# PER-LAYER ROUTING OVER HETEROGENEOUS EXPERTS FOR SHORTEST-PATH NEURAL ALGORITHMIC REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Neural Algorithmic Reasoning (NAR) asks whether a neural processor can carry a classical algorithm beyond its training scale. Shortest-path algorithms differ in how their inner step aggregates and composes path lengths, yet most NAR processors commit to a single aggregation geometry, and an end-to-end selector trained on task loss alone falls short of the best single algebra on four of seven CLRS-30 tasks. We introduce PSAR, a processor that pools three attention algebras (softmax, $\alpha$-entmax, tropical max-plus) and a triangular multiplicative pair-state update (TMU), and selects one expert per hint step with a hard router trained in stages over frozen, then jointly tuned, experts. Under teacher-forced evaluation on three CLRS-30 shortest-path tasks, no single expert is best on all three. PSAR, trained with the same recipe on each task, matches or exceeds the strongest attention algebra on every task, but falls below standalone TMU on Dijkstra. On Floyd–Warshall, most of the observed gain over attention is already present in the pair-state expert: standalone TMU is $13.0$ percentage points above the strongest attention algebra at $N{=}32$. On the same task at $N{=}48$ and $64$, PSAR outperforms a parameter-matched TMU by $7.2$ and $8.1$ pp, and surpasses its own step schedule retrained as a fixed routing rule by $16.1$ and $12.6$ pp. However, when PSAR and the parameter-matched TMU are trained and evaluated on their own predictions, the TMU leads on Floyd–Warshall. For shortest-path NAR, these results identify the pair-state update as a primitive worth adding to attention-based processors, and show that the ordering between processors can reverse with the hint contract under which they are trained and compared.

## 1 INTRODUCTION

Neural Algorithmic Reasoning (NAR) studies whether neural networks can be trained to execute classical algorithms in a way that transfers beyond the training distribution (Veličković & Blundell, 2021). The standard stress test is length generalization on the CLRS-30 benchmark (Veličković et al., 2022): models are trained on small graphs (typically $N{=}16$) and evaluated on substantially larger inputs, with in-distribution versus out-of-distribution (OOD) accuracy gaps of ten to sixty percentage points routinely observed. Modern NAR architectures such as Triplet-GMPNN (Ibarz et al., 2022) attain strong in-distribution scores yet continue to degrade on larger $N$, a pattern consistent with the representation fitting an input–output map at training scale rather than internalising the step that the target algorithm actually performs.

Several orthogonal axes have been linked to this failure – operational–semantic alignment between the processor's aggregation primitive and the algorithm's inner step, discrete state tracking, adaptive computation depth, and compositional program structure – and this paper isolates the first. The attention layers used by recent processors predominantly implement normalized weighted-sum aggregation (Vaswani et al., 2017), whereas graph algorithms draw their inner aggregation from a wider family. Sparse, support-adaptive normalisers such as $\alpha$-entmax (Peters et al., 2019; Correia et al., 2019) modify the support of this weighted aggregation, while tropical $(\max, +)$ attention (Hashemi et al., 2025) provides a max-over-sums operation that is sign-dual to the classical $(\min, +)$ relaxation used in shortest-path dynamic programming (Dudzik & Veličković, 2022). Whether any one of these aggregation primitives suffices across the algorithmic tasks in CLRS-30, or whether



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.

selecting among them per step helps, is an open empirical question – distinct from summary-aggregator selection (Corso et al., 2020) or head-level soft mixing within standard attention (Jin et al., 2025). A controlled survey across the seven selected tasks of CLRS-30 (§3) shows that no single attention algebra is uniformly best and that an end-to-end per-head selector trained on task loss alone falls short of the per-task best on four of the seven tasks. We therefore build PSAR (§4), which pools the three attention algebras with a pair-state primitive, the triangular multiplicative update (TMU) of AlphaFold, and selects one expert per hint step with a hard router trained in stages over frozen, then jointly tuned, experts. We evaluate it on Dijkstra, Bellman–Ford and Floyd–Warshall against each expert alone and against controls that probe different factors, and organise the results around three questions: how much comes from the pair-state primitive (§6), what the routed pool adds beyond it (§7), and whether the ordering between processors survives a change of the hint contract under which they are trained and compared (§8).

**Contributions.** We make four contributions in this paper:

- **A routed heterogeneous processor.** PSAR (§4) pools three attention algebras from two aggregation semirings (softmax, $\alpha$-entmax, tropical max-plus) and the triangular multiplicative update (TMU; Jumper et al., 2021) of AlphaFold, and selects one expert per hint step with a hard router trained in stages. The same staged pool supports matched controls: a homogeneous pool, constant, step-only and retrained fixed-schedule policies, a routing-free mixture and a parameter-matched specialist. Its in-pool preference-distillation term is a regulariser; switching off every routing-supervision loss moves accuracy at $N{=}32$ by under $0.4$ pp (Appendix I.1).
- **Pair-state structure accounts for most of the Floyd–Warshall gain over attention.** Under teacher forcing on three shortest-path tasks, no single expert is best on all three; PSAR matches or exceeds the strongest attention algebra on every task but falls below standalone `TMU` on Dijkstra. On Floyd–Warshall at $N{=}32$, standalone `TMU` is $13.0$ pp above the strongest attention algebra, accounting for most of PSAR's $14.9$-pp margin over attention (§6). This is a further instance, alongside triplet messages and triangular attention (§2), of third-node composition being useful for shortest-path NAR under the tested configuration; the comparison is not capacity-isolated (§6).
- **Beyond the pair-state expert on Floyd–Warshall.** At $N{=}48$ and $64$, the two sizes not used for checkpoint selection, PSAR is above a parameter-matched `TMU` by $7.2$ and $8.1$ pp and above its own step schedule retrained as a fixed routing rule by $16.1$ and $12.6$ pp, on every seed (§7). The advantage is bounded: a step-only router is not separable from PSAR at four seeds, a routing-free mixture is not separable at $N{=}64$, and on Dijkstra and Bellman–Ford routing adds little.
- **The ordering reverses with the hint contract.** When PSAR and the parameter-matched `TMU` are retrained, checkpoint-selected and evaluated on their own predictions, the `TMU` leads on Floyd–Warshall, in two harnesses; evaluating the teacher-forced checkpoints on their own predictions does not by itself reverse the ordering (§8).

## 2 Related Work

**NAR on CLRS-30.** Neural Algorithmic Reasoning studies neural execution of classical algorithms with OOD length generalisation (Veličković & Blundell, 2021; Veličković et al., 2022).

Representative strong methods include Triplet-GMPNN (Ibarz et al., 2022), Open-Book NAR (Li et al., 2024), and DNAR (Rodionov & Prokhorenkova, 2025). Language-model composition methods such as TransNAR (Bounsi et al., 2024) on CLRS-Text (Markeeva et al., 2026) are a separate axis we do not target (§9). These methods commit to a single aggregation primitive on the GNN side.

The pair-state primitive we adopt as PSAR's fourth expert is the triangular multiplicative update of AlphaFold (Jumper et al., 2021). Its core operation, a channel-wise contraction $\sum_k a_{ik} b_{kj}$ over a third node, is the matrix-multiplication layer of PPGN (Maron et al., 2019); NAR processors already compose over a third node through triplet messages (Ibarz et al., 2022; Jung & Ahn, 2023); the Edge Transformer (Bergen et al., 2021), evaluated on CLRS by Müller et al. (2024), fuses $(i, k)$ and $(k, j)$ pair values under triangular attention; and FloydNet keeps an all-pairs relational state and updates

each target pair through pivot candidates (Yu et al., 2026), evaluated under the CLRS-30 protocol, which supplies no hints at test time, so its scores are not comparable with the teacher-forced ones here. PSAR does not introduce this operation, and these processors were not run in our harness. TMU differs from the triplet messages of Triplet-GMPNN, a learned function of the three nodes, the three edges and graph features maximised over $k$, in multiplying the two legs and summing over $k$; we place it as one expert in a routed pool of row-wise attention algebras. It is also distinct from edge-conditioned softmax attention (Diao & Loynd, 2023), which keeps node-to-node $Q/K/V$ scoring, and from semiring-path GNNs such as NBFNet (Zhu et al., 2021) (§4.1).

**Attention algebras and the relational-bias axis.** Softmax (Vaswani et al., 2017) and $\alpha$-entmax (Peters et al., 2019; Correia et al., 2019) both produce normalized weighted aggregations, with $\alpha$-entmax additionally allowing adaptive sparse support. Tropical max-plus attention (Hashemi et al., 2025) instead aggregates with a max over sums; under sign reversal, this operation is dual to the classical $(\min, +)$ semiring used in shortest-path relaxation (Dudzik & Veličković, 2022).

In contrast to within-family aggregator selection (Corso et al., 2020; Jin et al., 2025), our setting concerns a discrete choice among substantially different aggregation primitives. Graph-aware transformers (Ying et al., 2021; Rampášek et al., 2022; Diao & Loynd, 2023) inject edge-biased pairwise structure within softmax attention, which is a design axis distinct from aggregation choice and can in principle compose with it.

We retain Relational Attention (`RelAttn`, port in Appendix F.5) as a strong cross-axis baseline (§6.1). We ran one cross-axis composition (PSAR-on-`RelAttn`) and report it as a negative result in Appendix F.6; it is not part of the empirical claims of this paper.

**Mixture-of-Experts and routing.** Per-token / per-head expert routing in MoE systems (Shazeer et al., 2017; Fedus et al., 2022; Jin et al., 2025) typically learns a load-balanced gate over experts *within* a fixed semiring. Table 4 (Appendix A) contrasts PSAR with the closest mixture / routing methods along five axes: pool homogeneity, routing granularity, teacher source, test-time routing and target domain. What is specific to PSAR is a pool whose experts span different semirings and include a non-attention pair-state primitive, combined by staged hard selection over separately pretrained experts. DARTFormer also assembles a heterogeneous pool, but its nine efficient-attention mechanisms stay within the $(+, \times)$ semiring and its architecture is fixed by search rather than selected at test time; hard top-1 or top-$k$ dispatch is shared with Switch Transformer, StableMoE, MoH and MoA. PSAR also carries an in-pool preference teacher, which we do not count as a differentia because removing it moves accuracy at $N{=}32$ by under $0.4$ pp (Appendix I.1).

## 3 Motivating Observations

Modern attention-based NAR processors commit to a single algebra at design time, yet the graph algorithms they emulate draw their inner step from a wider family of semirings (§2). We test two empirical questions before proposing any routing solution: whether this asymmetry matters at the task level, and whether an end-to-end per-head selector can recover the per-task best from task loss alone. We evaluate the three single-step algebras alongside a controlled per-head selector across seven selected CLRS-30 tasks (4 seeds, OOD at $N{=}32$; full tables in Appendix E). Three observations emerge. **(O1)** The per-task best single algebra is not constant: all three of softmax, $\alpha$-entmax, and tropical win on at least one task in the panel. **(O2)** A controlled end-to-end per-head algebra selector (`E2E`/`E2E-Div`, Straight-Through Gumbel-Softmax (Jang et al., 2017; Maddison et al., 2017)) under-performs the per-task best by $+16.9$ pp regret on Dijkstra and significantly trails on $4/7$ tasks (CIs exclude zero on Dij/MCO/MST-Prim/SCC; cross-task router JSD $0.08$–$0.09$, so its routing barely differs between tasks). On Bellman–Ford and BFS the regret CI includes zero, and on Floyd–Warshall the selector *outperforms* the best single algebra ($-5.9$ pp regret, CI excluding zero for `E2E`), so the collapse is task-specific. Adding a Switch-style load-balance loss and Gumbel temperature annealing does not rescue it on Dijkstra ($0.646 \pm 0.031$ against PSAR's $0.939$ under the same own-OOD-dev protocol; Appendix I.3). That selector differs from PSAR in its pool (no `TMU`), routing granularity, size ($0.22$M against $1.42$M parameters) and training schedule, while PSAR with every routing-supervision loss switched off still reaches $0.937$ on Dijkstra (Appendix I.1); the preference supervision is therefore not what avoids the collapse, and which of the other differences does is not isolated here. **(O3)** The per-sample oracle-winning algebra shifts with $N$ on Dij/BF/FW,

with consistent oracle-over-three-experts gains at large $N$ (Appendix Figure 3). These observations motivate pooling heterogeneous primitives and testing separately what the primitives and the routing among them each contribute (§4).

# 4 METHOD: PSAR ARCHITECTURE AND TRAINING

We adopt the Encode–Process–Decode backbone (Veličković et al., 2022) and replace only the processor's attention layer with PSAR (per-layer specialised aggregation routing); encoder and decoder remain the CLRS-native ones. The processor applies one layer per hint step, so per-layer routing is per-step routing.

## 4.1 EXPERT POOL: THREE ALGEBRA FAMILIES AND ONE STRUCTURAL PRIMITIVE

With $s_{ij} = q_i \cdot k_j/\sqrt{d}$ as the raw score over neighbors $j \in \mathcal{N}(i)$, the three single-step attention algebras are

$$o_i^{\mathrm{sm}} = \sum_j \mathrm{softmax}(s_{i,\cdot})_j \, v_j \qquad \text{softmax (Vaswani et al., 2017),} \tag{1}$$

$$o_i^{\mathrm{em}} = \sum_j \mathrm{entmax}_\alpha(s_{i,\cdot})_j \, v_j \qquad \alpha\text{-entmax (Peters et al., 2019),} \tag{2}$$

$$o_i^{\mathrm{tr}} = \max_j \left\{ q'_i \cdot k'_j + v_j \right\} \qquad \text{tropical (Hashemi et al., 2025).} \tag{3}$$

Softmax and $\alpha$-entmax both produce normalized weighted sums (entmax admits exact zeros; $\alpha$ is learnable per head, Appendix F.4); tropical instead uses $(\max, +)$ with an independent $(q', k')$ projection pair, so its aggregation is a max over sums, sign-dual to the classical $(\min, +)$ shortest-path recurrence (Dudzik & Veličković, 2022). These three differ in normalisation and aggregation operator, and we treat them as a discrete choice set rather than a soft mixture.

To complement the three single-step algebras with a non-attention structural primitive that maintains explicit pair representations, we add a fourth expert based on the triangular multiplicative update (TMU) (Jumper et al., 2021), ported from protein structure prediction to algorithmic reasoning. Given node embeddings $z \in \mathbb{R}^{N\times d}$ and edge features $e \in \mathbb{R}^{N\times N\times d_e}$, the TMU expert first builds an explicit pair state

$$z_{ij}^{(0)} = \mathrm{LN}\big(W_\ell z_i + W_r z_j + W_e e_{ij}\big) \in \mathbb{R}^{d_p}, \tag{4}$$

forms two gated paths $a_{ij} = \sigma(\mathrm{gate}_a(z_{ij}^{(0)})) \odot \mathrm{proj}_a(z_{ij}^{(0)})$ and $b_{ij} = \sigma(\mathrm{gate}_b(z_{ij}^{(0)})) \odot \mathrm{proj}_b(z_{ij}^{(0)})$, and updates the pair state through third-node triangular composition

$$\Delta z_{ij} = \tfrac{1}{\sqrt{N}} \textstyle\sum_k a_{ik} \odot b_{kj}, \qquad z_{ij}^{(1)} = \mathrm{LN}\big(z_{ij}^{(0)} + \sigma(\mathrm{tri_gate}(z_{ij}^{(0)})) \odot \mathrm{tri_out}(\Delta z_{ij})\big), \tag{5}$$

followed by a residual feed-forward *transition* block. The pair output is then contracted to a node update $o_i^{\mathrm{tmu}} = \mathrm{LN}\big(z_i + W_p \, \mathrm{mean}_j(z_{ij}^{(1)})\big)$. TMU uses neither $Q/K/V$ projections nor a softmax normalisation: third-node aggregation through the elementwise (Hadamard) product is the operative composition, and inter-node interaction is mediated by an explicit pair state rather than by attention over node values. This provides a transitive composition primitive that single-step row-wise algebras lack, and is mechanistically distinct from edge-conditioned softmax attention (Diao & Loynd, 2023) which retains node-to-node $Q/K/V$ scoring.

**Expert pool.** The four experts $\{\texttt{Softmax}, \texttt{Entmax}, \texttt{Tropical}, \texttt{TMU}\}$ form PSAR's native method pool: three single-step attention algebras over node-pair scores plus one pair-state structural primitive. We refer to this pool as the ***4-expert pool*** throughout.

## 4.2 ROUTING AMONG THE FOUR EXPERTS

Each of the four experts is first pre-trained as an independent single-expert processor and loaded as a *frozen* branch in PSAR. A lightweight MLP router takes the graph-pooled node embeddings $(\mathrm{mean}_i h_i^{(t)}, \max_i h_i^{(t)})$ concatenated with a Fourier encoding of hint step $t$ and small graph statistics,

producing per-step logits $\boldsymbol{\ell}^{(t)} \in \mathbb{R}^4$ and routing distribution $\mathbf{p}^{(t)} = \phi_{\text{act}}(\boldsymbol{\ell}^{(t)}/\tau_{\text{router}})$ (we use $\phi_{\text{act}} = \text{softmax}$ in the canonical configuration). The layer output combines experts by the shared per-step routing weights:

$$o_i^{(t)} = p_{\text{sm}}^{(t)} o_i^{\text{sm}} + p_{\text{em}}^{(t)} o_i^{\text{em}} + p_{\text{tr}}^{(t)} o_i^{\text{tr}} + p_{\text{tmu}}^{(t)} o_i^{\text{tmu}}. \tag{6}$$

At training time the router output is discretised by a Straight-Through argmax — the forward pass uses the one-hot top-expert selection while gradients flow through the soft distribution $\mathbf{p}^{(t)}$ — with routing temperature $\tau_{\text{router}}$ decayed from $1.0 \to 0.3$ over the final training stage to sharpen the soft distribution. *At test time the router uses a hard top-1 selection (one-hot) with no soft mixing across experts;* the soft mixture form in Equation (6) is used only as the differentiable training-time relaxation of this discrete inference rule, consistent with the discrete choice-set motivation of §4.1. A direct re-evaluation of the released checkpoints under soft inference is reported in Appendix I.5 (at $N$=32 hard routing is 5.8 and 9.1 pp higher on BF and FW and within seed std on Dij; on Dij at $N$=64 soft inference is 9.7 pp higher). The finer per-node routing variant $\boldsymbol{\ell}_i^{(t)} \in \mathbb{R}^4$ is examined as an ablation (Appendix I.3) and does not improve over per-step routing at matched training budget.

We use in-pool expert preferences as a routing regulariser during joint fine-tuning. The teacher is built online inside the training loop: at each hint step of every sample (training size $N$=16), the current hidden state of each in-pool expert branch is decoded through the *shared* hint decoder to give per-node, per-expert step losses $L_{t,i}^{(k)}$, from which we form the per-node teacher $q_i^{(t)} = \text{softmax}_k(-L_{t,i}^{(k)}/\tau_q)$ with $\tau_q = 0.5$; the teacher is detached, so no gradient flows through it. Since Stage 1 freezes the expert branches together with the encoder and decoder, the teacher coincides with a frozen-expert preference during that stage and co-evolves with the model from Stage 2a onward. The regulariser is $\mathcal{L}_{\text{KL}} = \frac{1}{NT}\sum_{i,t} \text{KL}(\mathbf{p}^{(t)} \,\|\, q_i^{(t)})$, with the graph-level routing distribution broadcast to every node. No OOD-size expert losses are used to construct the teacher, and at test time the router does not re-evaluate expert accuracy as $N$ grows; its routing policy is input-conditioned through the hidden state but regularised toward training-scale preferences. Switching off every routing-supervision loss ($\lambda_{\text{KL}}$=$\lambda_{\text{oracle}}$=$\lambda_z$=0) moves accuracy at $N$=32 by under 0.4 pp on each task (Appendix I.1), so these terms act as a regulariser rather than as the source of the gain; how the gain divides among the pool, expert pretraining and joint training is not resolved. We keep $\lambda_{\text{KL}}$=1 to match earlier checkpoints.

### 4.3 Training objective and three-stage schedule

The full training loss is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{KL}}\,\mathcal{L}_{\text{KL}} + \lambda_{\text{oracle}}\,\mathcal{L}_{\text{CE}}^{\text{oracle}} + \lambda_z\,\mathcal{L}_z, \tag{7}$$

where $\mathcal{L}_{\text{task}}$ is the standard CLRS step- and output-type loss aggregation (Veličković et al., 2022), $\mathcal{L}_{\text{KL}}$ is the preference distillation term of §4.2, $\mathcal{L}_{\text{CE}}^{\text{oracle}}$ is a cross-entropy between the router output and the per-step argmax of $-L_t^{(k)}$, and $\mathcal{L}_z$ is a log-partition stabilisation term that prevents router logits from collapsing. *Canonical configuration of the final* PSAR uses only the preference distillation term $\mathcal{L}_{\text{KL}}$ as the active routing-supervision channel: $\lambda_{\text{oracle}}$=0 and $\lambda_z$=0 in all three training stages, and the seven auxiliary regularisers implemented in the codebase (expert disentanglement, load-balance, cross-$N$ alignment, algebra-structure, abstention, phase smoothness, anchor) are deactivated ($\lambda_*$=0). The decomposition equation above lists the full set of routing-related supervision terms studied in the ablations of Appendix I; their independent contributions in the canonical configuration are zero by construction.

**Three-stage schedule.** Stage 1 (router only, frozen experts/enc/dec, $\lambda_{\text{KL}}$=1, 15 ep); Stage 2a (unfreeze enc/dec, experts frozen, $\lambda_{\text{KL}}$=0.3, 30 ep); Stage 2b (joint, $\lambda_{\text{KL}}$=0, anneal $\tau_{\text{router}}$:1.0→0.3, 70 ep). All parameter groups share a single learning rate (constant $3 \times 10^{-4}$ in Stages 1 and 2a and $10^{-4}$ in Stage 2b; Appendix F.3); the codebase's differential-LR option is left disabled in the canonical configuration. Stages 2a and 2b are subject to early stopping with patience 30; of the twelve main runs, eight complete the full 15+30+70 epoch budget and four stop earlier. Model selection uses OOD accuracy at $N$=32.

# 5 EXPERIMENTAL SETUP

**Tasks and protocol.** We evaluate PSAR on three canonical shortest-path tasks from CLRS-30 (Veličković et al., 2022): Dijkstra, Bellman–Ford, and Floyd–Warshall. They share shortest-path semantics but distinct inductive requirements (greedy / global / triadic relaxation), exposing three behaviours of PSAR: on Dij it trails the standalone `TMU` specialist, on BF it is not separable from `Softmax`, and on FW it is above every single expert. Models are trained on $N{=}16$ and evaluated on a held-out CLRS eval set (seed 88888, independent of train/val/test) over the CLRS-range grid $N \in \{16, 32, 48, 64\}$, which corresponds to the official CLRS-30 OOD test size of $N{=}64$ (Veličković et al., 2022) augmented with two intermediate points. Four training seeds per (model, task); we report 4-seed mean $\pm$ sample std. Model selection uses OOD accuracy at $N{=}32$.

**Baselines and reporting.** We organise baselines by design axis. Same-axis baselines, used in §6: single-algebra `Softmax`, `Entmax`, `Tropical`; width-varied `Softmax-140`/`Entmax-140` at $d_{\text{model}}{=}140$ (Table 6); end-to-end per-head selector `E2E`/`E2E-Div` (controlled negative, Observation O2). Cross-axis: `RelAttn` (Diao & Loynd, 2023) on the relational-bias axis (port in Appendix F.5), used in §6.1. One cross-axis composition (PSAR-on-`RelAttn`) was run and is reported as a negative result (Appendix F.6). Native-framework references ($N \in \{16, 32, 64\}$): Triplet-GMPNN (Ibarz et al., 2022), DNAR (Rodionov & Prokhorenkova, 2025), Open-Book NAR (Li et al., 2024). These are cited for broader CLRS-30 context only and are **not comparable** to the numbers reported here (see the evaluation contract below); we neither tabulate our numbers alongside them nor claim any improvement over them. Pairwise comparisons use paired-bootstrap 95% CI ($10^4$ resamples) and Cohen's $d$. In the accuracy tables, **bold marks entries where PSAR exceeds all three same-axis single algebras on the point estimate**; it is not a significance marker, and significance is reported separately as intervals. All same-axis baselines share $d_{\text{model}}{=}192$, $H{=}4$ heads, AdamW (Loshchilov & Hutter, 2019) with learning rate $3 \times 10^{-4}$ and weight decay $10^{-4}$, cosine-annealed learning rate, gradient clip $\ell_2{=}1.0$. Full hyperparameters and per-step inference cost are detailed in Appendices F and G.

**Evaluation contract.** Unless stated otherwise (§8, Appendix N), results are obtained under teacher forcing: the ground-truth hint is supplied at each step, and the output is decoded after the step that consumed $\text{hints}[T{-}2]$ (Appendix F.3); the reference systems above are evaluated without hints at test time. Copying that last fed hint to the output, with no processor, scores 1.000 on Dijkstra and 0.930–0.956 on Floyd–Warshall at $N \geq 32$, above every trained model here (Table 1); on Dijkstra at $N \geq 32$ the output equals the last fed hint on every held-out sample, so Dijkstra output accuracy measures retention. Two implementation choices deviate from the reference CLRS code: padded steps of short Dijkstra and Bellman–Ford trajectories enter the hint loss, and pair-level quantities are read out without a per-pair term (Appendix N). The accuracies therefore measure how much of a supplied trace a processor preserves and transforms beyond the training size; they order processors under one fixed contract, and when each model consumes its own hint predictions after the first step (self-rollout), every model falls below 0.19 on Dijkstra and Floyd–Warshall at $N \geq 32$.

# 6 WHAT COMES FROM THE PRIMITIVE

Table 1 compares, at $N{=}32$ on the shared held-out split, the three single-step attention algebras, the standalone pair-state expert `TMU`, two fixed-routing controls over the same four-expert pool and PSAR. No single expert is best on all three tasks (`TMU` on Dijkstra and Floyd–Warshall, `Softmax` on Bellman–Ford). Replacing them by `TMU` is the largest single change: it is 10.3 pp above the best attention algebra on Dijkstra and 13.0 pp above it on Floyd–Warshall, and 11.6 pp below `Softmax` on Bellman–Ford. PSAR matches or exceeds the strongest attention algebra on every task (+7.0 and +14.9 pp on Dijkstra and Floyd–Warshall; +2.1 pp on Bellman–Ford, with a paired interval that includes zero, Appendix H.2), falls below standalone `TMU` on Dijkstra and 1.9 pp above it on Floyd–Warshall, so 13.0 of its 14.9 pp over attention on Floyd–Warshall are present in `TMU` alone. The fixed-routing controls are weak: `StaticTask` learns four task-level logits that stay within $[-0.38, +0.50]$ of their zero initialisation, so its gate selects a seed-dependent expert, and `Uniform` ($p_k{=}1/4$) selects `Softmax` at every step through the hard tie-break. The informative routing controls are in §7.

| Method | Dijkstra | Bellman–Ford | Floyd–Warshall |
|---|---|---|---|
| *Single-step attention algebra baselines* | | | |
| `Softmax` | $0.778 \pm 0.068$ | $0.881 \pm 0.036$ | $0.203 \pm 0.020$ |
| `Entmax` | $0.854 \pm 0.024$ | $0.873 \pm 0.018$ | $0.238 \pm 0.020$ |
| `Tropical` | $0.850 \pm 0.049$ | $0.723 \pm 0.133$ | $0.199 \pm 0.054$ |
| *Pair-state expert (standalone)* | | | |
| `TMU` | $0.957 \pm 0.018$ | $0.765 \pm 0.166$ | $0.368 \pm 0.018$ |
| *Four-expert pool with fixed routing (controls)* | | | |
| `StaticTask` (learned task-constant gate) | $0.832 \pm 0.038$ | $0.681 \pm 0.250$ | $0.169 \pm 0.105$ |
| `Uniform` ($p_k$=1/4; hard tie-break ⇒ always `Softmax`) | $0.781 \pm 0.180$ | $0.857 \pm 0.041$ | $0.117 \pm 0.036$ |
| *Proposed* | | | |
| **PSAR** | $\mathbf{0.924 \pm 0.027}$ (+7.0) | $\mathbf{0.902 \pm 0.041}$ (+2.1) | $\mathbf{0.387 \pm 0.013}$ (+14.9) |
| *No processor (reference under the teacher-forced contract)* | | | |
| Copy last fed hint $\text{hints}[T-2]$ | 1.000 | 0.812 | 0.930 |

Table 1: PSAR versus single-step attention algebra baselines, a single-expert pair-state primitive ablation (`TMU`), and fixed-routing controls over the same four-expert pool at $N$=32 on the shared held-out split (eval seed 88888; §5), 4-seed mean ± sample std. The copy row emits the last hint fed to the processor as the output; it is not a trained model and is excluded from bold, and on Bellman–Ford it is lowered by padded steps that follow the informative hint (Appendix N). Parenthesised deltas in the PSAR row are pp gains over the best single-algebra attention baseline (`Softmax`/`Entmax`/`Tropical`) per column. Bold marks gains over the three attention algebras only: the standalone `TMU` expert exceeds PSAR on Dijkstra, and most of the Floyd–Warshall gain over attention is already present in standalone `TMU`. Appendix H.2 reports seed-level paired-bootstrap intervals for the contrast against the best single algebra and extends PSAR to 8 seeds: the mean moves by at most 1.1 pp on any task, and the interval excludes zero on Dijkstra and Floyd–Warshall but *not* on Bellman–Ford.

### 6.1 Length extrapolation (CLRS-range $N$-scaling)

Figure 1 plots the CLRS-range grid (train $N$=16, official OOD size $N$=64; per-$N$ numbers in Appendix H). Over the best attention algebra, PSAR is $-0.3$, $+7.0$, $+13.1$ and $+8.2$ pp on Dijkstra at $N$=16, 32, 48 and 64, so the gain appears only beyond the training size, while standalone `TMU` is higher at every $N$; on Bellman–Ford it grows from $+0.2$ to $+7.1$ pp over `Softmax`; on Floyd–Warshall it is $+14.9$ pp at $N$=32 and $+7.0$ pp at $N$=64, and it stays above standalone `TMU` at the selection-free sizes ($+10.3$ and $+8.8$ pp at $N$=48 and 64). The cross-axis `RelAttn` baseline retains a lead on Floyd–Warshall at every $N$, which we discuss next.

**Relational bias as an orthogonal mechanism.** `RelAttn` (Diao & Loynd, 2023) conditions queries, keys and values on edge features, a relational bias outside the native pool. It is 3.2 and 4.8 pp above PSAR on Floyd–Warshall at $N$=32 and 64 and 30–32 pp below it on Dijkstra (Appendix K); the one composition we ran, PSAR-on-`RelAttn`, degrades the same-axis result (Appendix F.6), and a within-family `GraphBias` intensity scan does not close the Floyd–Warshall gap (Appendix L). We treat it as a boundary of the native pool rather than as a comparator we claim to beat.

## 7 What the Routed Pool Adds beyond the Pair-State Expert

**Floyd–Warshall.** Each control in Table 2 targets one factor, though none isolates it exactly (`TMU-1.4M`, for example, removes both the heterogeneous pool and the routing). A parameter-matched single `TMU` and PSAR's own step schedule retrained as a fixed routing rule are below PSAR on every seed at both sizes. A routing-free learned mixture of the four experts is below it at $N$=48 and not separable at $N$=64, and a router that sees only the step index is not separable at four seeds and is separable at $N$=48 only with eight. A same-recipe pool of four `TMU` copies is 3.9 pp $[+2.1, +5.6]$ below PSAR, and a separately trained always-`TMU` policy is 5.15 pp $[+3.53, +7.12]$ below it (own OOD-dev at $N$=32; Appendices I.4 and J.4). Floyd–Warshall is therefore the clearest regime in which the jointly trained routed heterogeneous system exceeds both a widened single primitive and a retrained fixed step schedule under the teacher-forced contract; the step-only and routing-free controls bound how much of that advantage can be attributed specifically to input-conditioned routing.

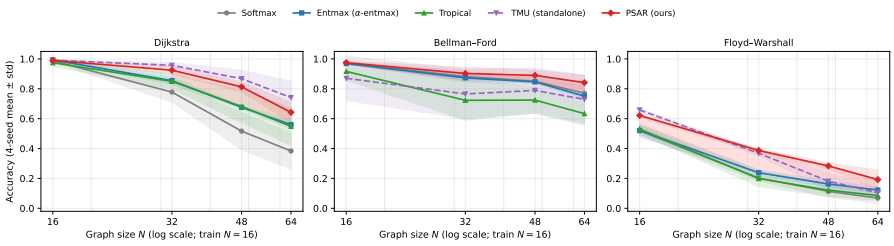



Figure 1: Three-task CLRS-range $N$-scaling at $N \in \{16, 32, 48, 64\}$ (train $N$=16, 4 seeds, shaded $\pm$std; shared held-out split, CLRS eval seed 88888), comparing PSAR against the three attention algebras and standalone `TMU` (dashed) on (**left**) Dij, (**middle**) BF, (**right**) FW; red shading marks where PSAR exceeds the best attention algebra. The cross-axis `RelAttn` baseline is discussed at the end of §6.

| Control | primary factor tested | $N$=48 | $N$=64 |
|---|---|---|---|
| `TMU-1.4M` | heterogeneous system vs. one widened primitive | $+7.16$ $[+3.14, +11.07]$ | $+8.09$ $[+5.47, +10.45]$ |
| `TrainedSchedule` | input-conditioned vs. fixed step-indexed selection | $+16.07$ $[+11.21, +20.81]$ | $+12.58$ $[+8.16, +16.96]$ |
| `SoftMix` | hard adaptive selection vs. static convex mixing | $+3.41$ $[+1.77, +5.13]$ | $-0.24$ $[-3.38, +2.91]$ |
| `StepOnly`, 4 seeds | router inputs beyond the step index | $+4.46$ $[-0.54, +11.40]$ | $+2.49$ $[-5.09, +10.08]$ |
| `StepOnly`, 8 seeds | router inputs beyond the step index | $+4.05$ $[+0.79, +7.85]$ | $+2.95$ $[-2.06, +7.89]$ |

Table 2: PSAR minus each control on Floyd–Warshall (pp, 95% hierarchical paired-bootstrap interval; shared held-out split, teacher forcing, the two sizes not used for checkpoint selection). Each control primarily probes one factor, but the comparisons are not strict one-factor interventions. `TMU-1.4M` has four seeds, `TrainedSchedule` and `SoftMix` eight; the first two are below PSAR on every seed. The eight-seed `StepOnly` row was run after the four-seed intervals had been seen (Appendix J.4).

**Dijkstra and Bellman–Ford.** Routing adds little on the other two tasks. On Dijkstra the router selects `TMU` on 96% of steps averaged over seeds and on every step on three of four seeds; a separately trained always-`TMU` policy is not separable from PSAR ($+1.3$ pp $[-2.6, +5.2]$), a ten-bin step schedule read off the router reproduces it, and standalone `TMU` stays above both. On Bellman–Ford, where `TMU` alone is weak, PSAR is 20.7 pp $[+4.2, +37.1]$ above a pool of four `TMU` copies but not separable from `Softmax`; its seeds concentrate on different experts, so the seed-averaged routing mass (63% `TMU`) reflects disagreement between seeds rather than a within-seed blend (Appendix M).

**Primitive mismatch across seven tasks.** Under one protocol (each run's own best OOD-dev accuracy at $N$=32, which is also the selection split; four seeds; Table 20), the pool's largest shortfall to the per-task best single expert is 6.6 pp, against 17.7–35.2 pp for any single expert used on every task. It exceeds every single expert on Floyd–Warshall and BFS, is not separable from the best one on Bellman–Ford and SCC, and trails a specialist on Dijkstra, MST-Prim and Matrix-Chain-Order (Table 22). The pool reduces worst-case primitive mismatch rather than dominating specialists; its processor block has 3.6–5.6 times the parameters of a single-expert block (Table 9).

## 8 Processor Rankings Depend on the Hint Contract

Every result above is teacher-forced (§5). Splitting pointer targets by whether they differ from the last supplied hint (a split chosen after the results were seen; Appendix N) shows where the margins lie. On Floyd–Warshall PSAR's margin over the parameter-matched `TMU` at $N$=48 and 64 lies on output entries that already equal that hint ($+7.5$ and $+8.4$ pp) rather than on the 4–5% that the last step changes ($+0.2$ and $+1.0$ pp, intervals containing zero). On Dijkstra at $N \geq 32$ the output equals that hint on every entry, so output accuracy measures retention; on the changed entries of the per-step predictions PSAR is above standalone `TMU` ($+11.7$ pp $[+3.9, +19.4]$ and $+11.5$ pp $[+5.3, +17.5]$), so its Dijkstra deficit is a deficit in retention.

Table 3 asks whether the Floyd–Warshall ordering against the parameter-matched expert survives a change of contract. Changing only the evaluation, by running the teacher-forced checkpoints

| Harness | trained and evaluated with | $N$=48 | $N$=64 |
|---|---|---|---|
| main | ground-truth hints | $+7.16$ $[+3.14, +11.07]$ | $+8.09$ $[+5.47, +10.45]$ |
| main | own predictions | $-5.84$ $[-8.83, -2.83]$ | $-3.44$ $[-6.17, -0.24]$ |
| reference-form read-out | ground-truth hints | $+7.95$ $[+1.55, +14.56]$ | $+7.15$ $[+2.09, +13.56]$ |
| reference-form read-out | own predictions | $-5.39$ $[-7.58, -3.14]$ | $-5.68$ $[-7.60, -3.13]$ |

Table 3: PSAR minus the parameter-matched TMU on Floyd–Warshall (pp, 95% hierarchical paired-bootstrap interval, four seeds per arm). Main harness: the ground-truth row uses the models of Table 1; the own-predictions row retrains both models under that contract, with experts pretrained and checkpoints selected under it. Reference-form harness: an edge read-out with a per-edge term and per-sample length masking, new held-out instances, and checkpoints selected on autonomous OOD-dev accuracy as preregistered; under teacher-forced selection the own-prediction intervals include zero (Table 38).

on their own predictions, leaves PSAR ahead ($+4.2$ and $+2.7$ pp; Appendix N). Retraining both models under that contract reverses the ordering, and the reversal recurs in a second harness with a reference-form edge read-out and per-sample length masking on new instances; each comparison and its decision rule were fixed before the runs. The teacher-forced advantage is therefore a property of the models together with the contract they are trained under, not an intrinsic ordering of routed heterogeneous computation over a single wider primitive.

## 9 Limitations

The main evaluation covers three shortest-path tasks of CLRS-30; non-graph algorithm families, evaluation beyond teacher-forced CLRS, and language-model backbones such as TransNAR (Bounsi et al., 2024) on CLRS-Text (Markeeva et al., 2026) are untested, and we do not contest the benchmark-wide state of the art on CLRS-30.

The headline accuracies are teacher-forced; under self-rollout every processor here falls below $0.29$ at $N \geq 32$, and the ordering of §8 depends on the contract. The padded hint steps on Dijkstra and Bellman–Ford and the restricted pair read-out apply to every model in the main harness. No attention baseline is matched to PSAR's parameter count (Appendix C); in the reference-form harness a parameter-matched TMU is $11.9$ and $21.0$ pp above PSAR on Dijkstra at $N$=48 and $64$ (ground-truth hints, teacher-forced selection; Appendix N), and Triplet-GMPNN, the Edge Transformer and FloydNet were not run in our harness. The implementation executes all four experts at every step and keeps the selected output; top-1 selection would permit running only the selected expert, which we do not exploit (31 ms per step against $0.55$ ms for `Softmax` at $N$=32; Appendix G).

## 10 Conclusion

We introduced PSAR, which pools three attention algebras and the triangular multiplicative update (TMU; Jumper et al., 2021) and selects one expert per hint step with a hard router trained in stages. Under teacher forcing on three CLRS-30 shortest-path tasks, no single expert is best on all three, and PSAR matches or exceeds the strongest attention algebra on every task while falling below standalone TMU on Dijkstra. On Floyd–Warshall, most of the observed gain over attention is already present in the pair-state expert: standalone TMU is $13.0$ pp above the strongest attention algebra at $N$=32. The jointly trained routed pool additionally exceeds a parameter-matched TMU and a retrained fixed schedule on Floyd–Warshall under teacher forcing, although a step-only router at four seeds and a routing-free mixture at $N$=64 are not separable from it. When PSAR and the parameter-matched TMU are retrained, checkpoint-selected, and evaluated on their own predictions, the TMU leads, while evaluating the teacher-forced checkpoints on their own predictions leaves PSAR ahead. For shortest-path NAR, the results identify pair-state structure as a useful primitive under the tested configurations and show that the ordering between processors can depend on the hint contract under which they are trained and compared.

### Reproducibility statement

The canonical configuration, data splits, hint contract, router inputs, optimiser settings and bootstrap procedure are specified in Appendix F.3; model and training hyperparameters are in Appendices F.1 and F.2, compute in Appendix G, and checkpoint provenance and evaluation seeds in Appendix O. The evaluation contract and its deviations from the reference CLRS implementation are described in §5 and Appendix N. Code for all models, the training entry points, the job scripts and the analysis scripts behind the tables is provided as supplementary material.

### AI use statement

Large language models were used to assist in editing the grammar of this paper. The authors prompted the model with text from the paper and received suggestions for rewording, grammar, and clarity. The authors reviewed and edited the suggestions before incorporating them into the final text. The use of AI was limited to language assistance; all scientific content, analysis, and conclusions were generated by the authors.

## References

Leon Bergen, Timothy O' Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1390–1402. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/0a4dc6dae338c9cb08947c07581f77a2-Paper.pdf.

Wilfried Bounsi, Borja Ibarz, Andrew Dudzik, Jessica B Hamrick, Larisa Markeeva, Alex Vitvitskyi, Razvan Pascanu, and Petar Veličković. Transformers meet neural algorithmic reasoners. *arXiv preprint arXiv:2406.09308*, 2024.

Jason Ross Brown, Yiren Zhao, Ilia Shumailov, and Robert D Mullins. DARTFormer: Finding the best type of attention. *arXiv preprint arXiv:2210.00641*, 2022.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2174–2184, 2019.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in neural information processing systems*, 33: 13260–13271, 2020.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

Cameron Diao and Ricky Loynd. Relational attention: Generalizing transformers for graph-structured tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cFuMmbWiN6.

Andrew J Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. *Advances in neural information processing systems*, 35:20635–20647, 2022.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Esmail Gumaan. Mixture of attention schemes (MoAS): Learning to route between MHA, GQA, and MQA. *arXiv preprint arXiv:2512.20650*, 2025.

Baran Hashemi, Kurt Pasque, Chris Teska, and Ruriko Yoshida. Tropical attention: Neural algorithmic reasoning for combinatorial algorithms. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 98631–98656. Curran Associates, Inc., 2025. doi: 10.52202/085713-3295. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8e0be7875ffe7dcfea07e35c79e1a923-Paper-Conference.pdf.

Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Joseph Dudzik, Matko Bošnjak, Alex Vitvitskyi, Yulia Rubanova, Andreea Deac, Beatrice Bevilacqua, Yaroslav Ganin, Charles Blundell, and Petar Veličković. A generalist neural algorithmic learner. In *Learning on graphs conference*. PMLR, 2022.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng YAN. Moh: Multi-head attention as mixture-of-head attention. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=eYtgs9k75o.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Yeonjoon Jung and Sungsoo Ahn. Triplet edge attention for algorithmic reasoning. *arXiv preprint arXiv:2312.05611*, 2023.

Hefei Li, Chao Peng, Chenyang Xu, and Zhengfeng Yang. Open-book neural algorithmic reasoning. *Advances in Neural Information Processing Systems*, 37:9819–9836, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Larisa Markeeva, Sean Michael McLeish, Borja Ibarz, Wilfried Bounsi, Olga Kozlova, Alex Vitvitskyi, Charles Blundell, Tom Goldstein, Avi Schwarzschild, and Petar Veličković. The CLRS-text algorithmic reasoning language benchmark. *Journal of Data-centric Machine Learning Research*, 2026. ISSN XXXX-XXXX. URL https://openreview.net/forum?id=1sZ8RsVEoX.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Luis Müller, Daniel Kusuma, Blai Bonet, and Christopher Morris. Towards principled graph transformers. *Advances in Neural Information Processing Systems*, 37:126767–126801, 2024.

Ben Peters, Vlad Niculae, and André FT Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, 2019.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in neural information processing systems*, 35:14501–14515, 2022.

Gleb Rodionov and Liudmila Prokhorenkova. Discrete neural algorithmic reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Inrv8EXylW.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7), 2021.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS algorithmic reasoning benchmark. In *International Conference on Machine Learning*, pp. 22084–22102. PMLR, 2022.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkwZSG-CZ.

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? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Jingcheng Yu, Mingliang Zeng, and Qiwei Ye. FloydNet: A learning paradigm for global relational reasoning. *arXiv preprint arXiv:2601.19094*, 2026.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token. *arXiv preprint arXiv:2210.05144*, 2022.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural Bellman-Ford networks: A general graph neural network framework for link prediction. *Advances in neural information processing systems*, 34:29476–29490, 2021.

## A Nearest mixture and routing methods

Table 4 contrasts PSAR with the nearest mixture and routing methods along the five design axes of §2.

| Method | Pool homogeneity | Granularity | Teacher | Inference | Domain | Reference |
|---|---|---|---|---|---|---|
| Switch Transformer | homogeneous (FFN) | per-token | load balance | hard top-1 | NLP | (Fedus et al., 2022) |
| Mixture-of-Softmaxes | homogeneous (softmax) | per-token | none | soft | NLP | (Yang et al., 2018) |
| MoH | homogeneous (softmax heads) | per-token | load balance | hard top-$k$ | NLP | (Jin et al., 2025) |
| MoA | homogeneous (attn variants) | per-token (heads) | load balance | hard top-$k$ | NLP | (Zhang et al., 2022) |
| MoAS | homogeneous (softmax variants) | per-token | load balance | soft | NLP | (Gumaan, 2025) |
| DARTFormer | heterogeneous (9 efficient attn) | search-time | none (DARTS) | fixed after search | NLP | (Brown et al., 2022) |
| StableMoE | homogeneous (FFN) | per-token | router-distillation | hard top-1 | NLP | (Dai et al., 2022) |
| MHTA (Tropical) | homogeneous (tropical heads) | n/a (no routing) | none | concat | NAR | (Hashemi et al., 2025) |
| PNA | homogeneous (aggregators) | n/a (no routing) | none | sum-concat | GNN | (Corso et al., 2020) |
| **PSAR (ours)** | **heterogeneous (3 algebras + TMU)** | per-layer | online in-pool expert prefs (auxiliary) | **hard top-1** | NAR (CLRS-30 path) | this work |

Table 4: Mixture / routing methods nearest to PSAR, contrasted along five design axes. PSAR's differentiation is the combination of (i) experts spanning different semirings, including a non-attention pair-state primitive (TMU), and (ii) staged hard per-step selection over separately pretrained experts. The in-pool preference teacher is listed for completeness: switching off every routing-supervision loss moves accuracy at $N{=}32$ by under $0.4$ pp (Appendix I.1). Switch Transformer / MoH / MoA / MoAS route inside a fixed softmax algebra; DARTFormer searches over nine efficient-attention mechanisms with DARTS and then fixes the resulting architecture, so its pool is heterogeneous but stays within one semiring and it performs no test-time routing; MHTA uses heads of a single tropical attention; PNA concatenates fixed aggregators without routing. Hard selection at inference is not a differentiating axis — Switch and StableMoE dispatch hard top-1, MoH / MoA select hard top-$k$ — and is listed only for completeness. We do not claim novelty along any single axis (heterogeneous expert pools, oracle distillation, and hard routing each have prior instances), but the combination has not, to our knowledge, been studied for attention-algebra selection in NAR.

| Task (Ch.) | Inner aggregation | Semiring view | Hint trajectory |
|---|---|---|---|
| BFS (§22.2) | layer-wise OR over neighbors | Boolean $(\vee, \wedge)$ | step-variable, $\leq N$ |
| **Dijkstra (§24.3)** | **per-node priority relaxation** | **tropical** $(\min, +)$ | **step-synchronous,** $N-1$ |
| **Bellman–Ford (§24.1)** | **global edge relaxation** | **tropical** $(\min, +)$ | **step-synchronous,** $N-1$ |
| **Floyd–Warshall (§25.2)** | **all-pairs DP update** | **tropical** $(\min, +)$ **on pair grid** | **step-synchronous,** $N$ |
| MST-Prim (§23.2) | priority-frontier extraction | tropical $(\min, +)$ on cut | step-synchronous, $N-1$ |
| SCC (§22.5) | DFS-based post-order decomposition | sequential | step-variable |
| Matrix-Chain-Order (§15.2) | interval dynamic programming | sum-product / minimisation | step-variable |

Table 5: Seven selected CLRS-30 tasks surveyed in §3; main method evaluation focuses on the three canonical shortest-path tasks (Dijkstra, Bellman–Ford, Floyd–Warshall; bold rows). Chapter references are to (Cormen et al., 2009). Inner aggregation and semiring view describe the dominant arithmetic of each algorithm's per-step update. Hint trajectory length governs the recurrent depth $T$ of the processor.

## B CLRS-30 TASK DESCRIPTIONS

Table 5 summarizes the seven selected CLRS-30 tasks included in our initial survey (§3); bold rows mark the three canonical shortest-path tasks (Dijkstra, Bellman–Ford, Floyd–Warshall) used for main method evaluation (§5). For each we record (i) the algorithm's inner aggregation pattern, (ii) the semiring that this pattern corresponds to if interpreted as repeated tensor operations, and (iii) whether the algorithm's ground-truth hint trajectory is step-synchronous or step-variable with respect to node count $N$. We follow the CLRS-30 evaluation protocol (Veličković et al., 2022) for all data generation and ground-truth hints. String, sorting, geometry, and divide-and-conquer task families are outside our scope.

## C WIDTH-VARIED SINGLE-ALGEBRA BASELINES

We additionally train single-algebra baselines (`Softmax-140`, `Entmax-140`) at $d_{\text{model}}$=140, a width calibrated against an earlier, smaller PSAR configuration ($\approx 2.2 \times 10^5$ parameters). The canonical PSAR processor reported throughout this paper is substantially larger ($1.42 \times 10^6$ parameters), so these runs vary width *within* the single-algebra family rather than equalising parameter count against PSAR. Every entry in Table 6, PSAR included, is the run's own best OOD-dev accuracy at $N$=32, the protocol of Appendix I.4; it is not the shared held-out split of Table 1, which is why PSAR reads 0.939 rather than 0.924 on Dijkstra. Under this protocol, on Dijkstra `Entmax-140` (0.804) is 13.5 pp below PSAR (0.939); on Floyd–Warshall PSAR (0.394) leads the stronger of the two (0.224) by 17.0 pp; on Bellman–Ford PSAR (0.923) is 1.9 pp above the stronger `Softmax-140` (0.904), consistent with BF being a single-algebra-friendly regime. Because the parameter budgets differ by a factor of six to seven, we do not read this comparison as separating the routing contribution from raw capacity.

**Where $d_{\text{model}}$=140 comes from.** The width was calibrated so that a single-algebra processor would match the parameter count of an earlier, smaller PSAR configuration: target 215,833, achieved 218,397 at $d_{\text{model}}$=140, within 1.2%, holding head count, FFN ratio, and depth fixed. The realised runs land at 219,660–219,664 parameters on Dijkstra, 218,818–218,822 on Bellman–Ford and 201,261–201,265 on Floyd–Warshall (`Softmax-140`–`Entmax-140`; the spec differs by task). The canonical PSAR configuration reported in the main body is 1,423,794 parameters — roughly six to seven times larger — because the expert pool and router grew after this calibration was performed, and the calibration was not repeated. We therefore retain `Softmax-140` and `Entmax-140` as a width-varied single-algebra control and do not use them to separate the routing contribution from the capacity contribution.

## D EMPIRICAL OBSERVATION FIGURES

The two figures supporting Observations O2 and O3 are reproduced here.

Table 6: Width-varied single-algebra baselines at $N$=32 (4 seeds, mean $\pm$ sample std; every entry, PSAR included, is the run's own best OOD-dev accuracy over training). `Softmax-140` and `Entmax-140` are single-algebra processors at $d_{\text{model}}$=140, i.e. 2.0–2.2 $\times$ $10^5$ parameters, well below the canonical PSAR's $1.42 \times 10^6$, so this table varies width within the single-algebra family and does *not* equalise parameter count with PSAR.

| Task | `Softmax-140` | `Entmax-140` | PSAR |
|---|---|---|---|
| Dijkstra | $0.704 \pm 0.100$ | $0.804 \pm 0.062$ | $0.939 \pm 0.018$ |
| Bellman–Ford | $0.904 \pm 0.032$ | $0.885 \pm 0.044$ | $0.923 \pm 0.049$ |
| Floyd–Warshall | $0.192 \pm 0.016$ | $0.224 \pm 0.022$ | $0.394 \pm 0.010$ |

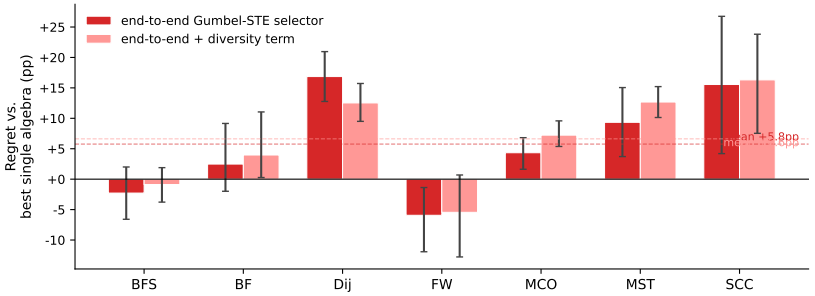



Figure 2: Per-task regret of the end-to-end per-head algebra selector at $N$=32, OOD-dev, 4-seed mean with paired-bootstrap 95% CI ($10^4$ resamples). Positive regret indicates the selector trails the per-task best single-algebra baseline. Dashed horizontal lines mark the cross-task mean regret. This survey-era baseline is used only as motivation for selecting among algebras; it does not include the final PSAR model, TMU, or RelAttn.

# E Detailed empirical observation tables

The numbers underlying Observations O1 and O2 are tabulated below. Both tables report 4-seed mean OOD-dev accuracy at $N$=32 under the protocol of §5; pairwise-bootstrap CIs use $10^4$ resamples over seed pairs.

# F Implementation details

## F.1 Model hyperparameters

All same-axis baseline configurations share the following settings. Node and edge encoders are 2-layer MLPs with hidden dimension 128 mapping input features to the processor dimension $d_{\text{model}} = 192$. The processor uses $H = 4$ attention heads with per-head dimension 48, and runs for $T$ recurrent steps equal to the task's maximum hint trajectory length (up to $N$ for most path algorithms). Dropout is 0.0; layer normalization is pre-norm. The decoder is a 2-layer MLP per output head, tied to the CLRS output types (pointer, mask, categorical, scalar). The three single-step attention branches of the end-to-end selector `E2E`/`E2E-Div` (Observation O2) and of PSAR's algebra experts (§4) share (i) a per-head learnable graph-distance decay $\lambda_h$ initialized to 0.9 (Appendix F.4). The graph distance $d_G(i, j)$ is the hop distance on the input adjacency matrix, computed per sample by breadth-first search and capped at 8 (unreachable pairs at the cap); it uses no hint or target information and (ii) an independent $(Q', K')$ projection pair for the tropical branch with per-head dimension 32. PSAR's TMU expert uses pair dimension $d_p$ and a feed-forward transition multiplier separately calibrated for capacity matching (§4.2); it shares neither $\lambda_h$ nor $(Q', K')$. Routing-mechanism-specific parameters of PSAR are documented together with the routing specification noted in §4.2.

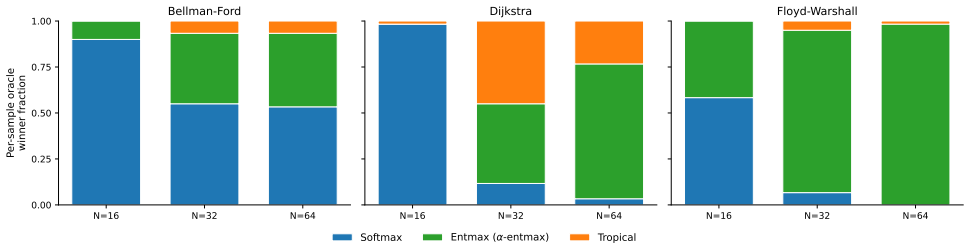



Figure 3: Per-sample oracle-winner composition across $N$ for three tasks. Each bar shows the fraction of 60 held-out samples won by softmax, $\alpha$-entmax, and tropical under the per-task best four trained single-algebra checkpoints. This survey-era draw uses 60 samples at the same evaluation seed and the survey checkpoints; the oracle statistics of Appendix M use the 64-sample split and the common-width checkpoints, so the two are not the same measurement. This three-algebra survey is used only as motivation for selecting among algebras; it does not include the final PSAR model, TMU, or RelAttn.

| Task | `Softmax` | `Entmax` | `Tropical` | Best | Best vs. second |
|---|---|---|---|---|---|
| BFS | 0.791 | 0.819 | **0.824** | `Tropical` | +0.005 (n.s., $d$=0.03) |
| Bellman–Ford | **0.901** | 0.885 | 0.762 | `Softmax` | +0.017 (sig., $d$=0.34) |
| Dijkstra | 0.644 | 0.761 | **0.798** | `Tropical` | +0.037 (n.s., $d$=0.51) |
| Floyd–Warshall | 0.214 | **0.229** | 0.189 | `Entmax` | +0.015 (n.s., $d$=0.47) |
| Matrix-Chain-Order | 0.571 | 0.572 | **0.750** | `Tropical` | +0.179 (sig., $d$=11.5) |
| MST-Prim | 0.732 | 0.741 | **0.751** | `Tropical` | +0.011 (n.s., $d$=0.36) |
| SCC | 0.898 | **0.912** | 0.889 | `Entmax` | +0.014 (n.s., $d$=0.20) |

Table 7: Best single-algebra processor by per-task OOD-dev accuracy at $N$=32. **Provenance note:** this survey table differs from the main results in *two* respects, and both must be held in mind when comparing them. First, it reports each run's own best OOD-dev accuracy over training, whereas Tables 1, 11 and 32 re-score checkpoints on the shared held-out split, which exists only for the three canonical tasks. Second, the single-algebra checkpoints here come from the original seven-task survey sweep, not from the later common-$d_{\text{model}}$=192 sweep used for the main results; on Dijkstra the two sweeps differ by more than ten points for the same algebra (`Entmax` 0.761 here vs 0.890 under the common sweep). The consequence is that per-task winners are only comparable within this table: it ranks Dijkstra as `Tropical`-best, while the main-results provenance ranks it `Entmax`-best. This table supports Observation O1 — that no single algebra wins the whole panel — and should not be used to read off absolute accuracies. Bold marks the per-task winner. "Best vs. second" reports the absolute gap to the runner-up with paired-bootstrap statistical significance and Cohen's $d$.

### F.2 Training hyperparameters

Single-expert and baseline models are trained with AdamW (Loshchilov & Hutter, 2019) at learning rate $3 \times 10^{-4}$, weight decay $10^{-4}$, $\beta = (0.9, 0.999)$, and $\epsilon = 10^{-8}$ (the latter two are the optimiser defaults); the three-stage PSAR schedule, with constant learning rates, is given in §4.3 and Appendix F.3. The learning rate follows cosine annealing over the epoch budget; there is no warmup phase. Batch size is 32 instances. The training budget is 200 epochs with early stopping at patience 30, between roughly 3 minutes and 1.1 GPU-hours per (model, task, seed) cell on a single NVIDIA A100 GPU, depending on the expert and task (Appendix G). Gradients are clipped at $\ell_2$ norm 1.0. For the end-to-end per-head selector `E2E`/`E2E-Div` (Observation O2), the Straight-Through Gumbel-Softmax temperature is fixed at $\tau = 1.0$ without annealing; the optional diversity regularizer coefficient (`E2E-Div`) is $\lambda_d = 10^{-3}$.

### F.3 Canonical configuration and evaluation contract

This subsection specifies the canonical PSAR run of Table 1 at the level needed to re-implement it; every statement is read off the code.

| Task | E2E regret (pp) | E2E 95% CI | E2E-Div regret (pp) | E2E-Div 95% CI |
|---|---|---|---|---|
| BFS | $-2.3$ | $[-6.6, +2.0]$ | $-0.9$ | $[-3.8, +1.9]$ |
| Bellman–Ford | $+2.5$ | $[-2.0, +9.2]$ | $+4.0$ | $[+0.3, +11.0]$ |
| Dijkstra | $+16.9$ | $[+12.8, +21.0]$ | $+12.5$ | $[+9.5, +15.7]$ |
| Floyd–Warshall | $-5.9$ | $[-11.9, -1.4]$ | $-5.4$ | $[-12.8, +0.7]$ |
| Matrix-Chain-Order | $+4.3$ | $[+1.6, +6.8]$ | $+7.2$ | $[+5.4, +9.6]$ |
| MST-Prim | $+9.3$ | $[+3.7, +15.1]$ | $+12.7$ | $[+10.1, +15.2]$ |
| SCC | $+15.6$ | $[+4.2, +26.8]$ | $+16.3$ | $[+7.5, +23.8]$ |
| Mean regret | $+5.8$ | – | $+6.6$ | – |
| Win count | $2/7$ | – | $2/7$ | – |

Table 8: Per-task regret of the end-to-end per-head selector relative to the per-task best single-algebra baseline at $N$=32, point-estimate 4-seed mean and paired-bootstrap 95% CI. Supports Observation O2 and Figure 2.

**Data and splits.** Instances come from the CLRS-30 samplers (`dm-clrs` 2.0.3). Each training seed $\sigma$ draws 1,000 training graphs at $N$=16 (sampler seed $\sigma$), 64 in-distribution validation graphs at $N$=16 ($\sigma$+1000), 64 OOD-dev graphs at $N$=32 ($\sigma$+2000) and 64 CLRS test graphs at $N$=64 ($\sigma$+3000); the shared held-out split draws 64 graphs per $N$ at sampler seed 88888. An epoch is one pass over the training graphs in batches of 32.

**Hint contract.** For a hint trajectory of length $T$ the processor is applied $T-1$ times. At step $t \in \{0, \dots, T-2\}$ the ground-truth hint $\mathrm{hints}[t]$ is encoded and added to the encoded inputs, the step's hidden state is decoded into a prediction of $\mathrm{hints}[t+1]$, and the task output is decoded from the hidden state after the last step, whose input contained $\mathrm{hints}[T-2]$. Evaluation uses the same contract, which is what teacher forcing means here. $T$ is the longest trajectory in the split; shorter trajectories are zero-padded and the forward pass does not truncate them, so their final steps consume padding hints; Dijkstra and Bellman–Ford trajectories end early on some graphs, while Floyd–Warshall trajectories all have the same length. $\mathcal{L}_{\mathrm{task}}$ is the output loss plus 0.5 times the mean of the $T-1$ hint losses; accuracy is the fraction of correctly predicted pointer entries of the output.

**Expert inputs.** Every expert receives the same node features (encoded inputs plus the encoded current hint), edge features, hop-distance matrix and hidden state entering the step, and projects the concatenation of node features and hidden state to $d_{\mathrm{model}}$ before its own computation.

**Router.** With $s_t = t/(T-2)$ the step fraction, the router input at step $t$ concatenates (i) the mean- and max-pooled hidden state entering the step ($2d_{\mathrm{model}} = 384$); (ii) four graph statistics through a LayerNorm: the mean and the maximum over all node pairs of the hop distance (capped at 8, unreachable pairs at the cap), and the mean and the maximum over nodes of the number of nodes at hop distance one; (iii) a step encoding of $s_t$: $\sin(2^k \pi s_t)$ and $\cos(2^k \pi s_t)$ for $k \in \{0, 1, 2, 3\}$ and a one-hot of $\lfloor 3 s_t \rfloor$ clipped to $\{0, 1, 2\}$ (11 dimensions); and (iv) three adjacency features through a LayerNorm: edge density and the mean in- and out-degree divided by $N$. The 402-dimensional input passes through $\mathrm{Linear}(402, 384)$, LayerNorm, GELU and $\mathrm{Linear}(384, 4)$, i.e. 157,074 router parameters. `StepOnly` (Appendix J.4) keeps only (iii), so its router has 6,916 parameters; the difference, 150,158, equals the difference between the two full models' parameter counts (1,388,547 − 1,238,389). The logits are divided by $\tau_{\mathrm{router}}$ and mapped through a softmax to $\mathbf{p}^{(t)}$; the forward pass uses the one-hot argmax and the backward pass the Straight-Through form $\mathrm{stopgrad}(\mathbf{1}_{\arg\max} - \mathbf{p}^{(t)}) + \mathbf{p}^{(t)}$. $\tau_{\mathrm{router}}$ is 1.0 in Stages 1 and 2a and decays linearly per epoch from 1.0 to 0.3 over Stage 2b.

**Stages and optimiser.** A fresh AdamW optimiser is built at each stage over the parameters trainable in that stage, with constant learning rate $3 \times 10^{-4}$ in Stages 1 and 2a and $10^{-4}$ in Stage 2b, weight decay $10^{-4}$ and gradient clipping at $\ell_2$ norm 1.0. OOD-dev accuracy at $N$=32 is evaluated after every epoch of every stage; the reported checkpoint is the epoch with the highest value across all stages, and early stopping with patience 30 on that value is active in Stages 2a and 2b and ends training.

**Hierarchical paired bootstrap.** For contrasts on the shared held-out split, each of the $10^4$ replicates draws as many seed pairs as the comparison has (four, or eight where stated) with replacement and one set of 64 instance indices with replacement, applied to every drawn seed and to both arms of each pair, and records the mean over drawn seeds of the mean per-instance paired difference; the interval is the 2.5th and 97.5th percentile of these replicates. Every hierarchical interval in this paper uses this procedure.

### F.4 Entmax and Gumbel-Softmax numerics

We compute $\alpha$-entmax with the bisection algorithm of Peters et al. (2019) (50 iterations). In the entmax expert, $\alpha$ is a learnable per-head parameter $\alpha_h = 1 + \sigma(a_h) \in (1, 2)$, initialised at $a_h = 0.5$, following the adaptive-$\alpha$ formulation of Correia et al. (2019); the canonical router activation $\phi_{\text{act}}$ is a softmax, and the $\text{entmax}_{1.5}$ router activation appears only in the early iteration ablation (Appendix I.3). For the end-to-end per-head selector `E2E`/`E2E-Div` (Observation O2), the Straight-Through Gumbel-Softmax estimator (Jang et al., 2017; Maddison et al., 2017) uses one-hot argmax in the forward pass and the softmax Jacobian in the backward pass at $\tau = 1.0$ without annealing.

### F.5 Relational Attention port: JAX → PyTorch

Our port of Relational Attention (Diao & Loynd, 2023) re-implements the attention layer of the authors' 229-line JAX module in PyTorch and plugs it into our CLRS encoder, decoder, and training harness in place of the processor. The attention arithmetic follows the reference: node and edge projections are summed into edge-conditioned queries, keys, and values ($Q_{ij} = W_Q h_i + W_{eQ} e_{ij}$, and analogously $K_{ij}$ with $h_j$ and $V_{ij}$ with $h_j$), scored by scaled dot products with a softmax over $j$, followed by an output projection, a post-norm residual, and a feed-forward block. The port departs from the reference CLRS configuration in three respects: it omits the global graph-feature branch (the reference default concatenates graph features onto node and edge inputs), its feed-forward hidden width is $4\,d_{\text{model}}$ rather than the reference default of 32, and it disables the per-layer edge-update MLP that the reference enables by default. We therefore use `RelAttn` only as a same-framework comparator under our training and evaluation protocol, and make no claim that the port reproduces the published Relational Attention numbers.

### F.6 Cross-axis composition: PSAR-on-RelAttn (negative result)

We ran the cross-axis composition PSAR-on-`RelAttn` — replacing PSAR's same-axis four-expert pool with a `RelAttn`-augmented backbone underneath the same per-layer hard-routing architecture — on the three canonical shortest-path tasks (4 seeds, $N$=32 OOD-dev, model-selection scale). The composition does *not* compose additively with same-axis routing (all values in this paragraph are each run's own best OOD-dev accuracy, so the plain `RelAttn` values differ from its held-out numbers in Table 32): it loses to plain `RelAttn` on Bellman–Ford ($\Delta = -5.1$ pp, $0.877 \pm 0.063$ vs `RelAttn` $0.927 \pm 0.039$) and on Floyd–Warshall ($\Delta = -14.2$ pp, $0.285 \pm 0.131$ vs `RelAttn` $0.426 \pm 0.007$; FW seed s4 collapses to 0.099, inflating std). Only Dijkstra gains over plain `RelAttn` ($\Delta = +18.4$ pp, $0.833 \pm 0.089$ vs `RelAttn` $0.649 \pm 0.084$); however, plain PSAR (native-pool) already exceeds plain `RelAttn` on Dij by $+29.0$ pp under the same protocol (0.939 vs 0.649; $+30.4$ pp on the held-out split of Table 32), so substituting the backbone strictly degrades the same-axis champion. We report this as a negative result for the tested composition: it did not transplant the native-pool router onto an edge-conditioned `RelAttn` substrate and did not close the FW gap to plain `RelAttn` (§6.1).

## G Compute cost breakdown

This appendix reports two cost layers for PSAR: (i) measured per-step inference latency for the architectural building blocks at $d_{\text{model}}$=192, $H$=4, batch 8 on a single CUDA device, and (ii) the end-to-end pipeline budget that aggregates expert pretrain and router/joint training cells.

The three single-step attention branches (`Softmax`, `Entmax`, `Tropical`) run in parallel for every input at the same $d_{\text{model}}$; the TMU branch is reported separately because its asymptotic regime ($N^3$)

Table 9: Measured per-step inference latency (mean over 30 repeats after 5 warmups) at $d_{\text{model}}$=192, $H$=4, batch 8. Params count processor blocks only; full canonical PSAR models have 1.39–1.42M parameters (Table 18). The implementation executes all four experts at every step and keeps the selected output, so hard routing does not reduce computation. Per-block costs do not sum to the PSAR total because (a) PSAR runs the four experts plus the router head without sharing intermediate activations, and (b) the `Entmax` bisection is the dominant per-step cost at small $N$ ($\sim$ 23 ms regardless of $N$). PSAR parameters are roughly 5.6$\times$ a single-algebra block.

| Block | Params | $N$=32 ms | $N$=64 ms |
|---|---|---|---|
| `Softmax` | 0.22M | 0.55 | 0.53 |
| `Entmax` | 0.22M | 23.47 | 24.14 |
| `Tropical` | 0.30M | 0.63 | 0.87 |
| `TMU` | 0.34M | 1.05 | 2.67 |
| `RelAttn` | 0.63M | 0.74 | 1.91 |
| **PSAR** | **1.24M** | **31.08** | **34.18** |

Table 10: End-to-end training pipeline budget, measured as Slurm wall-clock per job on one NVIDIA A100 GPU (mean over the listed cells, range in parentheses). PSAR-equivalent total per (task, seed) cell, summing Stage 0 over the four experts and Stage 1+2a+2b: $\approx 1.1 + 0.6 \approx 1.7$ GPU-hr, against 0.14–0.58 GPU-hr for one single-algebra baseline cell, i.e. an end-to-end ratio of roughly 3–12$\times$ ($\approx 6\times$ against the mean single-algebra cell). The routing teacher is constructed online inside the training loop (§4.3), so its cost is already contained in the Stage 1+2a+2b row rather than forming a separate precomputation stage. The TMU branch's $O(N^3 d_p)$ pair update is the component whose latency grows fastest with $N$ (Table 9), although at the measured sizes the entmax bisection still dominates.

| Stage | Component | Per-cell GPU-hr, mean (range) | Cells (task $\times$ seed $\times$ expert) |
|---|---|---|---|
| 0 | Single-expert pretrain: `Softmax` / `Entmax` / `Tropical` / `TMU` | 0.14 / 0.58 / 0.19 / 0.18 (0.06–1.13) | $3 \times 4 \times 4 = 48$ |
| 1+2a+2b | Router and joint training (15+30+70 ep) | 0.62 (0.34–0.90) | $3 \times 4 = 12$ |
| Width-varied controls (`Softmax-140` / `Entmax-140`, $d_{\text{model}}$=140) | | 0.13 / 0.56 (0.05–1.08) | $3 \times 4 \times 2 = 24$ |
| Cross-axis port (`RelAttn`) | | 0.12 (0.05–0.25) | $3 \times 4 = 12$ |

differs. For PSAR, measured latency stays under 50 ms per step at every size we profiled (31.1, 34.2 and 41.5 ms at $N$=32, 64 and 128). Floating-point operations for one complete forward pass of PSAR (all $T-1$ steps, a batch of 8 Dijkstra graphs, counting the operators for which the PyTorch profiler reports FLOPs) are $1.2 \times 10^{11}$, $8.8 \times 10^{11}$ and $7.0 \times 10^{12}$ at $N$=32, 64 and 128.

## H FULL RESULTS: PER-$N$ ACCURACY ACROSS THE THREE TASKS

Table 11 reports the 4-seed mean $\pm$ std for the three canonical shortest-path tasks across the CLRS-range grid $\{16, 32, 48, 64\}$, covering `Softmax`, `Entmax` ($\alpha$-entmax), `Tropical`, the `TMU` pair-state expert (single-expert ablation), the cross-axis `RelAttn` baseline, and PSAR (PSAR hard routing). These columns underlie Figure 1, Table 1, and the scope analysis in §6.1.

### H.1 EXTENSION TO FOUR ADDITIONAL CLRS TASKS (PARTIAL COVERAGE)

The main results focus on the three canonical shortest-path tasks (Dij/BF/FW). To probe coverage breadth, we extend the canonical PSAR configuration — identical architecture, expert pool, and three-stage schedule — to four additional CLRS-30 tasks: BFS (`bfs`), Prim's MST (`mst_prim`), matrix-chain order (`matrix_chain_order`), and strongly connected components (`strongly_connected_components`). All 4 seeds $\times$ 4 tasks completed within budget.

### H.2 SEED EXPANSION TO 8 SEEDS

The headline gain in Table 1 is reported on 4 seeds. To check seed-tail sensitivity on the canonical PSAR configuration, we extended training to 8 seeds on all three tasks (s5–s8 added to s1–s4) and evaluated the four new checkpoints under exactly the protocol of Table 1. The single-algebra

| Task | Config | $N$=16 | 32 | 48 | 64 |
|---|---|---|---|---|---|
| Dij | `Softmax` | $0.987 \pm 0.007$ | $0.778 \pm 0.068$ | $0.516 \pm 0.125$ | $0.384 \pm 0.120$ |
| Dij | `Entmax` | $0.993 \pm 0.003$ | $0.854 \pm 0.024$ | $0.676 \pm 0.010$ | $0.560 \pm 0.029$ |
| Dij | `Tropical` | $0.974 \pm 0.026$ | $0.850 \pm 0.049$ | $0.681 \pm 0.108$ | $0.548 \pm 0.126$ |
| Dij | `TMU` | $0.991 \pm 0.003$ | $0.957 \pm 0.018$ | $0.867 \pm 0.057$ | $0.741 \pm 0.109$ |
| Dij | `RelAttn` | $0.974 \pm 0.020$ | $0.621 \pm 0.082$ | $0.414 \pm 0.101$ | $0.326 \pm 0.105$ |
| Dij | PSAR | $0.990 \pm 0.008$ | $\mathbf{0.924 \pm 0.027}$ | $\mathbf{0.813 \pm 0.033}$ | $\mathbf{0.642 \pm 0.068}$ |
| BF | `Softmax` | $0.972 \pm 0.008$ | $0.881 \pm 0.036$ | $0.850 \pm 0.023$ | $0.771 \pm 0.052$ |
| BF | `Entmax` | $0.969 \pm 0.003$ | $0.873 \pm 0.018$ | $0.846 \pm 0.009$ | $0.751 \pm 0.016$ |
| BF | `Tropical` | $0.916 \pm 0.059$ | $0.723 \pm 0.133$ | $0.724 \pm 0.089$ | $0.633 \pm 0.077$ |
| BF | `TMU` | $0.870 \pm 0.149$ | $0.765 \pm 0.166$ | $0.789 \pm 0.149$ | $0.729 \pm 0.162$ |
| BF | `RelAttn` | $0.971 \pm 0.003$ | $0.897 \pm 0.041$ | $0.893 \pm 0.020$ | $0.833 \pm 0.040$ |
| BF | PSAR | $\mathbf{0.975 \pm 0.010}$ | $\mathbf{0.902 \pm 0.041}$ | $\mathbf{0.890 \pm 0.036}$ | $\mathbf{0.841 \pm 0.048}$ |
| FW | `Softmax` | $0.519 \pm 0.030$ | $0.203 \pm 0.020$ | $0.114 \pm 0.038$ | $0.068 \pm 0.036$ |
| FW | `Entmax` | $0.521 \pm 0.043$ | $0.238 \pm 0.020$ | $0.163 \pm 0.020$ | $0.122 \pm 0.020$ |
| FW | `Tropical` | $0.530 \pm 0.030$ | $0.199 \pm 0.054$ | $0.122 \pm 0.043$ | $0.085 \pm 0.031$ |
| FW | `TMU` | $0.658 \pm 0.010$ | $0.368 \pm 0.018$ | $0.181 \pm 0.057$ | $0.104 \pm 0.030$ |
| FW | `RelAttn` | $0.688 \pm 0.015$ | $0.419 \pm 0.010$ | $0.306 \pm 0.007$ | $0.241 \pm 0.007$ |
| FW | PSAR | $\mathbf{0.621 \pm 0.013}$ | $\mathbf{0.387 \pm 0.013}$ | $\mathbf{0.284 \pm 0.021}$ | $\mathbf{0.192 \pm 0.063}$ |

Table 11: Per-$N$ OOD accuracy for the three canonical shortest-path tasks: 4-seed mean $\pm$ sample std across the CLRS-range grid $\{16, 32, 48, 64\}$. `TMU` is reported as a native-pool single-expert ablation and `RelAttn` as a cross-axis comparator (§6.1); neither is used for bold determination. Bold highlights entries where PSAR exceeds all three same-axis single algebras (`Softmax`, `Entmax`, `Tropical`). Differences quoted in the text are computed from unrounded means and can differ by 0.1 pp from differences of the rounded entries (e.g. +13.1 pp on Dijkstra at $N$=48).

| Task | best_ood_dev mean $\pm$ std | best_test mean $\pm$ std |
|---|---|---|
| `bfs` | $0.919 \pm 0.108$ | $0.659 \pm 0.126$ |
| `mst_prim` | $0.951 \pm 0.024$ | $0.783 \pm 0.098$ |
| `matrix_chain_order` | $0.723 \pm 0.098$ | $0.498 \pm 0.176$ |
| `strongly_connected_components` | $0.924 \pm 0.046$ | $0.273 \pm 0.206$ |

Table 12: PSAR on four additional CLRS-30 tasks (4 seeds, OOD-dev at $N$=32 and held-out test; all models are trained at $N$=16). The architecture trains and evaluates without modification on all four tasks. We report this as a coverage check rather than a positive method-quality claim: per-task best-native and `RelAttn` comparators on these four tasks are not run within this paper's budget, so the same-axis $\Delta$ used for the three canonical tasks is unavailable here. We report the raw numbers to demonstrate (a) the four-expert pool transfers without architectural change to non-shortest-path tasks (`bfs`, `strongly_connected_components`) and to a non-graph dynamic-programming task (`matrix_chain_order`), and (b) the seed std remains comparable to the canonical three tasks on 3/4 tasks (`bfs` is the high-variance outlier, consistent with its known seed sensitivity in the original CLRS-30 benchmark). This raw-accuracy view is superseded by Appendix J, which reports the same tasks against every single expert with per-seed values, paired contrasts and checkpoint provenance; it is retained here only because the held-out test column is not duplicated there.

baselines were trained at 4 seeds only, so an 8-seed *paired* contrast against them does not exist; the $n$=8 column therefore reports PSAR's own seed stability, and the paired interval stays at $n$=4.

# I Ablations

## I.1 Loss-component decomposition: $\lambda_{\mathrm{KL}}$, $\mathcal{L}_{\mathrm{CE}}^{\mathrm{oracle}}$, $\mathcal{L}_z$

We decompose the routing-supervision contribution by training the canonical PSAR architecture under three loss-term configurations, holding architecture / schedule / seed pool / hyperparameters fixed and varying only the active routing-supervision losses (4 seeds $\times$ 3 tasks $\times$ 3 cells = 36 runs):

- *CE only*: only oracle cross-entropy active ($\lambda_{\mathrm{KL}}$=0, $\lambda_{\mathrm{oracle}}$=1, $\lambda_z$=0).
- $\mathcal{L}_z$ *only*: only the log-partition stabiliser active ($\lambda_{\mathrm{KL}}$=0, $\lambda_{\mathrm{oracle}}$=0, $\lambda_z$=0.1).
- *All off*: *all* explicit routing-supervision losses off ($\lambda_{\mathrm{KL}}$=$\lambda_{\mathrm{oracle}}$=$\lambda_z$=0).

| Task | PSAR $n$=4 | PSAR $n$=8 | shift | best single algebra ($n$=4) | paired $\Delta$ ($n$=4), 95% CI |
|---|---|---|---|---|---|
| Dijkstra | $0.924 \pm 0.027$ | $0.935 \pm 0.035$ | $+1.07$ pp | `Entmax` $0.854 \pm 0.024$ | $+6.99$ pp, $[+2.88, +11.11]$ |
| Bellman–Ford | $0.902 \pm 0.041$ | $0.893 \pm 0.041$ | $-0.91$ pp | `Softmax` $0.881 \pm 0.036$ | $+2.11$ pp, $[-0.04, +4.05]$ |
| Floyd–Warshall | $0.387 \pm 0.013$ | $0.391 \pm 0.014$ | $+0.35$ pp | `Entmax` $0.238 \pm 0.020$ | $+14.94$ pp, $[+13.06, +17.89]$ |

Table 13: Seed sensitivity of PSAR at $N$=32. Every entry is measured on the shared held-out split (CLRS eval seed 88888, 64 samples, hard routing) — the same protocol as Table 1, so the $n$=4 column reproduces that table exactly. Extending PSAR from 4 to 8 seeds moves the mean by at most 1.1 pp on any task ($+1.07$ Dij, $-0.91$ BF, $+0.35$ FW), and the direction is not uniform: Bellman–Ford moves *down*. The paired interval is a bootstrap over the four matched seed-level differences ($10^4$ resamples), which is deliberately conservative in that it propagates between-seed variance rather than pooling instances. Under it the gain over the best single algebra excludes zero on Dijkstra and Floyd–Warshall but **not on Bellman–Ford**, where the interval is $[-0.04, +4.05]$ and one of the four seeds (s2, $-0.98$ pp) is negative. This is consistent with the reading of Bellman–Ford throughout this paper as the single-algebra-friendly regime in which PSAR matches rather than beats the dominant primitive.

Table 14: Loss-component decomposition at $N$=32 OOD-dev (4-seed mean $\pm$ sample std). Every entry, *including the canonical row*, is the run's own best OOD-dev accuracy, so all differences below are within-protocol; this is why the canonical row here reads 0.939 rather than the 0.924 of Table 1, which is measured on the shared held-out split (Appendix O). Each cell varies only the active routing-supervision losses; everything else matches the canonical configuration. A paired bootstrap over the four matched seeds ($10^4$ resamples) gives, against canonical: all off $-0.17$ pp Dij $[-3.47, +2.34]$, $-0.39$ pp BF $[-1.07, +0.29]$, $-0.32$ pp FW $[-1.90, +0.60]$; CE only $-2.00$ pp Dij $[-11.72, +3.54]$, $+0.18$ pp BF $[-2.44, +2.81]$, $-0.03$ pp FW $[-1.06, +1.14]$; $\mathcal{L}_z$ only $+1.40$ pp Dij $[+0.18, +2.47]$, $-1.78$ pp BF $[-4.57, +1.00]$, $+0.17$ pp FW $[-0.79, +1.13]$. Eight of the nine intervals contain zero; the one exception is $\mathcal{L}_z$ only on Dijkstra, where replacing $\lambda_{\mathrm{KL}}$ with the log-partition stabiliser is *better* than the canonical configuration. Turning all explicit routing-supervision losses off moves accuracy by less than 0.4 pp on every task. We therefore conclude that the routing-supervision losses are not the measured driver of the headline gain; the gain remains after they are removed, and how it divides among the pool, expert pretraining and joint training is not resolved. Appendix I.4 separately tests whether that pool must be heterogeneous. The KL term is active only in Stages 1 and 2a ($\lambda_{\mathrm{KL}}$=1 then 0.3) and is off in Stage 2b, where model selection lands; it can still shape the reported model through the parameters it produces before Stage 2b, which is why the ablation, not the Stage-2b coefficient, is the relevant evidence. We retain $\lambda_{\mathrm{KL}}$=1 in the canonical configuration to match earlier reproducibility checkpoints, not because it is optimal.

| Cell | Dijkstra | Bellman–Ford | Floyd–Warshall |
|---|---|---|---|
| CE only | $0.919 \pm 0.107$ | $0.925 \pm 0.023$ | $0.394 \pm 0.014$ |
| $\mathcal{L}_z$ only | $0.953 \pm 0.027$ | $0.906 \pm 0.032$ | $0.396 \pm 0.013$ |
| All routing losses off | $0.937 \pm 0.047$ | $0.919 \pm 0.045$ | $0.391 \pm 0.016$ |
| **PSAR canonical** | $\mathbf{0.939 \pm 0.018}$ | $\mathbf{0.923 \pm 0.049}$ | $\mathbf{0.394 \pm 0.010}$ |

**Adding the KL term and adding `TMU`.** Two further contrasts use the earlier three-algebra pool at the same width, on the shared held-out split at $N$=32 with the hierarchical bootstrap of Appendix F.3 (Table 15). Adding the KL term to that pool, with architecture, parameter count, seeds, schedule and learning rates fixed, changes accuracy by amounts whose intervals contain zero on all three tasks. Adding `TMU` to the routed pool, with the KL term active in both arms, raises accuracy by 13.3 pp on Floyd–Warshall and 3.1 pp on Bellman–Ford, with intervals above zero; on Dijkstra the interval contains zero. The second contrast also changes the pool size, the router width and the competition among experts, so it measures extending the pool rather than `TMU` in isolation.

### I.2 Teacher temperature $\tau_q$ sensitivity

We vary the teacher softmax temperature $\tau_q$ across $\{0.1, 0.3, 1.0, 2.0\}$ on all three canonical shortest-path tasks (4 seeds, $N$=32 OOD-dev; 48 runs total). Aggregate accuracy is reported in Table 16:

Table 15: Adding the KL term to the three-algebra pool, and adding TMU to the routed pool ($N$=32, shared held-out split, four matched seeds; pp with 95% hierarchical bootstrap interval).

| Contrast | Dijkstra | Bellman–Ford | Floyd–Warshall |
|---|---|---|---|
| +KL (three algebras) | $+3.9$ $[-4.5, +12.2]$ | $-1.4$ $[-4.6, +1.7]$ | $-1.0$ $[-2.2, +0.2]$ |
| +TMU (KL in both arms) | $+8.0$ $[-2.3, +19.2]$ | $+3.1$ $[+0.6, +5.7]$ | $+13.3$ $[+11.7, +14.7]$ |

| Task | $\tau_q$=0.1 | $\tau_q$=0.3 | $\tau_q$=1.0 | $\tau_q$=2.0 |
|---|---|---|---|---|
| Dijkstra | $0.907 \pm 0.028$ | $0.866 \pm 0.048$ | $0.896 \pm 0.034$ | $0.910 \pm 0.042$ |
| Bellman–Ford | $0.782 \pm 0.189$ | $0.866 \pm 0.053$ | $0.882 \pm 0.044$ | $0.881 \pm 0.049$ |
| Floyd–Warshall | $0.194 \pm 0.052$ | $0.247 \pm 0.063$ | $0.238 \pm 0.061$ | $0.247 \pm 0.064$ |

Table 16: Teacher-temperature $\tau_q$ sensitivity at $N$=32 OOD-dev (4-seed mean $\pm$ sample std, 48 runs). **Scale caveat:** this sweep was run on the earlier 486,443-parameter PSAR version, not the canonical 1,423,794-parameter configuration. It shows that the moderate-$\tau_q$ band is not sharply tuned *at that scale*; it is not a measurement of the canonical model's $\tau_q$ sensitivity.

The moderate band $\tau_q \in \{0.3, 1.0, 2.0\}$ is stable on all three tasks: per-task mean spreads across this band are small (Dij 0.044, BF 0.016, FW 0.009). The only stability flag (seed std $> 0.1$) is BF at $\tau_q$=0.1, where seed s3 collapses to 0.512 vs. s1/s2 $\approx 0.91$. The canonical configuration uses $\tau_q = 0.5$, which is between the 0.3 and 1.0 grid points; this value is not directly measured in the sweep, so the canonical-configuration stability rests on interpolation between neighbouring grid points at a smaller model scale rather than on a direct observation.

### I.3 Router-design variants of an earlier version

We report router-design variants of the earlier three-algebra version of PSAR that preceded the canonical configuration, all trained under matched hyperparameters (4 seeds, $N$=32, each run's own OOD-dev split; Table 17). ***KL router*** is the initial preference-distilled router (softmax router activation, per-step routing, $\lambda_{\mathrm{KL}} = 1/0.3/0.1$ over the three stages, plus a small Stage-2b disentanglement penalty $\lambda_{\mathrm{disent}}$=0.05 that all four PSAR columns share); ***KL + extras*** adds to it adjacency-statistic router features (graph density and mean in/out-degree), a warm-started router, a load-balance term, a re-weighting of the hint loss against the output loss in Stages 2a/2b ($\alpha_{\text{hint-output}}$=0.7/0.3) and a differential learning rate (router $\times 2$, experts $\times 0.1$), and is included as a cautionary negative; ***KL + CE + $\mathcal{L}_z$*** instead adds to the KL router an oracle cross-entropy term $\mathcal{L}_{\mathrm{CE}}^{\mathrm{oracle}}$, a log-partition regulariser $\mathcal{L}_z$, a load-balance term and the $\mathrm{entmax}_{1.5}$ router activation; ***Per-node*** keeps the KL + CE + $\mathcal{L}_z$ recipe and replaces per-hint-step routing with per-node routing logits $(B, N, 3)$.

### I.4 Is the pool required to be heterogeneous?

The four-expert pool could help simply by being a pool — four branches and a router, regardless of what is inside them. We test this directly by replacing the heterogeneous pool with four copies of the *same* expert. We choose TMU for the duplicated branch because it is the strongest single expert on two of the three tasks. Everything else — router, three-stage schedule, seeds, evaluation — is held at the canonical configuration, and each of the four branches is initialised from the same Stage-0 TMU checkpoint. Because a TMU branch carries more parameters than an attention branch, the homogeneous pool ends up 18–19% *larger* than PSAR. The experts stay frozen through Stages 1 and 2a, so until Stage 2b the four copies compute the same function and neither the task loss nor the teacher distinguishes them; the control therefore does not test independently initialised or trained homogeneous experts, and a $4\times$Softmax pool on Bellman–Ford was not run.

### I.5 Hard vs. soft routing at inference

The canonical PSAR configuration uses hard top-1 argmax routing at inference and in the training forward pass (§4.2); the soft distribution carries the gradient. We re-evaluated the same PSAR checkpoints under *soft* routing at inference (per-step softmax-weighted convex combination of the

Table 17: Architectural iteration ablation and Fair-E2E control at $N{=}32$ on each run's own OOD-dev split (4 seeds, mean $\pm$ std). The four PSAR columns vary the active loss terms, router inputs and routing granularity around the earlier three-algebra pool. **Scale caveat:** these four columns come from the earlier three-algebra PSAR version with roughly $4.9$–$5.2 \times 10^5$ parameters ($486{,}443$ for the KL router and KL + CE + $\mathcal{L}_z$, $487{,}217$ for KL + extras and $519{,}211$ for Per-node, Dijkstra configuration), not the canonical $1{,}423{,}794$-parameter configuration of the main body. They are an internally matched comparison *among variants at that scale*, and the *KL router* column is not a re-measurement of the canonical configuration — its Dijkstra mean $0.902$ is neither the canonical $0.924$ of Table 1 nor its $0.939$ under the own-run protocol. The canonical-scale loss-component decomposition is Appendix I.1. *KL + extras* regresses all three tasks and was abandoned. *Per-node* routing is $4.9$ pp below KL + CE + $\mathcal{L}_z$ on Dijkstra, $0.9$ pp above it on Bellman–Ford and $0.6$ pp below it on Floyd–Warshall, the latter two within one seed std; we retain per-step routing as the canonical configuration. The teacher-granularity axis (per-node teacher distribution vs per-step broadcast) is held fixed at the canonical configuration (per-node teacher) rather than separately ablated; *Per-node* is the closest available proxy under this paper's compute budget, and isolating the teacher-granularity axis is left for follow-up. **Fair E2E** (rightmost column; $216{,}988$ parameters, i.e. smaller again than the four PSAR variant columns) augments the plain end-to-end selector (Observation O2) with a Switch-Transformer-style load-balance auxiliary loss and Gumbel-Softmax temperature annealing, supplying the "fair" supervision missing from the original `E2E` baseline; on Dijkstra it still trails the canonical PSAR by $29.3$ pp under the same own-OOD-dev protocol ($0.646$ vs $0.939$), so load balancing and annealing do not rescue the collapse. BF s4 is a 22-pp outlier ($0.71$ vs. $0.89$–$0.96$ on s1–s3) that inflates the BF std.

| Task | KL router | KL + extras | KL + CE + $\mathcal{L}_z$ | Per-node | Fair E2E |
|---|---|---|---|---|---|
| Dijkstra | $0.902 \pm 0.038$ | $0.766 \pm 0.125$ | $\mathbf{0.909 \pm 0.021}$ | $0.860 \pm 0.061$ | $0.646 \pm 0.031$ |
| Bellman–Ford | $0.868 \pm 0.052$ | $0.759 \pm 0.097$ | $0.881 \pm 0.037$ | $\mathbf{0.890 \pm 0.028}$ | $0.876 \pm 0.117$ |
| Floyd–Warshall | $0.246 \pm 0.038$ | $0.151 \pm 0.081$ | $\mathbf{0.255 \pm 0.046}$ | $0.249 \pm 0.015$ | $0.311 \pm 0.024$ |

Table 18: Heterogeneous vs. homogeneous four-expert pool (4 seeds, $N{=}32$, each run's own OOD-dev split; paired bootstrap over the four matched seeds, $10^4$ resamples). Both arms are our own models measured under one protocol, but that protocol is *not* the shared held-out split of Table 1, so these numbers are not directly comparable to it. The answer is task-dependent. On Bellman–Ford and Floyd–Warshall the heterogeneous pool is ahead with the interval excluding zero, decisively so on Bellman–Ford, where the homogeneous pool collapses on two of four seeds ($0.639$, $0.407$). On Dijkstra the interval includes zero and the homogeneous pool is nominally ahead. The pattern tracks how strong the duplicated expert is on each task: the standalone `TMU` specialist scores $0.974$ on Dijkstra, $0.801$ on Bellman–Ford and $0.380$ on Floyd–Warshall, so on Dijkstra four copies of `TMU` are four copies of an expert that is already the right answer, whereas on Bellman–Ford they are four copies of a weak one and the pool has no alternative to fall back on. We therefore read heterogeneity as mattering relative to a `TMU`-only pool where `TMU` alone is insufficient (Bellman–Ford, and Floyd–Warshall by a smaller margin), and *not* on Dijkstra.

| Task | PSAR (heterogeneous) | 4×`TMU` (homogeneous) | params (het. / hom.) | paired $\Delta$ | 95% CI |
|---|---|---|---|---|---|
| Dijkstra | $0.939 \pm 0.018$ | $0.943 \pm 0.036$ | 1,423,794 / 1,691,030 | $-0.44$ pp | $[-2.21, +2.04]$ |
| Bellman–Ford | $0.923 \pm 0.049$ | $0.717 \pm 0.244$ | 1,422,640 / 1,684,494 | $+20.65$ pp | $[+4.17, +37.13]$ |
| Floyd–Warshall | $0.394 \pm 0.010$ | $0.355 \pm 0.017$ | 1,388,547 / 1,650,401 | $+3.85$ pp | $[+2.14, +5.56]$ |

four experts; same checkpoints, same evaluation pipeline, only the inference-time mixing rule differs). The re-evaluation covers $48$ records (3 tasks $\times$ 4 seeds $\times$ 4 $N$-points).

## J SEVEN-TASK COMPARISONS AND ROUTING CONTROLS

This appendix collects the comparator panel over seven CLRS-30 tasks, its per-seed values and paired contrasts, the constant one-hot policy grid, the output-level surrogate-teacher pilot, the encoder/decoder initialisation ablation, and the routing controls referenced from §7.

| Task | $N$ | hard mean $\pm$ std | soft mean $\pm$ std | $\Delta$ (hard $-$ soft) |
|---|---|---|---|---|
| Dijkstra | 16 | $0.990 \pm 0.008$ | $0.982 \pm 0.017$ | $+0.81$ pp |
| Dijkstra | 32 | $0.924 \pm 0.027$ | $0.928 \pm 0.032$ | $-0.38$ pp |
| Dijkstra | 48 | $0.813 \pm 0.033$ | $0.817 \pm 0.071$ | $-0.46$ pp |
| Dijkstra | 64 | $0.642 \pm 0.068$ | $0.739 \pm 0.078$ | $-9.69$ pp |
| Bellman–Ford | 16 | $0.975 \pm 0.010$ | $0.946 \pm 0.021$ | $+2.91$ pp |
| Bellman–Ford | 32 | $0.902 \pm 0.041$ | $0.844 \pm 0.084$ | $+5.80$ pp |
| Bellman–Ford | 48 | $0.890 \pm 0.036$ | $0.841 \pm 0.069$ | $+4.89$ pp |
| Bellman–Ford | 64 | $0.841 \pm 0.048$ | $0.782 \pm 0.095$ | $+5.87$ pp |
| Floyd–Warshall | 16 | $0.621 \pm 0.013$ | $0.502 \pm 0.035$ | $+11.97$ pp |
| Floyd–Warshall | 32 | $0.387 \pm 0.013$ | $0.296 \pm 0.022$ | $+9.14$ pp |
| Floyd–Warshall | 48 | $0.284 \pm 0.021$ | $0.192 \pm 0.022$ | $+9.21$ pp |
| Floyd–Warshall | 64 | $0.192 \pm 0.063$ | $0.129 \pm 0.027$ | $+6.36$ pp |

Table 19: Hard vs. soft routing at inference for PSAR (4 seeds, OOD-dev). Hard routing is the canonical reporting convention. On BF and FW hard routing wins at every grid point, by $+2.9$ to $+5.9$ pp on BF and $+6.4$ to $+12.0$ pp on FW; on Dijkstra hard and soft are within seed std at $N \in \{16, 32, 48\}$ and soft routing is $+9.7$ pp better than hard at $N{=}64$. The re-evaluation changes only the inference-time mixing rule; the canonical router activation is a softmax (§4.2), so soft inference is the softmax-weighted convex combination of the four expert outputs at the same temperature. Hard top-1 is the rule the models were trained with and is retained as the canonical configuration; the Dijkstra difference at $N{=}64$ is reported as observed and not interpreted.

| Task | `Softmax` | `Entmax` | `Tropical` | `TMU` | PSAR |
|---|---|---|---|---|---|
| Dijkstra | $0.809 \pm 0.081$ | $0.890 \pm 0.042$ | $0.861 \pm 0.034$ | $\underline{0.974 \pm 0.005}$ | $0.939 \pm 0.018$ |
| Bellman–Ford | $\underline{0.923 \pm 0.037}$ | $0.914 \pm 0.023$ | $0.787 \pm 0.137$ | $0.801 \pm 0.189$ | $0.923 \pm 0.049$ |
| Floyd–Warshall | $0.205 \pm 0.021$ | $0.242 \pm 0.013$ | $0.203 \pm 0.052$ | $\underline{0.380 \pm 0.017}$ | $0.394 \pm 0.010$ |
| BFS | $0.823 \pm 0.109$ | $0.835 \pm 0.066$ | $\underline{0.877 \pm 0.082}$ | $0.525 \pm 0.153$ | $0.919 \pm 0.108$ |
| MST-Prim | $0.737 \pm 0.041$ | $0.781 \pm 0.040$ | $0.842 \pm 0.028$ | $\underline{0.976 \pm 0.008}$ | $0.951 \pm 0.024$ |
| Matrix-Chain-Order | $0.570 \pm 0.012$ | $0.587 \pm 0.028$ | $\underline{0.789 \pm 0.013}$ | $0.782 \pm 0.025$ | $0.723 \pm 0.098$ |
| SCC | $\underline{0.946 \pm 0.041}$ | $0.925 \pm 0.045$ | $0.841 \pm 0.112$ | $0.938 \pm 0.038$ | $0.923 \pm 0.046$ |

Table 20: Seven-task comparator panel at $N{=}32$ (4 seeds, mean $\pm$ sample std). Every entry is the run's own best OOD-dev accuracy over training, with checkpoints selected on OOD-dev; the same four seeds are used for every method. Underline marks the best single expert per task. Expert checkpoints: `Softmax`/`Entmax` from the common-width ($d_{\text{model}}{=}192$) baseline runs, `Tropical` and `TMU` from standalone runs using the same expert configurations as PSAR's branches. Per-seed values are in Table 21 and paired contrasts in Table 22.

### J.1 Seven-task comparator panel

Table 20 gives the panel, Table 21 its per-seed values and Table 22 the paired contrasts. Measured as the best single expert's mean minus each method's mean on every task (negative means above the best single expert), PSAR is at 3.5, 0.0, $-1.4$, $-4.2$, 2.5, 6.6 and 2.3 pp on Dijkstra, Bellman–Ford, Floyd–Warshall, BFS, MST-Prim, Matrix-Chain-Order and SCC (maximum 6.6, mean 1.3). The maxima for the single experts are 23.9 (`Softmax`, MST-Prim), 20.2 (`Entmax`, Matrix-Chain-Order), 17.7 (`Tropical`, Floyd–Warshall) and 35.2 pp (`TMU`, BFS), and their means 12.2, 9.9, 9.5 and 7.0 pp. These are point estimates on the checkpoint-selection split; the paired intervals are in Table 22.

### J.2 Encoder/decoder initialization

Each expert processor branch is initialized from the processor state of its OOD-dev-selected standalone Stage-0 checkpoint. The shared encoder and decoder are *randomly* initialized and trained from Stage 2a onward; they are not preloaded from any specialist. A four-seed ablation replaced that choice by initializing the shared encoder/decoder from the `Softmax` specialist, under the same

| Task | Method | s1 | s2 | s3 | s4 |
|---|---|---|---|---|---|
| Dijkstra | Softmax | 0.891 | 0.714 | 0.861 | 0.771 |
| Dijkstra | Entmax | 0.838 | 0.873 | 0.923 | 0.925 |
| Dijkstra | Tropical | 0.822 | 0.880 | 0.846 | 0.896 |
| Dijkstra | TMU | 0.980 | 0.974 | 0.967 | 0.974 |
| Dijkstra | PSAR | 0.964 | 0.923 | 0.932 | 0.937 |
| Bellman–Ford | Softmax | 0.947 | 0.939 | 0.938 | 0.867 |
| Bellman–Ford | Entmax | 0.943 | 0.918 | 0.907 | 0.888 |
| Bellman–Ford | Tropical | 0.896 | 0.875 | 0.782 | 0.596 |
| Bellman–Ford | TMU | 0.956 | 0.925 | 0.781 | 0.542 |
| Bellman–Ford | PSAR | 0.978 | 0.927 | 0.929 | 0.859 |
| Floyd–Warshall | Softmax | 0.188 | 0.204 | 0.234 | 0.192 |
| Floyd–Warshall | Entmax | 0.247 | 0.254 | 0.224 | 0.244 |
| Floyd–Warshall | Tropical | 0.251 | 0.132 | 0.227 | 0.202 |
| Floyd–Warshall | TMU | 0.360 | 0.402 | 0.381 | 0.375 |
| Floyd–Warshall | PSAR | 0.379 | 0.397 | 0.402 | 0.397 |
| BFS | Softmax | 0.681 | 0.934 | 0.876 | 0.801 |
| BFS | Entmax | 0.786 | 0.899 | 0.884 | 0.772 |
| BFS | Tropical | 0.755 | 0.920 | 0.903 | 0.930 |
| BFS | TMU | 0.540 | 0.670 | 0.310 | 0.580 |
| BFS | PSAR | 0.759 | 0.952 | 0.974 | 0.991 |
| MST-Prim | Softmax | 0.779 | 0.742 | 0.746 | 0.680 |
| MST-Prim | Entmax | 0.777 | 0.795 | 0.823 | 0.728 |
| MST-Prim | Tropical | 0.825 | 0.868 | 0.865 | 0.812 |
| MST-Prim | TMU | 0.964 | 0.977 | 0.982 | 0.981 |
| MST-Prim | PSAR | 0.955 | 0.918 | 0.952 | 0.978 |
| Matrix-Chain-Order | Softmax | 0.581 | 0.577 | 0.568 | 0.553 |
| Matrix-Chain-Order | Entmax | 0.558 | 0.610 | 0.612 | 0.567 |
| Matrix-Chain-Order | Tropical | 0.779 | 0.781 | 0.806 | 0.788 |
| Matrix-Chain-Order | TMU | 0.757 | 0.797 | 0.764 | 0.808 |
| Matrix-Chain-Order | PSAR | 0.577 | 0.757 | 0.772 | 0.784 |
| SCC | Softmax | 0.967 | 0.886 | 0.976 | 0.956 |
| SCC | Entmax | 0.920 | 0.873 | 0.982 | 0.924 |
| SCC | Tropical | 0.688 | 0.829 | 0.914 | 0.933 |
| SCC | TMU | 0.923 | 0.986 | 0.948 | 0.896 |
| SCC | PSAR | 0.924 | 0.958 | 0.858 | 0.954 |

Table 21: Per-seed values behind Table 20 (best OOD-dev accuracy at $N{=}32$, seeds 1–4).

OOD-dev selection protocol and otherwise identical recipe. It lowers accuracy by 8.2 pp on Dijkstra and 17.0 pp on Floyd–Warshall and leaves Bellman–Ford unchanged (Table 25).

| Task | vs `Softmax` | vs `Entmax` | vs `Tropical` | vs `TMU` |
|---|---|---|---|---|
| Dijkstra | $+13.0$ $[+7.2, +18.7]$ | $+4.9$ $[+1.0, +9.7]$ | $+7.8$ $[+4.2, +11.7]$ | $-3.5$ $[-4.7, -2.2]$ |
| Bellman–Ford | $+0.0$ $[-1.1, +2.1]$ | $+0.9$ $[-1.6, +2.9]$ | $+13.6$ $[+6.7, +21.8]$ | $+12.2$ $[+1.2, +24.3]$ |
| Floyd–Warshall | $+18.9$ $[+17.4, +20.2]$ | $+15.2$ $[+13.7, +17.0]$ | $+19.1$ $[+14.5, +24.3]$ | $+1.4$ $[+0.2, +2.1]$ |
| BFS | $+9.6$ $[+3.8, +16.2]$ | $+8.4$ $[+0.2, +17.8]$ | $+4.2$ $[+1.8, +6.6]$ | $+39.4$ $[+25.0, +56.9]$ |
| MST-Prim | $+21.4$ $[+17.6, +26.7]$ | $+17.0$ $[+12.6, +22.0]$ | $+10.8$ $[+6.9, +14.8]$ | $-2.5$ $[-4.6, -0.7]$ |
| Matrix-Chain-Order | $+15.3$ $[+4.8, +21.8]$ | $+13.6$ $[+5.4, +20.0]$ | $-6.6$ $[-15.7, -1.1]$ | $-5.9$ $[-14.1, -0.4]$ |
| SCC | $-2.3$ $[-8.9, +4.3]$ | $-0.1$ $[-8.6, +6.5]$ | $+8.3$ $[-1.7, +18.3]$ | $-1.5$ $[-6.8, +3.6]$ |

Table 22: Paired contrasts of PSAR against each single expert on the seven-task panel (pp difference, paired bootstrap over the four matched seeds, $10^4$ resamples). Bold marks intervals excluding zero — in either direction. PSAR is significantly ahead on some cells and significantly behind on others; in particular it trails `TMU` on Dijkstra and MST-Prim and trails `Tropical` on Matrix-Chain-Order.

| Task | learned router | force `Softmax` | force `Entmax` | force `Tropical` | force `TMU` |
|---|---|---|---|---|---|
| Dijkstra | **0.924** | 0.057 | 0.176 | 0.035 | 0.861 |
| Bellman–Ford | **0.902** | 0.022 | 0.117 | 0.268 | 0.752 |
| Floyd–Warshall | **0.387** | 0.043 | 0.101 | 0.032 | 0.347 |
| BFS | **0.937** | 0.152 | 0.082 | 0.127 | 0.671 |
| MST-Prim | **0.947** | 0.058 | 0.043 | 0.042 | 0.947 |
| Matrix-Chain-Order | **0.722** | 0.114 | 0.037 | 0.274 | 0.286 |
| SCC | **0.927** | 0.047 | 0.102 | 0.094 | 0.730 |

Table 23: Constant one-hot policy grid. The learned router is compared against pinning it to a single expert for every step, on the same checkpoints, evaluated on the shared held-out split (CLRS eval seed 88888, 64 samples, $N$=32, 4 seeds). Bold marks tasks where the learned policy is at least as good as the best constant policy. Pinning the router at inference is not a separately trained constant policy; trained constant policies are reported in Table 26. Note that forcing an expert inside the jointly trained pool is not the same as running that expert standalone: the in-pool branches share an encoder and decoder that were trained against the learned routing distribution, so a forced in-pool branch can score far below its standalone counterpart (e.g. forced `Softmax` on Bellman–Ford). Standalone specialist numbers are in Table 20.

### J.3 Standalone specialists versus in-pool branches

Two quantities in this paper are easy to conflate and are not comparable. A *standalone specialist* is a single-expert model trained end to end with its own encoder and decoder (Table 20). An *in-pool branch* is the same architectural primitive sitting inside the four-expert pool, where the encoder and decoder are shared across all four branches and were trained against the learned routing distribution rather than against that branch alone.

The consequence is that pinning the router to one in-pool branch does not recover the corresponding standalone specialist, and can fall far below it: on Bellman–Ford, forcing the in-pool `Softmax` branch scores near zero while the standalone `Softmax` specialist is the best single model on that task. This is a statement about shared-representation coupling, not about the primitive itself. Accordingly, we use the forced-policy grid (Table 23) only to ask whether a *constant* policy reproduces the *learned* policy within one trained pool, and we use the comparator panel (Table 20) for all claims about how PSAR compares with specialists.

### J.4 Further routing controls

**`StaticTask`: seed-level uncertainty.** Re-evaluating the hard-routing `StaticTask` checkpoints of Table 1 on the same held-out instances (CLRS eval seed 88888, 64 samples, $N$=32) and resampling hierarchically (outer: the four matched seed pairs; inner: the shared held-out instances, with identical indices for both members of a pair; $10^4$ resamples), PSAR minus `StaticTask` is $+9.2$ pp $[+5.3, +13.4]$ on Dijkstra, $+22.1$ pp $[+5.0, +43.7]$ on Bellman–Ford and $+21.8$ pp $[+12.5, +29.2]$ on Floyd–Warshall. All twelve seed-level differences are positive. The learned gate logits of the twelve checkpoints lie in $[-0.38, +0.50]$; their argmax is `Tropical`, `Tropical`, `Softmax`, `TMU` on Dijkstra, `TMU`, `TMU`, `Entmax`, `TMU` on Bellman–Ford and `Softmax`, `TMU`, `Tropical`, `Softmax` on Floyd–Warshall (seeds 1–4), and on 11 of the 12 runs the own-OOD-dev

| Task | hint-based teacher (canonical) | output-level surrogate | paired $\Delta$ |
|---|---|---|---|
| Dijkstra | $0.939 \pm 0.018$ | $0.964 \pm 0.019$ | $+2.5$ pp |
| Bellman–Ford | $0.923 \pm 0.049$ | $0.953 \pm 0.016$ | $+3.0$ pp |

Table 24: Output-level surrogate preference teacher (4 seeds, $N$=32 OOD-dev, matched seeds). The teacher is built from each in-pool branch's loss on the final output only ($\alpha_{\text{hint-output}}$=0), so no intermediate hint targets enter the preference construction; every other component matches the canonical recipe. We report this as parity evidence at $n$=4, not as an improvement. **Scope:** this does not produce a hint-free model — the underlying CLRS training and evaluation remain trace-supervised and teacher-forced — so we make no claim of applicability to benchmarks without intermediate traces. Extending output-level preference construction to further tasks is future work.

| Task | random init (canonical) | `Softmax`-specialist init | $\Delta$ |
|---|---|---|---|
| Dijkstra | $0.939 \pm 0.018$ | $0.857 \pm 0.030$ | $-8.2$ pp |
| Bellman–Ford | $0.923 \pm 0.049$ | $0.925 \pm 0.038$ | $+0.1$ pp |
| Floyd–Warshall | $0.394 \pm 0.010$ | $0.224 \pm 0.042$ | $-17.0$ pp |

Table 25: Shared encoder/decoder initialization ablation (4 seeds, best OOD-dev at $N$=32). Initializing the shared encoder/decoder from a single specialist costs 8.2 pp on Dijkstra and 17.0 pp on Floyd–Warshall and leaves Bellman–Ford unchanged. This provides supporting context for the canonical random-initialization choice but does not establish global optimality: only one alternative initializer was tried.

accuracy is within 3.5 pp of the separately trained constant policy of that expert on the same seed (Table 26), so `StaticTask` behaves as a constant-expert policy whose expert varies with the seed.

**Per-seed decomposition of the constant one-hot grid.** Splitting Table 23 by seed (same protocol) gives 28 task–seed cells. The learned policy is strictly higher than the best constant arm in 16 cells ($+0.05$ to $+73.3$ pp), identical in 9 cells, in each of which the router selects that expert at every step, and lower in 3 cells (by at most 0.15 pp). The number of seeds with a strictly positive difference is $1/4$ on Dijkstra, $2/4$ on Bellman–Ford, $3/4$ on Floyd–Warshall, BFS and SCC, $4/4$ on Matrix-Chain-Order and $0/4$ on MST-Prim. The Dijkstra aggregate of $+6.4$ pp comes entirely from seed 1, the only Dijkstra seed whose router does not select `TMU` at every step.

**Separately trained constant one-hot policies.** The grid above pins a jointly trained router at inference. To test constant policies that are trained as such, for each task, seed and expert we train a model with the same four-branch pool, the same Stage-0 expert initialisation, a randomly initialised shared encoder/decoder and the canonical three-stage recipe, replacing the input-conditioned router with a state-independent four-logit router initialised at $\pm 20$ in favour of that expert and frozen (48 runs). For the `TMU` policy the four logits remained nominally trainable owing to a stage-level implementation setting; across its twelve runs their maximum drift was 0.003 and the selected expert was `TMU` at every step, so the realised policy is exactly always-`TMU`. Table 26 uses the own-OOD-dev protocol, under which PSAR scores 0.939, 0.923 and 0.394. PSAR has the higher mean in all twelve task–expert cells and is not lower on any seed in ten of them; the exceptions are Dijkstra against always-`Tropical` ($3/4$ seeds, interval includes zero) and against always-`TMU` ($2/4$ seeds, $+1.3$ pp $[-2.6, +5.2]$). On Dijkstra, a trained always-`TMU` policy is therefore not separable from the learned router, consistent with the router selecting `TMU` at every step on three of the four seeds. Relative to its standalone run (Table 20), every fixed policy except always-`Tropical` on Dijkstra is lower, by 1.8 to 11.8 pp, so training inside the pool costs accuracy whichever expert is used. On Dijkstra always-`TMU` is 4.8 pp below standalone `TMU` (0.926 against 0.974) and PSAR is 3.5 pp below it; on Bellman–Ford and Floyd–Warshall, where the router combines two experts, PSAR matches or exceeds the best standalone expert.

**Step-indexed schedules.** A constant policy is the simplest input-independent control; step-indexed schedules are the next. For each task and seed, all schedule construction and selection use only that run's own OOD-dev split; each schedule is derived and selected once, then applied unchanged on the held-out split at $N$=32 and $N$=64. We evaluate (a) the ten-bin majority projection, which

Table 26: Separately trained constant one-hot policies versus PSAR ($N$=32, 4 seeds, mean ± sample std; each run's own best OOD-dev accuracy, the protocol of Appendix I.4). $\Delta$ is PSAR minus the fixed policy with a paired-bootstrap 95% CI over the four matched seeds; the last column counts seeds on which PSAR is not lower.

| Task | fixed policy | accuracy | $\Delta$ (pp) | 95% CI | seeds |
|---|---|---|---|---|---|
| Dijkstra | `Softmax` | $0.791 \pm 0.151$ | $+14.76$ | $[+5.13, +27.97]$ | 4/4 |
| | `Entmax` | $0.853 \pm 0.027$ | $+8.64$ | $[+7.29, +10.75]$ | 4/4 |
| | `Tropical` | $0.896 \pm 0.046$ | $+4.33$ | $[-0.59, +9.25]$ | 3/4 |
| | `TMU` | $0.926 \pm 0.054$ | $+1.34$ | $[-2.56, +5.25]$ | 2/4 |
| Bellman–Ford | `Softmax` | $0.889 \pm 0.038$ | $+3.43$ | $[+2.44, +5.22]$ | 4/4 |
| | `Entmax` | $0.871 \pm 0.061$ | $+5.19$ | $[+3.16, +6.84]$ | 4/4 |
| | `Tropical` | $0.743 \pm 0.187$ | $+18.07$ | $[+7.96, +30.68]$ | 4/4 |
| | `TMU` | $0.683 \pm 0.265$ | $+24.00$ | $[+6.25, +41.75]$ | 4/4 |
| Floyd–Warshall | `Softmax` | $0.125 \pm 0.040$ | $+26.91$ | $[+22.06, +29.93]$ | 4/4 |
| | `Entmax` | $0.212 \pm 0.052$ | $+18.16$ | $[+13.91, +23.50]$ | 4/4 |
| | `Tropical` | $0.120 \pm 0.050$ | $+27.40$ | $[+22.13, +31.65]$ | 4/4 |
| | `TMU` | $0.342 \pm 0.013$ | $+5.15$ | $[+3.53, +7.12]$ | 4/4 |

Table 27: Learned routing minus step-indexed schedules (pp, hierarchical paired-bootstrap 95% CI over matched seeds and shared held-out instances, $10^4$ resamples; CLRS eval seed 88888, 64 samples; positive means the learned router is higher).

| Schedule, $N$ | Dijkstra | Bellman–Ford | Floyd–Warshall |
|---|---|---|---|
| `TMU`-heavy, 32 | $+7.3$ $[-1.9, +18.3]$ | $+34.1$ $[+27.4, +40.6]$ | $+3.0$ $[-1.2, +7.2]$ |
| `TMU`-heavy, 64 | $+18.8$ $[+13.2, +25.3]$ | $+43.5$ $[+35.5, +51.4]$ | $+2.3$ $[-4.0, +8.0]$ |
| majority, 32 | $+0.0$ $[-0.2, +0.4]$ | $+2.5$ $[+0.0, +6.3]$ | $+6.7$ $[-0.1, +17.1]$ |
| majority, 64 | $-1.6$ $[-5.0, +0.3]$ | $+3.0$ $[-2.5, +10.4]$ | $+5.8$ $[-0.8, +12.5]$ |

divides normalised execution time into ten step-fraction bins and assigns each bin the expert the learned router selects most often on OOD-dev, and (b) twelve `TMU`-heavy schedules (`TMU` for the first or last 30% or 50% of steps, crossed with each attention expert elsewhere), of which the one with the highest OOD-dev accuracy per run is applied. At $N$=32 the task-level means are 0.924, 0.902, 0.387 for the learned router, 0.852, 0.561, 0.358 for the selected `TMU`-heavy schedule and 0.924, 0.878, 0.321 for the majority projection (Dijkstra, Bellman–Ford, Floyd–Warshall). Table 27 gives all contrasts. The interval excludes zero for the `TMU`-heavy schedule on Bellman–Ford at both sizes and on Dijkstra at $N$=64; every other contrast is unresolved at four seeds, and the Bellman–Ford majority contrast at $N$=32 has a lower bound of exactly zero. On Dijkstra the ten-bin projection reproduces the learned router (equal means at $N$=32, nominally 1.6 pp higher at $N$=64), so a step-only schedule suffices there. These are within-checkpoint interventions: encoder, decoder, experts and instances are unchanged; a separately trained fixed-schedule model is the `TrainedSchedule` control below.

**A step-only router and a parameter-matched single expert.** Two Floyd–Warshall controls target the margin at $N$=48 and $N$=64 on the shared held-out split. `StepOnly` keeps the four-expert pool, the three-stage schedule and every loss term of PSAR, and restricts the router input to the Fourier step encoding; it has 1,238,389 parameters against PSAR's 1,388,547 on this task. `TMU-1.4M` is the single `TMU` expert at $d_{\text{model}}$=328, $d_p$=164 and transition multiplier 6, giving 1,416,125 parameters, 2.0% above PSAR. Against `TMU-1.4M` both intervals exclude zero and all four seeds are positive at both sizes; against `StepOnly` both intervals include zero and the seed-level differences change sign (Table 28). An eight-seed extension, run after the four-seed intervals had been seen, gives $+4.05$ pp $[+0.79, +7.85]$ and $+2.95$ pp $[-2.06, +7.89]$ against `StepOnly` and $+9.21$ pp $[+5.91, +12.46]$ and $+9.56$ pp $[+7.29, +11.71]$ against `TMU-1.4M`, all sixteen seed-level differences positive.

**Removing routing, and training the schedule.** Two further Floyd–Warshall controls use eight seeds, with the comparisons, sample size and decision rule fixed in advance. `SoftMix` keeps the pool and the three-stage recipe and removes routing: four learnable task-level weights pass through a softmax and combine the expert outputs convexly, in training and at test time; it has 1,231,477 parameters, the difference to PSAR being the router. `TrainedSchedule` keeps selection but conditions only on the step: the ten-bin majority projection of each run's own router, derived on that

Table 28: PSAR against a step-only router and a parameter-matched single TMU expert on Floyd–Warshall under teacher forcing, shared held-out split (eval seed 88888, 64 samples), hierarchical paired bootstrap over the four matched seeds with shared inner instances, $10^4$ resamples. The criterion was fixed before evaluation: routing counts as isolated only if all four lower bounds are positive. The last column counts seeds on which PSAR is higher.

| contrast | $N$ | $\Delta$ (pp) | 95% CI | seeds |
|---|---|---|---|---|
| PSAR − StepOnly | 48 | +4.46 | $[-0.54, +11.40]$ | 3/4 |
| | 64 | +2.49 | $[-5.09, +10.08]$ | 2/4 |
| PSAR − TMU-1.4M | 48 | +7.16 | $[+3.14, +11.07]$ | 4/4 |
| | 64 | +8.09 | $[+5.47, +10.45]$ | 4/4 |

| Seed | Softmax | Entmax | Tropical | TMU |
|---|---|---|---|---|
| 1 | 0.214 | 0.318 | 0.228 | 0.240 |
| 2 | 0.249 | 0.275 | 0.289 | 0.187 |
| 3 | 0.240 | 0.291 | 0.281 | 0.188 |
| 4 | 0.298 | 0.306 | 0.152 | 0.245 |
| 5 | 0.278 | 0.279 | 0.190 | 0.253 |
| 6 | 0.272 | 0.319 | 0.216 | 0.193 |
| 7 | 0.277 | 0.260 | 0.264 | 0.199 |
| 8 | 0.256 | 0.299 | 0.243 | 0.203 |
| mean | 0.261 | 0.293 | 0.233 | 0.214 |

Table 29: The four mixture weights SoftMix learns on Floyd–Warshall, read off each run's selected checkpoint. A seed counts as degenerate if fewer than two experts carry weight above 0.05; 0 of 8 seeds are, so every run keeps a genuine mixture rather than collapsing onto one expert. The criterion and this report were fixed before these runs existed. SoftMix has 1,231,477 parameters against PSAR's 1,388,547 on this task, the difference being the router, so its margin over SoftMix at $N$=48 cannot be separated from that capacity gap.

run's development split, becomes the routing rule and the model is retrained end to end (Tables 30 and 31).

Against SoftMix, PSAR is +3.41 pp $[+1.77, +5.13]$ at $N$=48, positive on seven of eight seeds and significant after Holm correction, and −0.24 pp $[-3.38, +2.91]$ at $N$=64; neither the preregistered rule for a routing contribution (both lower bounds positive) nor the equivalence rule (90% intervals within ±2.0 pp) is met. Against TrainedSchedule, PSAR is +16.07 pp $[+11.21, +20.81]$ and +12.58 pp $[+8.16, +16.96]$, positive on all eight seeds and significant after Holm correction at both sizes: the step projection that is not separable from the learned router on Dijkstra inside the checkpoint does not reproduce it on Floyd–Warshall when trained as the routing rule. Every SoftMix run keeps all four mixture weights above 0.15 (Table 29).

### J.5 Scope of the contribution

Taken together, the evidence in this appendix supports a two-part contribution, and we state its limits explicitly. First, the Floyd–Warshall result is closely associated with introducing the pair-state primitive (TMU): PSAR exceeds every single expert there, while most of its margin over the best attention algebra is already present in standalone TMU. Because the primitive comparison is not capacity-isolated, we interpret this as evidence for the usefulness of pair-state structure under the tested configurations rather than as a causal estimate of the primitive alone. Second, heterogeneous per-step routing over that pool is characterised rather than advocated: Table 22 shows regimes where the pool exceeds all singles (Floyd–Warshall, BFS), regimes where it reduces to a specialist and trails it (Dijkstra and MST-Prim behind TMU, Matrix-Chain-Order behind Tropical), and regimes where the difference is not statistically distinguishable at four seeds (Bellman–Ford, SCC). Standalone specialists retain higher peak accuracy in specialist-aligned regimes, and Relational Attention remains a distinct boundary on Floyd–Warshall under the tested composition (Appendix F.6).

| Contrast | $\Delta$ (pp) | sd | 95% CI | seeds $> 0$ |
|---|---|---|---|---|
| PSAR $-$ `SoftMix`, $N$=48 | $+3.41$ | 2.60 | $[+1.77, +5.13]$ | 7/8 |
| PSAR $-$ `SoftMix`, $N$=64 | $-0.24$ | 4.84 | $[-3.38, +2.91]$ | 4/8 |
| PSAR $-$ `TrainedSchedule`, $N$=48 | $+16.07$ | 7.42 | $[+11.21, +20.81]$ | 8/8 |
| PSAR $-$ `TrainedSchedule`, $N$=64 | $+12.58$ | 6.68 | $[+8.16, +16.96]$ | 8/8 |

Table 30: Two controls that remove input-conditioned routing while keeping the heterogeneous pool. `SoftMix` replaces the router with four learnable weights shared by every sample and every step and mixes the experts convexly during training and at test time; `TrainedSchedule` replaces it with a fixed step-indexed schedule, read off that run's own out-of-distribution development split and then trained as the routing rule. Both are trained end to end with the same three-stage recipe on Floyd–Warshall. Teacher-forced accuracy on the shared held-out split; hierarchical paired bootstrap over the matched seeds with instance indices shared across seeds, $10^4$ resamples. The comparisons, sample size and decision thresholds were fixed before any of these runs existed.

| Contrast | 90% CI (pp) | Verdict at $\delta = 2.0$ pp |
|---|---|---|
| PSAR $-$ `SoftMix`, $N$=48 | $[+2.02, +4.87]$ | not equivalent |
| PSAR $-$ `SoftMix`, $N$=64 | $[-2.92, +2.38]$ | inconclusive |
| PSAR $-$ `TrainedSchedule`, $N$=48 | $[+11.98, +20.10]$ | not equivalent |
| PSAR $-$ `TrainedSchedule`, $N$=64 | $[+8.85, +16.28]$ | not equivalent |

Table 31: Equivalence test for the same contrasts. An interval containing zero does not establish equivalence, so we state a margin in advance: the two are called equivalent only if the whole 90% interval lies inside $\pm 2.0$ pp and the point estimate does too, and not equivalent only if the whole interval lies outside it; otherwise the test is inconclusive. The margin was chosen before these runs as smaller than every effect the paper attributes to the pool.

We therefore claim a bounded characterisation of when heterogeneous routing helps, not a general routing advantage.

## K Native-pool axis-ceiling table and visualisation

This appendix presents the full native-pool scope comparison referenced from §6.1: Table 32 reports raw accuracies across the CLRS-range $N \in \{32, 64\}$, and Figure 4 provides a four-bar visualisation at $N$=32.

| Task | $N$ | best single algebra | best native expert | PSAR | `RelAttn` | $\Delta$ (PSAR $-$ `RelAttn`) |
|---|---|---|---|---|---|---|
| **Dij** | 32 | 0.854 (`Entmax`) | 0.957 (`TMU`) | 0.924 | 0.621 | $\mathbf{+30.4}$ |
| **Dij** | 64 | 0.560 (`Entmax`) | 0.741 (`TMU`) | 0.642 | 0.326 | $\mathbf{+31.7}$ |
| BF | 32 | 0.881 (`Softmax`) | 0.881 (`Softmax`) | 0.902 | 0.897 | $+0.6$ |
| BF | 64 | 0.771 (`Softmax`) | 0.771 (`Softmax`) | 0.841 | 0.833 | $+0.9$ |
| FW | 32 | 0.238 (`Entmax`) | 0.368 (`TMU`) | 0.387 | 0.419 | $-3.2$ |
| FW | 64 | 0.122 (`Entmax`) | 0.122 (`Entmax`) | 0.192 | 0.241 | $-4.8$ |

Table 32: PSAR vs reference points on the three canonical shortest-path tasks (4-seed mean OOD-dev). The CLRS-range rows ($N \in \{32, 64\}$) underlie main-text scope claims. Best native expert adds `TMU` to the algebra pool; on Dij and on FW at $N$=32 the native argmax differs from the algebra argmax because `TMU` is the strongest single expert. **Dij**: PSAR dominates `RelAttn` by $+30$ to $+32$ pp at $N$=32, 64 (the native pool matches Dij's structure). **BF**: near-tie at $N \in \{32, 64\}$ ($\Delta \in \{+0.6, +0.9\}$ pp), as expected for a single-`Softmax`-friendly oracle. **FW**: `RelAttn` retains a 3 to 5 pp lead at $N \in \{32, 64\}$ — the cross-axis scope boundary of native-pool routing. We report the FW gap as a within-pool boundary against the relational-bias axis, not a closed comparison: cross-axis composition (PSAR-on-`RelAttn`, Appendix F.6) is run as a negative-result diagnostic, not assumed additive.

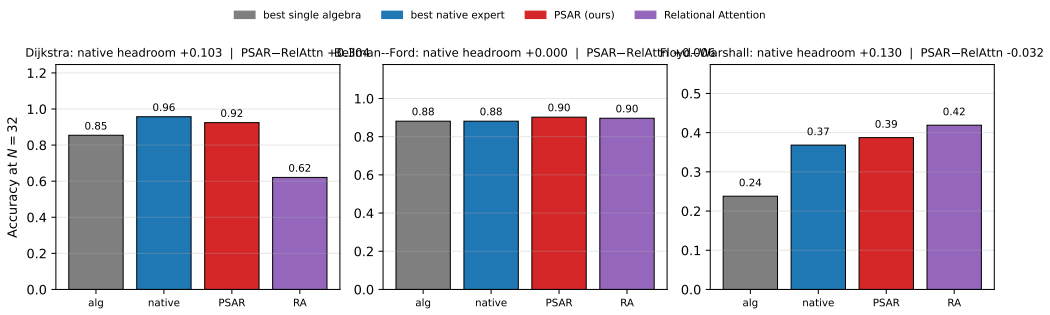



Figure 4: Native-pool ceiling vs observed `RelAttn` gap at $N$=32. Dij/BF: four bars within the populated per-node oracle headroom, PSAR near or above same-pool comparators. FW: `RelAttn` remains ahead of PSAR by 3.2 pp at $N$=32 and 4.8 pp at $N$=64 (Table 32); this gap is outside the native-pool axis PSAR targets, and the one cross-axis composition we ran does not close it (Appendix F.6).

## L Within-family intensity scaling diagnostic

This diagnostic addresses the conjecture that the FW gap to `RelAttn` reduces to a stronger intensity of the `GraphBias` submodule shared by PSAR's three single-step attention experts.

A natural conjecture is that the FW residual is reducible to a stronger *intensity* of the `GraphBias` submodule that all three of PSAR's experts already share. That is, perhaps PSAR only needs to turn up the distance-decay and edge-bias strength on FW to match `RelAttn`. We directly test this by exposing `GraphBias` intensity as a router-learned scalar $s = 2\sigma(\ell_s) \in (0, 2)$ centred at the fixed behaviour ($s$=1), and evaluate at two levels of evidence. (i) Learned intensity (paired 4-run pilot): the router converges to $s$=1.39 on Dij (amplify) and $s$=0.60 on FW (attenuate), with best OOD-dev accuracy dropping $-14.9$pp on Dij and $-2.9$pp on FW versus the paired fixed-intensity baseline. (ii) Fixed intensity (diagnostic): setting $s$ to values in each task's hypothesised-helpful direction — $s \in \{0.5, 0.8\}$ on Dij and $s \in \{1.2, 1.5\}$ on FW — *all four* settings underperform $s$=1 by $-0.88$ to $-9.77$pp. Since opposite-direction fixed scales also underperform, the failure is not the learner picking the wrong sign; the intensity dimension itself carries no measured headroom over the fixed value in this pilot. This does not support the account that `RelAttn`'s FW advantage reduces to a stronger same-family structural bias; `RelAttn`'s edge-conditioned $Q/K/V$ (Diao & Loynd, 2023) remains a distinct mechanism under the tested configurations.

## M Per-seed routing mass

Panels (a) and (c) of Figure 5 use the three attention algebras only. For each task and $N \in \{16, 32, 48, 64\}$ we record the per-sample argmax algebra over the 64 samples of the shared held-out split under all four single-algebra seeds ($64 \times 4 = 256$ oracle observations per task and $N$; eval seed 88888, Appendix O), call the algebra with the highest winrate at each $N$ its winner, and define the *oracle regime entropy* $H_{\text{oracle}} = -\sum_a f_a \log f_a / \log 3 \in [0, 1]$ over the fraction $f_a$ of the four $N$ points won by algebra $a \in \{\texttt{Softmax}, \texttt{Entmax}, \texttt{Tropical}\}$. Only on Dij does the winner change with $N$ (`Softmax` at $N$=16, `Entmax` at $N \geq 32$ with at most half of the per-sample wins), giving $H_{\text{oracle}} = 0.51$; BF is `Softmax`-best and FW `Entmax`-best at every $N$, giving $H_{\text{oracle}} = 0$. PSAR's mean gain over the best single algebra across the grid is $+7.0$ pp on Dij, $+3.3$ pp on BF and $+10.8$ pp on FW, so the highest-gain task has $H_{\text{oracle}} = 0$: the entropy is descriptive, excludes `TMU`, and does not predict PSAR's gain across the three tasks.

Table 33 decomposes the seed-averaged routing mass of §7 into its four constituent seeds. All entries are measured on the same protocol as the aggregate: 64 samples of the shared held-out evaluation split (CLRS eval seed 88888) at $N$=32, from the OOD-dev-selected checkpoints.

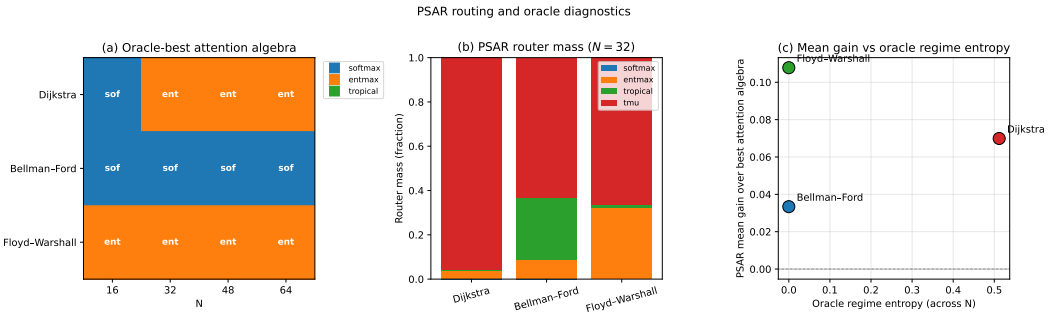



Figure 5: Routing and oracle diagnostics (descriptive). (a) Per-$N$ oracle-best-algebra heatmap over $\{\texttt{Softmax}, \texttt{Entmax}, \texttt{Tropical}\}$. (b) Router mass (4-seed mean; 64 samples of the shared held-out split at $N$=32) over PSAR's four experts. (c) Per-task mean gain over best single algebra (CLRS-range grid) vs oracle-regime entropy $H_{\text{oracle}}$.

| Task | Seed | Softmax | Entmax | Tropical | TMU |
|---|---|---|---|---|---|
| Dijkstra | 1 | 0.00 | 16.11 | 0.00 | 83.89 |
| Dijkstra | 2 | 0.00 | 0.00 | 0.00 | 100.00 |
| Dijkstra | 3 | 0.00 | 0.00 | 0.00 | 100.00 |
| Dijkstra | 4 | 0.00 | 0.00 | 0.00 | 100.00 |
| *mean* | — | 0.00 | 4.03 | 0.00 | 95.97 |
| Bellman–Ford | 1 | 0.00 | 4.95 | 41.67 | 53.39 |
| Bellman–Ford | 2 | 0.00 | 0.00 | 0.00 | 100.00 |
| Bellman–Ford | 3 | 0.00 | 16.67 | 0.00 | 83.33 |
| Bellman–Ford | 4 | 0.00 | 13.28 | 70.83 | 15.89 |
| *mean* | — | 0.00 | 8.72 | 28.12 | 63.15 |
| Floyd–Warshall | 1 | 0.00 | 9.32 | 0.00 | 90.68 |
| Floyd–Warshall | 2 | 0.00 | 62.90 | 0.00 | 37.10 |
| Floyd–Warshall | 3 | 0.00 | 55.90 | 3.23 | 40.88 |
| Floyd–Warshall | 4 | 1.01 | 0.00 | 0.86 | 98.14 |
| *mean* | — | 0.25 | 32.03 | 1.02 | 66.70 |

Table 33: Per-seed routing mass (%) over the four-expert pool at $N$=32 on the shared held-out split. On Dijkstra the policy is TMU-dominated on all four seeds. On Bellman–Ford and Floyd–Warshall the seed-averaged row is not representative of any individual seed: BF seed 2 routes entirely to TMU while BF seed 4 routes mostly to Tropical (70.83%, with TMU at 15.89%), and FW seeds 2–3 are Entmax-leaning (62.90%, 55.90%) while FW seeds 1 and 4 are TMU-dominated (90.68%, 98.14%). Softmax receives no mass on Bellman–Ford under any seed. The averaged mixture therefore reflects disagreement between seeds about which expert to use, not a within-seed blend; we make no claim that a single trained router combines experts on these two tasks.

## N THE HINT CONTRACT: A COPY BASELINE AND SELF-ROLLOUT

Every accuracy in the body is teacher-forced: the processor consumes the ground-truth hint at each of the $T-1$ steps and the output is decoded from the state produced after $\text{hints}[T-2]$ (Appendix F.3). This appendix reports two contrasts that bound what such an accuracy can mean.

**Copying the last fed hint.** A baseline with no processor emits $\text{hints}[T-2]$ as its output. On Dijkstra the last fed hint coincides with the output array on every held-out sample at $N \in \{32, 48, 64\}$, so this baseline scores 1.000 there, and 0.970 at $N$=16; on Floyd–Warshall it scores 0.873, 0.930, 0.951 and 0.956 at $N \in \{16, 32, 48, 64\}$. On both tasks it is above every trained model in this paper at $N \geq 32$. On Bellman–Ford it scores 0.268, 0.812, 0.250 and 0.214, for a reason that lies in our implementation: the recurrence runs $T-1$ steps for every sample, where $T$ is the padded trajectory length of the split, and the per-sample trajectory lengths are computed but used neither to truncate the

recurrence nor to mask the hint loss. Floyd–Warshall's pivot loop runs a fixed number of iterations, so its trajectories all have the same length and none is padded; Dijkstra and Bellman–Ford stop when the queue empties or no distance changes, so their short samples are zero-padded and those padded steps enter the hint loss as ordinary targets. At $N{=}64$ most Bellman–Ford samples are fed padding after the informative hint, which is why copying loses the answer there and not on the other two tasks. It applies to every model reported here, which does not mean that it affects them equally, and it is a deviation from the reference CLRS implementation, which freezes a sample's predictions past its own length and passes the lengths into the hint loss. The teacher-forced numbers therefore measure how much of a supplied trace a processor preserves and transforms at sizes beyond training. They order processors under one fixed contract; they do not measure execution of the algorithm, and they are not comparable with systems evaluated without hints at test time.

**Entries that change and entries that are kept.** Most target entries equal a hint the model has already been given, which is why copying scores well. We split every pointer target by whether it differs from its reference, using the ground truth only to define the split: at each step the reference is the hint fed at that step, and for the output it is the last non-padding hint fed to the model, $\mathrm{hints}[T{-}2]$ for a full-length trajectory and the final hint $\mathrm{hints}[L{-}1]$ for a sample of length $L < T$, after which padding is fed. Unlike the copy baseline, this reference ignores padding, so on Dijkstra at $N \geq 32$ every output entry equals it, and on Bellman–Ford all but a few do. Padded steps are excluded. Copying the reference scores 1 on kept entries and 0 on changed ones. The split was chosen after the results in the body had been seen, and its intervals are exploratory.

On Floyd–Warshall, changed entries are 7.0, 4.9 and 4.4% of the output at $N{=}32$, 48 and 64, and every model is far below the copy on the kept entries (0.21 to 0.40 at $N{=}32$; Table 34), so most output errors are entries that the last fed hint already had right. PSAR's margins lie on those kept entries (Table 35). Against the parameter-matched `TMU` at $N{=}48$ and 64 the output margin is $+7.5$ and $+8.4$ pp on kept entries and $+0.2$ and $+1.0$ pp on changed entries, whose intervals contain zero; the per-step predictions split the same way. Against `Entmax`, the strongest attention algebra on this task, the changed-entry differences at $N{=}48$ and 64 are negative and their intervals contain zero. Weighted by their share of the output, kept entries also carry almost all of standalone `TMU`'s lead over `Entmax` at $N{=}32$. Under this contract the Floyd–Warshall differences between processors, the pool's margin over the parameter-matched expert included, are carried by the kept entries: they measure how well supplied pointer entries survive the recurrence and the read-out, and on the entries that the pivot step changes no difference from the parameter-matched expert or from `Entmax` is resolved at $N{=}48$ and 64.

On Dijkstra at $N \geq 32$ the output equals the reference on every entry, so output accuracy there measures retention only. The per-step predictions do contain changed entries, and on those PSAR is above standalone `TMU` ($+11.7$ pp $[+3.9, +19.4]$ at $N{=}48$, $+11.5$ pp $[+5.3, +17.5]$ at $N{=}64$) while it is below it on the output ($-5.5$ pp $[-9.3, -2.1]$ and $-9.9$ pp $[-17.1, -2.4]$), so its deficit to `TMU` in the body is a deficit in retention. Against each attention algebra its changed-entry differences at $N \geq 32$ are not separable from zero. On Bellman–Ford, PSAR and `Softmax` are not separable on either per-step stratum at any $N$.

**A second deviation: the edge-level read-out.** Quantities that live on node pairs are decoded here as $\hat{D}_{ij} = a(h_i) + b(h_j)$, a sum of one score for the source and one for the target, with no term that depends on the pair. Every matrix of that form satisfies $\hat{D}_{ij} + \hat{D}_{kl} = \hat{D}_{il} + \hat{D}_{kj}$ for all $i, j, k, l$, so none of them equals $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, whose two sides are 0 and 2; widening the model does not change this. The reference implementation decodes the same quantities as $a(h_i) + b(h_j) + c(e_{ij})$, and the per-edge term removes the restriction (Veličković et al., 2022). On Floyd–Warshall the distance and reachability hints are both decoded this way. Under teacher forcing both are supplied as ground truth at every step, so the restriction never corrupts the state that is carried forward; when the model consumes its own predictions it does, and the restriction bears directly on autonomous execution. The hint loss on both quantities is computed and backpropagated under either contract, so the restriction shapes the shared representation during training as well; we identified it after these runs were trained and did not retrain under a less restricted read-out.

**Self-rollout.** Under the opposite contract the model receives the ground-truth hint only at $t{=}0$ and consumes its own argmax hint prediction at every later step. Table 39, Table 40 and Table 41 give both

| | $N$=32 | | $N$=48 | | $N$=64 | |
|---|---|---|---|---|---|---|
| Model | kept | changed | kept | changed | kept | changed |
| Copy of the reference | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 |
| `Softmax` | 0.210 | 0.122 | 0.116 | 0.077 | 0.069 | 0.044 |
| `Entmax` | 0.244 | 0.175 | 0.166 | 0.117 | 0.124 | 0.095 |
| `Tropical` | 0.209 | 0.069 | 0.126 | 0.038 | 0.088 | 0.028 |
| `TMU` | 0.376 | 0.238 | 0.186 | 0.077 | 0.107 | 0.040 |
| `TMU-1.4M` | 0.396 | 0.245 | 0.217 | 0.094 | 0.114 | 0.052 |
| `StepOnly` | 0.355 | 0.140 | 0.248 | 0.065 | 0.173 | 0.049 |
| `SoftMix` | 0.361 | 0.177 | 0.254 | 0.099 | 0.195 | 0.075 |
| PSAR | 0.401 | 0.159 | 0.293 | 0.096 | 0.198 | 0.062 |
| Changed share of output entries | 7.0% | | 4.9% | | 4.4% | |

Table 34: Floyd–Warshall output pointer accuracy under teacher forcing, split by whether the target entry equals the last hint fed to the model (kept) or differs from it (changed). Four-seed mean of per-instance accuracy on the shared held-out split, same checkpoints as the body. Copying the reference is the copy baseline of this appendix.

| Contrast | Entries | $N$=48 | $N$=64 |
|---|---|---|---|
| PSAR − `TMU-1.4M` | output, kept | $+7.5$ $[+3.5, +11.6]$ | $+8.4$ $[+5.8, +10.8]$ |
| | output, changed | $+0.2$ $[-2.4, +3.4]$ | $+1.0$ $[-1.0, +3.0]$ |
| | per step, kept | $+3.9$ $[+2.0, +5.5]$ | $+5.7$ $[+4.7, +6.7]$ |
| | per step, changed | $+1.5$ $[-2.1, +5.0]$ | $+0.3$ $[-2.3, +2.8]$ |
| PSAR − `Entmax` | output, kept | $+12.7$ $[+10.0, +16.3]$ | $+7.5$ $[+1.4, +13.5]$ |
| | output, changed | $-2.1$ $[-11.4, +4.1]$ | $-3.3$ $[-10.8, +2.3]$ |
| | per step, kept | $+10.2$ $[+8.7, +12.2]$ | $+7.0$ $[+3.9, +10.2]$ |
| | per step, changed | $-1.8$ $[-7.1, +3.5]$ | $-2.8$ $[-7.2, +1.5]$ |

Table 35: PSAR minus two comparators on Floyd–Warshall within each stratum (pp, positive favours PSAR), teacher-forced, shared held-out split, four matched seeds, hierarchical bootstrap with instance indices shared across seeds, $10^4$ resamples. *Per step*: the pointer hint predicted at every step, split against the hint fed at that step. Exploratory: the split was chosen after the results in the body had been seen.

protocols for the same checkpoints. Every processor, PSAR included, falls below $0.19$ on Dijkstra and Floyd–Warshall and below $0.29$ on Bellman–Ford at $N \geq 32$. The ordering is not preserved: on Dijkstra PSAR is below all three single algebras at every $N \geq 32$ under self-rollout, while it is above all three under teacher forcing. One ordering does survive: on Floyd–Warshall PSAR stays above the parameter-matched `TMU` under self-rollout too, at $N$=48 ($+4.2$ pp, $[+2.2, +6.9]$) and at $N$=64 ($+2.7$ pp, $[+1.0, +4.7]$), by the same hierarchical paired bootstrap over the four matched seeds. The margin under teacher forcing minus the margin under self-rollout, for the same checkpoints, is $+3.0$ pp $[-2.9, +8.9]$ at $N$=48 and $+5.3$ pp $[+3.6, +7.5]$ at $N$=64 (computed after the results were seen). These models were trained with ground-truth hints at every step, so self-rollout evaluates them off their training distribution, and the collapse mixes that mismatch with the models themselves.

**Training under the same contract.** The collapse mixes how well a processor executes on its own with the fact that it never saw its own predictions during training. A Dijkstra pilot separates them: two single-expert processors (`Softmax` and `TMU`), two seeds each, identical except that one arm receives the ground-truth hint at every training step and the other its own predicted hint distribution from the first step on, gradients flowing through the feedback. Both arms freeze each sample's state past its own trajectory length and mask the hint loss accordingly, as the reference implementation does, which makes these numbers internally comparable but not comparable with the Dijkstra numbers in the body. Both select checkpoints on an autonomous development metric, and the criterion was fixed beforehand: recovery at the training size requires autonomous accuracy at $N$=16 to reach $0.90$.

Aligning the contract improves autonomous accuracy several-fold and does not reach that bar. On the shared held-out split, autonomous accuracy rises from $0.409$ to $0.836$ at $N$=16, from $0.210$ to $0.404$

| Contrast | $N$ | $\Delta$ (pp) | 95% CI | seeds $> 0$ |
|---|---|---|---|---|
| PSAR − `TMU-1.4M` | 48 | $-5.84$ | $[-8.83, -2.83]$ | 0/4 |
| PSAR − `TMU-1.4M` | 64 | $-3.44$ | $[-6.17, -0.24]$ | 1/4 |
| PSAR − `SoftMix` | 48 | $+4.38$ | $[-5.16, +13.84]$ | 2/4 |
| PSAR − `SoftMix` | 64 | $+1.25$ | $[-7.47, +9.84]$ | 2/4 |

Table 36: The two preregistered comparisons after retraining every arm under the corrected training contract on Floyd–Warshall, evaluated with the model consuming its own predicted hints. Positive favours the pool. The condition fixed in advance was that both intervals have a lower bound above zero; it is not met for either comparison, and against the parameter-matched single expert both intervals lie entirely below zero. Four seeds, shared held-out split, hierarchical paired bootstrap with instance indices shared across seeds, $10^4$ resamples.

at $N$=32 and from $0.058$ to $0.213$ at $N$=64; teacher-forced accuracy on the same checkpoints falls from $0.995$ to $0.782$ at $N$=16 and from $0.518$ to $0.217$ at $N$=64. The two contracts therefore trade against each other rather than one dominating, and the teacher-forced accuracies reported throughout this paper are specific to the contract they are measured under. At $N$=16 the aligned arm reaches $0.836$ against the preregistered $0.90$, so the criterion is not met; beyond the training size both arms remain far below the teacher-forced numbers. Taking the last epoch instead of the checkpoint selected on the autonomous development metric gives $0.931$ at $N$=16 and $0.195$ at $N$=64 on the same split, so at the training size the aligned contract does reach high autonomous accuracy; what remains is the fall-off above it. This is a two-expert, two-seed pilot on one task.

**The Floyd–Warshall comparison under the corrected contract.** The pilot above changes the contract for single experts on one task. We then retrained the whole Floyd–Warshall comparison under it: four single experts, the parameter-matched single pair-state expert, the non-routing convex mixture and PSAR, four seeds each, with the same hint contract, the same length masking and the same autonomous selection metric in every arm, and the same number of optimiser steps per epoch. Each arm consumes its own predicted hint distribution as in the pilot, and the pool's experts are pretrained and selected under that contract. On this task every trajectory has full length, so the length masking changes nothing; these runs differ from the models in the body in the hints fed during training and in the metric used for checkpoint selection and early stopping, not in length handling. The two comparisons and the decision rule were fixed before these runs: the pool's advantage counts as preserved only if PSAR minus the parameter-matched expert has a lower bound above zero at both $N$=48 and $N$=64, with the autonomous protocol primary.

It is not preserved, and the sign is the other way (Table 36). Against the parameter-matched expert PSAR is $-5.84$ pp $[-8.83, -2.83]$ at $N$=48, negative on all four seeds, and $-3.44$ pp $[-6.17, -0.24]$ at $N$=64: both intervals lie entirely below zero. Against the non-routing mixture the intervals contain zero at both sizes, $+4.38$ pp $[-5.16, +13.84]$ and $+1.25$ pp $[-7.47, +9.84]$, with two of four seeds positive at each. Under this contract the parameter-matched single expert is the strongest arm at every size (Table 37).

Both sides of one comparison inside this set come from the same training contract and the same checkpoints. Evaluated with ground-truth hints, PSAR is ahead of the parameter-matched expert only on the point estimate at $N$=32 and 48 ($+1.2$ pp $[-4.2, +9.2]$ and $+1.1$ pp $[-2.4, +6.6]$, one of four seeds positive) and by $+2.2$ pp $[+0.0, +5.5]$ at $N$=64; evaluated with its own predictions, it is clearly behind. The contrast between the two evaluations, (PSAR − parameter-matched expert) under ground-truth hints minus the same under its own predictions, is $+9.2$ pp $[+3.5, +17.8]$, $+7.0$ pp $[+2.0, +13.7]$ and $+5.7$ pp $[+0.9, +10.7]$ at $N \in \{32, 48, 64\}$, by the same bootstrap with instance indices shared across both evaluations; it was computed after these results were seen. Ground-truth hints are themselves off-distribution for these checkpoints (PSAR scores $0.184$ on its own hints and $0.136$ on ground-truth hints at $N$=48). The ordering between a routed heterogeneous pool and a single wider expert is therefore not a property of the two models alone but of the evaluation contract as well. This is four seeds on one task. These numbers are not comparable with the rest of the paper, whose models are trained on ground-truth hints throughout.

| | own predicted hints | | | | ground-truth hints | | | |
|---|---|---|---|---|---|---|---|---|
| Method | $N$=16 | $N$=32 | $N$=48 | $N$=64 | $N$=16 | $N$=32 | $N$=48 | $N$=64 |
| `Softmax` | 0.428 | 0.136 | 0.073 | 0.046 | 0.369 | 0.134 | 0.081 | 0.053 |
| `Entmax` | 0.523 | 0.209 | 0.150 | 0.100 | 0.441 | 0.177 | 0.127 | 0.080 |
| `Tropical` | 0.459 | 0.127 | 0.062 | 0.040 | 0.421 | 0.124 | 0.065 | 0.040 |
| `TMU` | 0.635 | 0.357 | 0.180 | 0.105 | 0.474 | 0.221 | 0.122 | 0.063 |
| `TMU-1.4M` | 0.659 | 0.394 | 0.243 | 0.148 | 0.502 | 0.218 | 0.125 | 0.072 |
| `SoftMix` | 0.378 | 0.213 | 0.141 | 0.101 | 0.288 | 0.174 | 0.120 | 0.086 |
| PSAR | 0.575 | 0.314 | 0.184 | 0.113 | 0.479 | 0.230 | 0.136 | 0.094 |

Table 37: Accuracy of every arm after retraining under the corrected contract, on the same checkpoints under both evaluation protocols. Four seeds, shared held-out split. The parameter-matched single pair-state expert is highest at every size when the model consumes its own predictions; under ground-truth hints the ordering between it and the pool is reversed at $N \geq 32$. These numbers are not comparable with the rest of the paper, which uses a different training contract.

**Two candidate explanations for the fall-off.** The restricted read-out above and the fixed training size could each account for the fall-off away from the training size. We screened both on Floyd–Warshall with the standalone pair-state expert under the corrected contract, in four cells: a control, a read-out that scores each pair jointly from the two node states and the input edge features, training drawn across sizes $4$ to $16$ instead of $16$ alone, and the two together. Two seeds per cell on a development split reserved for this screen, criteria fixed beforehand, direction only and no intervals.

Neither change helps. The joint read-out lowers the distance error on both seeds, from $0.208$ to $0.148$, so it does what it was built to do; autonomous accuracy moves by $-4.2$ pp at $N$=16 and $-3.6$ pp at $N$=32, with the two seeds disagreeing in sign. Spreading the training budget across sizes lowers accuracy on both seeds, by $4.9$ and $9.6$ pp. The two together are the worst cell, $-9.8$ and $-12.5$ pp on both seeds.

The pattern across the four cells runs opposite to the hypothesis: the cell with the lowest distance and reachability error, $0.123$ and $0.099$ against the control's $0.208$ and $0.205$, is also the cell with the lowest output accuracy, and the ordering is monotone across all four. The pointer hint, whose read-out we left unchanged, moves the other way. Predicting the intermediate scalar state better therefore did not translate into executing the algorithm better here. Two qualifications: the screen has two seeds, and its size-coverage cells draw only a fifth of their samples at $N$=16 because the sample budget is held fixed, so they test how a fixed budget is best spent across sizes rather than size coverage with more data.

**The comparison with a reference-form read-out.** A preregistered rerun changes the harness for every model in four ways. Edge scalar and mask read-outs take the reference decoder's form, $w_l^\top h_i + w_r^\top h_j + w_e^\top e_{ij}$, with $e_{ij}$ the encoded input edge features (the reference uses the processor's current edge features, and our edge-pointer read-out is unchanged). The recurrence stops and the hint loss is masked at each sample's own trajectory length. PSAR's step encoding is normalised by that length instead of by the longest trajectory in the split. The routing regulariser ignores steps after a sample has ended. Dijkstra has no edge scalar or mask hints, so the read-out change reaches Floyd–Warshall only, and Floyd–Warshall trajectories share one length, so the other three changes reach Dijkstra only. On each task and under each training contract, ground-truth hints or the model's own soft predictions, the four experts, the parameter-matched `TMU` and PSAR are retrained with four seeds and evaluated under both contracts on new instances (CLRS eval seed 20260924, 64 per $N$). The preregistered rule selects every checkpoint on autonomous OOD-dev accuracy, so that selection is the same under both contracts; Table 38 also reports selection on teacher-forced OOD-dev accuracy. The reference-form read-out adds one $\mathrm{Linear}(16, 1)$ edge term to each of the two Floyd–Warshall edge scalar and mask decoders, so both Floyd–Warshall parameter counts in that table are $34$ above the ones given in Appendix J.4; Dijkstra's are unaffected.

On Floyd–Warshall both orderings recur. With ground-truth hints PSAR is $+7.95$ and $+7.15$ pp above the parameter-matched expert at $N$=48 and 64, and with its own predictions $-5.39$ and $-5.68$ pp below it, each on all four seeds; the two contrasts differ by $+13.34$ pp $[+6.83, +22.26]$ and $+12.82$ pp $[+5.49, +20.40]$, and both differences hold under Holm correction over the four

| Task | $N$ | ground-truth hints | own predictions | difference |
|---|---|---|---|---|
| *Checkpoints selected on autonomous OOD-dev accuracy (preregistered rule)* | | | | |
| Floyd–Warshall | 48 | $+7.95$ $[+1.55, +14.56]$ | $-5.39$ $[-7.58, -3.14]$ | $+13.34$ $[+6.83, +22.26]$ |
| Floyd–Warshall | 64 | $+7.15$ $[+2.09, +13.56]$ | $-5.68$ $[-7.60, -3.13]$ | $+12.82$ $[+5.49, +20.40]$ |
| Dijkstra | 48 | $-49.70$ $[-70.20, -25.94]$ | $-23.01$ $[-27.16, -18.99]$ | $-26.69$ $[-46.97, -5.15]$ |
| Dijkstra | 64 | $-45.84$ $[-64.84, -24.88]$ | $-13.87$ $[-18.72, -10.25]$ | $-31.98$ $[-47.80, -12.58]$ |
| *Checkpoints selected on teacher-forced OOD-dev accuracy* | | | | |
| Floyd–Warshall | 48 | $+6.93$ $[+1.75, +10.86]$ | $-2.32$ $[-7.33, +2.72]$ | $+9.25$ $[-0.39, +16.24]$ |
| Floyd–Warshall | 64 | $+6.36$ $[+1.95, +10.93]$ | $-4.57$ $[-8.47, +0.70]$ | $+10.93$ $[+2.72, +19.09]$ |
| Dijkstra | 48 | $-11.87$ $[-16.37, -7.57]$ | $-23.07$ $[-26.27, -19.21]$ | $+11.21$ $[+8.59, +13.75]$ |
| Dijkstra | 64 | $-21.01$ $[-27.08, -14.09]$ | $-14.56$ $[-17.19, -11.30]$ | $-6.45$ $[-11.29, -1.79]$ |

Table 38: PSAR minus the parameter-matched TMU (pp) with an edge read-out of the reference form, length masking and per-sample step encoding, on new held-out instances (CLRS eval seed 20260924, 64 per $N$); four seeds per cell, hierarchical paired bootstrap with $10^4$ resamples and instance indices shared across seeds, 95% intervals. *Ground-truth hints*: both models trained and evaluated with ground-truth hints; *own predictions*: both trained and evaluated on their own soft predictions; *difference*: the first minus the second. The parameter-matched expert has 1,416,159 parameters against PSAR's 1,388,581 on Floyd–Warshall and 1,521,004 against 1,423,794 on Dijkstra. The comparisons, the first checkpoint rule and the decision thresholds were fixed before these runs; under that rule two Dijkstra PSAR runs trained on ground-truth hints select a checkpoint from the router-only stage (Appendix N).

| | Teacher-forced | | | Self-rollout | | |
|---|---|---|---|---|---|---|
| | $N$=32 | $N$=48 | $N$=64 | $N$=32 | $N$=48 | $N$=64 |
| Copy last fed hint | 1.000 | 1.000 | 1.000 | – | – | – |
| Softmax | 0.778 | 0.516 | 0.384 | 0.076 | 0.053 | 0.033 |
| Entmax | 0.854 | 0.676 | 0.560 | 0.101 | 0.073 | 0.061 |
| Tropical | 0.850 | 0.681 | 0.548 | 0.076 | 0.047 | 0.036 |
| TMU | 0.957 | 0.867 | 0.741 | 0.101 | 0.073 | 0.048 |
| PSAR | 0.924 | 0.813 | 0.642 | 0.068 | 0.030 | 0.016 |

Table 39: Dijkstra: the same checkpoints under the two hint protocols, on the shared held-out split. Teacher-forced supplies the ground-truth hint at every step; self-rollout supplies the model's own argmax hint prediction from the first step on. The first row is the copy baseline, which emits the last fed hint as the output and has no processor; it is a property of the protocol, not of a trained model, so it has a single value rather than a seed mean.

preregistered ones. On the earlier held-out instances each of these values moves by less than 1 pp. Under teacher-forced selection the signs are the same, and the difference at $N$=48 includes zero ($+9.25$ pp $[-0.39, +16.24]$).

On Dijkstra the parameter-matched expert leads under both contracts, so there is no ordering to reverse: under teacher-forced selection PSAR is $-11.87$ and $-21.01$ pp below it with ground-truth hints and $-23.07$ and $-14.56$ pp below it with its own predictions. Under the preregistered rule, two of the four PSAR runs trained on ground-truth hints select a checkpoint from the router-only Stage 1 (epochs 2 and 13), whose teacher-forced accuracy at $N$=16 is 0.03 and 0.27, against 1.00 for the checkpoints the same runs select on teacher-forced accuracy; a model trained on ground-truth hints has low and noisy autonomous accuracy, and in these two runs its maximum fell in Stage 1. The preregistered Dijkstra values ($-49.70$ and $-45.84$ pp with ground-truth hints) therefore reflect checkpoint selection rather than architecture.

## O Protocol history and checkpoint provenance

Several baseline checkpoints in this paper come from earlier project phases trained under an 8-point evaluation grid, while the final PSAR main results are reported on the unified CLRS-range grid

| | Teacher-forced | | | Self-rollout | | |
|---|---|---|---|---|---|---|
| | $N$=32 | $N$=48 | $N$=64 | $N$=32 | $N$=48 | $N$=64 |
| Copy last fed hint | 0.812 | 0.250 | 0.214 | – | – | – |
| `Softmax` | 0.881 | 0.850 | 0.771 | 0.257 | 0.187 | 0.144 |
| `Entmax` | 0.873 | 0.846 | 0.751 | 0.257 | 0.207 | 0.166 |
| `Tropical` | 0.723 | 0.724 | 0.633 | 0.199 | 0.158 | 0.118 |
| `TMU` | 0.765 | 0.789 | 0.729 | 0.201 | 0.157 | 0.102 |
| PSAR | 0.902 | 0.890 | 0.841 | 0.280 | 0.217 | 0.156 |

Table 40: Bellman–Ford: the same checkpoints under the two hint protocols, on the shared held-out split. Teacher-forced supplies the ground-truth hint at every step; self-rollout supplies the model's own argmax hint prediction from the first step on. The first row is the copy baseline, which emits the last fed hint as the output and has no processor; it is a property of the protocol, not of a trained model, so it has a single value rather than a seed mean.

| | Teacher-forced | | | Self-rollout | | |
|---|---|---|---|---|---|---|
| | $N$=32 | $N$=48 | $N$=64 | $N$=32 | $N$=48 | $N$=64 |
| Copy last fed hint | 0.930 | 0.951 | 0.956 | – | – | – |
| `Softmax` | 0.203 | 0.114 | 0.068 | 0.130 | 0.066 | 0.042 |
| `Entmax` | 0.238 | 0.163 | 0.122 | 0.149 | 0.087 | 0.062 |
| `Tropical` | 0.199 | 0.122 | 0.085 | 0.112 | 0.065 | 0.050 |
| `TMU` | 0.368 | 0.181 | 0.104 | 0.144 | 0.065 | 0.042 |
| `TMU-1.4M` | 0.388 | 0.212 | 0.111 | 0.137 | 0.067 | 0.039 |
| `StepOnly` | 0.343 | 0.239 | 0.167 | 0.127 | 0.071 | 0.047 |
| PSAR | 0.387 | 0.284 | 0.192 | 0.187 | 0.109 | 0.067 |

Table 41: Floyd–Warshall: the same checkpoints under the two hint protocols, on the shared held-out split. Teacher-forced supplies the ground-truth hint at every step; self-rollout supplies the model's own argmax hint prediction from the first step on. The first row is the copy baseline, which emits the last fed hint as the output and has no processor; it is a property of the protocol, not of a trained model, so it has a single value rather than a seed mean.

$N \in \{16, 32, 48, 64\}$ (§5). We summarise the protocol history and ckpt provenance of every method that appears in Table 1, the $N$-scaling figure (Figure 1), and the axis-ceiling table (Appendix K).

| Method | Trained under | Re-eval grid for paper claims | Selection |
|---|---|---|---|
| `Softmax`, `Entmax` | Common width ($d_{\text{model}}$=192) | CLRS-range eval (shared held-out) | best OOD $N$=32 |
| `Tropical` | PSAR-matched | CLRS-range eval (shared held-out) | best OOD $N$=32 |
| `Softmax-140`, `Entmax-140` | Width-varied ($d_{\text{model}}$=140) | native training grid only | best OOD $N$=32 |
| `TMU` | Native-pool pretraining | CLRS-range eval (shared held-out) | best OOD $N$=32 |
| `RelAttn` | Common width ($d_{\text{model}}$=192) | CLRS-range eval at $N \leq 64$ | best OOD $N$=32 |
| `StaticTask`, `Uniform` | Control-router training | CLRS-range eval (shared held-out) | best OOD $N$=32 |
| **PSAR** | PSAR routing training | CLRS-range eval (shared held-out) | best OOD $N$=32 |

Table 42: Protocol-history map. All methods are re-evaluated on the unified CLRS-range grid $N \in \{16, 32, 48, 64\}$ from a single eval-set seed (88888), regardless of the training-time evaluation grid. The width-varied baselines (`Softmax-140`, `Entmax-140`) are reported only at $N$=32 (Appendix C) using their own training-time validation; they were trained for a width comparison within the single-algebra family and not for the $N$-scaling claim. The checkpoint-directory mapping for each method is provided with the released code.

All re-evaluations use the same 64-sample held-out eval seed at each $N$ (CLRS `seed`=88888, independent of train/val/test/ood-dev). Checkpoints are selected on the $N$=32 OOD-dev split for the headline numbers.