ARC: ASSIGNMENT-STRUCTURED REPRESENTATION AND CONSTRUCTION FOR ASYMMETRIC TSP

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural constructive solvers efficiently generate tours for the asymmetric traveling salesman problem (ATSP), but maintaining solution quality beyond the training scale remains challenging. With direction-dependent costs, a cheap move to one node may force another node to use an expensive alternative. This competition arises because each node in a tour must have exactly one predecessor and one successor. Assignment captures this requirement through a one-to-one pairing of source and destination nodes, providing a global reference for local choices. When this structure remains implicit, neural policies must infer the changing competition from learned representations and tour states, even at unseen scales. We propose ARC (Assignment-structured Representation and Construction), which uses assignment structure both to represent directed costs and to guide tour construction. ARC derives source and destination dual potentials and gauge-adjusted edge costs from a full-instance assignment decomposition, providing complementary node and edge inputs. The adjustment shifts every complete tour's cost by the same constant, preserving which tours are cheaper. As construction proceeds, ARC updates a pressure signal for each remaining destination through residual assignment rebalancing. Frontier Sinkhorn Pressure (FSP) uses these signals to guide choices without fully solving the remaining assignment problem at every step. Experiments show that ARC outperforms the evaluated neural baselines on standard ATSP benchmarks and generalizes to larger unseen instances. Structured inputs reduce reliance on many starts, while FSP improves policies without retraining.

1 INTRODUCTION

Learning to construct tours has become an effective approach to the traveling salesman problem, with neural policies generating feasible solutions through sequential node selections (Kool et al., 2019; Kwon et al., 2020). Each decision extends a partial path and determines the options available for subsequent choices. For the asymmetric traveling salesman problem (ATSP), recent work has strengthened neural models' ability to handle direction-dependent costs. MatNet directly encodes cost matrices (Kwon et al., 2021), RADAR combines directional node features with Sinkhorn-normalized encoder attention (Yi et al., 2026), and Edge-aware Decoding incorporates directed-edge, closure, and static lookahead features into candidate scoring (Liang et al., 2026).

Although these advances strengthen directed-cost modeling, they do not by themselves resolve conflicts between local low-cost preferences and one-to-one connection requirements. Every node requires exactly one incoming and one outgoing edge, yet several sources may rely on the same low-cost destination, and some may have few comparably cheap alternatives. A candidate edge's value therefore depends on how cheaply the remaining nodes can obtain mutually compatible connections after it is selected. The opportunity cost of selecting a destination thus changes with the partial tour even when edge costs stay fixed. Without an explicit cost-based reference for this evolving competition, policies must infer its effect on completion costs from learned features and tour states, including at larger unseen scales where more sources compete for low-cost destinations.

Assignment offers a cost-based reference by evaluating source–destination connections jointly under one-to-one constraints, so each connection is assessed against the alternatives available to other nodes. Our analysis shows that assignment information makes reference-tour connections more distinguishable than raw costs or ordinary centering, suggesting a common structural basis for repre-

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.

sentation and construction. Realizing this idea poses two challenges. At the input stage, competition is jointly determined across the cost matrix, but must be expressed through compact node and edge features that preserve distinct source and destination roles under a consistent reference. During construction, the reference must adapt to remaining matching requirements within the computational budget of sequential decisions. A fixed initial signal cannot follow changing competition, while fully solving each residual assignment adds repeated computation.

To address these challenges, we introduce **ARC (Assignment-structured Representation and Construction)**, which couples assignment-structured representation with residual assignment rebalancing. A full-instance assignment decomposition yields paired dual node features and gauge-adjusted edge costs, exposing source and destination roles alongside connection preferences under a common reference. During construction, Frontier Sinkhorn Pressure (FSP) recursively updates destination pressure to reflect competition among the remaining sources without fully solving each residual assignment problem. The policy combines this guidance with its learned scores, using assignment information both to interpret the instance and to complete the partial tour.

On standard ATSP benchmarks, ARC achieves the lowest mean tour costs among the compared neural solvers and generalizes strongly from small training instances to larger test instances. Its structured inputs also reduce the construction budget needed for high-quality tours. In ATSP1000 evaluation with instances processed individually, ARC-Base without FSP finds better tours on average with 16 starts than RADAR does with 1,000, while taking about 32% less time per instance (Appendix C.1). FSP further improves existing solvers without retraining. Its recursive updates adapt guidance to the residual problem and improve large-scale tour quality.

Our main contributions are as follows: (1) We introduce ARC to unify representation and construction under an assignment reference, and derive paired dual node features and gauge-adjusted edge inputs. (2) We extend this reference to the residual problem through FSP, recursively rebalancing destination pressure to guide decisions as the remaining matching requirements change. (3) We demonstrate improved solution quality, strong size generalization and effective use of construction budgets, with FSP also improving existing solvers without retraining.

2 An Assignment-Structured Framework

2.1 ATSP and Sinkhorn Assignment

An asymmetric traveling salesman problem (ATSP) instance is defined by a directed cost matrix $C \in \mathbb{R}^{N\times N}$. The objective is to find a tour π that visits every node exactly once and returns to its start, minimizing $J(C,\pi) = \sum_{k=0}^{N-1} C_{\pi_k,\pi_{k+1}}$, where $\pi_N = \pi_0$.

Dropping the single-cycle constraint yields the assignment relaxation (Burkard et al., 2009): nonnegative edge weights with one incoming and one outgoing unit per node and no self-loops. An integral optimum can contain disjoint cycles and lower-bounds tour cost. For instance-wise standardized costs $\bar{C}$, entropy regularization gives a soft assignment Q with unit marginals; temperature $\tau > 0$ controls dispersion (Appendix A.1). Sinkhorn alternately scales rows and columns of $K_{ij} = \exp(-\bar{C}_{ij}/\tau)$, $K_{ii} = 0$, yielding at convergence

$$Q_{ij} = \exp\big(-\bar{C}_{ij}/\tau + \alpha_i + \beta_j\big), \qquad i \neq j. \tag{1}$$

The log scalings give cost-unit potentials $u_i = \tau\alpha_i$ and $v_j = \tau\beta_j$ (Sinkhorn & Knopp, 1967; Cuturi, 2013; Peyré & Cuturi, 2019). Each balanced edge weight reflects both cost and competition.

2.2 Rethinking Sinkhorn for Neural ATSP

Assignment information beyond balanced weights. An inexpensive edge may consume a destination needed by another source. Sinkhorn's joint row–column balancing offers a natural way to examine this competition; RADAR's gains from using it in encoder attention provide empirical motivation (Yi et al., 2026). Beyond attention balancing, we examine whether cost-derived assignment potentials and adjusted costs make useful connection preferences easier to read out. Two probes separate discrimination among cost-matched alternatives from ranking all $N-1$ candidate destinations. Assignment features distinguish reference-tour edges in the matched pairs better than raw costs or simple row/column statistics (Figure 1a). Across all tested sizes, adjusted costs also rank

reference successors first more often than raw costs or ordinary centering (Figure 1b). These probes make useful connection preferences visible before a neural policy constructs a tour (Appendix D.1). This motivates presenting assignment competition directly in the inputs, rather than requiring the network to recover these relationships implicitly from raw costs.

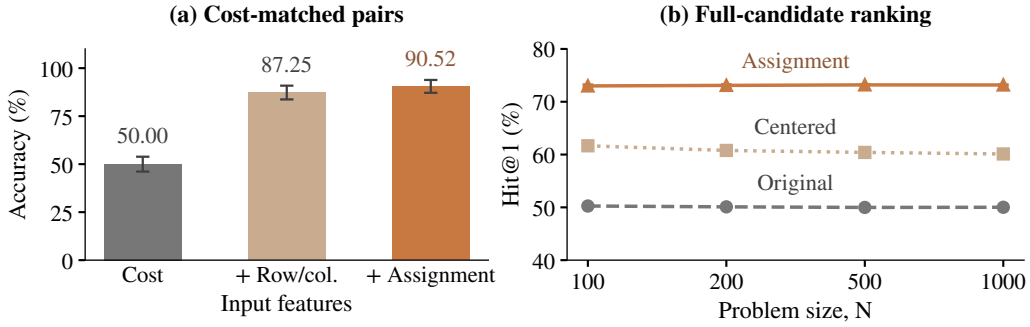


Figure 1: Assignment features help identify reference edges. (a) Cost-matched pairs (20 ATSP100 instances). (b) Reference-successor Hit@1 (1,000 instances per size). Error bars: 95% instance-bootstrap CIs.

Selected edges change the remaining requirements. Selecting $0 \to 1$ removes source 0's outgoing requirement and destination 1's incoming requirement, making node 1 the next source while retaining start node 0 for final closure (Figure 2). Node 1 still needs an outgoing edge, but the remaining destinations have lost source 0's support. Thus, fixed costs face changing matching conditions (Appendix A.3). This calls for explicit competition information on the complete instance and guidance that stays aligned with the remaining requirements during construction.

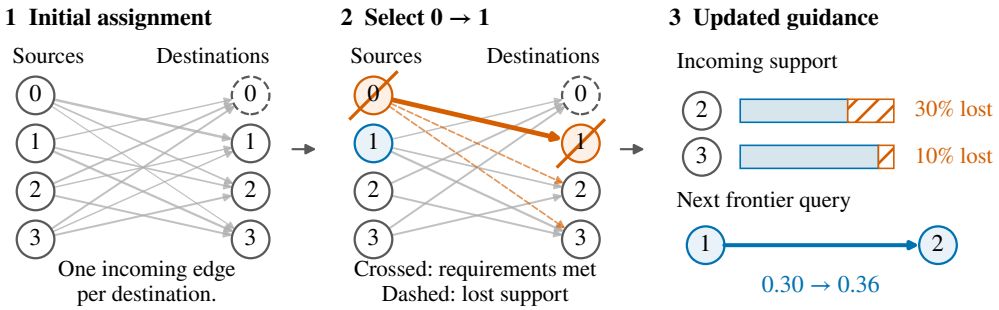


Figure 2: A selected edge changes the remaining assignment competition. One column correction raises the next-row structural weight of $1 \to 2$ from 0.30 to 0.36; these weights are not policy probabilities.

3 Assignment-structured Representation and Construction (ARC)

We introduce ARC, a neural framework that uses assignment competition to guide how an ATSP instance is represented and how its tour is constructed (Figure 3). ARC begins with the full cost matrix and exposes competing source and destination roles through paired node and edge inputs. During construction, the choices already made determine which connection requirements remain. Residual Assignment Rebalancing tracks these changes through Frontier Sinkhorn Pressure (FSP), providing guidance for the next decision. The neural policy combines this structural view with the partial path to select feasible successors under its action mask, connecting the full-instance representation to the connection requirements that remain along each construction trajectory.

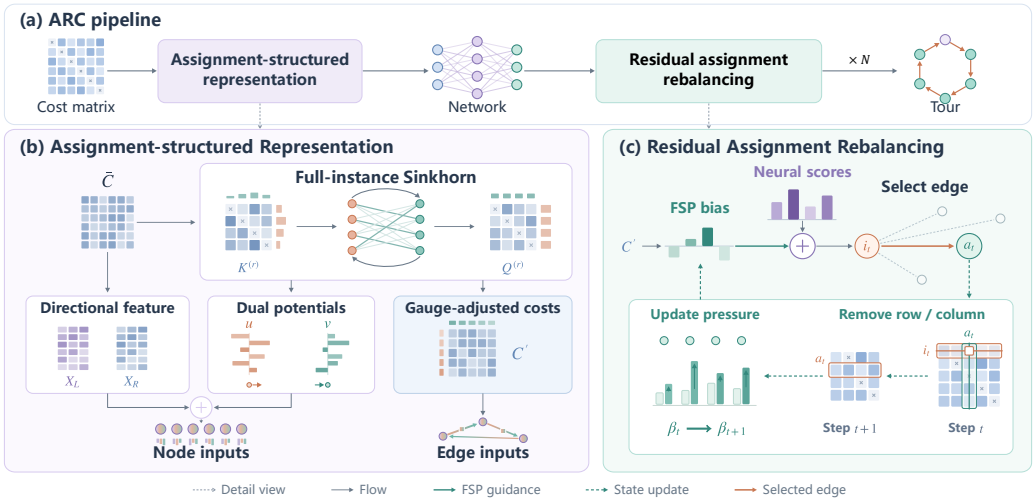


Figure 3: ARC: Assignment-structured Representation and Construction. (a) ARC connects a full-instance assignment representation to residual assignment rebalancing throughout tour construction. (b) Assignment potentials enter node features and adjust edge costs. (c) FSP biases neural scores using residual pressure, updated after each selection.

3.1 Assignment-structured Representation

Node and edge inputs provide complementary views of assignment competition. The node potentials identify source and destination matching roles, while Gauge expresses connection preferences under the same reference. Deriving both from one assignment decomposition lets the model interpret an edge relative to the endpoint roles that shape its value.

We apply finite Sinkhorn iterations to $K_{ij}^{(r)} = \exp(-\bar{C}_{ij}/\tau_r)$ for $i \neq j$, with $K_{ii}^{(r)} = 0$ and temperature $\tau_r > 0$. The resulting scaling vectors x, y define the soft coupling $Q^{(r)} = \mathrm{diag}(x)K^{(r)}\,\mathrm{diag}(y)$. Its cost-unit potentials are $u_i = \tau_r \log x_i$ and $v_j = \tau_r \log y_j$. Joint balancing makes u_i reflect source alternatives and competition, and v_j reflect destination demand and competing sources' alternatives. The paired edge input is

$$C'_{ij} = \bar{C}_{ij} - u_i - v_j = -\tau_r \log Q_{ij}^{(r)}, \qquad i \neq j. \tag{2}$$

Every complete tour uses each source and destination once, so the gauge adjustment subtracts the same $\sum_i u_i + \sum_j v_j$ from all tour costs, preserving their ordering (Appendix A.1). Lower C'_{ij} means greater matching weight after accounting for source and destination competition.

We retain SVD-based directional features z_i^{SVD} computed from $\bar{C}$ (Yi et al., 2026) and pair them with assignment potentials, so the node inputs encode both directed distance structure and matching competition. We fix $\mathrm{mean}(u) = 0$ by shifting u and v oppositely, preserving $u_i + v_j$. This removes the arbitrary offset between source and destination potentials without changing the adjusted edges. Only the node channels are normalized by their joint root mean square r (Appendix A.1):

$$h_i^0 = W_1 z_i^{\mathrm{SVD}} + W_2 \frac{[u_i, v_i]}{r + \epsilon_n}, \tag{3}$$

W_1, W_2 are learnable projections, and $\epsilon_n > 0$ stabilizes normalization. Joint RMS normalization sets the scale of the dual node channels. The edge adjustment retains the potentials in cost units, keeping C' consistent with the assignment decomposition and its edge-cost meaning in mixed node/edge attention and cost-dependent computation. The model learns by minimizing original tour cost, without requiring assignment labels (Algorithm 1).

3.2 Residual Assignment Rebalancing

How edge selection changes the residual assignment. For a path starting at s, ending at i_t, and leaving U_t unvisited, the remaining source and destination sets are $R_t = U_t \cup \{i_t\}$ and $A_t =$

$U_t \cup \{s\}$. The start remains a destination for closure. Selecting $a_t \in U_t$ consumes source row i_t and destination column a_t. The selected node remains a source and becomes $i_{t+1} = a_t$; the unvisited set loses a_t. For a nonnegative reference B_t with unit marginals on $R_t \times A_t$, removing the current source leaves each surviving column with mass $d_{t+1}(j) = 1 - B_t(i_t, j)$. Destinations that relied more on the removed source lose more support and require larger column-side compensation. FSP estimates this loss from the frontier row and accumulates log corrections in a destination vector (Appendix A.2). Normalizing the next source on demand avoids storing a full residual matrix.

From assignment representations to frontier states. FSP uses the representation's adjusted costs through the log kernel $L_{ij} = -C'_{ij}/\tau_s = (\tau_r/\tau_s)\log Q_{ij}^{(r)}$ for $i \neq j$. It masks self-loops with $L_{ii} = -\infty$ and uses a separate temperature $\tau_s > 0$. The representation temperature controls competition exposed to the encoder; the structural temperature controls the sharpness of the decoding preference. Keeping them separate lets guidance vary without changing the input decomposition. Separate finite Sinkhorn initialization of $\exp(L)$ produces α_0, β_0; the row constant $\alpha_0(i)$ cancels under row normalization. Across multiple starts, ARC reuses the full-instance encoding H and kernel L. Initial pressure provides a common starting reference, but each trajectory maintains its own copy of β_0. Different paths consume different source and destination requirements, so their pressure states evolve with their respective residual problems. The current-row score $g_t(j) = L_{i_t j} + \beta_t(j)$ defines the frontier query

$$\rho_t(j) = \frac{\exp g_t(j)}{\sum_{k \in A_t} \exp g_t(k)}, \qquad j \in A_t. \tag{4}$$

Destination states make scores path-dependent; normalization over A_t computes row scaling and retains the start for closure, while policy actions are restricted to U_t. After choosing a_t, FSP substitutes the cached pre-selection ρ_t for $B_t(i_t, \cdot)$:

$$\beta_{t+1}(j) = \beta_t(j) - \eta \log \max\{1 - \rho_t(j), \epsilon\}, \qquad j \in A_{t+1}, \tag{5}$$

where η sets correction strength and $\epsilon > 0$ is a numerical floor. The next query normalizes source a_t over A_{t+1} using β_{t+1}, accounting for column removal without a full residual assignment solve. Each step queries one current-source row and updates a destination vector, matching structural maintenance to the granularity of edge selection. The shared instance representation remains available without re-encoding, while accumulated support losses make guidance depend on the partial tour.

3.3 Structure-guided Decision

The decoder combines its learned score $s_t^\theta(j)$ with FSP guidance before nonlinear clipping. Because a shared offset can change score differences after clipping, we center the structural log probabilities over feasible candidates:

$$\widetilde{\ell}_t(j) = \log \rho_t(j) - \frac{1}{|U_t|} \sum_{k \in U_t} \log \rho_t(k). \tag{6}$$

For partial path $\pi_{:t}$, guidance weight λ, and clipping amplitude $c_{\text{clip}} > 0$, the masked policy is

$$p_\theta(a_t = j \mid \pi_{:t}, C) = \text{softmax}_{j \in U_t}\left[c_{\text{clip}} \tanh\Big(s_t^\theta(j) + \lambda \widetilde{\ell}_t(j)\Big)\right]. \tag{7}$$

The assignment reference coordinates local connection requirements, while the learned policy uses path context to assemble a single tour. Updating the path and pressure together connects assignment-aware inputs to construction until closure (Algorithm 2; complexity and overhead in Appendix C.6). This interface supports both policy training and inference-time guidance of existing policies.

4 Experiments

We evaluate whether explicit assignment competition improves tour construction across problem sizes and cost distributions. Synthetic ATSP benchmarks and real road networks test solution quality and generalization, while decoding-budget comparisons examine how effectively ARC uses multiple starts. Controlled ablations and representation analyses assess the paired node and edge inputs; fixed-policy comparisons and residual-state diagnostics test the value of FSP-guided construction.

4.1 EXPERIMENTAL SETUP

Datasets and baselines. We train synthetic-benchmark models on ATSP100 and test without fine-tuning on 1,000 instances per size $N \in \{100, 200, 500, 1000\}$ and 100 ATSP2000 instances. Road experiments use RRNCO's datasets. RADAR is the direct backbone baseline; LKH-3 (Helsgaun, 2017) supplies reference tours. Controlled studies use the official RADAR checkpoint or independently trained models; literature sources are marked in the tables.

Model variants. ARC-Base uses Dual and Gauge with FSP disabled. ARC-I adds inference-time FSP to the same trained weights; ARC-T trains and tests with FSP. ARC-I and ARC-T use $\lambda = 0.5$ on synthetic main benchmarks; Appendix B.2 specifies road settings.

Evaluation. Main-benchmark RADAR and ARC use greedy decoding with $P = N$ starts and no augmentation; ATSP2000 uses $P = 100$ for every neural solver. We report mean original tour cost, LKH-reference gap (%), and total evaluation time in minutes. Main-benchmark runs use one NVIDIA RTX 3090 GPU. Evaluation protocols are summarized in Appendix B.3.

4.2 MAIN RESULTS

Standard ATSP benchmarks. ARC-T achieves the lowest cost among the compared neural solvers at all four sizes (Table 1). ARC-I improves the fixed ARC-Base policy, and FSP training yields further gains: on ATSP1000, the reference gaps are 2.38%, 1.28%, and 1.03% for ARC-Base, ARC-I, and ARC-T. Across three independent training runs per configuration, ARC-T has lower mean cost than ARC-Base at every size (mean $\pm$ standard deviation in Appendix Table 12).

Table 1: Performance on standard ATSP benchmarks.

Method	ATSP100			ATSP200			ATSP500			ATSP1000		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
LKH-100†	1.5643	0.00	1.08	1.5721	0.00	2.44	1.5763	0.00	7.32	1.5739	0.00	21.92
LKH-1000†	1.5643	*	5.91	1.5721	*	16.40	1.5763	*	44.45	1.5739	*	102.60
GLOP†	1.8848	20.49	2.75	2.0584	30.93	3.07	2.2345	41.76	3.72	2.3429	48.86	7.29
UDC-x250†	1.5921	1.78	61.20	1.7577	11.81	122.40	2.3810	51.05	384.60	3.0734	95.27	1147.20
MatNet†	1.5969	2.08	0.02	1.6543	5.23	0.13	1.8610	18.10	1.34	2.1821	38.64	11.11
ELG†	1.5982	2.17	0.06	1.6423	4.47	0.27	1.7456	10.74	2.60	1.8441	17.17	22.31
ReLD†	1.5900	1.64	0.03	1.6310	3.75	0.15	1.7873	13.39	1.49	2.0723	31.67	12.18
RADAR	1.5753	0.70	0.04	1.5871	0.96	0.15	1.6072	1.96	1.48	1.6314	3.66	11.85
Edge-aware‡	1.5741	0.63	0.05	1.5852	0.83	0.38	1.6032	1.71	5.31	1.6170	2.73	43.44
ARC-Base	1.5729	0.55	0.04	1.5804	0.53	0.15	1.5926	1.04	1.49	1.6112	2.38	11.91
ARC-I	1.5723	0.51	0.04	1.5799	0.49	0.20	1.5887	0.79	2.13	1.5939	1.28	18.21
ARC-T	**1.5711**	**0.44**	0.04	**1.5783**	**0.39**	0.19	**1.5855**	**0.58**	2.12	**1.5900**	**1.03**	18.22

Note. † Results reported by Yi et al. (2026). ‡ Results reported by Liang et al. (2026). RADAR and ARC results are from evaluations in this work. MatNet denotes the MatNet-Single (Random) variant. Literature results retain the reference solutions and evaluation settings of their source publications. Gap is reported in percent; time is reported in minutes.

Using the decoding budget. Structured inputs reduce reliance on many starts: in the ATSP1000 single-instance evaluation, ARC-Base with 16 starts finds better tours on average than RADAR with 1,000, at about 32% lower measured latency. ARC-I also yields lower mean cost than ARC-Static using 768 rather than 1,000 starts, with lower latency (Appendix Figure 6). ARC-Static keeps the initial destination pressure fixed and shares trained weights with ARC-I, so the lower start requirement reflects the value of updating destination pressure during construction, beyond the benefit already provided by their shared representation.

Generalization to 2,000 nodes. To test size generalization at a fixed rollout budget, we evaluate 100 instances with 2,000 nodes. Each neural model is trained on instances with 100 nodes and evaluated using 100 starts without any fine-tuning. ARC-I and ARC-T improve on ARC-Base and RADAR, reaching reference gaps of 2.09% and 2.15%, respectively (Table 2). Residual guidance remains useful at twenty times the training size.

Table 2: Generalization to ATSP2000.

Method	Cost ↓	Gap (%) ↓	Time (min)
LKH-100	1.5716	0.00	36.64
LKH-1000	1.5716	0.00	61.05
RADAR	1.7023	8.32	2.97
ARC-Base	1.6648	5.93	2.97
ARC-I	**1.6044**	**2.09**	5.30
ARC-T	1.6054	2.15	5.37

Real-world road networks. Road-trained models test changes in city and location-sampling distributions. ARC-I improves on RADAR and ARC-Base across training-city, unseen-city, and shifted location-sampling splits; it also improves on reported RRNCO gaps on both OOD splits (Table 3). On these road benchmarks, matrix-only ARC supports coordinate-free matrix augmentation: using eight equivalent cost-matrix variants and 8×100 starts, ARC-I achieves the lowest neural gaps shown on ID, OOD-city, and OOD-cluster: 0.82%, 0.95%, and 0.94%, respectively (Appendix Table 13).

Table 3: In-distribution and out-of-distribution performance on real-world road networks.

Method	Input	Aug.	In-distribution		OOD (city)		OOD (cluster)	
			Gap ↓ (%)	Time (min)	Gap ↓ (%)	Time (min)	Gap ↓ (%)	Time (min)
LKH†	Matrix	—	0.00	96.00	0.00	96.00	0.00	96.00
RRNCO†	Matrix + coords	×8	1.80	0.38	2.27	0.38	2.25	0.38
RADAR	Matrix	×1	2.11	0.13	2.35	0.12	2.47	0.12
ARC-Base	Matrix	×1	1.97	0.20	2.19	0.20	2.15	0.19
ARC-I	Matrix	×1	1.92	0.32	2.14	0.27	2.00	0.34
ARC-T	Matrix	×1	1.97	0.33	2.31	0.29	2.28	0.34
ARC-I	Matrix	×8	**0.82**	1.31	**0.95**	1.18	**0.94**	1.19

Note. † Son et al. (2026). RADAR and ARC use the ATSP backbones evaluated in this work. All splits use $N = 100$. Bold: lowest displayed neural gap per split; LKH supplies the reference. Aug.: augmentation factor. Time: minutes per 1,280-instance split.

4.3 Ablation Study

Table 4: Ablation study of ARC.

(a) Component combinations							Mean tour cost ↓	
Representation	$N = 100$		$N = 200$		$N = 500$		$N = 1000$	
	FSP off	FSP on	FSP off	FSP on	FSP off	FSP on	FSP off	FSP on
RADAR	1.5751	1.5745	1.5872	1.5860	1.6071	1.6027	1.6314	1.6127
ARC-Base (w/o Gauge)	1.5744	1.5739	1.5872	1.5862	1.6125	1.6072	1.6470	1.6230
ARC-Base (w/o Dual)	1.5731	1.5726	1.5814	1.5808	1.5959	1.5914	1.6164	1.5978
ARC-Base	**1.5727**	**1.5722**	**1.5806**	**1.5800**	**1.5926**	**1.5885**	**1.6109**	**1.5939**

(b) Retrained input alternatives				Mean tour cost ↓
Representation	$N = 100$	$N = 200$	$N = 500$	$N = 1000$
RADAR	1.5754	1.5870	1.6068	1.6318
Mean+DC	1.5750	1.5875	1.6078	1.6321
ARC-Base	**1.5728**	**1.5807**	**1.5928**	**1.6114**

Component contributions. Gauge provides the largest representation gains (Table 4a). Dual's benefit depends on the edge representation: at $N = 500, 1000$, it further improves Gauge with FSP both off and on. This conditional benefit supports the common node/edge competition reference in Section 3.1. FSP also improves every fixed policy at all tested sizes.

Assignment representations vs. mean centering. ARC-Base outperforms Mean+DC at every size when both are independently trained with equal budgets and tested without FSP (Table 4b; Appendix D.2). The tested row–column means and double-centered edges do not reproduce this advantage, supporting the paired inputs' assignment competition reference (Section 3.1).

4.4 Cost Representation Analysis

Robustness to row and column shifts. Under row and column shifts that preserve tour rankings, ARC-Base's relative cost increase at $\lambda_g = 1$ is about $1/128$ of RADAR's (Appendix Figure 7). Scoring selected tours on original costs measures decision changes at a fixed routing objective.

Varying asymmetry. ARC-Base has the lowest mean cost across all six conditions: three asymmetry levels and $N = 50, 100$ (Table 5). Each level uses separate $N = 50$ training, a common decoder, and FSP off; encoder capacities differ (Appendix B.2). Its advantage over RADAR grows across the tested asymmetry levels at both sizes, persisting at twice the training size.

Table 5: Mean tour cost across asymmetry levels (lower is better).

N	σ	MatNet	ICAM	RRNCO	RADAR	ARC-Base
50	0.1	5.6840	5.7159	5.7487	5.6305	**5.6269**
50	0.2	5.4398	5.5394	5.6567	5.3570	**5.3333**
50	0.3	5.9114	4.9980	5.3522	4.6886	**4.5717**
100	0.1	8.0437	8.1124	8.2290	7.9735	**7.8704**
100	0.2	7.8936	7.9838	8.2262	7.9301	**7.5552**
100	0.3	9.3834	7.5300	8.1781	7.1724	**6.6494**

4.5 Improving Trained Solvers with FSP

Gains without retraining. FSP reduces mean reference gaps for all four frozen backbones at every size under their respective protocols. In the ARC-Base panel, the same trained weights are evaluated as ARC-Base and ARC-I (Figure 4; Appendix B.3).

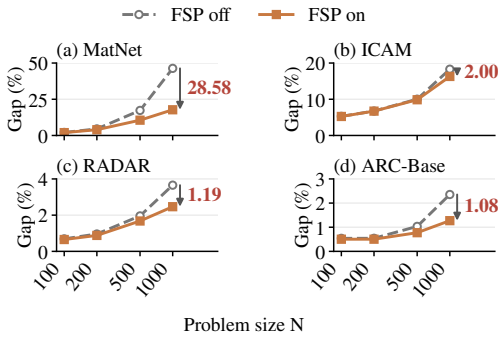


Figure 4: FSP gains without retraining.

Table 6: Guidance controls on ATSP1000.

	Cost change vs. None (‰) ↓	
Guidance	RADAR	ARC-Base
None	0.00	0.00
Random bias	439.69	42.41
LogK-only	8.05	−10.33
Static pressure	−10.83	−10.32
FSP	**−11.47**	**−10.57**

Updating guidance as the residual problem changes. Recursive FSP has the lowest mean cost on both fixed ATSP1000 backbones (Table 6). It improves on Static at $N = 500, 1000$ for all three independently trained ARC-Base policies (Appendix Table 17(c)). On ARC-Base, it also improves on LogK-only, whose kernel already includes Gauge-adjusted assignment information.

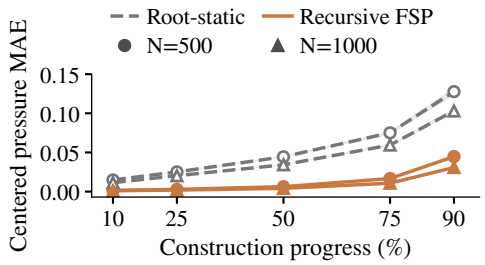


Figure 5: FSP improves pressure tracking. Shared ARC-I trajectories and residual problems; recomputed references; 50 instances/size; 95% instance-bootstrap bands.

FSP during tour construction. Recursive maintenance aligns guidance with remaining connection requirements. FSP tracks pressure more accurately than fixed initial pressure at all five depths and both scales (Figure 5). The difference grows late: at 90% progress, centered pressure MAE falls by about 65%–70% (Appendix D.3). In separate frozen-RADAR tests at $N = 500, 1000$, reading guidance in the final third gives the largest mean cost reduction. All windows update state throughout; only the reading period varies (Appendix D.5). Together, these observations show the stage-dependent value of residual guidance during construction.

5 Related Work

Constructive neural solvers and state representations. Learned construction builds on Pointer Networks (Vinyals et al., 2015), reinforcement learning (Bello et al., 2017; Nazari et al., 2018), and attention (Kool et al., 2019); POMO (Kwon et al., 2020) exploits equivalent tours through multiple starts. Graph embeddings represent partial solutions (Khalil et al., 2017), LEHD models remaining-node relationships (Luo et al., 2023), BQ-NCO reduces states through bisimulation (Drakulic et al., 2023), and ReLD improves light decoders' use of static embeddings (Huang et al., 2025). ARC scores candidates using explicit competition among remaining sources and destinations.

Asymmetric representations and structured decoding. MatNet encodes cost matrices (Kwon et al., 2021); UniCO extends this formulation through problem reductions (Pan et al., 2025). RADAR combines directional SVD embeddings with Sinkhorn-normalized encoder attention (Yi et al., 2026). ARC adds assignment-derived dual node features and gauge-adjusted edges. Edge-aware Decoding uses directed-edge, closure and static lookahead cues (Liang et al., 2026); FSP tracks destination competition through residual assignment rebalancing.

Assignment structure and dual information. Classical assignment algorithms solve one-to-one matching (Kuhn, 1955; Munkres, 1957); matrix scaling and entropic transport compute soft matchings and potentials (Sinkhorn & Knopp, 1967; Cuturi, 2013) through marginal corrections (Benamou et al., 2015). Learned duals warm-start matching algorithms (Dinitz et al., 2021), while NeuroLKH predicts node penalties and edge scores to guide LKH search (Xin et al., 2021). ARC computes assignment potentials from costs to link node features, adjusted edges and residual guidance.

Generalization and inference enhancement. Generalization also depends on training and solver design (Joshi et al., 2021): Omni-VRP supports adaptation across sizes/distributions (Zhou et al., 2023), GLOP and UDC use decomposition (Ye et al., 2024; Zheng et al., 2024), and ELG combines global/local policies (Gao et al., 2024). DACT learns iterative improvement (Ma et al., 2021), DPDP guides restricted dynamic programming (Kool et al., 2022), and EAS adapts parameters per instance (Hottung et al., 2022); FSP adjusts construction scores using residual assignment pressure.

6 Conclusion

We have presented ARC, a unified framework that incorporates assignment structure into neural representation and sequential construction for the asymmetric traveling salesman problem. Assignment-structured representation exposes global matching preferences through node and edge inputs, while residual assignment rebalancing maintains destination pressure as the remaining requirements change. Together, these interfaces allow a neural policy to use one-to-one competition throughout tour construction. Experiments demonstrate improved tour quality, strong cross-size generalization and effective use of construction budgets, with FSP also improving existing neural solvers without retraining. Future work will explore learned guidance strengths, temperatures and update strategies to adapt structural guidance and computational effort to the residual problem.

AI USE STATEMENT

We used generative AI tools to assist with language editing, literature searches, mathematical analysis, code implementation, analysis and interpretation of experimental results, and figure preparation. The authors developed the core research ideas and methodology, designed the experiments and code architecture, and led manuscript writing and figure design and refinement. They reviewed all AI-assisted materials and made the final scientific and editorial decisions.

ETHICS STATEMENT

This work studies neural ATSP solvers using synthetic instances and previously released road-network distance matrices (Appendix B.1). The experiments evaluate solution quality and computational efficiency. Deployment in real-world routing systems would require additional assessment of safety, privacy, and fairness. Third-party data and software are subject to their respective licenses and attribution requirements.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A provide the mathematical derivations and pseudocode for ARC. The appendices document the datasets, implementation and experimental settings, evaluation protocols, and additional analyses, including results across independent training runs. Source code is provided in the Supplementary Material.

REFERENCES

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *ICLR Workshop*, 2017. URL https://arxiv.org/abs/1611.09940.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015. doi: 10.1137/141000439. URL https://epubs.siam.org/doi/10.1137/141000439.

Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, 2009. ISBN 978-0-89871-663-4. URL https://tugraz.elsevierpure.com/en/publications/assignment-problems/.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL https://papers.nips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f445ba15f0f05c26e1d24f908ea78d60-Abstract.html.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

Alan Frieze and Gregory B. Sorkin. The probabilistic relationship between the assignment and asymmetric traveling salesman problems. *SIAM Journal on Computing*, 36(5):1435–1452, 2007. doi: 10.1137/S0097539701391518. URL https://www.math.cmu.edu/users/af1p/Texfiles/atsp.pdf.

Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6914–6922, 2024. doi: 10.24963/ijcai.2024/764. URL https://www.ijcai.org/proceedings/2024/0764.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53 (2):217–288, 2011. doi: 10.1137/090771806. URL https://epubs.siam.org/doi/10.1137/090771806.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017.

André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.05126.

Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/447ac93bf22099aa346a45577376492d-Abstract-Conference.html.

Chaitanya K. Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning TSP requires rethinking generalization. In *27th International Conference on Principles and Practice of Constraint Programming*, volume 210 of *Leibniz International Proceedings in Informatics*, pp. 33:1–33:21, 2021. doi: 10.4230/LIPIcs.CP.2021.33. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CP.2021.33.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1803.08475.

Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, volume 13292 of *Lecture Notes in Computer Science*, pp. 190–213. Springer, 2022. doi: 10.1007/978-3-031-08011-1_14. URL https://wouterkool.github.io/publication/deep-policy-dynamic-programming/.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955. doi: 10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/10.1002/nav.3800020109.

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

Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/29539ed932d32f1c56324cded92c07c2-Abstract.html.

Li Liang, Jinbiao Chen, and Zizhen Zhang. Edge-aware decoding for neural asymmetric routing. *arXiv preprint arXiv:2606.02136*, 2026. URL https://arxiv.org/abs/2606.02136.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1c10d0c087c14689628124bbc8fa69f6-Abstract-Conference.html.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/5c53292c032b6cb8510041c54274e65f-Abstract.html.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957. doi: 10.1137/0105003. URL https://epubs.siam.org/doi/10.1137/0105003.

MohammadReza Nazari, Afshin Oroojlooy, Lawrence V. Snyder, and Martin Takáč. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf.

Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. UniCO: On unified combinatorial optimization via problem reduction to matrix-encoded general TSP. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3acefc84f1355f5f974195bb9f6014a4-Abstract-Conference.html.

Gabriel Peyré and Marco Cuturi. Computational optimal transport with applications to data sciences. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, February 2019. ISSN 1935-8237. doi: 10.1561/2200000073. URL https://doi.org/10.1561/2200000073.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. URL https://msp.org/pjm/1967/21-2/pjm-v21-n2-p14-s.pdf.

Jiwoo Son, Zhikai Zhao, Federico Berto, Chuanbo Hua, Zhiguang Cao, Changhyun Kwon, and Jinkyoo Park. RRNCO: Towards real-world routing with neural combinatorial optimization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 91794–91818, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/9446c291a8744a125a0bda5b18f4d5a1-Paper-Conference.pdf.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://papers.nips.cc/paper_files/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://link.springer.com/article/10.1007/BF00992696.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/3d863b367aa379f71c7afc0c9cdca41d-Abstract.html.

Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 20284–20292, 2024. doi: 10.1609/aaai.v38i18.30009. URL https://ojs.aaai.org/index.php/AAAI/article/view/30009.

Hang Yi, Ziwei Huang, Yining Ma, and Zhiguang Cao. RADAR: Learning to route with asymmetry-aware distance representations. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d28077e5ff52034cd35b4aa15320caea-Abstract-Conference.html.

Zhi Zheng, Changliang Zhou, Xialiang Tong, Mingxuan Yuan, and Zhenkun Wang. UDC: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0b8e4c8468273ee3bafb288229c0acbc-Abstract-Conference.html.

Changliang Zhou, Xi Lin, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Instance-conditioned adaptation for large-scale generalization of neural routing solver. *IEEE Transactions on Intelligent Transportation Systems*, 27(7):8270–8284, 2026. ISSN 1524-9050. doi: 10.1109/TITS.2026.3674538. URL https://doi.org/10.1109/TITS.2026.3674538.

Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 42769–42789, 2023. URL https://proceedings.mlr.press/v202/zhou23o.html.

A METHOD AND ALGORITHM DETAILS

A.1 ASSIGNMENT-STRUCTURED REPRESENTATION

For instance-wise standardized costs $\bar{C}$, the assignment relaxation is

$$\begin{aligned} \min_{X \geq 0} \quad & \sum_{i \neq j} \bar{C}_{ij} X_{ij} \\ \text{s.t.} \quad & X\mathbf{1} = \mathbf{1}, \qquad X^\top \mathbf{1} = \mathbf{1}, \\ & X_{ii} = 0, \qquad \forall i. \end{aligned} \tag{8}$$

Its feasible set $\mathcal{A}$ enforces one unit of incoming and outgoing mass per node, excluding self-loops. An integral optimum exists but can have multiple cycles, so its cost lower-bounds the optimal tour under $\bar{C}$ (Frieze & Sorkin, 2007). For independent uniform $[0, 1]$ costs, the assignment–tour gap vanishes with high probability as size grows (Frieze & Sorkin, 2007). The entropically regularized problem is

$$Q = \arg\min_{X \in \mathcal{A}} \left[\sum_{i \neq j} \bar{C}_{ij} X_{ij} + \tau \sum_{i \neq j} X_{ij} (\log X_{ij} - 1) \right], \qquad \tau > 0. \tag{9}$$

where $0 \log 0 = 0$. Alternating row and column scaling computes the coupling in Equation (1) (Sinkhorn & Knopp, 1967; Cuturi, 2013).

Subtracting source and destination potentials preserves complete-tour cost ordering because each tour uses every source and destination once:

$$\sum_{(i,j) \in \pi} \bar{C}_{ij} = \sum_{(i,j) \in \pi} C'_{ij} + \sum_i u_i + \sum_j v_j. \tag{10}$$

The fixed offset changes cost values, without changing their ordering; it does not imply that a neural policy is invariant to the transformed inputs. After fixing $\mathrm{mean}(u) = 0$ by opposite shifts of u and v, the joint scale used only for the node channels is

$$r = \sqrt{\frac{1}{2N} \sum_i (u_i^2 + v_i^2)}. \tag{11}$$

Algorithm 1 normalizes all N^2 finite entries, including the diagonal, and uses RADAR's randomized SVD features (Yi et al., 2026; Halko et al., 2011). In both algorithms, $\mathrm{LogSinkhorn}(L, T; 0, 0)$ starts log scalings at zero and performs T row-then-column updates with unit marginals on finite entries.

Edge adjustment uses the cost-unit potentials before joint RMS normalization, which is applied only to node features. Both projections include biases. Road inputs use Min–Max normalization (Appendix B.2).

A.2 TOUR CONSTRUCTION WITH FSP

For the ideal balanced reference in Section 3.2, column correction rescales surviving destinations by reciprocal remaining mass. With a floor $\epsilon > 0$, its log increment is

$$-\log \max\{d_{t+1}(j), \epsilon\} = -\log \max\{1 - B_t(i_t, j), \epsilon\}. \tag{12}$$

Destinations that lose more support receive stronger correction. FSP substitutes the cached frontier row for $B_t(i_t, \cdot)$ in this increment, giving Equation (5); it does not recompute a balanced residual matrix at every decision.

Algorithm 2 shares (H, L, β_0) across trajectories, each with its own pressure and sets. Let $V = \{0, \ldots, N-1\}$.

The two branches use independent temperatures, iteration counts and initial scalings. The start remains in A_t for closure, while actions lie in U_t. Centering applies to log probabilities over U_t, not the pressure vector. The update uses the cached frontier distribution from before action selection.

Algorithm 1 Assignment-structured Representation

Input: finite cost matrix $C \in \mathbb{R}^{N\times N}$; rank k, temperature τ_r, iterations T_r; learned affine maps W_1, W_2.
Output: node embeddings h^0 and gauge-adjusted edge inputs C'.

$\bar{C} \leftarrow (C - \mathrm{mean}(C))/(\mathrm{std}_{\mathrm{sample}}(C) + 10^{-9})$.
$z^{\mathrm{SVD}} \leftarrow \mathrm{SVDFeatures}(\bar{C}, k)$.
$L^{(r)}_{ij} \leftarrow -\bar{C}_{ij}/\tau_r$ for $i \neq j$; $L^{(r)}_{ii} \leftarrow -\infty$.
$(\alpha^{(r)}, \beta^{(r)}) \leftarrow \mathrm{LogSinkhorn}(L^{(r)}, T_r; 0, 0)$.
$u \leftarrow \tau_r \alpha^{(r)}$, $v \leftarrow \tau_r \beta^{(r)}$.
$m \leftarrow \mathrm{mean}(u)$; $u \leftarrow u - m$, $v \leftarrow v + m$.
$C'_{ij} \leftarrow \bar{C}_{ij} - u_i - v_j$ for all i, j *cost-unit potentials.*
$r \leftarrow \sqrt{\frac{1}{2N}\sum_i (u_i^2 + v_i^2)}$; $\widehat{h}^{\mathrm{dual}}_i \leftarrow [u_i, v_i]/(r + 10^{-8})$.
$h^0_i \leftarrow W_1 z^{\mathrm{SVD}}_i + W_2 \widehat{h}^{\mathrm{dual}}_i$ for all i.
return (h^0, C').

Algorithm 2 ARC Tour Construction with FSP

Input: C, θ, P starts; representation (k, τ_r, T_r); FSP $(\tau_s, T_s, \eta, \epsilon, \lambda)$; clipping c_{clip}.
Output: the lowest-original-cost closed tour among the P trajectories.

$(h^0, C') \leftarrow$ Algorithm 1$(C; k, \tau_r, T_r)$; $H \leftarrow \mathrm{Encoder}_\theta(h^0, C')$.
$L_{ij} \leftarrow -C'_{ij}/\tau_s$ for $i \neq j$; $L_{ii} \leftarrow -\infty$.
$(\alpha_0, \beta_0) \leftarrow \mathrm{LogSinkhorn}(L, T_s; 0, 0)$ *independent FSP initialization.*
for each start $s \in \{0, \ldots, P-1\}$ **in parallel do**
  $\pi \leftarrow [s]$, $i_0 \leftarrow s$, $U_0 \leftarrow V \setminus \{s\}$, $A_0 \leftarrow V$, $R_0 \leftarrow V$, $\beta^{(s)}_0 \leftarrow \mathrm{copy}(\beta_0)$.
  for $t = 0, \ldots, N-2$ **do**
    $g_t(j) \leftarrow L_{i_t j} + \beta^{(s)}_t(j)$ if $j \in A_t \setminus \{i_t\}$; otherwise $g_t(j) \leftarrow -\infty$.
    $\ell_t(j) \leftarrow g_t(j) - \mathrm{logsumexp}_{k\in A_t} g_t(k)$; $\rho_t(j) \leftarrow \exp \ell_t(j)$.
    $\widetilde{\ell}_t(j) \leftarrow \ell_t(j) - |U_t|^{-1} \sum_{k\in U_t} \ell_t(k)$ for $j \in U_t$; set it to zero otherwise.
    $s^\theta_t \leftarrow \mathrm{DecoderScores}_\theta(H, s, i_t, U_t)$.
    $q_t(j) \leftarrow c_{\mathrm{clip}} \tanh\big(s^\theta_t(j) + \lambda \widetilde{\ell}_t(j)\big)$; then set $q_t(j) \leftarrow -\infty$ for $j \notin U_t$.
    $p_t \leftarrow \mathrm{softmax}(q_t)$; $a_t \leftarrow \arg\max_j p_t(j)$ (or sample from p_t when specified).
    $(\beta^{(s)}_{t+1}, A_{t+1}) \leftarrow \mathrm{FSPUpdate}(\beta^{(s)}_t, A_t, a_t, \rho_t)$.
    $R_{t+1} \leftarrow R_t \setminus \{i_t\}$; $U_{t+1} \leftarrow U_t \setminus \{a_t\}$; $i_{t+1} \leftarrow a_t$; append a_t to π.
  end for; append s and store the closed tour $\pi^{(s)}$ with cost $J(C, \pi^{(s)})$.
end for; **return** the stored tour minimizing $J(C, \pi^{(s)})$.

Subprocedure $\mathrm{FSPUpdate}(\beta, A, a, \rho)$
$A^+ \leftarrow A \setminus \{a\}$; $\beta^+ \leftarrow \mathrm{copy}(\beta)$.
$\beta^+(j) \leftarrow \beta(j) - \eta \log \max\{1 - \rho(j), \epsilon\}$ for $j \in A^+$.
return (β^+, A^+).

The selected destination is removed before its surviving peers are updated; the next query uses the new source and new destination set. After $N-1$ selections, $U = \varnothing$ and $A = \{s\}$, so closure is

deterministic and needs no further policy query. ARC-Base omits FSP initialization, queries and updates and sets $\lambda = 0$. Training samples actions using the same construction routine and the POMO objective; the training switches appear in Table 7.

Kernel scaling equivalence. For fixed support and marginals, subtracting row and column potentials changes the assignment objective by a constant. Consequently, at the same temperature, $\bar{C}$ and C' yield the same fully converged entropic assignment coupling. Their kernels differ only by positive diagonal scalings, which are absorbed by the converged Sinkhorn factors (Sinkhorn & Knopp, 1967; Cuturi, 2013). This equivalence concerns the converged coupling; ARC uses Gauge as a neural input representation and initializes FSP separately with finite iterations before maintaining its state recursively.

A.3 Running Example

Figure 2 illustrates how consuming a source changes guidance for its successor, using the synthetic balanced assignment

$$Q = \begin{pmatrix} 0 & 0.6 & 0.3 & 0.1 \\ 0.2 & 0 & 0.3 & 0.5 \\ 0.4 & 0.2 & 0 & 0.4 \\ 0.4 & 0.2 & 0.4 & 0 \end{pmatrix}, \qquad u = (0.2, 0.4, 0.1, 0.3), \quad v = (0.3, -0.2, 0.1, 0).$$

Both marginals equal one. Define off-diagonal costs by $C_{ij} = -\log Q_{ij} + u_i + v_j$ and forbid self-edges. Selecting $0 \to 1$ removes row 0 and column 1, leaving

$$Q_{\{1,2,3\},\{0,2,3\}} = \begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.4 & 0 & 0.4 \\ 0.4 & 0.4 & 0 \end{pmatrix}, \qquad d = (1, 0.7, 0.9).$$

Destinations 2 and 3 lose 30% and 10% of their incoming support. The one-sided correction multiplies surviving columns by $(1, 1/0.7, 1/0.9)$, restoring their sums. Normalizing the next source row gives

$$\rho_1(0, 2, 3) = \frac{(0.2,\ 0.3/0.7,\ 0.5/0.9)}{0.2 + 0.3/0.7 + 0.5/0.9} \approx (0.1689,\ 0.3619,\ 0.4692).$$

The weight toward destination 2 rises from 0.30 to 0.36 without changing edge costs. This is a structural query after one column correction, rather than a fully balanced residual solve. It includes destination 0 for closure; the neural policy can select only nodes 2 and 3 at this step.

B Experimental Details

B.1 Datasets and Reference Solutions

Standard ATSP and size generalization. The standard benchmark contains 1,000 ordered official ATSP test matrices at each $N \in \{100, 200, 500, 1000\}$. Training uses generated ATSP100 instances. The ATSP2000 evaluation uses 100 fixed instances from the uniform-tmat generator with integer min-plus closure. All neural methods and reference solvers evaluate the same matrices within each experiment. Standard costs are closed-tour objectives on the original integer scale divided by 10^6.

Local four-scale references use LKH-3 (Helsgaun, 2017), with ten runs and 1,000 maximum trials. ATSP2000 uses one run with either 100 or 1,000 maximum trials and no time limit, with fixed instance-specific randomness. All ATSP2000 gaps use the LKH-1000 mean 1.57155144. LKH references are heuristic solutions. Published baseline gaps retain their original reference values; local ARC gaps use the unrounded local reference means.

Road networks. The released RRNCO data (Son et al., 2026) provide three splits of 1,280 ATSP100 instances each: ID uses the training cities, OOD (city) uses unseen cities, and OOD (cluster) changes the location sampling distribution. Training draws from the released data for 80 cities. Costs are closed-tour distances in km on the original directed distance matrices. The reference means are 38.387, 38.903 and 12.170 km for ID, OOD (city) and OOD (cluster), respectively, as reported for LKH in RRNCO's Table 1.

Varying asymmetry. For $\sigma \in \{0.1, 0.2, 0.3\}$, generate $C_{ij} = \|x_i - x_j\|_2 Z_{ij}$, with independent uniform coordinates $x_i \in [0,1]^2$ and independent ordered-edge multipliers $Z_{ij} \sim \mathcal{N}(1, \sigma^2)$. Matrices are generated in float64 and stored in float32. Negative edges remain unmodified; no symmetrization, shortest-path closure or integer scaling is applied. Six fixed test sets contain 1,000 instances each, one per (σ, N) with $N \in \{50, 100\}$. Each encoder trains separately at $N = 50$ for each σ and is evaluated on both sizes at that same asymmetry. Costs use the original Euclidean–Gaussian scale.

B.2 Architecture, Training, and Inference

Table 7 collects the standard ARC settings. The backbone follows RADAR (Yi et al., 2026); training uses POMO REINFORCE (Kwon et al., 2020; Williams, 1992) with the mean reward across starts as the baseline and Adam (Kingma & Ba, 2015). Rewards use original closed-tour costs. ARC-Base and ARC-I share weights; ARC-T is trained separately with FSP enabled.

Table 7: Standard ARC architecture, optimization and structural settings.

Setting	Value
Embedding / encoder / attention	256 dimensions / 5 layers / 8 heads of dimension 32
Feed-forward / mixed-score hidden size	512 / 16
SVD	Rank 10; randomized SVD of $\bar{C}$
Training data and budget	ATSP100; 2,100 epochs; 10,000 instances/epoch
Batch / POMO starts	64 / 100
Adam learning rate / weight decay	4×10^{-4} / 10^{-6}
Learning-rate schedule	Factor 0.1 at epoch 2001
Clipping c_{clip} (train / test)	10 / 50
Representation (τ_r, T_r)	$(0.1, 5)$; gauge fixed before joint RMS
FSP $(\tau_s, T_s, \eta, \epsilon)$	$(1, 10, 1, 10^{-6})$
Inference	Greedy $P = N$; float32; no augmentation
Configuration	**FSP during training / inference**
ARC-Base	Off / off
ARC-I	Off / on ($\lambda = 0.5$); ARC-Base weights
ARC-T	On / on ($\lambda = 0.5$ in both)

Road-network training and model selection. The road ARC models use Min–Max inputs without coordinates and train for 200 epochs of 100,000 instances, with effective batch size 256 and float32 arithmetic. Adam uses learning rate 4×10^{-4} and weight decay 10^{-6}, with a factor-0.1 decay at epochs 180 and 195. ARC-Base and ARC-I share the selected epoch-198 checkpoint trained with FSP disabled. ARC-T is a separate training run with FSP enabled ($\lambda = 0.5$, $\tau_s = 1$) and uses its terminal epoch-200 checkpoint. Inference parameters for ARC-I are selected on a 512-instance ID validation set disjoint from all final 1,280-instance test splits; $\lambda = 1$, $\tau_s = 0.1$ is then fixed across the three splits. The default $(0.5, 1)$ setting is included as a control.

Greedy decoding uses 100 starts per input. Eightfold matrix augmentation combines four fixed node permutations with original and transposed orientations, selecting the best of 8×100 tours by original cost. The local RADAR baseline uses a 128-dimensional, 12-layer, eight-head encoder and rank-10 SVD; its epoch-199 checkpoint was selected by the inherited ID monitor. The independent validation split above applies to the new ARC runs, not that historical baseline.

Common-decoder asymmetry experiment. All encoders use the RADAR decoder architecture (256 dimensions, eight heads of dimension 32), with independently trained decoder weights and FSP disabled. Table 8 lists the encoder differences. ICAM denotes the Instance-Conditioned Adaptation Model (Zhou et al., 2026). Training uses 1,200 epochs, 10,000 fresh ATSP50 instances per epoch, batch size 64 and the same POMO objective. Adam uses learning rate 4×10^{-4}, weight decay 10^{-6}, $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, with no learning-rate decay, gradient clipping or mixed

precision. All reported checkpoints are fixed at epoch 1101. Training/test logit clipping is 10/50; evaluation is greedy with $P = N$, batch size 64 and no augmentation. Selected tours are scored by summing original float32 entries in float64, including closure. These experiments compare encoder configurations under a common decoder architecture; encoder capacities differ.

Table 8: Encoder configurations for the common-decoder asymmetry experiment.

Encoder	Inputs and architecture
RADAR	SVD; five layers, 256 dimensions, eight heads.
ARC-Base	RADAR backbone with paired potentials and gauge-adjusted edges.
MatNet	One-hot inputs (100 indices); five layers, 256 dimensions, 16 heads of dimension 16; identity adapters.
ICAM	Top-50 row/column costs; six layers, 128 dimensions; $\log_2 N$ encoder scaling; stable softmax weighting.
RRNCO	Min–Max costs and sampled distance embeddings; six layers, 128 dimensions; coordinates and polar features disabled.

Note. ICAM and RRNCO use separate bias-free $128 \to 256$ row and column adapters.

B.3 Baselines and Evaluation Protocols

Published and local baselines. The LKH, GLOP, UDC-x250, MatNet-Single (Random), ELG and ReLD rows in Table 1 come from RADAR's Table 1, with adaptations defined in its Section 5.1 and Appendix B.1 (Yi et al., 2026). Edge-aware uses the complete decoder's ATSP result from Table 1 of Liang et al. (2026). LKH-100/1000 denotes 100/1,000 search trials. UDC-x250 uses $\alpha = 50$; GLOP uses RADAR-retrained MatNet locally; ELG uses matrix/random-embedding processing. MatNet-Single (Random) is the single-set random-embedding adaptation. Neural baselines reported with z-score inputs retain that preprocessing. Published costs, gaps and timing scopes are preserved; times are displayed in minutes. In Table 1, boldface follows the displayed-precision convention. The road table compares all displayed neural methods per split using retained source values; LKH is the reference and runtime is not emphasized.

RRNCO's road-network reference uses coordinates and eightfold coordinate augmentation on an A6000. The local RADAR evaluation uses greedy 100-start decoding, batch size 32 and a TITAN RTX. ARC road times are the reported total evaluation times per 1,280-instance split, converted from seconds to minutes. The four ARC configurations run sequentially on one RTX 3090 (24 GB) in FP32, with batches of 32 original instances; augmentation expands each batch to 256 variants before decoding. Their wall timer starts before first-batch preprocessing and ends after all tours have been decoded and independently checked against original costs. It includes augmentation construction, Min–Max normalization, SVD and potential preprocessing, decoding, best-tour selection and augmentation mapping, but excludes checkpoint loading, prior test-set disk reading and result-file writing. The ARC-Base ID time includes approximately one second of initial CUDA warm-up.

Local evaluation and timing. The main RADAR and ARC evaluations use RTX 3090 GPUs with batches $1000/400/50/3$ in increasing size order. RADAR uses its official epoch-2100 checkpoint. Factorial, RADAR state and window studies use TITAN RTX and batches $64/16/4/8$. Randomized SVD uses a fixed batch-indexed rule, shared within each comparison. In the main block, the timer covers the complete batch loop: environment setup, transfer, reset, preprocessing, encoding, decoding, best-of-POMO extraction and CPU score copying. Checkpoint/data loading and serialization are excluded. The RADAR road timings sum synchronized rollout calls, including environment setup, preprocessing, encoding, decoding and reward computation; transfer before rollout, subsequent best-tour extraction and independent tour checks are outside the timer. Timings are from one evaluation pass.

ATSP2000 keeps the ATSP100-trained checkpoints, caps every neural method at $P = 100$ (starts 0–99), and uses batch size 1 on TITAN RTX with float32 arithmetic. Costs are divided by 10^6 before z-score normalization; SVD randomness is fixed by instance index. Candidate selection uses original integer tour costs. Neural times include transfer, preprocessing, decoding, tour transfer,

validity checks and raw-cost ranking, excluding data/checkpoint loading and output I/O. LKH time sums solver subprocess wall times, including input parsing; up to four CPU tasks may run concurrently, so this sum is accumulated task time. Allocated-budget and profiling timers are defined in Appendices C.2 and C.6.

Inference-time FSP across backbones. Figure 4 fixes policy weights within each off/on pair, using 1,000 instances per size, $P = N$ and no augmentation. Table 9 records the differences between backbones. FSP uses the settings in Table 7, except for λ. MatNet and RADAR supply standardized costs to FSP; ARC supplies C'. ICAM (Zhou et al., 2026) keeps its raw scaled-cost neural inputs and both learned edge-cost terms, and standardizes a separate cost matrix for FSP. Each pair shares preprocessing, batch partitions and random-seed rules. The four panels measure within-backbone improvements rather than a common-protocol ranking.

Table 9: Decoding settings for inference-time FSP transfer.

Backbone	Decoding	Batches	λ
MatNet-Single (Random)	Softmax sampling	$3/3/3/3$	0.5
ICAM	Greedy	$3/3/1/1$	0.1
RADAR	Greedy	$64/16/4/8$	1.0
ARC-Base (on: ARC-I)	Greedy	$64/16/4/8$	0.5

Note. Batch sizes are listed in the order $N = 100, 200, 500, 1000$. All backbones use a clipping value of 50 during testing, with the same random draws for each paired evaluation with FSP off and on. MatNet uses a clipping value of 10 during training.

Metrics and statistical inference. Let $\bar{c}_m$ be the mean original-cost objective for method m. Reference gaps are $100(\bar{c}_m/\bar{c}_{\text{ref}} - 1)$ in percent. The asymmetry comparison reports mean original tour costs. State-control contrasts use $1000(\bar{c}_m - \bar{c}_{\text{initial}})/\bar{c}_{\text{initial}}$ in per mille; the main mean-control table instead uses the no-FSP mean. Guidance windows report the reduction $1000\overline{c_{\text{no FSP}} - c_w}/b_N$, where b_N is the observed no-FSP mean. Each denominator and sign is labeled at its point of use.

Confidence intervals quantify test-instance variation conditional on the evaluated models. Shift, state and window contrasts use pointwise paired 95% percentile intervals from 10,000 instance-bootstrap resamples (Efron & Tibshirani, 1993), retaining common instance indices across conditions. Shift experiments average the three fixed perturbations within each instance; shift and state ratios recompute both means per resample, whereas window reductions hold b_N fixed. The probe, deployment-tracking and start-budget studies specify their own sampling units below. Cross-backbone curves report means, while Table 12 reports variation across independent training runs rather than test-instance uncertainty.

C Additional Results and Computational Analysis

C.1 POMO Start Budgets and Measured Latency

On 200 ATSP1000 instances, ARC-Base with 16 starts achieves a lower mean tour cost than RADAR with 1,000 starts while reducing measured end-to-end latency by 31.6% (Table 10). The guided comparison shows that updating destination pressure achieves better solution quality with fewer starts and lower measured latency than keeping the initial pressure fixed. Figure 6 shows all 32 measured configurations.

The sweep uses the same 200 held-out ATSP1000 instances for all methods, an RTX 3090 and batch size 1. ARC-Base, ARC-Static and ARC-I share weights; Static fixes the initial destination pressure, while ARC-I updates it. The number of POMO starts is $K \in \{16, 32, 64, 128, 256, 512, 768, 1000\}$, with three shared start subsets at each partial budget and one full-start evaluation. Start-subset repeats are averaged within each instance before aggregating instances or computing the 10,000-resample paired intervals. Gaps use the same cohort's unrounded LKH mean, 1.5726286. These are separate runs from the standard main table and the 1,000-instance guidance controls.

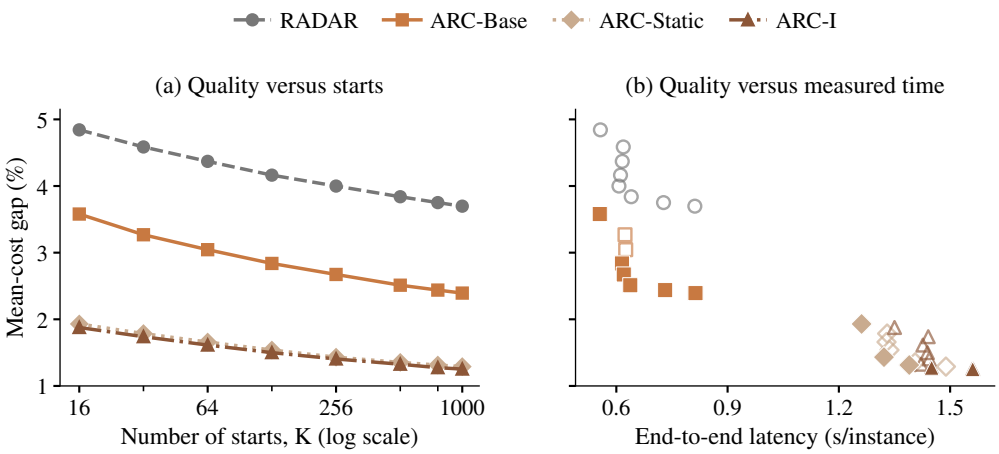


Figure 6: POMO budget and measured end-to-end latency on 200 ATSP1000 instances (batch size 1, shared starts). (a) Mean-cost gap versus starts. (b) Gap versus latency; filled markers are observed non-dominated points. Each point is a separate configuration.

Table 10: Measured configurations highlighted in the main results.

Method	Starts	Gap (%)	Latency (s/instance)
ARC-Base	16	3.579	0.5554
RADAR	1000	3.698	0.8117
ARC-I	768	1.274	1.4501
ARC-Static	1000	1.290	1.4886

Latency is measured wall time per instance, including CPU preparation, host-to-device transfer, representation and encoder computation, root initialization, decoding, guidance, tour selection and returned-tour transfer. Model/data loading and result-file I/O are excluded. The time panel shows the 32 measured configurations without interpolation; non-dominance is computed over their observed mean gaps and latencies. ARC-Static retains non-dominated points at $K = 16, 256, 768$. At $K = 1000$, ARC-I improves Static by 0.0373 gap percentage points (paired 95% interval $[0.0181, 0.0570]$) with 0.0723 additional seconds per instance. The two guided methods have overlapping measured latency ranges only from about 1.350 to 1.489 seconds; comparisons use actual configurations, not a continuous time-budget frontier.

C.2 Solution Quality under Allocated Decoding Budgets

Under the allocated decoding-budget protocol, ARC-I reduces mean tour cost relative to RADAR by 1.17% at $N = 500$ and 2.34% at $N = 1000$ (Table 11). This separate experiment uses 1,000 matched instances per size on TITAN RTX, with batch size 4. ARC-I performs one greedy $P = N$ pass; its per-batch decode time becomes RADAR's allocation for repeated sampled $P = N$ passes, retaining the best tour per instance. SVD seeds stay fixed within a batch while action-sampling seeds change. A first pass always runs; a further pass is admitted only if its projected duration, estimated from the preceding pass, fits the remaining allocation.

Cost change is $100(\bar{c}_{\text{ARC-I}} - \bar{c}_{\text{RADAR}})/\bar{c}_{\text{RADAR}}$. Whole-pass admission leaves part of RADAR's allocation unspent; the table reports both allocated and actual time. Timers are CUDA-synchronized and cover reset, preprocessing, encoding, decoding and best-tour extraction, excluding model and instance loading. The comparison uses the recorded endpoints under this allocation rule.

Table 11: Performance under ARC-I's allocated decode-time budget.

N	Allocated	RADAR (sampled)		ARC-I (greedy)		ARC-I vs. RADAR
	budget (min)	Time (min)	Mean cost	Time (min)	Mean cost	Cost change (%) ↓
500	5.05	3.65	1.6072	5.05	1.5885	-1.17
1000	26.25	18.87	1.6319	26.25	1.5938	-2.34

C.3 Results across Training Runs

ARC-T has lower average cost than ARC-Base at every size across three independent training runs per configuration (Table 12). Each run uses greedy full-POMO decoding without augmentation on the same four 1,000-instance test sets. Entries are the mean and sample standard deviation of the three run-level mean costs; ARC-T uses independently trained FSP-enabled policies.

Table 12: Three independent training runs evaluated under the same test protocol. Entries report the across-run mean ± standard deviation of mean tour cost.

Method	ATSP100	ATSP200	ATSP500	ATSP1000
ARC-Base	1.57254 ± 0.00028	1.58035 ± 0.00033	1.59158 ± 0.00138	1.60723 ± 0.00396
ARC-T	$\mathbf{1.57146} \pm 0.00035$	$\mathbf{1.57895} \pm 0.00055$	$\mathbf{1.58661} \pm 0.00106$	$\mathbf{1.59097} \pm 0.00106$

C.4 Road-network Guidance and Augmentation

The validation-selected ARC-I configuration achieves lower mean tour cost than ARC-Base on ID, OOD-city, and OOD-cluster inputs, both with and without matrix augmentation (Table 13). Eight-fold augmentation also reduces mean cost for every model and guidance setting in the table. Appendix B.2 gives the training, model-selection, and augmentation protocols.

Table 13: Full road augmentation comparison. Each entry is mean cost (km) / gap (%) to LKH on 1,280 ATSP100 instances per split.

Model	(λ, τ_s)	Aug.	ID	OOD-city	OOD-cluster
ARC-Base	—	×1	39.144 / 1.97	39.756 / 2.19	12.432 / 2.15
ARC-I	$(0.5, 1.0)$	×1	39.142 / 1.97	39.752 / 2.18	12.431 / 2.14
ARC-I	$(1.0, 0.1)$	×1	39.123 / 1.92	39.736 / 2.14	12.413 / 2.00
ARC-Base	—	×8	38.724 / 0.88	39.292 / 1.00	12.298 / 1.05
ARC-I	$(0.5, 1.0)$	×8	38.718 / 0.86	39.292 / 1.00	12.297 / 1.04
ARC-I	$(1.0, 0.1)$	×8	38.702 / 0.82	39.272 / 0.95	12.284 / 0.94
ARC-T	$(0.5, 1.0)$	×1	39.144 / 1.97	39.800 / 2.31	12.447 / 2.28
ARC-T	$(0.5, 1.0)$	×8	38.721 / 0.87	39.287 / 0.99	12.303 / 1.09

C.5 Robustness to Row and Column Shifts

Assignment-structured representations reduce sensitivity to row and column shifts that preserve tour ordering. At the strongest tested shift, $\lambda_g = 1$, the relative tour-cost increases are 0.12%, 0.11%, and 0.08% for ARC-Base, ARC-I, and ARC-T, respectively, compared with 15.40% for RADAR (Figure 7).

All four methods use 1,000 ATSP100 instances, greedy $P = 100$, batch size 64 and the common SVD seed rule.

For each instance, draw independent standard normal row and column vectors a, b and center each. Set $G_{ij} = (a_i + b_j)/\sigma_{\text{off}}$, where σ_{off} is the population standard deviation over nonself edges. Supply $\bar{C} + \lambda_g G$ to both SVD and potential construction, without subsequent input renormalization. The shift applies to every matrix entry. Use $\lambda_g \in \{0, 0.25, 0.5, 1\}$ and three fixed perturbation draws, sharing each direction across methods and nonzero strengths. The zero-shift input is evaluated once. Every tour receives the same additive shift; best-tour selection and scoring use the original matrix.

For method m, the ordinate is $100[\mu_m(\lambda_g)/\mu_m(0) - 1]$, where μ_m averages the three seed costs within each instance and then averages instances.

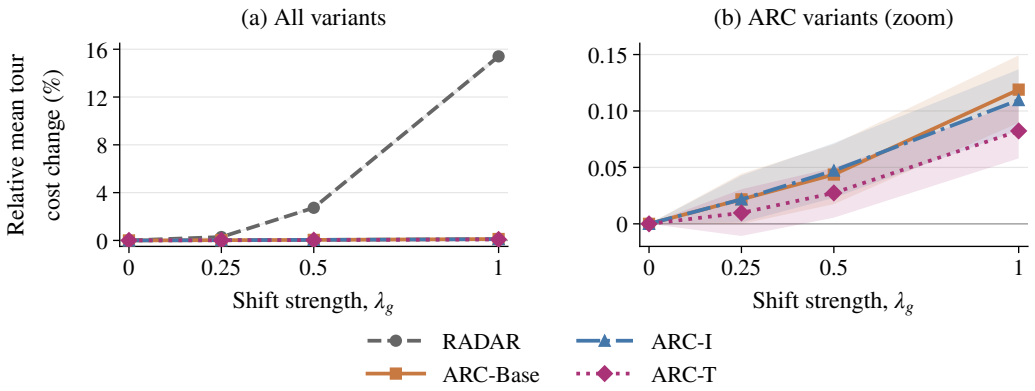


Figure 7: ARC remains less sensitive to row and column shifts across the tested strengths. Curves show relative cost increase (%) against each method's unperturbed mean; shading denotes paired 95% intervals. The right panel magnifies the ARC curves using a separate vertical range. Paired comparisons do not resolve differences among the ARC variants.

C.6 Runtime and Memory Analysis

Let T be the total number of Sinkhorn initialization iterations across the two branches. The corresponding preprocessing costs $O(TN^2)$. FSP frontier queries and state updates require $O(N)$ work per decision, giving $O(N^2)$ structural computation per trajectory and $O(PN^2)$ across P trajectories. The shared kernel and trajectory-specific destination states require $O(N^2 + PN)$ additional memory. The measurements below quantify this overhead. ARC's structural computation is concentrated in tour construction. At $N = 1000$, potential and Gauge preprocessing account for 0.2% of ARC-I's measured stage time, while frontier queries and pressure updates account for 30.0% and 9.9%, respectively (Table 15).

Under full-POMO profiling, ARC-I takes 1.7196 seconds per instance and uses 10,177 MiB of peak allocated memory; RADAR takes 1.1214 seconds and uses 10,086 MiB (Table 14). Appendix C.1 evaluates solution quality and measured end-to-end latency across different start counts.

Profiling uses warmed-up CUDA events on TITAN RTX, greedy $P = N$, batches $64/16/4/8$ and cohorts of $256/256/100/50$ instances. The FSP weight is 1 for RADAR+FSP and 0.5 for ARC-I. Table 15 reports module shares within each configuration's profiling run. Latency is in seconds per instance; A/R lists allocated/reserved memory in MiB. P/G denotes potential/Gauge preprocessing; Query and Update are the per-step structural query and pressure update. Other includes the remaining overhead.

D Representation and FSP Analysis

D.1 Probing Assignment Information

Cost-controlled reference-edge discrimination. Figure 1a uses a fixed 70/10/20 instance split of 100 ATSP100 instances and LKH reference tours computed with 100 runs and 100 trials. Can-

Table 14: Runtime and memory by configuration. Latency is per instance.

Method	$N = 100$		$N = 200$		$N = 500$		$N = 1000$	
	Latency	A/R	Latency	A/R	Latency	A/R	Latency	A/R
RADAR	0.0034	889/924	0.0176	856/904	0.1719	1,320/1,898	1.1214	10,086/11,182
ARC-Base	0.0035	943/978	0.0172	898/938	0.2174	1,362/1,930	1.0916	10,102/11,214
RADAR+FSP	0.0058	936/986	0.0310	898/940	0.3006	1,374/1,934	1.7196	10,118/11,218
ARC-I	0.0057	957/1,000	0.0368	929/980	0.3437	1,342/1,962	1.7196	10,177/11,246

Table 15: Reported module shares (%) of profiled latency.

N	Method	SVD	P/G	Encoder	Attention	Query	Update	Other
100	RADAR	17.0	–	33.4	37.7	–	–	11.9
	ARC-Base	15.9	2.0	32.2	38.0	–	–	12.1
	RADAR+FSP	9.5	–	19.2	29.0	23.2	4.2	14.9
	ARC-I	9.8	1.9	19.5	23.9	24.5	5.9	14.5
200	RADAR	3.5	–	23.6	52.8	–	–	20.1
	ARC-Base	3.6	1.5	24.4	52.9	–	–	17.6
	RADAR+FSP	2.5	–	15.3	29.6	28.0	4.3	20.3
	ARC-I	1.7	1.0	11.3	27.8	30.4	9.5	18.3
500	RADAR	0.6	–	14.0	66.9	–	–	18.5
	ARC-Base	0.6	0.6	11.0	62.0	–	–	25.8
	RADAR+FSP	0.3	–	8.0	39.7	32.2	6.1	13.7
	ARC-I	0.3	0.3	7.0	36.6	33.1	5.8	17.0
1000	RADAR	0.2	–	9.0	81.3	–	–	9.5
	ARC-Base	0.1	0.3	9.7	83.4	–	–	6.4
	RADAR+FSP	0.1	–	5.4	49.7	30.0	9.9	4.9
	ARC-I	0.1	0.2	5.4	49.8	30.0	9.9	4.7

didate pairs differ in cost by at most 0.05 input-instance standard deviations and are balanced between cheaper and more expensive alternatives, retaining 306 pairs from the 20 test instances. Cost uses the original edge cost. The Row/col. addition (Stats) concatenates four quantities: the source-row mean and minimum, and the destination-column mean and minimum. The Assignment addition (Dual+Gauge) concatenates $[u_i, v_i, u_j, v_j, u_iu_j, u_iv_j, v_iu_j, v_iv_j]$ and the adjusted cost $C'_{ij} = \bar{C}_{ij} - u_i - v_j$. The potentials use five Sinkhorn iterations at temperature 0.1. Both additions retain the original cost feature. Self-edges are excluded from the candidate set and the statistics. Combined features concatenate the named components; Random uses fixed-seed independent Gaussian scores. A separate L2-regularized linear readout is fitted for each feature set, using ordered candidate concatenation, both candidate orders during training, and feature standardization fitted only on training pairs. Accuracy measures reference-edge identification, with half credit for score ties. Intervals resample the test instances 5,000 times while holding the fitted readout fixed.

Full-candidate reference-successor ranking. Figure 1b ranks all nonself destinations for each source on 1,000 instances at each of the four sizes. Original (Raw) uses standardized costs; Centered (Mean+DC) subtracts the row and column means and adds back the overall mean; Assignment (Gauge) uses the same five-iteration, temperature-0.1 assignment adjustment. The rankings use these edge scores directly, without a fitted readout; lower scores rank first and exact ties are resolved by destination index. References follow the LKH protocol in Appendix B.1 and include the closing edge of each tour. These are heuristic reference successors, not certified optimal actions for a construction prefix. Hit@1 is computed within each instance and then averaged equally across instances; intervals use 10,000 instance-bootstrap resamples.

Dual and Gauge each achieve higher test accuracy and ROC-AUC than Stats on the same cost-controlled pairs (Figure 8). Both endpoint potentials and adjusted edge costs therefore expose useful reference-edge information beyond local row and column summaries.

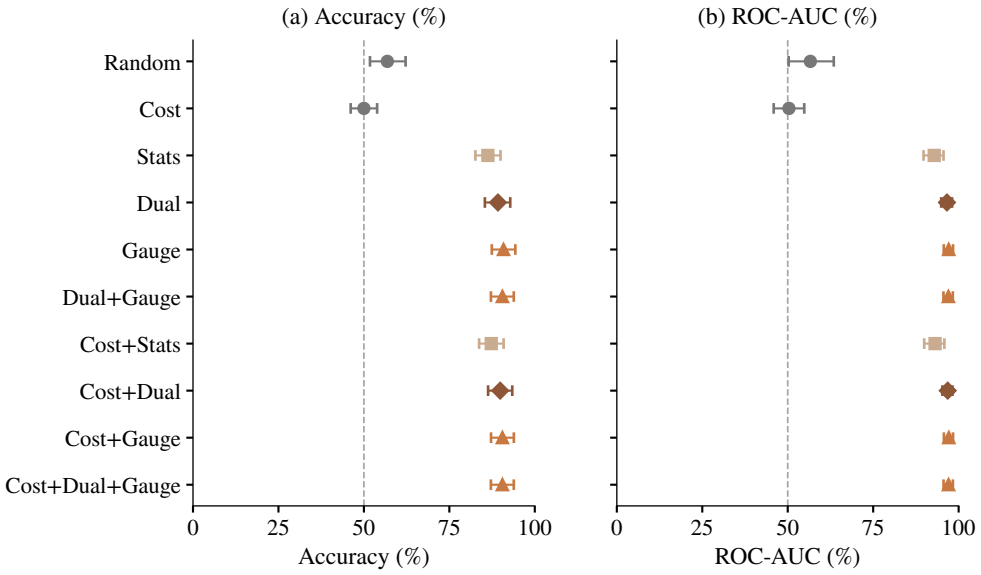


Figure 8: Cost-controlled reference-edge discrimination with additional features. Accuracy and ROC-AUC use the same 306 pairs from 20 ATSP100 test instances; bars show 95% instance-bootstrap intervals.

Assignment has the highest mean MRR, Hit@5, and Hit@10 at all four tested sizes (Figure 9). Its ranking advantage extends beyond Hit@1.

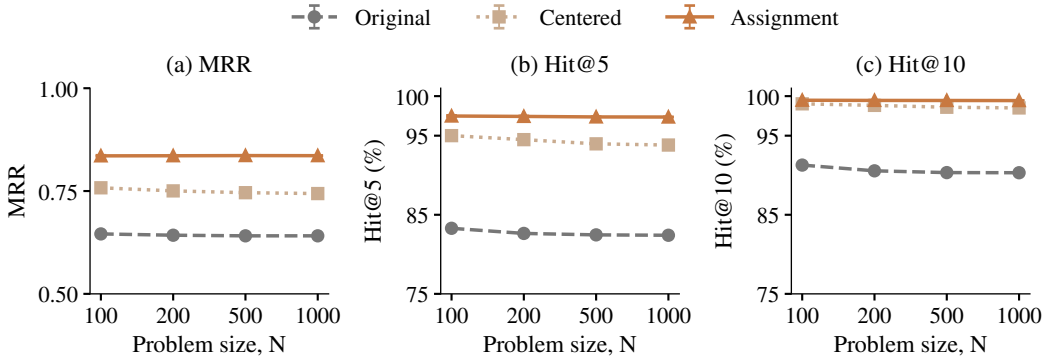


Figure 9: Full-candidate reference-successor ranking beyond Hit@1. MRR, Hit@5, and Hit@10 use all $N - 1$ nonself destinations on 1,000 instances per size; bars show 95% instance-bootstrap intervals.

D.2 Representation Ablations and Alternatives

Table 4 tests whether assignment-specific node and edge inputs help beyond either component alone or ordinary mean centering. The 2^3 design crosses Dual, Gauge and inference-time FSP. RADAR supplies the no-Dual/no-Gauge checkpoint; Dual-only, Gauge-only and Dual+Gauge are separately trained to epoch 2100. Each FSP off/on pair holds its policy weights fixed. RADAR uses validation-

selected $\lambda = 1$; the three trained representation backgrounds use $\lambda = 0.5$, fixed across sizes. The factorial follows the shared TITAN RTX protocol in Appendix B.3 and excludes ARC-T.

The component means identify Gauge as the largest representation benefit and Dual's large-scale benefit as conditional on Gauge. At $N = 100$, Dual also helps without Gauge. FSP lowers mean cost on every tested background.

Independently retrained Mean+DC. For a standardized finite matrix X, define $r_i = \mathrm{mean}_j\, X_{ij}$, $c_j = \mathrm{mean}_i\, X_{ij}$ and $m = \mathrm{mean}_{ij}\, X_{ij}$, including the input diagonal. Mean+DC sends $[r_i - m, c_i]$ through the same joint RMS and node projection and supplies $E_{ij} = X_{ij} - r_i - c_j + m$ to the two encoder edge channels. SVD-r10 is computed before double centering; attention Sinkhorn and policy feasibility masks are unchanged.

Mean+DC trains from scratch with the standard ATSP100 budget in Table 7, with FSP disabled. Its terminal epoch-2100 checkpoint is fixed by the budget. ARC-Base resumes the same training run from an intermediate checkpoint; updates before and after resumption together follow the stated training budget. The two runs share the budget, not identical generated training streams. Table 4b evaluates all three models on 1,000 ordered instances per size, greedy $P = N$, no augmentation and FSP disabled, reconstructing costs from the original integer matrices. This is a separate evaluation block from part (a). ARC-Base has lower cost than Mean+DC at every size, supporting assignment-derived inputs over this mean-based alternative under the common training budget.

D.3 Tracking Residual Assignment States

Recursive FSP follows the recomputed residual reference more closely throughout deployed ATSP500/1000 construction, with the largest separation from fixed initial pressure (Root-static) late in the trajectory (Figure 5). At 90% depth, centered pressure MAE falls from 0.1277 to 0.0447 at $N = 500$ and from 0.1030 to 0.0308 at $N = 1000$. Guidance Spearman rises from 0.9485 to 0.9921 and from 0.9657 to 0.9961, respectively; paired differences in guidance and pressure Spearman have positive 95% intervals at all five depths in both sizes.

Deployment trajectories and references. The diagnostic uses the first 50 held-out instances at each size and the closed tours selected by greedy, unaugmented, full-POMO ARC-I. States are compared at progress fractions $0.1, 0.25, 0.5, 0.75, 0.9$ on the same trajectories, kernels and residual supports, totaling 500 residual problems. Root-static restricts the initial pressure to each support; FSP maintains pressure with ten root iterations, $\tau_s = 1$, $\eta = 1$, $\epsilon = 10^{-6}$ and $\lambda = 0.5$. The reference promotes the cached float32 kernel to float64 and recomputes Sinkhorn scaling to convergence on the same residual problem.

Pressure and guidance metrics. Before computing MAE or RMSE, we center each method's cost-unit potentials over the active destinations A_t, including the reserved start. Guidance metrics compare the centered structural correction $\lambda(L_{i_t j} + \beta_t(j))$ over policy candidates U_t; they do not compare the complete neural-policy logits. Spearman uses average ranks for ties, and guidance top-10 overlap uses $k = \min(10, |U_t|)$. Each instance contributes one value per method and depth; means and paired differences use 10,000 instance-bootstrap resamples with the trajectories fixed.

Recursive FSP also lowers pressure RMSE and improves guidance top-10 overlap with the residual reference (Figure 10). The benefit appears in both state accuracy and the candidate preferences supplied to the policy.

Residual rebalancing controls. Table 16 compares pressure maintenance on frozen official RADAR and ARC-Base at $N = 1000$, $P = 8$, using 64 paired test instances per backbone. Static keeps the initial pressure, FSP updates it recursively, Periodic-rebalanced periodically re-solves the residual assignment, and Converged-residual rebalances it to convergence at every step. RADAR uses a period of 50 and marginal tolerance 10^{-5}. For ARC-Base, each method independently selects λ by mean cost on the same 32 independently generated validation instances, disjoint from the fixed random 64-instance test set, from $\{0.05, 0.1, 0.25, 0.5, 1.0, 1.25, 1.5, 1.75, 2.0\}$. All four methods select $\lambda = 1.0$; no other parameters are tuned in this comparison.

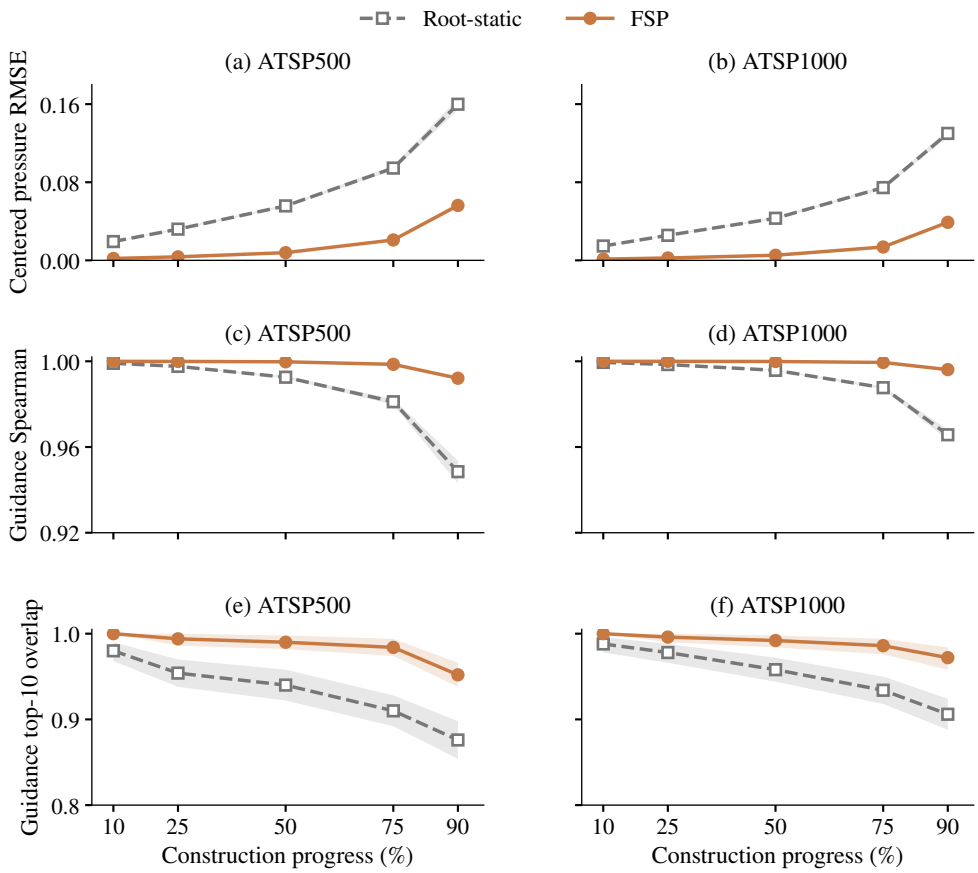


Figure 10: Additional deployment-tracking metrics on 50 instances per scale. Centered pressure RMSE, guidance Spearman correlation, and guidance top-10 overlap use the same trajectories and recomputed residual references as Figure 5. Bands show 95% instance-bootstrap intervals.

Table 16: Quality–computation trade-off for residual rebalancing on ATSP1000. Each backbone uses 64 paired test instances with $P = 8$ under the settings in Appendix D.3. Bold marks the lowest mean cost within each panel; time ratios are relative to FSP. All ARC-Base methods use $\lambda = 1.0$.

(a) Frozen official RADAR

Method	Mean cost ↓	Time (s/instance)	Time / FSP
Static	1.620156	1.087	0.95
FSP	1.617771	1.150	1.00
Periodic-rebalanced	1.617949	1.277	1.11
Converged-residual	**1.617480**	7.492	6.52

(b) Frozen ARC-Base

Method	Mean cost ↓	Time (s/instance)	Time / FSP
Static	1.604320	1.672	0.95
FSP	1.602683	1.758	1.00
Periodic-rebalanced	**1.602589**	1.803	1.03
Converged-residual	1.602729	6.946	3.95

FSP lowers mean tour cost relative to Static on both backbones and is the fastest of the three residual-update variants (Table 16). Converged-residual attains the lowest RADAR cost at $6.52\times$ FSP's runtime. On ARC-Base, the three residual-update methods have closely matched costs, with Periodic-rebalanced lowest and Converged-residual taking $3.95\times$ FSP's runtime. These results place FSP at a favorable quality–computation trade-off.

D.4 Destination Alignment and State Updates

Updating destination pressure improves the large-scale mean solution quality of both frozen backbones (Table 6). Figure 11 gives RADAR's paired increments relative to Static across all four sizes. Each backbone's control block evaluates 1,000 instances per size with greedy full-POMO decoding, no augmentation and fixed policy weights; ARC uses its FSP-off-trained epoch-2100 model. The RADAR and ARC control blocks retain their own None denominators.

Two LogK kernels and pressure controls. RADAR uses the raw standardized-cost kernel $L = -\bar{C}/\tau_s$; ARC uses the assignment-adjusted kernel $L = -C'/\tau_s$, where $C' = \bar{C} - u - v$ comes from five representation iterations at $\tau_r = 0.1$. Both use structural temperature $\tau_s = 1$. LogK-only centers the current kernel row over feasible policy candidates and adds it before tanh clipping, without a destination-pressure state. Thus ARC's LogK-only control retains assignment information in its kernel. Static adds fixed β_0; FSP recursively maintains β_t. RADAR uses $\lambda = 1$ and ARC uses $\lambda = 0.5$, with clipping 50; ATSP500/1000 batches are 4/8.

Random bias replaces the structural correction by a centered Gaussian vector matched to its standard deviation on each rollout's feasible candidates. State updates still follow the constructed trajectory. RADAR's Shuffled control instead preserves β_0 and permutes only $\beta_t - \beta_0$ across destinations independently at each step before masking. Lagged guidance uses the newest snapshot at least $\ell = \max(1, \mathrm{round}((N-1)/3))$ steps old, taking snapshots every ℓ steps and using β_0 initially. Both structural controls feed their queried frontier probabilities into later updates.

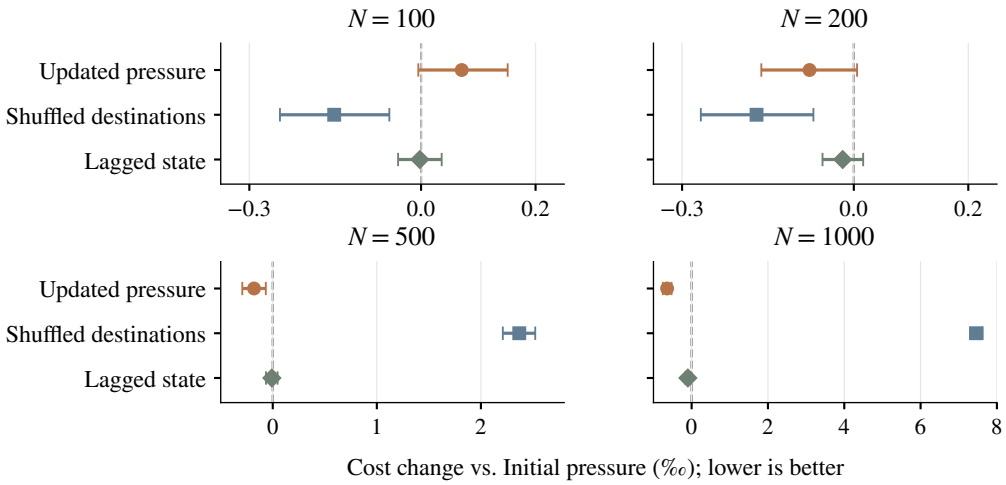


Figure 11: FSP (Updated pressure) lowers mean cost relative to Static pressure (Initial pressure) at $N = 500, 1000$. Points show $1000(\bar{c}_m - \bar{c}_{\text{initial}})/\bar{c}_{\text{initial}}$; horizontal bars are paired 95% intervals. Negative values favor the variant over Initial pressure; the zero line denotes equal mean cost. Horizontal ranges differ across sizes.

The Updated pressure intervals lie below zero at $N = 500, 1000$, and include zero at $N = 100, 200$. Shuffling raises cost relative to Initial pressure at $N = 500, 1000$, but slightly lowers it at $N = 100, 200$. Lagged guidance stays close to the initial state, with a small reduction at $N = 1000$. The additional benefit of fresh, destination-aligned updates is supported at the two larger sizes, alongside the benefit of fixed pressure.

Repeatability across training runs. FSP improves on Static at $N = 500, 1000$ for all three independently trained ARC-Base policies (Table 17(c)). Each policy is frozen and evaluated on 1,000 instances per size with greedy $P = N$ decoding and no augmentation. For Run 1, FSP also improves on LogK-only at both sizes (Table 17(d)). Paired 95% instance-bootstrap intervals are reported separately for each fixed checkpoint.

Table 17: Additional guidance controls (1,000 instances/size; Table 6 protocols). Panels (a,b): mean costs, with column minima in bold. Panels (c,d): absolute cost reductions [paired 95% CI]; positive favors FSP. Panel (c) reports Static-to-FSP reductions across three independently trained ARC-Base policies. Each interval conditions on one fixed checkpoint.

(a) ATSP500 guidance controls

Guidance	RADAR Cost ↓	ARC-Base Cost ↓
None	1.60714	1.59251
Random bias	1.84148	1.60663
LogK-only	1.61701	1.58859
Static pressure	1.60300	1.58862
FSP	**1.60271**	**1.58839**

(b) RADAR destination-pressure controls

Guidance	ATSP500 Cost ↓	ATSP1000 Cost ↓
FSP	**1.60271**	**1.61267**
Shuffled pressure	1.60680	1.62574
Lagged pressure	1.60299	1.61354

(c) ARC-Base: Static-to-FSP reduction [paired 95% CI]

Training run	ATSP500	ATSP1000
Run 1	$0.000229\,[0.000112, 0.000348]$	$0.000404\,[0.000252, 0.000559]$
Run 2	$0.000115\,[0.000006, 0.000227]$	$0.000312\,[0.000163, 0.000466]$
Run 3	$0.000131\,[0.000015, 0.000248]$	$0.000330\,[0.000182, 0.000480]$

(d) ARC-Base: FSP vs. LogK-only reduction [paired 95% CI]

Run 1	$0.000198\,[0.000028, 0.000368]$	$0.000399\,[0.000214, 0.000578]$

D.5 Guidance across Construction Stages

Figure 12 compares the three reading windows at all four sizes.

Reading FSP guidance in the final third gives the largest observed mean cost reduction among the three windows at $N = 500, 1000$: 1.86 and 6.59 per mille, respectively (Table 18). The comparison uses frozen RADAR with full-POMO greedy decoding, $\lambda = 1$, and 1,000 instances per size, following Appendix D.4. Early, middle, and late windows cover $[0, 1/3)$, $[1/3, 2/3)$, and $[2/3, 1]$ of $t/(N-1)$. State updates continue throughout every rollout; only the policy correction is gated.

Table 18: Guidance-window cost reductions (‰) relative to no FSP. Entries are mean reductions [paired 95% intervals]; positive values favor guidance.

N	Early	Middle	Late
100	$0.14\,[0.03, 0.25]$	$0.23\,[0.11, 0.34]$	$0.21\,[0.13, 0.30]$
200	$0.22\,[0.11, 0.33]$	$0.40\,[0.29, 0.52]$	$0.48\,[0.39, 0.57]$
500	$0.07\,[-0.09, 0.24]$	$0.90\,[0.74, 1.05]$	$1.86\,[1.71, 2.00]$
1000	$1.14\,[0.91, 1.38]$	$3.30\,[3.09, 3.51]$	$6.59\,[6.39, 6.79]$

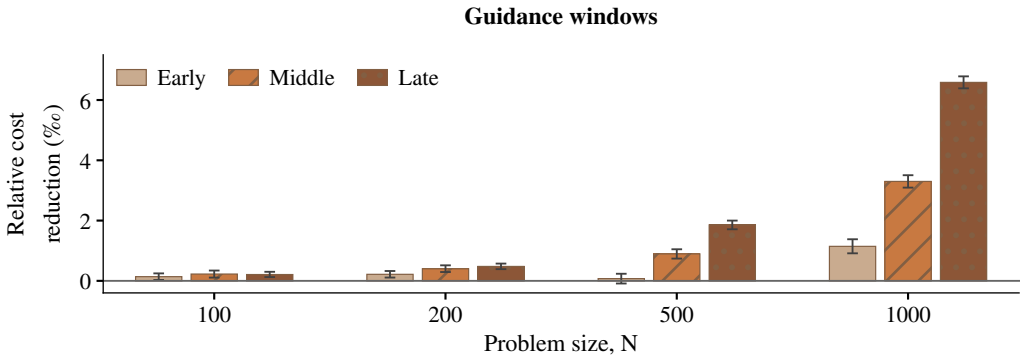


Figure 12: Guidance-window cost reductions on 1,000 instances per size; error bars show paired 95% instance-bootstrap intervals. State updates remain active throughout; only the guidance reading window changes.

Each interval compares a window with no guidance, using the fixed observed no-FSP mean as denominator (Appendix B.3). Different gated policies construct their own trajectories; the reductions are not additive, and the intervals do not test differences between windows.