and 0.0300 in parameter-score space. These terms are computed from the complete vector moments. Under exact conditioning, the parity-specific means agree between the native and averaged scores, so this common between-parity term dilutes the pooled reduction; empirical estimates include their sampling uncertainty. Parent-level summaries preserved the parent weights of the principal evaluation. Averaging latency and allocated memory used the same sample identities, with all failed numerical or resource checks retained.

The policy-gradient check kept complete RLOO groups intact, including groups with mixed parities. We did not classify a batch by one draw's parity or treat individual within-group gradients as independent observations. Paired all-length training compared the standard parity rule with an all-singleton rule under a common GPU-hour budget. The all-singleton-to-standard target-time ratio was 1.10; its final reference-gap difference was 0.18%. These full-run comparisons assessed whether the paired even branch mattered after accounting for the occurrence and cost of odd solutions.

D.12 Sparse Graphs and the Quadratic Table

For every timed graph–checkpoint–trajectory tuple we recorded n, m, L, the actual K, K/n, and $K^2/(n+m)$. The stress panel varied vertex count and average degree with fixed generation rules, including sparse instances with long accepted sets. It was specified before inspecting speedups and was reported separately from the original task suites. For each implementation, we reported the largest successful n and K observed on the prespecified grid under the registered hardware and time limits, with their configuration identifiers. These were grid-dependent endpoints rather than claims of a global maximum feasible size. Out-of-memory, timeout, and numerical failures were distinguished at every attempted grid point. Geometry bins were determined from the recorded graph quantities, not selected after observing which method won.

The separate-start, shared generic, and shared specialized evaluators used common input orders and the same numerical tolerance. The separate-start baseline processed starts sequentially, accumulated their contributions, and released each start's temporary differentiation graph instead of storing all K graphs at once. Any required second pass or recomputation was charged, and the implementation details stated which arrays were retained, so its memory was not inflated by unnecessary simultaneous tapes. Timing included table preparation, all posterior and score calculations, transfers, and synchronization. We measured table-specific allocation, peak full-step allocated and reserved memory, and complete-step time. The principal latency ratio used paired successful evaluations; its coverage was reported with separate timeout, out-of-memory, and numerical-failure frequencies, together with the elapsed and allocated work of those failures. An out-of-memory result was not encoded as zero throughput or an arbitrarily large speedup.

Matched short training runs in each stress regime compared shared SOM with native RLOO, additional samples, and the feasible separate-start evaluator. Their final quality and time-to-target established whether a score-computation saving reached learning. Among paired successful stress runs, the SOM-to-native target GPU-hour ratio was 1.076; the share of attempted shared evaluations ending in a resource or numerical failure was 10.000%. We retained regimes in which the table dominated or SOM did not improve the complete training comparison.

D.13 A Practical Local Estimator and Equal-Time Comparisons

The practical local control started from a native trajectory and repeatedly chose an adjacent position according to a schedule fixed independently of the current order. Given the remaining order, it chose between the two adjacent orientations with probabilities proportional to their native trajectory probabilities. Both orientations reached the same active set after the pair, so their likelihood ratio could be evaluated from the affected local normalization terms. This heat-bath transition preserved $p_\theta(Y \mid S, G)$. Starting from a native conditional draw avoided a burn-in assumption; finite chain averages of the native score preserved its expectation. Each native solution retained its place in the RLOO group, with its chain average replacing only its own score; chain states were not counted as independent reward samples. With positive rates and self-transitions, the composite kernel of a complete sweep covering every adjacent position is ergodic on the terminal permutations; its limiting score average is s_S, not SOM's $\bar{s}$. This limiting target does not imply a favorable finite-budget mixing time. The method approximated that terminal average by correlated samples rather than implementing the theorem's repeated exact conditional-expectation operators.

The implemented control recorded the transition rule, sweep schedule, trajectory likelihoods, number of score evaluations, and cost of updating the score after a swap. Any cached incremental implementation was checked against a fresh native-score calculation. It did not use unweighted permutations or differentiate through the transition probabilities. Fixed sweep counts and additional-sample group sizes were selected on disjoint calibration and validation data, with equal tuning allowances. A per-instance loop was not stopped according to its current gradient or a favorable outcome. Workloads used the calibrated fixed allocation; deviations from the intended time match were reported.

At frozen checkpoints, the time allowance included forward computation, sampling, all local transitions or conditional averages, the complete parameter backward, and the handling of the RLOO group. We compared independent complete-batch replicates and recorded both covariance trace and mean discrepancy. The cheap-local-to-SOM parameter covariance ratio at the matched allowance was 1.175, and its relative mean discrepancy from the independent reference was 0.009. The additional-sample control received the same forward-reuse opportunity. Full learning runs

then used a common end-to-end GPU-hour deadline, test budget, and validation target; all seeds contributed to attainment and restricted mean time. This measurement, rather than the exact-layer theorem, supported the practical efficiency comparison.

D.14 Additional diagnostic figures

Section 4.2 reports the residual, pairing, and parity conclusions, and Section 4.3 compares low-cost alternatives. The complete diagnostic panels are retained here to show all evaluated records, graph families, parity strata, and budget allocations.

Fig. 7 separates small-graph exact-terminal residuals from the matched-draw pairing comparison. The left panel uses the native conditional probabilities when enumerating each terminal set; score and parameter-score residuals are distinct quantities. The right panel reports variance and cost after checking the complete preimage of each proposed record. These panels do not establish exact-terminal coverage on complete task-scale graphs. Protocols and mean-error checks are given in Appendices D.9 and D.10.

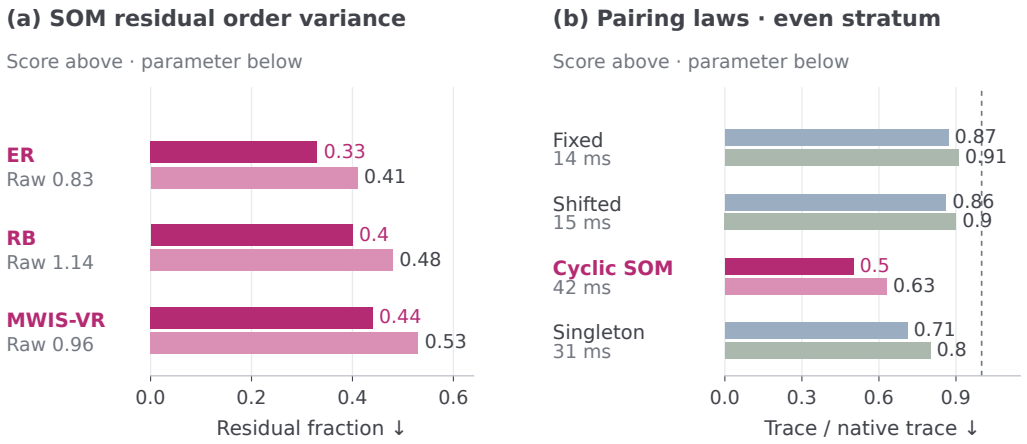


Figure 7: Order averaging. Left: score and parameter-score residual fractions on the enumerable panel. Right: pairing covariance ratios and averaging cost under matched draws; lower is better.

Fig. 8 preserves odd-length and zero-variation cases in the workload rather than resampling for even lengths. Its left panel reports the two strata and their pooled result; Appendix D.11 describes the moment-based pooling and singleton-on-even control. The right panel compares complete equal-budget learning runs. Appendix D.13 specifies the heat-bath transition, fixed sweep allocation, and independent-sample control; local chain states are not treated as independent reward draws.

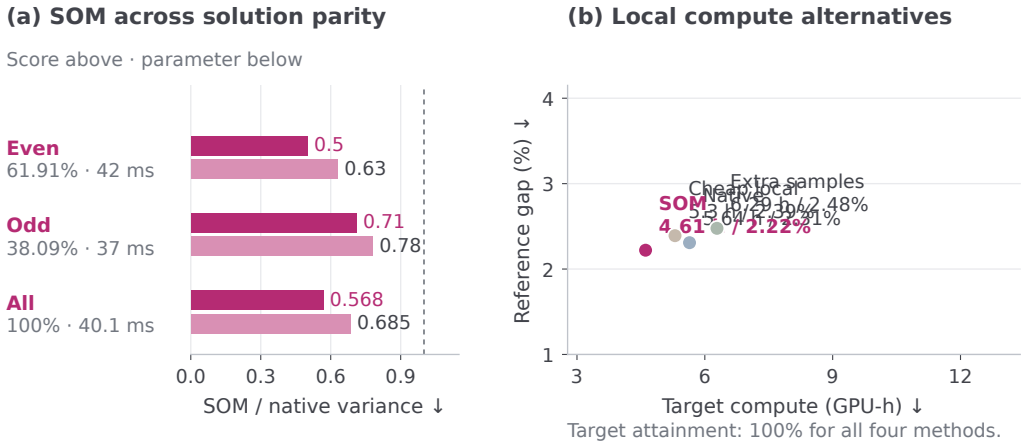


Figure 8: Parity and budget allocation. Left: native-normalized covariance by parity, with occurrence and latency. Right: quality and target time of equal-budget estimators. Lower ratios, gaps and times are better.

E Uncertainty, Reproducibility, and Comparison Fairness

E.1 Configuration and reproducibility records

Experiments use eight NVIDIA H20 devices in one node, host configuration 2 x 64-core x86 CPUs; 1 TiB RAM, and NVLink intra-node. The software stack is PyTorch 2.6.0, PyG 2.6.1, CUDA 12.4, NCCL 2.21.5, and Megatron-Core 0.12.0. Primary learning and timing use one GPU; the one-to-eight-GPU scaling panel is a separate workload. The weighted split VR-target-v3 groups 24/6/7 training/validation/test parents; the unweighted split RB-ER-target-v3 uses 4,096/256/512 instances per suite. Table 1 gives the matched model and budget settings.

The analysis uses separate tables for weighted parents, MIS instances, learning runs and curves, timing repeats, sampling sweeps, tuning controls, and solver time limits. Tables and plots use the same keyed observations, so changing a comparison unit updates every dependent summary. Aggregation retains missingness and run status rather than filling failed observations with zero.

Reproduction requires dataset manifests, parent/topology/weight identities, split assignments, run seeds, checkpoints, machine/software manifests and configuration digests. Returned independent sets and recomputed objectives must be linked to the reference solutions and any certificates. Solver commits, build options, timing boundaries and exit status are separate requirements for reproducing runtime comparisons; reporting a version number or a gap does not itself certify feasibility or optimality.

E.2 Paired uncertainty and all comparison units

Weighted summaries give equal weight to each of the seven parents after averaging each neural method's five seed-level values. A classical solver has one deterministic-run column, not five fictitious training seeds. For each of 2,000 bootstrap replicates, we resample parents and training-seed indices independently; the same seed-index sample is used across all resampled parents because one trained model evaluates every parent. Paired method differences use the same parent and seed selections. This crossed bootstrap avoids treating a reused checkpoint as an independent model for each graph. MIS uses the same crossed construction with 512 independent instance identities per suite. These percentile intervals are descriptive, unadjusted intervals; they are not simultaneous guarantees across all methods or suites.

The weighted post-search difference is -0.02 percentage points, with interval [-0.46, 0.45] crossing zero. This is a paired pipeline comparison against native CHILS, not against the strongest learned initializer. The interval concerns the mean on these parents, not uniform performance across graphs or a certified improvement in the MWIS optimum.

Table 2: MWIS-VR pipeline gaps (%) with crossed parent/seed bootstrap intervals. PRE is the decoder/initial incumbent; POST includes CHILS except for degree-greedy.

Method	PRE mean [95% interval]	POST mean [95% interval]
Degree-greedy	9.87 [6.36, 13.36]	9.87 [6.36, 13.36]
CHILS	9.86 [6.51, 13.12]	2.20 [1.45, 2.98]
REINFORCE	5.68 [5.16, 6.17]	4.14 [3.63, 4.62]
RLOO	3.80 [3.29, 4.29]	2.31 [1.83, 2.74]
Extra samples	4.03 [3.45, 4.57]	2.48 [1.95, 2.98]
Fixed pairs	4.58 [4.01, 5.14]	3.05 [2.56, 3.51]
Alternating pairs	3.88 [3.33, 4.38]	2.36 [1.87, 2.80]
Replay	4.16 [3.65, 4.65]	2.64 [2.17, 3.09]
SOM	3.72 [3.24, 4.22]	2.22 [1.75, 2.65]

Table 3: Paired differences on the same references. Positive gain favors SOM. Intervals retain both positive and negative paired differences; no uniform dominance is inferred.

Comparison	Gain (pp) [95% interval]
MWIS: CHILS − SOM	-0.02 [-0.46, 0.45]
rbsmall: KaMIS − SOM	-1.06 [-1.37, -0.76]
rblarge: KaMIS − SOM	-1.50 [-1.88, -1.13]
er: KaMIS − SOM	-1.06 [-1.41, -0.70]

Table 4: All seven MWIS parent means. Neural columns are post-CHILS. Gain is CHILS minus SOM (percentage points); negative values retain losses. References are uncertified; a reported gap is not an optimality certificate.

Parent	n	CHILS	RLOO	SOM	Gain (pp)
P01	500	2.58	2.85	2.79	-0.21
P02	1200	1.06	1.75	1.77	-0.71
P03	3500	1.10	1.63	1.67	-0.57
P04	6500	2.39	2.69	2.51	-0.13
P05	12000	3.73	2.78	2.81	0.92
P06	24000	1.11	1.51	1.28	-0.17
P07	48000	3.43	2.95	2.71	0.72

Table 5: MIS gaps (%) and crossed instance/seed bootstrap intervals. Each suite has 512 instances; each learned method has five training-seed columns. Classical solvers are not duplicated into training seeds. Decoders are native; no CHILS is used.

Method	RB200–300	RB800–1200	ER700–800
Degree-greedy	8.94 [8.52, 9.36]	13.12 [12.63, 13.67]	10.58 [10.10, 11.04]
KaMIS	2.06 [1.91, 2.22]	3.86 [3.68, 4.07]	3.11 [2.90, 3.32]
X^2GNN	3.71 [3.45, 3.98]	6.57 [6.12, 7.08]	5.49 [5.07, 5.90]
pCQO-MIS	4.85 [4.52, 5.21]	7.34 [6.84, 7.86]	5.79 [5.32, 6.25]
RLOO	5.76 [5.37, 6.14]	9.23 [8.50, 9.95]	7.03 [6.45, 7.59]
SOM	3.12 [2.83, 3.42]	5.36 [4.97, 5.77]	4.18 [3.81, 4.51]

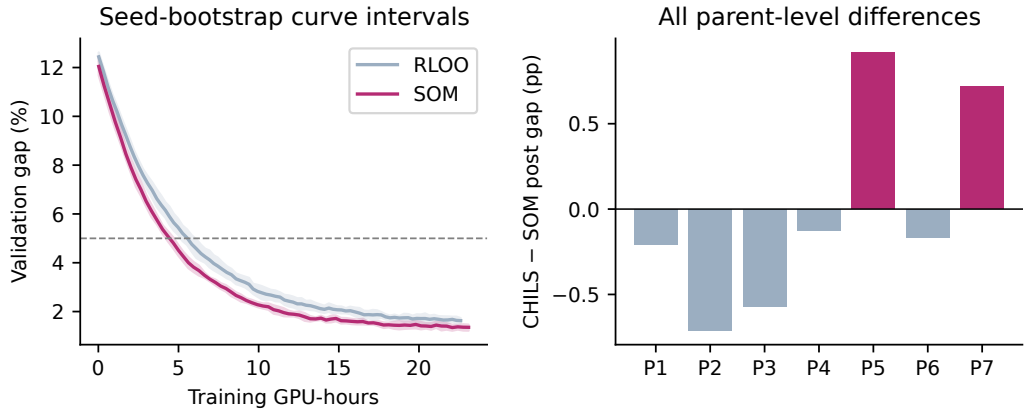


Figure 9: Uncertainty across seeds and parents. Left: validation curves and seed-bootstrap intervals from five explicitly stored curves, with the fixed target. Right: every weighted parent's post-search gap difference. Run-level first-hit time is computed independently of the averaged curve.

The per-instance analysis retains all 1,536 MIS instance identities and every associated seed-level value. Table 6 shows the first four by identifier rather than selecting favorable examples. A MIS reference gap is computed from the integer feasible cardinality and its common reference; reporting the gap does not make an uncertified reference an optimum.

Table 6: Instance-level MIS accounting. r is the common integer reference cardinality; k is the single KaMIS cardinality. SOM reports the sum of five integer cardinalities and their mean gap. Every displayed gap is calculated from these integers; an uncertified reference is not labeled optimal.

Suite	ID	n	r	k	KaMIS gap	SOM $\sum k$	SOM gap
rbsmall	I0001	232	71	69	2.817	349	1.690
rbsmall	I0002	295	71	70	1.408	348	1.972
rbsmall	I0003	203	54	53	1.852	261	3.333
rbsmall	I0004	225	58	58	0.000	278	4.138
rblarge	I0001	1,045	176	174	1.136	847	3.750
rblarge	I0002	813	156	153	1.923	756	3.077
rblarge	I0003	1,156	159	156	1.887	787	1.006
rblarge	I0004	1,086	148	145	2.027	717	3.108
er	I0001	735	56	53	5.357	272	2.857
er	I0002	757	58	56	3.448	269	7.241
er	I0003	782	72	68	5.556	335	6.944
er	I0004	752	72	68	5.556	331	8.056

E.3 CHILS POSTPROCESSING, NATIVE DECODERS, AND RESOURCE FAIRNESS

Figure 3(a) reports the complete weighted pipeline. Each neural method prepares features, encodes the graph, samples its candidate initialization, and then runs CHILS using only the remainder of the same 60-second wall-clock allowance. Native CHILS uses its own initializer under the same deadline and CPU-thread allocation. PRE in Table 2 denotes the initial incumbent before CHILS refinement; POST is the final incumbent. The SOM-versus-CHILS difference therefore measures the incremental value of learned initialization in a pipeline containing CHILS, not a replacement solver defeating CHILS without its assistance. The SOM-versus-RLOO comparison retains identical initialization/CHILS interfaces and hardware allocations.

Figure 3(b) uses each solver's native unit-weight decoder and no CHILS wrapper. Reference values are treated as uncertified throughout, retaining improvements beyond an old reference rather than clipping negative gaps. KaMIS uses a one-thread heuristic-mode configuration, seed 0, and limits of 1, 5, 10, 30, 60 and 300 seconds. Exact solver commits and options are required to reproduce this

Table 7: Resource allocation. Equal wall time is not equal heterogeneous hardware. CPU threads and GPU allocation are reported separately; preparation is charged inside the deadline.

Pipeline	Wall (s)	CPU threads	GPU allocation	Search after decode
Native CHILS	60	1	none	native CHILS
Neural MWIS	60	1	one H20	CHILS, remaining time
KaMIS / greedy	30	1	none	native
Learned MIS	30	1	one H20	native, no CHILS

comparison; hardware and wall limits alone do not specify the executable. Table 8 retains the entire time sweep. Rankings at 30 seconds cannot be extrapolated to longer searches, certified optima, or other hardware.

Table 8: KaMIS time-limit sensitivity under a one-thread configuration. References remain fixed across time. The 30-second point is not a claim of optimality or a universal ranking.

Limit (s)	RB200–300	RB800–1200	ER700–800
1	8.63 [8.25, 9.00]	10.04 [9.65, 10.47]	9.40 [8.96, 9.85]
5	6.16 [5.86, 6.46]	7.82 [7.49, 8.17]	7.15 [6.80, 7.51]
10	4.10 [3.89, 4.32]	5.83 [5.57, 6.11]	5.06 [4.76, 5.36]
30	2.06 [1.91, 2.22]	3.86 [3.68, 4.07]	3.11 [2.90, 3.32]
60	1.13 [1.01, 1.24]	2.25 [2.12, 2.38]	1.89 [1.73, 2.04]
300	0.22 [0.18, 0.27]	0.53 [0.48, 0.59]	0.46 [0.38, 0.53]

E.4 Native scoring cost and time-to-target accounting

Table 9 adds the missing native RLOO row on the same one-GPU workload. The native complete step is 101.89 ms, including 26.53 ms for raw scoring; the specialized SOM step is 118.21 ms. A forward/sampling/backward/update subtotal omits raw scoring and cannot be used as the complete RLOO baseline. The shared-versus-rebuild comparison and SOM-versus-native comparison answer different questions.

Table 9: Synchronized one-GPU step breakdown (ms), same batch, eight draws and workload. Score includes raw scoring for RLOO and conditional averaging for SOM. The five stages sum to the total; a non-score subtotal is not a complete RLOO step.

Method	Forward	Sample	Score	Backward	Update	Total
Native RLOO	18.33	12.23	26.53	35.62	9.18	101.89
Separate SOM	18.28	12.23	188.61	35.64	9.16	263.92
Shared AD	18.51	12.23	66.17	35.69	9.17	141.78
Shared SOM	18.38	12.25	42.82	35.59	9.16	118.21

Table 10: Run-level cost accounting, aggregated over five runs. GPU-h equals GPU count times (committed updates times step seconds plus other wall seconds)/3600 for every raw run. The displayed step is the all-update average; other time is the sum of counted validation calls, I/O and setup. Integer per-run counts are shown separately; the mean count need not be integer.

Backbone / method	GPU-h [95% interval]	Mean step (ms)	Mean updates	Mean other (s)
GraphGPS / RLOO	5.64 [5.21, 6.15]	97.85	204000	369.14
GraphGPS / SOM	4.61 [4.34, 4.87]	118.15	138000	282.24
Graphormer / RLOO	9.33 [8.58, 10.00]	145.72	226000	615.61
Graphormer / SOM	6.82 [6.51, 7.12]	165.28	146000	441.71
GatedGCN / RLOO	4.42 [4.16, 4.72]	63.00	248000	316.07
GatedGCN / SOM	3.73 [3.59, 3.87]	77.48	170000	238.22

For every learning run the table generator checks the identity

$$H = \frac{D}{3600}\left(U\bar{t}_{\text{step}} + t_{\text{other}}\right),$$

where D is the allocated GPU count, U is the committed update count, and $\bar{t}_{\text{step}}$ is the measured all-update mean in seconds. Validation, initialization, data loading and checkpoint work contribute to t_{other}, not a second addition of already included step stages. For GraphGPS the step ratio is 1.21 and the update-count ratio is 0.68; the target-cost ratio remains 0.82. Per-run integer counts and additive cost components are listed in Appendix E.8; means of products are not replaced by products of means.

These bookkeeping identities do not explain why an optimizer takes fewer updates. In particular, a covariance-trace reduction at one frozen checkpoint is not a causal law for time-to-quality, which also depends on the gradient mean, learning rate, curvature, exploration and checkpoint selection. The exact conditioning theorem is therefore not presented as a derivation of the empirical learning-speed factor.

E.5 Sampling allocation and validation-only tuning controls

Table 11 and Fig. 10 sweep independent draws $N \in \{2, 4, 8, 16, 32\}$ for native RLOO and SOM at a 24 GPU-hour deadline. Both reuse one actor forward pass and charge sampling, scoring, full backward, validation and updates. The entire sweep is reported, not only the best point. A completed nonattainer is administratively censored at the deadline and remains in the attainment denominator; an early resource failure would require a separately named failure-inclusive score rather than being silently treated as observed target-free execution.

Table 11: Sampling sweep under a 24 GPU-hour deadline. RMST includes all five assigned runs; a completed nonattainer contributes the deadline but is not a successful hit. Additional samples reuse the same forward pass.

Method	N	RMST [95% interval]	Hit (%)	Gap [95% interval]	Step (ms)
RLOO	2	9.28 [9.05, 9.58]	100	3.22 [3.07, 3.34]	83.28
RLOO	4	6.52 [6.20, 6.96]	100	2.43 [2.34, 2.49]	87.14
RLOO	8	5.64 [5.21, 6.15]	100	2.31 [2.20, 2.41]	97.85
RLOO	16	6.29 [6.03, 6.66]	100	2.48 [2.37, 2.60]	122.43
RLOO	32	7.92 [7.56, 8.26]	100	2.63 [2.56, 2.70]	158.01
SOM	2	6.33 [6.16, 6.55]	100	2.70 [2.63, 2.79]	87.30
SOM	4	5.17 [4.97, 5.32]	100	2.02 [1.92, 2.11]	101.55
SOM	8	4.61 [4.34, 4.87]	100	2.22 [2.06, 2.33]	118.15
SOM	16	5.70 [5.45, 5.95]	100	1.56 [1.45, 1.68]	155.92
SOM	32	7.68 [7.29, 8.10]	100	1.98 [1.89, 2.10]	228.39

Table 12 retains native RLOO, separately retuned RLOO, larger-N RLOO, an independently trained value baseline, a finite heat-bath chain, fixed-configuration SOM, and separately retuned SOM. Each method receives eight validation-only trials; the fixed rows isolate estimator substitution, whereas retuned rows compare achievable performance under the same tuning allowance. Retuning narrows the fixed-configuration gap; the paired ratios and their intervals are reported separately in Appendix E.8. They do not prove that reduced covariance alone causes a particular update-count reduction.

The local-layer theorem isolates the effect of averaging range on a specified exact projection sequence. The finite heat-bath chain is the practical comparator, not an implementation of those exact layers. Neither the star result nor the number of orders in the cyclic family provides a graph-uniform finite-budget guarantee. Likewise, fixed rates exclude policies that re-encode changing residual graphs; the existing controlled re-encoding panel examines that opportunity cost without claiming coverage of all autoregressive architectures.

The comparison does not cover unordered-set, multi-sample variational, diffusion, GFlowNet or all other neural solvers. A valid added baseline must match the reward objective, inference task and budget, or be labeled as a different objective/solver. Complete baseline coverage and a variance-only causal explanation are not established by these controls.

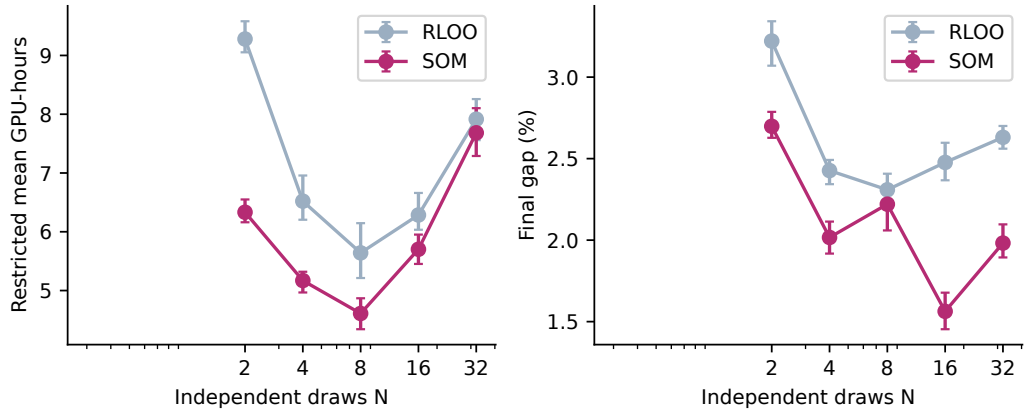


Figure 10: Sampling-allocation sweep. Both methods receive the same total training deadline. Error bars are seed-bootstrap intervals; larger N is not assumed to improve time-to-quality monotonically.

Table 12: Controlled tuning and variance-reduction comparisons. All rows use the same encoder and eight validation-only tuning trials. Variance is a frozen-checkpoint ratio, not a model of optimizer dynamics. A learned value baseline adds its acquisition and evaluation to the charged budget.

Method	LR	N	GPU-h [95% interval]	Gap [95% interval]	Var.
RLOO-fixed	1.0×10^{-4}	8	5.64 [5.21, 6.15]	2.31 [2.20, 2.41]	1.03
RLOO-retuned	2.0×10^{-4}	8	4.61 [4.39, 4.87]	2.19 [2.06, 2.36]	0.93
RLOO-N32	1.0×10^{-4}	32	7.92 [7.56, 8.26]	2.63 [2.56, 2.70]	0.30
RLOO-value-baseline	1.0×10^{-4}	8	5.25 [5.04, 5.46]	2.37 [2.29, 2.44]	0.84
Heat-bath	1.0×10^{-4}	8	5.30 [5.14, 5.49]	2.39 [2.25, 2.54]	0.77
SOM-fixed	1.0×10^{-4}	8	4.61 [4.34, 4.87]	2.22 [2.06, 2.33]	0.70
SOM-retuned	2.0×10^{-4}	8	4.15 [3.87, 4.44]	2.14 [1.97, 2.30]	0.70

Table 13: Fixed-update diagnostic at 250,000 updates. This comparison gives the same update opportunities, not the same compute. No causal attribution to covariance alone follows from it.

Method	Gap at 250k updates [95% interval]
RLOO-fixed	4.12 [3.81, 4.50]
RLOO-retuned	3.15 [2.91, 3.43]
RLOO-N32	3.65 [3.49, 3.81]
RLOO-value-baseline	3.66 [3.46, 3.93]
Heat-bath	3.45 [3.34, 3.59]
SOM-fixed	2.80 [2.63, 2.94]
SOM-retuned	2.54 [2.40, 2.73]

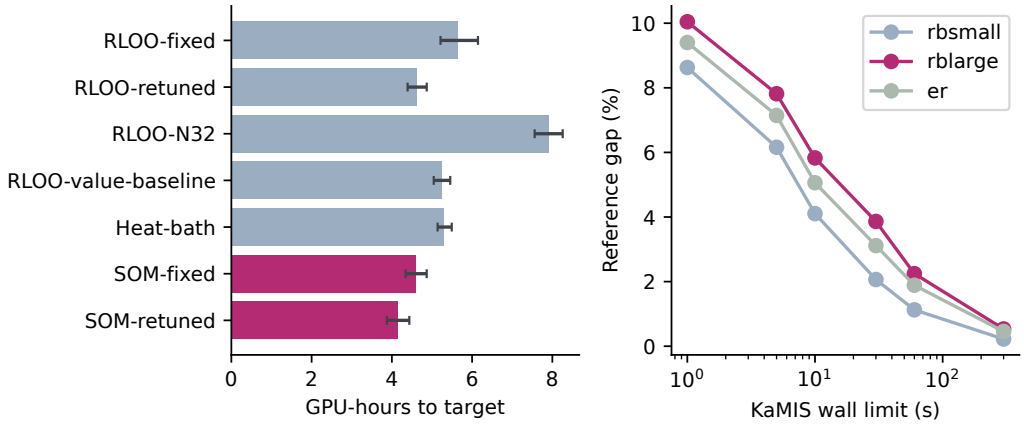


Figure 11: Sensitivity controls. Left: common-target cost after matched tuning; right: native KaMIS sensitivity to the allowed wall time. The panels have different comparison units and are not a joint solver ranking.

E.6 Scale, adverse regimes, and interpretation

Table 14: Complete-parent test gaps by graph size; this is not patch evaluation. Small: 500–1,200 vertices; medium: 3,500–12,000; large: 24,000–48,000. With only two or three parents per stratum, intervals are descriptive and cannot establish uniform generalization.

Size / parents	CHILS	RLOO + CHILS	SOM + CHILS
small / 2	1.82 [1.06, 2.58]	2.30 [1.67, 2.91]	2.28 [1.61, 2.85]
medium / 3	2.40 [1.10, 3.73]	2.37 [1.71, 2.87]	2.33 [1.72, 2.83]
large / 2	2.27 [1.11, 3.43]	2.23 [1.40, 3.10]	2.00 [1.16, 2.85]

Table 14 separates complete test parents by size instead of pooling them with training patches. Larger-graph generalization remains uncertain with only two or three parents in a stratum. The sparse stress regime retains its SOM/native target ratio of 1.076 and failure frequency 10.000%; it is not removed to improve the headline.

A near-one shared/separate implementation ratio on RB or ER does not logically contradict a training-quality difference from RLOO: the former compares two evaluations of the same cyclic estimator, whereas the latter changes the estimator. Nevertheless, modest sharing savings and large cyclic tables limit practical appeal, and only full end-to-end budgets can establish a useful trade-off. The distributed appendix similarly keeps cross-framework throughput separate from within-framework estimator gains. The conclusions concern the stated pipeline and uncertainty rather than treating every positive point estimate as a general solver advantage.

E.7 Integer objectives and the identity of a comparison unit

Table 6 reports the actual arithmetic behind the instance-level MIS gaps. Each instance has one integer reference cardinality $r \leq n$. A classical solver contributes one returned integer k, whereas a neural entry averages five seed-level gaps with that same r. Thus the neural mean is $100(1 - \sum_s k_s/(5r))$, not the result of rounding a continuous gap into an integer solution after aggregation. Each time-limit observation retains the same reference and an integer incumbent; longer runs cannot lose the best solution found so far. The full instance-level export retains n, r, k as well as the derived gap.

Feasible objective values and an uncertified reference do not establish optimality. For measured MIS data, the selected vertex IDs must pass an independent edge-conflict check and their integer cardinality must equal the recorded objective. For weighted data, the recorded objective must equal

the sum of original vertex weights. These are separate checks from the arithmetic identities above; a plausible gap alone cannot certify that an independent set exists in the stated graph.

E.8 Event-level timing, checkpoint identity, and target sensitivity

Validation occurs at update zero and after each block of 10,000 committed updates, rather than at equal GPU-hour intervals. Each event records cumulative compute, validation, I/O and setup seconds. Learning curves align the same update checkpoint across seeds and plot its mean elapsed GPU-hours; their vertical bands therefore describe seed variation at a common update, not a fixed wall-clock time. Only checkpoints observed for all five seeds enter these averaged curves; later per-run events remain in the run-level records. Target time is the first recorded validation event meeting the threshold, never an interpolation through the mean curve.

Table 15: Every primary GraphGPS run. Updates are integers and validation calls occur at update 0 and every 10,000 committed updates. Validation, I/O and setup are distinct additive costs; target time is read from the first recorded crossing.

Method	Seed	Updates	Calls	Validation s	I/O s	Setup s	GPU-h
RLOO	0	240,000	25	300.22	56.52	67.03	6.518
RLOO	1	210,000	22	262.11	45.17	81.53	5.999
RLOO	2	190,000	20	240.65	42.14	92.25	5.179
RLOO	3	180,000	19	223.03	39.49	48.94	5.198
RLOO	4	200,000	21	232.91	48.89	64.82	5.312
SOM	0	140,000	15	187.64	33.47	78.34	4.824
SOM	1	140,000	15	177.12	27.26	51.28	4.776
SOM	2	130,000	14	169.46	34.42	47.63	4.359
SOM	3	130,000	14	167.37	31.93	66.42	4.131
SOM	4	150,000	16	194.43	38.56	105.87	4.959

Table 16: Paired ratios of mean target costs with common seed resampling. Fixed and separately tuned comparisons are separate estimands; no mean of per-seed ratios is substituted.

Comparison	Speed ratio [95% interval]
RLOO-fixed / SOM-fixed	1.224 [1.129, 1.303]
RLOO-retuned / SOM-retuned	1.110 [1.067, 1.163]
RLOO-$N = 32$ / SOM-fixed	1.717 [1.623, 1.814]
Heat-bath / SOM-fixed	1.150 [1.084, 1.247]

Every row in Table 15 has an integer committed-update count. Table 10 instead reports means across those runs, so its mean update count need not be integer. Per-run accounting uses the product of that run's update count and its own average step time; a product of aggregate means need not reproduce the mean of products. The increasing validation count contributes its own measured duration rather than assigning the same overhead to runs of different lengths.

The fixed-configuration speed ratio is 1.224, with interval [1.129, 1.303]. For separately tuned methods it is 1.110, with interval [1.067, 1.163]. The tuned improvement is smaller than the fixed-configuration comparison and is reported separately, rather than assigning the larger ratio to the tuned methods. The heat-bath comparison has ratio 1.150, with interval [1.084, 1.247]. All these ratios use paired resampling of seed identities, and every assigned run remains in the attainment denominator.

Table 17: Sensitivity to three targets selected before testing. Restricted mean cost uses the common 24 GPU-hour horizon and every assigned seed; attainment is reported separately. A nonattainer is not called a successful hit at the horizon.

Gap target (%)	Configuration	GPU-h [95% interval]	Hit (%)
7.0	rloo-n8	3.522 [3.233, 3.855]	100
7.0	som-n8	2.859 [2.613, 3.106]	100
7.0	rloo-retuned	2.826 [2.652, 3.011]	100
7.0	som-retuned	2.506 [2.272, 2.748]	100
7.0	rloo-n32	4.936 [4.527, 5.358]	100
7.0	heat	3.259 [3.100, 3.404]	100
5.0	rloo-n8	5.641 [5.213, 6.146]	100
5.0	som-n8	4.610 [4.342, 4.869]	100
5.0	rloo-retuned	4.613 [4.392, 4.870]	100
5.0	som-retuned	4.155 [3.875, 4.435]	100
5.0	rloo-n32	7.915 [7.559, 8.258]	100
5.0	heat	5.301 [5.140, 5.493]	100
3.0	rloo-n8	9.870 [9.321, 10.527]	100
3.0	som-n8	7.894 [7.547, 8.139]	100
3.0	rloo-retuned	7.755 [7.044, 8.588]	100
3.0	som-retuned	6.988 [6.517, 7.458]	100
3.0	rloo-n32	14.333 [13.673, 14.993]	100
3.0	heat	9.095 [8.930, 9.251]	100

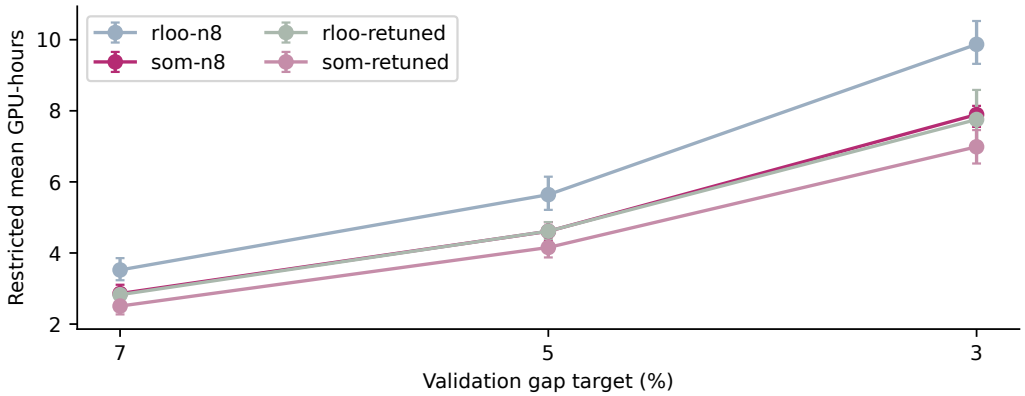


Figure 12: Target sensitivity under the same administrative horizon. A stricter validation target changes both elapsed cost and attainment. Fixed and separately tuned RLOO/SOM pairs remain distinct comparisons.

E.9 CONCRETE COMPARISON WORKLOADS AND VALIDATION SELECTION

The one-step implementation microbenchmark uses a batch of 32 graphs with $n = 256$, $m = 2{,}048$, accepted length $L = 64$, and eight independent draws per graph. It compares native scoring, separate-start SOM, shared generic AD and shared specialized SOM on the same inputs. The 16-graph numerical-audit panel in Appendix D.5 is a separate panel with eight retained orders per graph. Neither microbenchmark is the 48,000-vertex complete-graph test. The policy-scope inference latency is a one-solution measurement at $n = 256, L = 64$; it is not the end-to-end complete-parent time. Table 18 reports the latter's preparation and remaining search allocation.

Table 18: Per-parent preparation/decode and remaining CHILS time. Each pair sums to the same 60-second deadline. GPU allocation is reported separately; equal wall limits are not equal heterogeneous resource costs.

Parent	CHILS prep	Search	RLOO prep	Search	SOM prep	Search
P01	0.001	59.999	0.004	59.996	0.004	59.996
P02	0.001	59.999	0.006	59.994	0.007	59.993
P03	0.001	59.999	0.015	59.985	0.015	59.985
P04	0.002	59.998	0.027	59.973	0.024	59.976
P05	0.002	59.998	0.044	59.956	0.046	59.954
P06	0.004	59.996	0.089	59.911	0.093	59.907
P07	0.006	59.994	0.157	59.843	0.168	59.832

The tuning design separates architecture, estimator and sampling changes. Fixed rows use learning rate 10^{-4} and eight independent draws. The separately tuned RLOO and SOM rows select from learning rates $\{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}\}$ and gradient-clipping thresholds $\{0.5, 1.0\}$, giving eight trials per method. The independent N sweep uses $\{2, 4, 8, 16, 32\}$ and is not retrospectively merged into a larger tuning budget for one method. The selection criterion is validation restricted-mean target cost, with final validation gap breaking ties; test graphs do not select configurations. The value-baseline and heat-bath controls receive the same trial count, charging their additional work.

The local heat-bath control uses a fixed left-to-right/right-to-left alternating adjacent-position sweep. Candidate sweep budgets are $\{1, 2, 4, 8\}$, with cached and fresh score updates checked on the same calibration inputs. Its selected allocation is fixed before a measured batch and does not stop when a draw happens to produce a favorable gradient. A cost match must include the entire forward, transitions, score, backward and update path; a kernel-only match is not an equal-training-cost comparison.

The training protocol and the executable identity must be distinguished. The selected solver commit, build options, graph ordering, feature normalization and exact attention implementation must be pinned in the run manifest before execution; a version-like placeholder cannot substitute for these identities. Likewise, a large-graph attention claim requires a separate capacity and memory trace. The present cost comparisons are tied to their declared workloads and do not establish that every graph, model size and parallel setting fits the same memory budget.

E.10 SMALL-REFERENCE COVERAGE, TASK-SCALE GRADIENTS, AND TABLE GEOMETRY

Table 19: Composition of the small-terminal panel. Parents are independent source units within each family; MWIS-VR uses the seven held-out parents and their induced diagnostic subgraphs, not 20 additional independent test parents. Eligible draws retain their original occurrence frequency.

Family	Parents	Draws	Eligible	Coverage (%)	L cap
ER	20	168	123	73.21	8
RB	20	170	117	68.82	8
MWIS-VR	7	162	104	64.20	8

The exact-terminal panel contains 47 independent source parents across three families and 344 eligible terminal draws out of 500. Only seven of those source parents are MWIS-VR parents; induced

Table 20: Frozen-checkpoint whole-batch parameter-gradient covariance ratios. Each row uses 24 independent replicates of 512 estimator batches at one graph-family/checkpoint stratum. These total-covariance ratios are not exact-terminal residual fractions.

Family	Checkpoint updates	SOM/native trace [95% interval]
MWIS-VR	50,000	0.681 [0.660, 0.700]
MWIS-VR	150,000	0.677 [0.659, 0.694]
MWIS-VR	250,000	0.676 [0.656, 0.696]
RB	50,000	0.672 [0.653, 0.690]
RB	150,000	0.708 [0.689, 0.728]
RB	250,000	0.694 [0.669, 0.718]
ER	50,000	0.714 [0.688, 0.741]
ER	150,000	0.703 [0.688, 0.719]
ER	250,000	0.703 [0.674, 0.729]

Table 21: Exact geometry behind the sparse-cost panel. FP64 single-table bytes equal $8K^2$; this is not full allocator peak memory. Step ratios use paired successful repeats; failed shared attempts remain in the failure denominator and have no invented successful latency.

Regime	n	m	L	K	$K^2/(n+m)$	Table GiB	Step ratio	Fail
mwis-vr	48,000	1,200,000	10,560	5,280	22.338	0.207710	0.480	0/10
rbsmall	256	8,192	64	32	0.121	0.000008	0.950	0/10
rblarge	1,024	65,536	192	96	0.138	0.000069	0.870	0/10
er	768	24,576	94	47	0.087	0.000016	0.940	0/10
stress	4,096	8,192	2,868	1,434	167.347	0.015321	1.110	1/10

Table 22: Moments for reproducing the pooled ratio. The common between-parity component is $p_e p_o \|\mu_e - \mu_o\|^2$. Pool raw and averaged traces separately, then divide; displayed rounded ratios are not pooling inputs.

Parity	Count	Raw score	Score ratio	Raw param.	Param. ratio
even	317	1.080000	0.500000	0.960000	0.630000
odd	195	0.790000	0.710000	0.680000	0.780000

Table 23: Pooled results at higher precision. Agreement between a three-digit display and a naive weighted average would not establish equality of the underlying estimands.

Quantity	Value
score-ratio	0.568005690
parameter-ratio	0.685263829
score-fraction	0.006523859
parameter-fraction	0.030019470

subgraphs do not become additional independent parents. Coverage is 68.800%. The task-scale gradient panel uses checkpoints at 50,000, 150,000 and 250,000 updates, recording total reward-weighted parameter-gradient covariance separately from terminal residuals. The main gradient ratio uses the MWIS-VR/150,000-update stratum; other strata are not pooled to improve that number.

The table-memory column is exactly one FP64 $K \times K$ table, $8K^2/2^{30}$ GiB, not an allocator peak. Complete implementation memory includes forward/reverse tables, model state, sampling and temporary arrays. The graph tuples underlying the timing ratios are explicit in Table 21. An OOM attempt has a failure label and elapsed cost but no successful latency or target time. The evaluator columns compare shared with separate-start evaluation; the separately labeled learning-policy columns compare SOM with native RLOO. The stress learning ratio is conditional on paired successful runs and is presented with its separate failure fraction; it is not a failure-inclusive RMST estimate.

For parity pooling, raw and averaged within-stratum covariance traces are weighted separately and receive the same between-stratum term before taking their ratio. Tables 22–23 expose these inputs at higher precision. A coincidence in a three-digit display is not evidence that the raw moments were replaced by a weighted mean of ratios. Exact terminal residuals, transformed-score covariance and reward-weighted batch-gradient covariance remain different measurements throughout.

E.11 A NUMERICAL VALIDATION CONTRACT, NOT AN INFERRED CERTIFICATE

The FP64 implementation and its generic counterpart are compared on identical represented positive rates, sampled orders and original vertex coordinates, before centering. Reference audits start at 128-bit interval precision and double precision on the same record until the stopping condition is met. The resource-limited audit stops at 1,024 bits and reports a numerical failure if the condition remains unresolved. This cap limits the audit workload; it is not a theorem that 1,024 bits always suffices. Refinement count, time, output-conversion error and failed attempts belong in the numerical-audit record.

Appendix B.4 supplies an actual certificate path: represent the positive dyadic rates in common integer units, choose the seed grid so that $WT_0\delta \leq \min(\varepsilon, 1)/4$, propagate the rounded seeds through the integer transpose, and accept only if the bound plus the exact output-conversion error is at most ε. Generic and specialized evaluators are entitled to the same certificate construction. An FP64/high-precision discrepancy is an empirical accuracy check, not this deterministic per-call certificate. We therefore do not transfer the certified path's guarantee to an ordinary FP64 training trace without an accompanying certificate for every required call and its charged cost.

The reported reference error 2.8×10^{-7} and generic/specialized discrepancy 1.6×10^{-7} concern the stated audit panel. The distributed discrepancy 2.1×10^{-6} concerns the shared objective and normalization under the stated layouts. These diagnostics do not certify all task-scale records or all attention/parallel implementations.