EXPOSING LOCAL VULNERABILITIES IN NEURAL ROUTING SOLVERS WITH AN EFFICIENT BLACK-BOX ATTACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural routing solvers combine high solution quality with fast inference, yet a simple greedy probing attack exposes substantial local vulnerabilities: bounded coordinate perturbations subject to geometric constraints raise the mean solution gaps of representative neural solvers to roughly 4–35 times their original values on TSP instances with 100 nodes. To find stronger adversarial instances efficiently, we propose the **R**eference-Route **R**euse **A**ttack (RRA), a constrained black-box attack that maximizes the solution gap. Its Adam-ES update combines evolution strategies with adaptive momentum to aggregate candidate feedback within and across iterations. RRA further exploits local coordinate changes to reuse recent reference routes for candidate scoring, refreshing the reference after each update. At the same model-query budget, RRA increases the observed mean best-found gap over the greedy attack by 20–31% across three solvers on these instances. Relative to its full-reference variant, it reduces search-time reference solves by 96.94%, yielding 2.37–$2.52\times$ speedups on this TSP benchmark and 6.9–$12.2\times$ on CVRP with 50 customers. Fine-tuning a neural solver on these adversarial instances lowers its gaps under renewed fixed-budget attacks, linking local vulnerability analysis to model improvement.

1 INTRODUCTION

Neural solvers offer fast, learned heuristics for the Traveling Salesman Problem (TSP) and the Capacitated Vehicle Routing Problem (CVRP). They construct routes with attention, generate solutions through diffusion, or improve them through learned reconstruction and insertion (Kool et al., 2019; Kwon et al., 2020; Sun & Yang, 2023; Luo et al., 2025b). Progress is commonly measured by average solution quality on in-distribution test sets and generalization to out-of-distribution (OOD) instances. In real-world applications, however, small uncertainties in node locations can change the input to a deployed solver. *Does good average performance imply reliable solutions under such small changes?*

We probe this reliability with *adversarial attacks* that seek bounded coordinate perturbations increasing a fixed solver's gap to a reference solution. The resulting *adversarial instances* obey both local displacement and overall geometry constraints. Even a simple greedy attack substantially degrades solution quality. It evaluates a set of local perturbations, selects an improving proposal, and maps it to the constrained neighborhood. After 100 iterations on uniformly sampled TSP instances, POMO's mean gap rises from 0.29% to 4.30%, and Pointerformer's from 0.52% to 18.35% (Table 1). Figure 1 illustrates this degradation. These errors arise while both individual displacements and changes in the overall point distribution remain bounded.

This setting differs from generating harder training distributions to improve cross-distribution generalization, as in HAC (Zhang et al., 2022): our goal is to identify local vulnerabilities around in-distribution inputs while keeping the node set fixed. ROCO studies a complementary setting, restricting graph modifications to guarantee a no-worse optimum (Lu et al., 2023). For continuous coordinate perturbations, the optimum can change in either direction. Our attacks therefore track a reference cost and maximize the solution gap, so that changes in instance cost are distinguished from deterioration in solution quality.

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.

To construct stronger attacks with less computation, we propose the *Reference-Route Reuse Attack (RRA)*, which maximizes the solution gap through queries of returned routes and their costs. Its update component, Adam-ES, improves on greedy selection by using feedback from the entire candidate set and previous iterations. Evolution strategies (ES) combine candidate scores into a direction, and Adam accumulates and adaptively scales these directions across iterations. At the same model-query budget, this combined update yields higher mean best-found gaps than Greedy on the three backbones at 100 nodes.

RRA further improves attack efficiency by reusing a cache of recent reference routes: local coordinate changes preserve their feasibility and bound their cost change. It scores each candidate against the shortest cached route evaluated at the new coordinates. After the Adam-ES update, it computes a fresh reference, using that route both to measure the attained gap and to refresh the cache. We bound the resulting reference error in terms of coordinate displacement and the quality of the cached tours. The same principle extends to CVRP with a fixed depot, demands, and capacity.

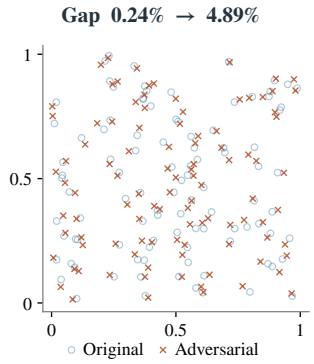


Figure 1: A greedy attack degrades POMO on TSP100 with small coordinate shifts.

Our contributions are:

- We expose local vulnerabilities across constructive, diffusion-based, and refinement-based routing solvers under fixed-node coordinate perturbations that conventional average-quality benchmarks do not reveal.
- We develop RRA, combining Adam-ES updates with reference-route reuse in one black-box attack, supported by an approximation bound and empirical analysis.
- We characterize recovery through additional inference and show that RRA-generated training data improves POMO under renewed attacks at a fixed generation budget.

2 Related Work

2.1 Neural Routing Solvers

End-to-end construction. Attention-based solvers encode the instance and construct a tour by selecting one unvisited node at a time (Kool et al., 2019). POMO explores multiple starting points, Pointerformer combines multiple pointers, and Sym-NCO exploits solution symmetries (Kwon et al., 2020; Jin et al., 2023; Kim et al., 2022). To improve generalization, BQ-NCO, LEHD, INViT, and DGL represent the remaining or local problem more explicitly (Drakulić et al., 2023; Luo et al., 2023; Fang et al., 2024; Xiao et al., 2025). Other approaches diversify solutions through collaborative ensembles (CNF) or strategy-conditioned decoding (PolyNet), condition policies on the instance (ICAM), or incorporate local structure (ELG) (Zhou et al., 2024b; Hottung et al., 2025; Zhou et al., 2026a; Gao et al., 2024). L2R learns to reduce the search space, while URS uses unified representations and problem-conditioned parameters for cross-problem generalization (Zhou et al., 2026b;c). TTPL learns an input projection for construction (Chen et al., 2025). For CVRP, MDAM learns diverse decoders and AMDKD distills policies across training distributions (Xin et al., 2021; Bi et al., 2022).

Neural search and refinement. Additional computation can improve an initial solution. LEHD and SIL reconstruct partial tours, L2C-Insert learns insertion decisions, and GLOP combines global partitioning with local construction (Luo et al., 2023; 2025a;b; Ye et al., 2024). DIFUSCO and FastT2T use diffusion to generate or revise solutions (Sun & Yang, 2023; Li et al., 2024). We attack the complete inference procedure to test whether these quality gains persist under local perturbations.

2.2 Adversarial Robustness and Black-Box Attacks

Geisler et al. (2022) attack TSP by inserting nodes while keeping the updated optimum known; ROCO constrains graph modifications to guarantee a no-worse optimum (Lu et al., 2023). HAC generates harder training distributions for curriculum learning (Zhang et al., 2022). CNF adversarially trains collaborative ensembles without restricting attribute changes to a small neighborhood (Zhou et al., 2024b). For multi-objective solvers, Liu et al. (2026) generate preference-conditioned adversarial instances through input gradients. Our query-only attack keeps the node set fixed and geometric changes bounded around in-distribution inputs. The perturbed instances' reference costs vary with their coordinates, requiring the solution gap to be tracked during an attack. Appendix C.6 tests a constrained adaptation of HAC's released generator.

Evolution strategies and zeroth-order methods estimate directions from function evaluations (Spall, 1992; Wierstra et al., 2014; Salimans et al., 2017). Paired queries and adaptive momentum support black-box attacks and optimization (Ilyas et al., 2018; Chen et al., 2019). Within RRA, Adam-ES uses these tools to combine feedback from local perturbations; feasible-route reuse reduces reference computation at the same model-query budget.

3 Attack Setting and Metrics

We formulate the attack for Euclidean TSP; Appendix F extends it to CVRP with fixed demands and capacity. An instance $X = (x_1, \ldots, x_n)$ contains n nodes, with $x_i \in [0,1]^2$ the coordinates of node i. Let Π_n be the set of permutations of $\{1, \ldots, n\}$, representing visiting orders $\pi = (\pi_1, \ldots, \pi_n)$, where π_i is the i-th visited node and $\pi_{n+1} = \pi_1$ closes the tour. The tour cost and optimal cost are

$$C(\pi, X) = \sum_{i=1}^{n} \|x_{\pi_i} - x_{\pi_{i+1}}\|_2, \qquad C^*(X) = \min_{\pi \in \Pi_n} C(\pi, X). \tag{1}$$

We denote a neural solver by $f_{\theta,c}$, where θ denotes its model parameters and c specifies decoding, starts, augmentation, ensembles, and refinement budget. The attacker queries the returned tour and its length $L_{\theta,c}(X) = C(f_{\theta,c}(X), X)$, without accessing parameters, gradients, or probabilities. An attack fixes the parameters, inference configuration, and random seed.

In the main TSP experiments, original instances follow the solver's training distribution. The coordinate query region is

$$\mathcal{B}_\epsilon(X) = \{X' \in [0,1]^{n\times 2} : \|X' - X\|_\infty \le \epsilon\}. \tag{2}$$

Here X' is a perturbed instance, ϵ is the maximum change to either coordinate of any node, and $\|\cdot\|_\infty$ denotes the largest absolute coordinate entry. The box bounds local movement; geometric constraints preserve the centroid and limit marginal-distribution drift and changes in mean nearest-neighbor distance. Together they define the feasible neighborhood $\mathcal{A}_\epsilon(X) \subseteq \mathcal{B}_\epsilon(X)$. Appendix B.1 specifies these constraints and the feasibility map.

We use the fast classical heuristic LKH-3 (Helsgaun, 2000; 2017) to obtain reference tours with length $C_{\mathrm{ref}}(X)$ and define the solution gap as

$$G_{\theta,c}(X) = 100\left(\frac{L_{\theta,c}(X)}{C_{\mathrm{ref}}(X)} - 1\right) \quad (\%). \tag{3}$$

The black-box attack approximately solves $\max_{X' \in \mathcal{A}_\epsilon(X)} G_{\theta,c}(X')$ within a fixed query budget: it changes the input coordinates to increase the solver's relative error, while keeping the solver fixed. Appendix A.3 checks the references against Concorde on clean and attacked instances.

Over T attack iterations, let X_t be the feasible instance obtained by perturbing and updating the previous instance X_{t-1}, starting from $X_0 = X$. Define $B_t = \max_{0 \le s \le t} G_{\theta,c}(X_s)$. We report the best-found gap B_T, its increase $\Delta G_T = B_T - B_0$, and the Average Adversarial Gap

$$\mathrm{AAG}_T = \frac{B_0/2 + \sum_{t=1}^{T-1} B_t + B_T/2}{T}. \tag{4}$$

B_T measures the strongest instance found; AAG_T summarizes the attack trajectory and is larger when high gaps are reached earlier. Metrics use the recorded update points and are averaged over instances. We also report $\kappa_T = \overline{B}_T/\overline{B}_0$, the ratio of mean best-found to initial gap, where bars denote instance means.

4 REFERENCE-ROUTE REUSE ATTACK

Writing X^0 for the original instance, we maximize its perturbed solution gap $G_{\theta,c}(X')$ within the constrained neighborhood $\mathcal{A}_\epsilon(X^0)$. RRA couples two components: Adam-ES uses candidate scores and historical directions to update coordinates, while reference-route reuse supplies these scores efficiently (Figure 2).

4.1 ADAM-ES UPDATES

The greedy attack retains only the winning displacement and discards information in the other candidate scores. To use this feedback more efficiently, the Adam-ES component uses all scores to estimate a direction, then accumulates directions across iterations through adaptive momentum.

At iteration t, for even M, we sample $M/2$ arrays $Z_{tj} \in \mathbb{R}^{n\times 2}$ with independent $\mathcal{U}[-\sigma,\sigma]$ entries, where σ sets the noise amplitude. Setting $Z_{t,j+M/2} = -Z_{tj}$ for $j = 1,\dots,M/2$ compares opposite displacements along the same direction, reducing sampling imbalance relative to independent draws (Salimans et al., 2017). The resulting M candidates are $X_{tj} = \mathrm{clip}_{\mathcal{B}_\epsilon(X^0)}(X_t + Z_{tj})$. With a fresh reference for each candidate, the score is

$$q_{tj} = G_{\theta,c}(X_{tj}) - \lambda R(X_{tj}; X^0), \qquad R(X'; X^0) = \frac{1}{n}\sum_{i=1}^{n}\sum_{a=1}^{2}\left(x'_{(i),a} - x^0_{(i),a}\right)^2. \tag{5}$$

Here $a \in \{1,2\}$ indexes the coordinate axes, $x_{(i),a}$ is the i-th smallest coordinate on axis a, and λ weights R, the sum of squared marginal 2-Wasserstein distances. This soft penalty guides search; recorded gaps omit it. Section 4.2 replaces the fresh-reference gap with a route-reuse estimate.

A greedy update accepts the highest-scoring candidate if its pre-map score improves the current score, then applies the feasibility map without a second acceptance test.

We standardize the candidate scores and estimate a direction from the weighted perturbations:

$$\widetilde{q}_{tj} = \frac{q_{tj} - \bar{q}_t}{\max(s_t, \tau)}, \qquad D_t = \frac{1}{M}\sum_{j=1}^{M}\widetilde{q}_{tj} Z_{tj}, \tag{6}$$

where $\bar{q}_t$ and s_t are the mean and population standard deviation of the M scores, D_t is their noise-weighted update direction, and τ is a numerical stabilizer. Thus every candidate contributes to the update, rather than only the winner.

To retain useful directions across iterations, we apply Adam (Kingma & Ba, 2015) to the estimated ascent direction rather than a model gradient. Let $\widehat{D}_t$ be D_t after neighborhood smoothing (Eq. (B3)). With t starting at zero and $m_{-1} = v_{-1} = 0$, the first- and second-moment estimates are

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\widehat{D}_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2)\widehat{D}_t^2. \tag{7}$$

The decay rates β_1, β_2 control accumulation: m_t retains directions, while v_t tracks their coordinate-wise squared magnitudes to scale the update. For step size η, bias-corrected ascent gives

$$X_{t+1} = \mathcal{P}_{X^0}\left(X_t + \eta \frac{m_t/(1-\beta_1^{t+1})}{\sqrt{v_t/(1-\beta_2^{t+1})} + \tau}\right). \tag{8}$$

Operations are elementwise. The map $\mathcal{P}_{X^0}$ enforces $\mathcal{A}_\epsilon(X^0)$ (Appendix B.1). A fresh LKH solve at X_{t+1} supplies the recorded gap. Algorithm 1 summarizes the procedure.

Our ES-sign control replaces Adam's accumulated moments with the current direction's element-wise sign (Eq. (B4)), isolating the effect of Adam. It and the greedy attack use the same candidate-generation scheme, constraints, and model-query budget as Adam-ES.

4.2 REFERENCE-ROUTE REUSE

A visiting order remains feasible when coordinates move, allowing nearby candidates to share reference routes. RRA exploits this property by maintaining a separate cache $\mathcal{R}_t$ of LKH tours for each

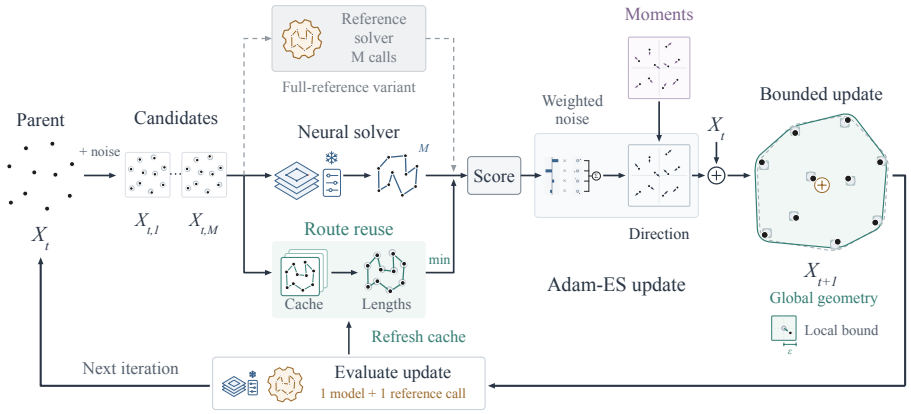


Figure 2: The RRA attack loop. Starting from X_t, the instance coordinates at iteration t, RRA samples M candidates, queries the neural solver, and scores the returned tours using cached references. Adam-ES aggregates candidate feedback and past directions to obtain the constrained update X_{t+1}. A fresh reference solve evaluates this update and refreshes the cache before the next iteration. The dashed branch shows full-reference candidate scoring.

original instance. The candidate reference is

$$U_t(X') = \min_{\pi \in \mathcal{R}_t} C(\pi, X'), \qquad \widehat{G}_t(X') = 100 \left(\frac{L_{\theta,c}(X')}{U_t(X')} - 1 \right). \tag{9}$$

Here $U_t(X')$ is the shortest cached-tour cost on the candidate and $\widehat{G}_t(X')$ is the resulting approximate gap. RRA substitutes $\widehat{G}_t$ into Eq. (5) to drive the Adam-ES update. After each update, it solves a fresh reference, records the gap, and adds the tour to a bounded cache of recent unique tours (Algorithm 1).

For N instances, T iterations, and M candidates per iteration, the full-reference variant (Adam-ES; shaded branch in Figure 2) uses $N[1+T(M+1)]$ reference solves during search, versus $N(T+1)$ for RRA. At $T = 100$ and $M = 32$, RRA reduces reference solves by 96.94% with the same model-query and candidate budgets.

The reused-reference gap lower-bounds the true optimality gap; its error depends on the cached tour's original suboptimality and coordinate drift (Appendix A.1), motivating nearby cache updates. Candidate-dependent errors can change the direction (Appendix A.2); Appendix C.3 illustrates reuse and measures score error, ranking agreement, and direction similarity.

5 Experiments

5.1 Experimental Setup

We compare attacks on the Attention Model (AM), POMO, and Pointerformer (Kool et al., 2019; Kwon et al., 2020; Jin et al., 2023), then apply RRA to other released solvers. AM uses greedy single-start decoding; POMO and Pointerformer select the shortest tour over all starting nodes. For LEHD, SIL, and L2C-Insert, we compare the base configuration with ten refinement iterations, denoted by the suffix -10. Other configurations and dataset generators are specified in Appendices B.2 and B.3.

The main attack comparison uses 128 common uniformly sampled 100-node TSP instances (TSP100), divided into eight blocks of 16. We process one block at a time per GPU. Within each block, neural queries for the 16 instances are batched on an RTX 3080 Ti; independent LKH-3 reference solves run in parallel across 12 CPU workers (Helsgaun, 2017). Each instance is attacked for 100 iterations with 32 candidates per iteration, $\epsilon = 0.03$, and the same random seed. We compare RRA with Greedy, ES-sign, and Adam-ES, which ablates route reuse by solving a fresh reference for every candidate. Additional experiments cover TSP20 and TSP50 (Appendix C.5). We also attack

Table 1: Attack strength and cost on TSP100 and CVRP50.

	TSP100				CVRP50			
Attack	AAG_{100} ↑	B_{100} ↑	Med. ↑	Time/s↓	AAG_{100} ↑	B_{100} ↑	Med. ↑	Time/s↓
	AM				ICAM			
Greedy	14.390	16.698	15.839	105.82	8.377	9.388	9.001	90.22
ES-sign	12.782	14.519	14.086	105.31	7.574	8.405	8.060	88.11
Adam-ES	**16.957**	**21.383**	19.364	78.41	8.808	10.411	**10.255**	81.07
RRA	16.453	20.078	**19.654**	**32.20**	**8.919**	**10.741**	10.194	**6.66**
	POMO				AMDKD-POMO			
Greedy	3.649	4.300	4.184	113.62	6.508	7.279	7.140	95.03
ES-sign	3.024	3.696	3.620	153.01	6.061	6.803	6.704	96.35
Adam-ES	**4.639**	**6.132**	**6.099**	89.23	**7.386**	**8.698**	**8.937**	90.90
RRA	3.981	5.385	5.228	**37.67**	7.376	8.683	8.828	**13.17**
	Pointerformer				MDAM			
Greedy	11.197	18.348	5.897	109.92	12.332	13.865	13.809	92.32
ES-sign	**24.058**	**29.282**	5.462	129.94	11.555	12.584	12.750	94.96
Adam-ES	18.045	23.935	**8.799**	87.11	13.814	15.600	15.379	88.54
RRA	19.508	24.018	8.133	**34.58**	**13.856**	**15.946**	**16.061**	**10.54**

16 CVRP50 instances using ICAM, AMDKD-POMO, and MDAM (Zhou et al., 2026a; Bi et al., 2022; Xin et al., 2021), covering multi-start and multi-decoder inference. These attacks perturb customer coordinates while fixing the depot, demands, and capacity, with HGS references (Vidal, 2022). Appendix F gives the CVRP50 protocol.

5.2 Attack Strength and Cost

Table 1 reports AAG_{100}, B_{100}, and its median (Med.), all in percent; time is the mean block wall-clock attack time divided by 16, in seconds. The initial TSP100 gaps are 4.4505%, 0.2917%, and 0.5239% for AM, POMO, and Pointerformer, respectively; the initial CVRP50 gaps are 2.3983%, 2.0244%, and 5.3426% for ICAM, AMDKD-POMO, and MDAM. All four attacks use the same model-query budget.

Adam-ES achieves the highest B_{100} on AM and POMO, whereas ES-sign leads on Pointerformer. Without Adam's accumulated moments, ES-sign trails Greedy on AM and POMO, showing that candidate aggregation alone need not improve an attack; the gains from more candidates in Table C9 further highlight sensitivity to the sampling budget. RRA achieves B_{100} of 20.078%, 5.385%, and 24.018% with 2.37–2.52× speedups over Adam-ES. At 100 seconds per block, RRA achieves 60%, 87%, and 106% higher gaps than Adam-ES on the three backbones (Figure 3). Single-worker replay links Adam-ES's lower reference time to fewer LKH trials (Appendix C.2). Paired 95% bootstrap intervals for RRA's gap gain over Greedy are above zero for AM and POMO, but span zero for Pointerformer (Appendix A.4). Rechecking Greedy's acceptance after feasibility mapping does not reverse the attack ranking on 16 diagnostic instances (Appendix C.9). Appendix C.10 compares the update rules with a shared route cache on the same 16 instances.

On CVRP50, RRA raises B_{100} over Greedy by 14.4% for ICAM, 19.3% for AMDKD-POMO, and 15.0% for MDAM. It closely matches or exceeds Adam-ES's gaps across all three models and reduces its runtime by 6.9–12.2×.

Pointerformer's RRA median is 8.133%, versus a mean of 24.018%, reflecting a small set of high-gap instances (Appendix C.8). Figure C6 shows one such instance: small displacements increase its tour length from 7.98 to 46.46, while the reference length changes from 7.98 to 8.02.

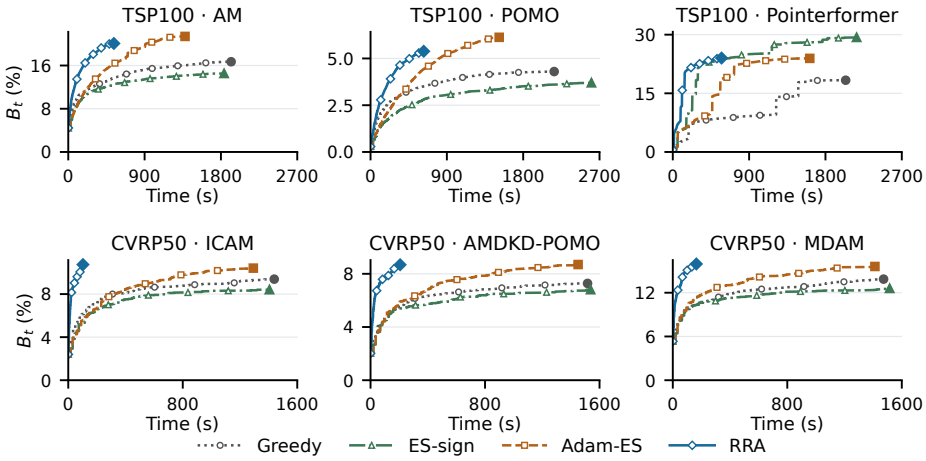


Figure 3: TSP100 and CVRP50 attack progress (time per 16-instance block).

5.3 Robustness across Solvers

Table 2 compares local robustness with accuracy on 1,280 Uniform instances, 10,240 instances from eight OOD distributions, and 16 TSPLIB instances with 51–442 nodes and known optima (Reinelt, 1991) (Appendix B.2). All configurations are attacked on the same 16 TSP100 originals. Time reports inference seconds for 1,280 TSP100 instances with batch size 64, taking the median of five runs.

Table 2: TSP solution quality and gaps found by RRA (%).

Method	Uniform↓	OOD↓	TSPLIB↓	B_0↓	B_{100}↓	AAG_{100}↓	κ_{100}	Time/s↓
AM	4.2831	20.6007	9.6876	4.8248	20.2442	16.9881	4.2	2.77
POMO	0.3646	2.5230	3.0959	**0.2401**	5.4419	3.8740	22.7	4.06
Pointerformer	0.5946	7.7320	5.4620	0.5960	68.5093	56.4949	114.9	**2.75**
Sym-NCO	0.3794	4.6957	3.6711	0.4480	5.4220	4.0276	12.1	5.68
BQ-NCO	**0.3354**	1.5265	**0.6817**	0.2891	5.9681	4.1426	20.6	35.54
LEHD	0.5920	2.2748	0.8368	0.3070	14.3993	9.7636	46.9	27.55
L2C-Insert	0.4519	**1.2408**	0.8559	0.2778	6.7808	4.7722	24.4	43.40
SIL	4.5897	5.7619	3.5351	6.0372	27.6820	23.1203	4.6	55.88
CNF (1 expert)	0.3959	1.7171	3.0730	0.2688	**5.1242**	**3.6903**	19.1	2.86
PolyNet	1.0358	11.4415	4.9986	0.6517	50.8034	33.5338	78.0	3.23
ICAM	0.8613	15.3527	6.2273	0.9679	11.5423	8.0238	11.9	2.82
ELG	1.6033	5.0968	4.2128	1.7003	17.5521	13.5090	10.3	7.53
INViT	6.0494	7.8134	6.4724	6.8274	23.4443	19.4800	3.4	27.01
DGL	2.2420	4.6771	1.9116	2.4112	15.5106	12.8585	6.4	26.41
L2R	2.9647	5.2447	2.9643	3.3509	16.3118	12.8321	4.9	29.07
URS	0.4492	3.7767	2.3361	0.3221	5.8871	4.0323	18.3	5.58
TTPL	1.9339	3.9462	2.1602	1.7766	16.7268	11.1518	9.4	32.29
CNF (3 experts)	0.2983	0.9828	2.8073	0.2429	5.3626	3.8487	22.1	**16.41**
LEHD-10	0.1744	0.6526	0.3429	0.1024	4.5264	3.3576	44.2	146.82
L2C-10†	0.1087	0.2786	**0.3342**	0.1072	1.7900	1.3571	16.7	194.04
SIL-10	1.1492	1.7010	1.3053	0.7501	9.4088	7.1802	12.5	291.98
GLOP	**0.0677**	**0.1372**	0.4467	**0.0402**	**1.5687**	**1.1561**	39.1	503.55
DIFUSCO	1.5338	6.3790	4.9122	1.6934	20.9796	18.4127	12.4	459.40
FastT2T	0.1927	5.6594	3.3424*	0.1569	12.7476	9.0612	81.2	115.75

Groups separate construction from additional inference computation. Bold and underline mark the best and second-best within each group. *FastT2T covers 15 TSPLIB instances (the 442-node case exceeds 12 GB), excluded from rankings. †L2C-10 uses per-instance maxima over evolving batch contexts, excluded from B_{100} and AAG rankings. See Appendix B.3 for deployment settings.

High conventional accuracy does not ensure local robustness. BQ-NCO improves on POMO's Uniform gap, 0.3354% versus 0.3646%, but has a larger best-found attack gap, 5.9681% versus 5.4419%. Pointerformer is fastest but reaches the largest attacked gap; GLOP has the lowest attacked gap at substantially higher inference cost. Across configurations, the mean best-found gap is 3.4–115 times the initial mean. The trajectories in Appendix C.1 show this growth. Fifteen CVRP50 configurations show the same pattern, with mean gaps rising to 2.3–5.5 times their initial values (Table F1).

5.4 Reference Accuracy and Attack Sensitivity

RRA's candidate gap estimates differ from fresh-reference gaps by 0.33–0.44 percentage points on average; rank correlations are 0.83–0.97 and update-direction cosines are 0.83–0.98 (Appendix C.3). At equal queries, reuse can weaken attacks on smaller TSP instances, most clearly at 20 nodes (Table C10). Increasing POMO's candidate count from 16 to 64 raises AAG_{100} from 3.6840% to 4.9429%, at a 3.2× time cost (Table C9). Under the same constraints, HAC-local, adapted from HAC's gradient-based generator (Zhang et al., 2022), reaches $B_{100} = 0.3264\%$ on 16 POMO instances, versus 5.4419% for RRA (Appendix C.6).

Coordinate checks remain within the prescribed displacement and geometry bounds (Appendix C.4). Cross-model transfer is uneven (Table C12), but joint RRA attacks raise the gap even when an ensemble returns its shortest member tour (Appendix E.4).

5.5 Inference-Time Recovery

Two-view inference reduces adversarial gaps on all three backbones but leaves them above clean-input gaps (Appendix E.1). We next test route repair and neural refinement.

Local route repair. We apply up to 10,000 improving 2-opt exchanges to the 16 originals and adversarial instances from Table 2, together with its full OOD and TSPLIB sets (Table 3). Model configurations and reference costs remain unchanged; TSPLIB uses official integer edge weights.

Table 3: Mean and median gaps before and after 2-opt (%).

		AM		POMO		Pointerformer	
Input (count)	Statistic	Neural	2-opt	Neural	2-opt	Neural	2-opt
Uniform (16)	Mean	4.8248	3.8095	0.2401	0.2125	0.5960	0.5161
	Median	5.2989	4.3132	0.0535	0.0000	0.3165	0.1815
Uniform adv. (16)	Mean	20.2442	11.4455	5.4419	5.2884	68.5093	6.3223
	Median	20.6972	12.0162	5.7425	5.6866	7.4377	5.7875
OOD (10,240)	Mean	20.6007	4.9388	2.5230	1.3990	7.7320	2.6391
	Median	8.4900	4.5888	1.2121	0.8573	2.2247	1.4885
TSPLIB (16)	Mean	9.6876	6.0257	3.0959	1.7193	5.4620	2.4139
	Median	7.3037	5.7548	0.4962	0.2301	1.4277	0.7325

The repaired adversarial medians remain 12.02%, 5.69%, and 5.79% for AM, POMO, and Pointerformer, above their OOD and TSPLIB medians. Pointerformer's much larger mean reduction, from 68.51% to 6.32%, reflects repair of exceptionally poor tours rather than comparable recovery on a typical attacked instance. Appendix C.7 gives route examples.

Learned reconstruction and insertion. On the 16 adversarial inputs found against each base and -10 solver, we vary neural refinement without classical local search. Table 4 reports gaps on both sets and full inference time on the -10-targeted inputs. L2C-10's fixed-batch replay differs from its evolving-batch maxima in Table 2 (Appendix E.2).

Ten refinement iterations reduce the mean gap on base-targeted inputs by over 94%. Inputs targeted at the -10 configurations are harder to repair: at ten iterations, LEHD and SIL retain gaps of 4.526% and 9.409%, compared with 0.446% and 1.638% on base-targeted inputs. Thus extra computation

Table 4: Neural refinement on adversarial inputs (gaps in %).

	LEHD			SIL			L2C-Insert		
Refinement iterations	0	5	10	0	5	10	0	5	10
Clean gap	0.307	0.146	0.102	6.037	1.265	0.750	0.278	0.116	0.107
Base-targeted gap	14.399	0.987	0.446	27.682	4.522	1.638	6.781	0.417	0.396
-10-targeted gap	4.761	4.543	4.526	13.491	9.901	9.409	2.313	0.969	0.895
Infer. s/16 inst.	1.2	3.8	6.5	2.4	8.2	13.0	2.1	7.2	8.8

improves recovery, but the strengthened solver remains vulnerable. Increasing FastT2T's diffusion steps from 1 to 50 similarly reduces its gap from 12.748% to 0.199% on fixed adversarial inputs (Appendix E.3).

5.6 Training Value of Adversarial Instances

Efficient attacks can also supply more training data. We fine-tune POMO with Adam-ES and RRA libraries under a 5,400-second generation budget, using random perturbations and unperturbed inputs as controls. All runs use 400 updates, the same mixture of fresh Uniform and library instances, and independent RRA evaluation on 128 held-out instances (Appendix D).

Table 5: Training value under a fixed generation budget (gaps in %).

Training library (originals)	Gen. (s)	Uniform↓	OOD↓	B_{100}↓	AAG_{100}↓
No training	—	<u>0.3646</u>	**2.5230**	5.2402	3.9214
Unperturbed (128)	—	0.3985	2.5455	5.5550	4.2016
Adam-ES (48)	4372.23	0.3793	2.5736	**4.8625**	<u>3.6622</u>
RRA (48)	1805.39	0.3689	2.5776	<u>4.8807</u>	3.6871
RRA (128)	4766.86	**0.3510**	<u>2.5255</u>	4.8855	**3.6247**
Random noise (128)	—	0.3915	2.6030	5.6258	4.2086

At matched library sizes, the attacks yield similar AAG. Within the generation budget, RRA covers 128 originals versus Adam-ES's 48, with a lower observed Uniform gap and similar OOD and re-attack performance (Table 5). Paired comparisons show lower AAG than both the unperturbed and random-noise controls (Appendix D.2).

6 Conclusion

High average accuracy does not ensure reliability under small coordinate changes. Even a greedy probing attack exposes large gaps near instances drawn from the training distribution. RRA combines Adam-ES updates, which exploit candidate feedback and accumulated directions, with efficient reuse of feasible reference routes. Across TSP and CVRP, this attack reveals local vulnerabilities missed by conventional evaluation. RRA also supplies a larger adversarial training pool within a fixed generation budget, and fine-tuning on this pool lowers the evaluated model's gaps under renewed fixed-budget attacks. Local robustness thus complements average accuracy and OOD evaluation, while identifying adversarial instances that can guide training toward more reliable routing solvers.

Ethics Statement

This work evaluates the adversarial robustness of learned combinatorial optimizers using public models and datasets. Adversarial examples can help identify and address unreliable behavior, but can also be used to degrade deployed solvers. The experiments use synthetic routing instances and public TSPLIB benchmarks and contain no personal data.

REPRODUCIBILITY STATEMENT

Appendix A provides the reference-reuse analysis, and Appendix B gives the attack algorithms, settings, the data generation procedure and the fixed deployment configurations used in the experiments. All reported statistics use the original instance as the unit, the attacks use fixed random seeds, and model queries, reference solves and wall-clock time are recorded separately (Appendix C). Appendix E specifies inference-budget comparisons, and Appendix F gives the CVRP settings and HGS reference budgets. The supplementary code implements the four attacks, a POMO interface, and LKH/HGS adapters, with configuration files and a command-line example. The POMO implementation and pretrained weights are included with attribution and the upstream license. Separate reproduction materials provide recorded configurations, per-instance results, model sources, and scripts for reconstructing the principal attack tables without model inference.

AI USE STATEMENT

Generative AI tools assisted with literature search, implementation, result organization, and editing. Numerical tables were checked against saved results, citations against source publications, and the compiled manuscript for notation and layout consistency. The authors remain responsible for the code, results, references, and manuscript.

REFERENCES

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA, 2006.

Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels A. Wouda, Leon Lan, Junyoung Park, Kevin Tierney, and Jinkyoo Park. RouteFinder: Towards foundation models for vehicle routing problems. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=QzGLoaOPiY.

Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 31226–31238, 2022. doi: 10.52202/068431-2264.

Jieyi Bi, Zhiguang Cao, Jianan Zhou, Wen Song, Yaoxin Wu, Jie Zhang, Yining Ma, and Cathy Wu. Towards efficient constraint handling in neural solvers for routing problems. In *International Conference on Learning Representations*, pp. 11907–11936, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1413fbda783a5bad7947c2f7396ba64b-Abstract-Conference.html.

Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. ZO-AdaMM: Zeroth-order adaptive momentum method for black-box optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/576d026223582a390cd323bef4bad026-Abstract.html.

Yuanyao Chen, Rongsheng Chen, Fu Luo, and Zhenkun Wang. Improving generalization of neural combinatorial optimization for vehicle routing problems via test-time projection learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 76543–76585, 2025. doi: 10.52202/085713-2570.

Darko Drakulić, 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, pp. 77416–77429. Curran Associates, Inc., 2023. doi: 10.52202/075280-3385.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. INViT: A generalizable routing problem solver with invariant nested view transformer. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12973–12992. PMLR, 2024. URL https://proceedings.mlr.press/v235/fang24c.html.

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 (IJCAI)*, pp. 6914–6922. International Joint Conferences on Artificial Intelligence Organization, 2024. doi: 10.24963/ijcai.2024/764.

Simon Geisler, Johanna Sommer, Jan Schuchardt, Aleksandar Bojchevski, and Stephan Günnemann. Generalization of neural combinatorial solvers through the lens of adversarial robustness. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://iclr.cc/virtual/2022/poster/7081.

Keld Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000. doi: 10.1016/S0377-2217(99)00284-2.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017. URL http://www.akira.ruc.dk/~keld/research/LKH-3/.

André Hottung, Mridul Mahajan, and Kevin Tierney. PolyNet: Learning diverse solution strategies for neural combinatorial optimization. In *International Conference on Learning Representations (ICLR)*, pp. 80417–80435, 2025. URL https://iclr.cc/virtual/2025/poster/29548.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2137–2146. PMLR, 2018. URL https://proceedings.mlr.press/v80/ilyas18a.html.

Yan Jin, Yuandong Ding, Xuanhao Pan, Kun He, Li Zhao, Tao Qin, Lei Song, and Jiang Bian. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8132–8140, 2023. doi: 10.1609/aaai.v37i7.25982.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1936–1949, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 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 (ICLR)*, 2019. URL https://arxiv.org/abs/1803.08475.

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, pp. 21188–21198. Curran Associates, Inc., 2020.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast T2T: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30179–30206, 2024. doi: 10.52202/079017-0950.

Wei Liu, Yaoxin Wu, Yingqian Zhang, Thomas Bäck, and Yingjie Fan. Adversarial instance generation and robust training for neural combinatorial optimization with multiple objectives. *arXiv preprint arXiv:2601.01665*, 2026. URL https://arxiv.org/abs/2601.01665.

Han Lu, Zenan Li, Runzhong Wang, Qibing Ren, Xijun Li, Mingxuan Yuan, Jia Zeng, Xiaokang Yang, and Junchi Yan. ROCO: A general framework for evaluating robustness of combinatorial optimization solvers on graphs. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://iclr.cc/virtual/2023/poster/11370.

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, pp. 8845–8864. Curran Associates, Inc., 2023. doi: 10.52202/075280-0387.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Boosting neural combinatorial optimization for large-scale vehicle routing problems. In *International Conference on Learning Representations*, pp. 14754–14780, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/264f2e10479c9370972847e96107db7f-Abstract-Conference.html.

Fu Luo, Xi Lin, Mengyuan Zhong, Fei Liu, Zhenkun Wang, Jianyong Sun, and Qingfu Zhang. Learning to insert for constructive neural vehicle routing solver. In *Advances in Neural Information Processing Systems*, volume 38, pp. 87244–87283, 2025b. doi: 10.52202/085713-2918.

Gerhard Reinelt. TSPLIB—a traveling salesman problem library. *ORSA Journal on Computing*, 3 (4):376–384, 1991. doi: 10.1287/ijoc.3.4.376.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017. URL https://arxiv.org/abs/1703.03864. Preprint, arXiv:1703.03864.

J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi: 10.1109/9.119632.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3706–3731, 2023. doi: 10.52202/075280-0164.

Thibaut Vidal. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140:105643, 2022. doi: 10.1016/j.cor.2021.105643.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014. URL https://www.jmlr.org/papers/v15/wierstra14a.html.

Yubin Xiao, Yuesong Wu, Rui Cao, Di Wang, Zhiguang Cao, Xuan Wu, Peng Zhao, Yuanshu Li, You Zhou, and Yuan Jiang. DGL: Dynamic global-local information aggregation for scalable VRP generalization with self-improvement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 8669–8677, 2025. doi: 10.24963/ijcai.2025/964.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12042–12049, 2021. doi: 10.1609/aaai.v35i13.17430.

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.

Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9136–9144, 2022. doi: 10.1609/aaai.v36i8.20899.

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, 2026a. doi: 10.1109/TITS.2026.3674538.

Changliang Zhou, Xi Lin, Zhenkun Wang, and Qingfu Zhang. Learning to reduce search space for generalizable neural routing solver. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), V.2*, pp. 6841–6852, Jeju Island, Republic of Korea, 2026b. ACM. doi: 10.1145/3770855.3818173.

Changliang Zhou, Canhong Yu, Shunyu Yao, Xi Lin, Zhenkun Wang, Yu Zhou, and Qingfu Zhang. URS: A unified neural routing solver for cross-problem zero-shot generalization. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026c. URL https://openreview.net/forum?id=pG4jfjf2g7.

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. MVMoE: Multi-task vehicle routing solver with mixture-of-experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61804–61824. PMLR, 2024a. URL https://proceedings.mlr.press/v235/zhou24c.html.

Jianan Zhou, Yaoxin Wu, Zhiguang Cao, Wen Song, Jie Zhang, and Zhiqi Shen. Collaboration! towards robust neural methods for routing problems. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121731–121764. Curran Associates, Inc., 2024b. doi: 10.52202/079017-3869.

A Reference Bounds and Verification

A.1 Perturbation Bound

We bound the excess cost of the cached reference in Eq. (9), even when perturbation changes the optimal visiting order. Let $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ be two coordinate sets for the same n nodes. The vectors $x_i, y_i \in \mathbb{R}^2$ locate node i before and after perturbation; $d_i = y_i - x_i$ is its displacement. Write $S(X, Y) = \sum_i \|d_i\|_2$ for the total Euclidean displacement. For any edge (i, j), the reverse triangle inequality gives

$$|\|y_i - y_j\|_2 - \|x_i - x_j\|_2| \le \|d_i - d_j\|_2 \le \|d_i\|_2 + \|d_j\|_2. \tag{A1}$$

Summing over a tour counts each node displacement twice. Thus, for every $\pi \in \Pi_n$,

$$|C(\pi, Y) - C(\pi, X)| \le 2S(X, Y), \qquad |C^*(Y) - C^*(X)| \le 2S(X, Y). \tag{A2}$$

To obtain the second inequality, let π^*_X be optimal at X. Its visiting order is also feasible at Y, so

$$C^*(Y) \le C(\pi^*_X, Y) \le C^*(X) + 2S(X, Y). \tag{A3}$$

Exchanging X and Y proves the reverse inequality.

Now consider a cached tour π_s computed at X_s, with suboptimality $e_s = C(\pi_s, X_s) - C^*(X_s)$. At a candidate X', its excess cost decomposes as

$$\begin{aligned} C(\pi_s, X') - C^*(X') &= C(\pi_s, X') - C(\pi_s, X_s) + e_s + C^*(X_s) - C^*(X') \\ &\le e_s + 4S(X_s, X'). \end{aligned} \tag{A4}$$

Taking the shortest tour in the cache $\mathcal{R}_t$ gives

$$0 \le U_t(X') - C^*(X') \le \min_{\pi_s \in \mathcal{R}_t} \{e_s + 4S(X_s, X')\}. \tag{A5}$$

Here $U_t(X')$ is the reused reference cost defined in Eq. (9). The bound separates source-tour error from coordinate drift. It is a worst-case bound; Appendix C.3 measures the approximation error and compares cache policies.

To express this cost error as a gap error, abbreviate the model cost, optimum, and reused cost at the same candidate X' as $L = L_{\theta,c}(X')$, $C^* = C^*(X')$, and $U = U_t(X')$. Define the dimensionless gaps $g^* = L/C^* - 1$ and $g_U = L/U - 1$, and the relative reference error $\rho = U/C^* - 1$. Substitution gives

$$g^* = \frac{L}{C^*} - 1 = (1 + \rho)(1 + g_U) - 1, \qquad g^* - g_U = \rho(1 + g_U). \tag{A6}$$

Multiplying g^* and g_U by 100 gives percentage gaps. Feasibility gives $\rho \ge 0$ and $1 + g_U = L/U > 0$. The bound is relative to the optimum; the next analysis compares against LKH.

A.2 Score Error and Search Direction

For candidate scoring, we examine how reference errors affect preferences and update directions. Fix one iteration t and the same M perturbations Z_{tj} from Eq. (6). Let q_{tj} use fresh LKH references and $\widehat{q}_{tj}$ use cached references, with score error $\xi_{tj} = q_{tj} - \widehat{q}_{tj}$. Bars denote means over candidates. Before standardization, the weighted directions differ by

$$\frac{1}{M}\sum_{j=1}^{M}(\widehat{q}_{tj} - \bar{\widehat{q}}_t)Z_{tj} - \frac{1}{M}\sum_{j=1}^{M}(q_{tj} - \bar{q}_t)Z_{tj} = -\frac{1}{M}\sum_{j=1}^{M}(\xi_{tj} - \bar{\xi}_t)Z_{tj}. \tag{A7}$$

A common score offset cancels, but errors that differ across candidates can change the direction. Standardization in Eq. (6) also changes its magnitude. Appendix C.3 therefore measures score accuracy, candidate ranking, and the resulting update direction on matched perturbations.

A.3 Concorde Verification

To check whether reference error affects the gaps in Table 2, we compare LKH with Concorde (Applegate et al., 2006) on its three primary backbones. For each 16-instance run, we check iterations 0, 20, 40, 60, 80, and 100, plus the returned historical-best coordinates: 336 instance–snapshot pairs.

Concorde receives edge weights $w_{ij} = \mathrm{round}(S_{\mathrm{scale}}\|x_i - x_j\|_2)$ with $S_{\mathrm{scale}} = 10^6$. Every tour contains n edges, so rounding changes its scaled cost by at most $n/2$. If Q^* and π_Q are the certified integer optimum and its tour, then

$$\frac{Q^* - n/2}{S_{\mathrm{scale}}} \le C^*(X) \le C(\pi_Q, X), \qquad C(\pi_Q, X) - C^*(X) \le \frac{n}{S_{\mathrm{scale}}}. \tag{A8}$$

All 336 solves finish within the 120-second limit and return tours matching the certified integer objective. We recompute tour lengths on the original floating-point coordinates. LKH and Concorde lengths agree to numerical precision; Table A1 shows the resulting best-found attack gaps.

Table A1: Best-found gaps with LKH and Concorde references (%).

Model	LKH	Concorde
AM	20.2442	20.2442
POMO	5.4419	5.4419
Pointerformer	68.5093	68.5093

A.4 Uncertainty of the Attack Comparison

Table A2 measures uncertainty across the 128 original instances in Table 1. We bootstrap the paired per-instance differences (RRA minus Greedy or Adam-ES) with 10,000 resamples and report 95% percentile intervals. These intervals are conditional on the fixed attack runs. Positive differences favor RRA. At TSP100, RRA–Greedy intervals are positive for AM and POMO but include zero for Pointerformer, whose rare high-gap instances widen the interval. Against Adam-ES, RRA has a clear equal-iteration deficit on POMO; the intervals include zero for AM and Pointerformer.

Table A2: Paired attack differences on TSP (percentage points).

Backbone	Nodes	ΔB_{100}	95% interval	ΔAAG_{100}	95% interval
RRA minus Greedy					
AM	100	$+3.3801$	$[+2.5844, +4.1749]$	$+2.0625$	$[+1.5397, +2.5882]$
POMO	100	$+1.0849$	$[+0.8536, +1.3146]$	$+0.3319$	$[+0.1408, +0.5225]$
Pointerformer	100	$+5.6699$	$[-6.7120, +20.7179]$	$+8.3115$	$[-0.4146, +19.7271]$
RRA minus Adam-ES					
AM	100	-1.3049	$[-3.9373, +0.5896]$	-0.5040	$[-1.7169, +0.3856]$
POMO	100	-0.7467	$[-0.9924, -0.5033]$	-0.6583	$[-0.8500, -0.4629]$
Pointerformer	100	$+0.0834$	$[-9.1194, +10.2654]$	$+1.4634$	$[-4.5634, +9.3900]$
AM	50	$+0.1811$	$[-0.7184, +1.4046]$	-0.0115	$[-0.5963, +0.7318]$
POMO	50	-0.8646	$[-1.0851, -0.6515]$	-0.8567	$[-1.0388, -0.6806]$
Pointerformer	50	-0.7399	$[-7.3668, +5.6606]$	-0.6009	$[-3.2352, +2.3124]$
AM	20	-0.3386	$[-0.6159, -0.0708]$	-0.3690	$[-0.5844, -0.1614]$
POMO	20	-0.6703	$[-0.8196, -0.5257]$	-0.5209	$[-0.6394, -0.4061]$
Pointerformer	20	-0.5910	$[-0.7571, -0.4379]$	-0.4784	$[-0.6101, -0.3575]$

B Experimental Setup

B.1 Attack Configuration

Table B1 specifies the shared attack settings. Algorithm 1 summarizes RRA. Its full-reference variant, labeled Adam-ES, solves a fresh reference for every candidate instead of using the cache.

Coordinate and distribution constraints. Bounding each node's displacement controls local movement but not how those movements accumulate across an instance. We therefore combine the coordinate box $\mathcal{B}_\epsilon(X^0)$ with three distribution constraints to define $\mathcal{A}_\epsilon(X^0)$.

For candidate coordinates $Y = (y_1, \ldots, y_n)$, we first preserve the centroid $\mu(Y) = n^{-1}\sum_i y_i$ by requiring $\mu(Y) = \mu(X^0)$. This excludes global translation, which leaves route lengths unchanged. Second, we limit marginal-distribution drift to $W_{\text{avg}}(Y, X^0) \le 0.015$. For axis $a \in \{1, 2\}$, sorting the coordinates gives the empirical Wasserstein distance:

$$W_1^{(a)}(Y, X^0) = \frac{1}{n}\sum_{i=1}^{n} |y_{(i),a} - x^0_{(i),a}|, \qquad W_{\text{avg}} = \tfrac{1}{2}\sum_{a=1}^{2} W_1^{(a)}. \tag{B1}$$

Third, we constrain local spacing, which the marginal distributions alone may miss. With mean nearest-neighbor distance $d_{\text{NN}}(Y) = n^{-1}\sum_i \min_{j\neq i} \|y_i - y_j\|_2$, we require

$$\left| \frac{d_{\text{NN}}(Y)}{d_{\text{NN}}(X^0)} - 1 \right| \le 0.10. \tag{B2}$$

The squared marginal-distance penalty in Eq. (5) guides candidate selection, whereas Eq. (B1) imposes a hard distribution-distance limit; Table C9 varies its weight, while Table C5 checks the constraints on returned instances.

Neighborhood smoothing. Nearby nodes jointly determine short tour edges. To favor locally coordinated movements over isolated coordinate changes, we average the direction from Eq. (6) over neighboring nodes before the moment update in Eq. (7):

$$\widehat{D}_{t,i} = (1-\alpha)D_{t,i} + \frac{\alpha}{k}\sum_{\ell \in \mathcal{N}_k(i)} D_{t,\ell}. \tag{B3}$$

Here $D_{t,i}$ is node i's direction, $\mathcal{N}_k(i)$ contains its k nearest other nodes in X_t, and α is the mixing weight. This smoothing is shared by Adam-ES and ES-sign. The moments and coordinate update follow Eqs. (7)–(8), with $\tau = 10^{-8}$. ES-sign removes the historical moments while retaining the same smoothed direction:

$$X_{t+1} = \mathcal{P}_{X^0}\left(X_t + \eta \frac{\widehat{D}_t}{|\widehat{D}_t| + \tau} \right). \tag{B4}$$

Table B1: Default attack configuration.

Parameter	Value
Iterations / candidates	$T = 100$ / $M = 32$ (16 antithetic pairs)
Displacement / noise	$\epsilon = 0.03$ / $\sigma = 0.01$
Learning rate / Adam moments	$\eta = 0.01$ / $(\beta_1, \beta_2) = (0.9, 0.999)$
Marginal penalty	$\lambda = 200$
Spatial smoothing	$k = 10$, $\alpha = 0.5$
RRA cache	$K_{\text{cache}} = 32$ unique recent reference tours
LKH	1 run, at most 1,000 trials, seed 1234

Algorithm 1 Reference-Route Reuse Attack (RRA).

Input: Original X^0, fixed solver $f_{\theta,c}$, T, M, ϵ
Output: Best coordinates, $B_{0:T}$, AAG, query counts, elapsed time

$X_0 \leftarrow X^0$; solve $\pi_{\text{ref}}(X^0)$; $\mathcal{R}_0 \leftarrow \{\pi_{\text{ref}}(X^0)\}$; $B_0 \leftarrow G(X^0)$
for $t = 0, \ldots, T-1$ **do**
  Draw paired noises Z_{tj}; form clipped candidates X_{tj}
  Query $L_{\theta,c}(X_{tj})$ for $j = 1, \ldots, M$
  Evaluate cached-route costs $U_t(X_{tj})$ and gaps $\widehat{G}_t(X_{tj})$ using Eq. (9)
  Substitute $\widehat{G}_t$ into Eq. (5); compute $\widehat{D}_t$ using Eqs. (6) and (B3)
  Update moments by Eq. (7) and coordinates by Eq. (8)
  Solve $\pi_{\text{ref}}(X_{t+1})$ and evaluate $G(X_{t+1})$
  Add the new reference; deduplicate and retain the latest K_{cache} tours
  $B_{t+1} \leftarrow \max\{B_t, G(X_{t+1})\}$; retain the corresponding coordinates
end for

Feasibility and cache maintenance. The map $\mathcal{P}_{X^0}$ clips a proposal to the coordinate box, finds a per-axis shift that restores the centroid after clipping, and contracts its displacement toward X^0 until the distribution constraints hold. Direction-query candidates obey the box; recorded updates obey the full set $\mathcal{A}_\epsilon(X^0)$. Greedy accepts an improving candidate before this map and records its freshly evaluated post-map gap; Appendix C.9 tests a second acceptance check. RRA identifies duplicate tours up to rotation and reversal and retains the most recent K_{cache} unique tours.

B.2 Evaluation Data

Attack comparisons use 128 fixed Uniform originals, split into eight blocks of 16. The first block is used for Table 2 and the auxiliary diagnostics. These originals are separate from attack-parameter development and training data. Conventional quality uses the frozen datasets in Table B2.

Table B2: Evaluation datasets.

Dataset	Nodes	Count	Generator
Uniform	100	1,280	Independent $\mathcal{U}([0,1]^2)$
Gaussian	100	1,280	Mean $(0.5, 0.5)$, std. 0.10
GMM-2	100	1,280	Centers $(0.25, 0.25)$, $(0.75, 0.75)$; std. 0.07
Cluster-3	100	1,280	Three random centers; std. 0.07
Mixed	100	1,280	50 clustered and 50 Uniform nodes
Expansion	100	1,280	Displacement away from a random line
Implosion	100	1,280	Contraction around a random center
Explosion	100	1,280	Radial displacement away from a center
Grid	100	1,280	Local replacement with a regular lattice
Uniform-20 / 50	20 / 50	1,280 each	Independent $\mathcal{U}([0,1]^2)$
TSPLIB	51–442	16	Euclidean instances with known optima

Gaussian and GMM samples are restricted to the unit square; GMM components are equiprobable. Cluster centers are uniform in $[0.2, 0.8]^2$. Mixed combines a single cluster of standard deviation 0.07 with Uniform nodes. The geometric generators modify Uniform coordinates around a random line, center, or local grid.

TSPLIB inputs are translated and isotropically scaled for neural inference; tours are scored in the original coordinates using official integer edge costs and known optima (Reinelt, 1991). Synthetic gaps use fixed LKH references evaluated with floating-point distances. OOD averages the eight equally sized distributions, and TSPLIB weights its 16 instances equally. Per-instance gaps, including negative values, enter these averages unchanged.

B.3 Deployment and Cost Measurement

AM uses single-start greedy decoding throughout. POMO and Pointerformer select the shortest tour across all starting nodes. All solvers use released weights. LEHD, SIL, and L2C-Insert use construction-only inference in their base rows and ten neural refinement iterations in the -10 rows. CNF evaluates all three experts in its main configuration; the single-expert row uses the first released expert. FastT2T uses one diffusion step and one guided rewrite, with classical 2-opt disabled. DIFUSCO uses 50 cosine-scheduled diffusion steps and one sample, without 2-opt. GLOP uses revision lengths 100, 50, 20, and 10 with five iterations each and width 140; TTPL applies a learned projection before LEHD. During attacks, stochastic samplers reset their random state before each decode: FastT2T per instance, and DIFUSCO, GLOP, and the -10 refinement variants per batch. Table 2's attacks use seed 42. Appendix E.4 uses seed 12345 for FastT2T in both joint and individual attacks.

For standard timing, we use 1,280 TSP100 instances, batch size 64, two warm-up batches, and five repetitions. GPU synchronization brackets input transfer, inference, and tour return; we report the median total time. Accuracy evaluation preserves each adapter's fixed batching. L2C-Insert uses batches of 16 for quality and attack evaluation. Its refinement procedure shares a node choice across the batch, so L2C-10's cost depends on the ordered inputs, $L_i(X_1, \ldots, X_{16})$. During attack, each candidate batch contains one proposal per original; recorded updates retain the same membership and order as their coordinates evolve. Reported histories are per-instance maxima over these joint batches. Larger instances use smaller batches, and TSPLIB is evaluated one instance at a time.

A model query is one complete solve under the chosen configuration, including all prescribed starts or experts. Each attack uses $N[1 + T(M + 1)]$ such queries during search: one initial solve and $M + 1$ per iteration. Final returned-coordinate verification costs another N queries. Only reference counts differ: Adam-ES uses $N[1 + T(M + 1)]$, whereas RRA uses $N(T + 1)$. Three repeated 16-instance POMO runs under the defaults in Table B1 produce identical AAG values, with median times of 1,543.74 seconds for Adam-ES and 620.56 seconds for RRA.

TSP attack time in Table 1 starts before initial inference and ends at the last recorded update, excluding model loading and final returned-coordinate verification. Model calls return tours and costs before the timer advances; LKH uses 12 workers for independent instances. CVRP table times additionally include the endpoint checks described in Appendix F.1; its curves stop at the final search update. Figure 3 averages the eight TSP blocks at each recorded time using their latest completed observations; the CVRP curves use one 16-instance block. At 100 seconds, RRA versus Adam-ES gaps are 13.3622% versus 8.3591% (AM), 2.4985% versus 1.3357% (POMO), and 11.5515% versus 5.6196% (Pointerformer).

C Attack Diagnostics

The following analyses explain the attack strength and cost reported in Section 5.2. We examine reference computation, geometric changes, and sensitivity to the attack settings before considering transfer, Greedy acceptance, and update rules under a shared route cache.

C.1 Attack Trajectories across Solvers

To show how the endpoint gaps in Table 2 develop, Figure C1 plots the mean best-found gap B_t defined in Section 3 over the 16 originals used for the common-protocol entries in Table 2. Curves

are grouped for readability and use a logarithmic gap axis. They show the growth and saturation of the gaps during search.

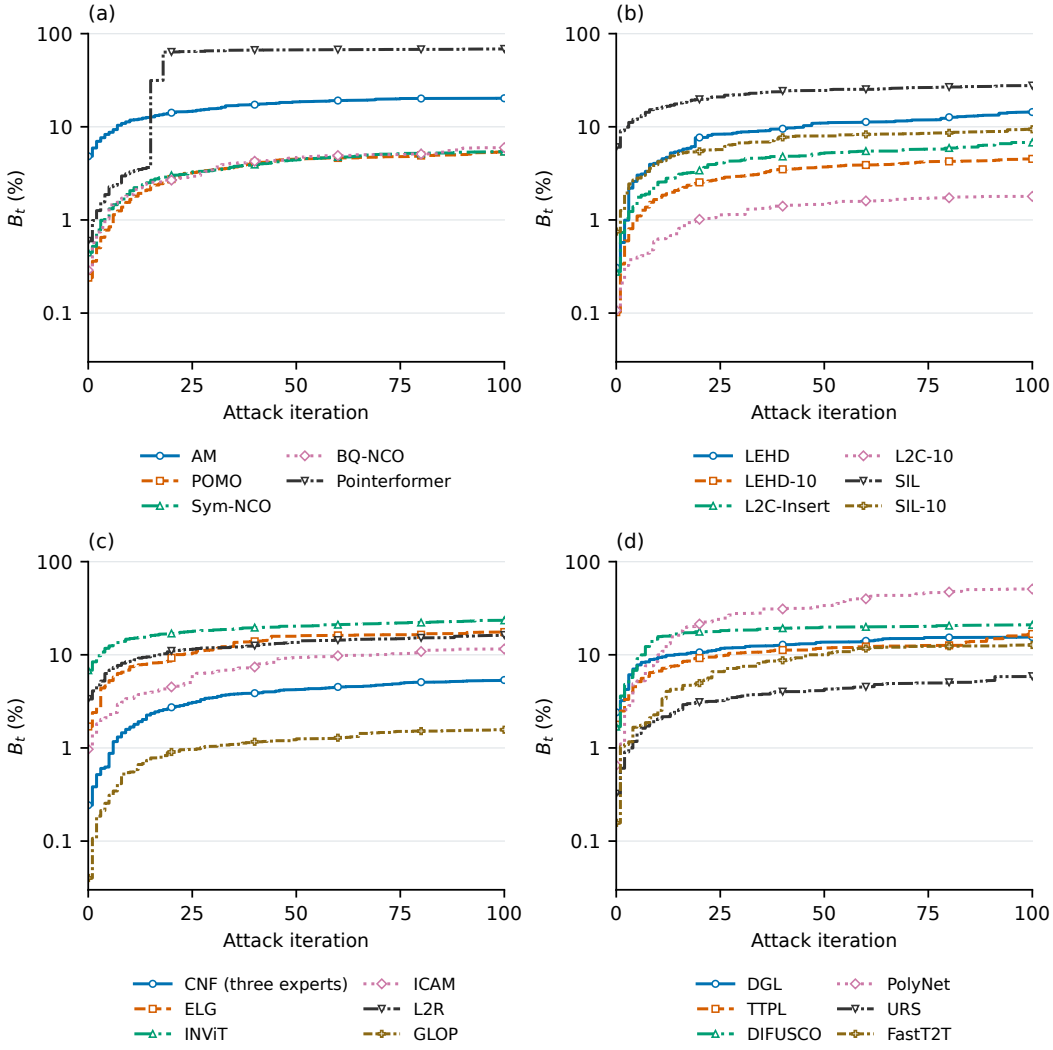


Figure C1: TSP100 attack trajectories across neural solvers. L2C-10 uses the joint-batch histories defined in Appendix B.3.

C.2 LKH Runtime and Reference Shortening

Why does Adam-ES take less time than the equal-query controls in Table 1? To separate changes in solver work from parallel scheduling, we replay the saved coordinates of 128 attacks per method and backbone through LKH twice, using one worker and the original per-instance seeds. Figure C2 shows the POMO trajectories. Mean process CPU time falls from 0.381 to 0.085 seconds per solve for Adam-ES, compared with 0.380 to 0.309 for ES-sign and 0.380 to 0.152 for Greedy. At iteration 100, LKH finishes during its ascent phase on 47.7% of Adam-ES inputs, versus 5.5% and 21.9% for ES-sign and Greedy; mean POMO trials fall from 1,000 to 523. AM and Pointerformer show similar Adam-ES CPU reductions, from 0.381 to 0.122 and from 0.382 to 0.126 seconds, respectively. Thus the runtime difference persists without parallel execution and accompanies a reduction in LKH's search effort. Replaying identical inputs with 1, 6, 12, and 18 workers also returns identical tour lengths.

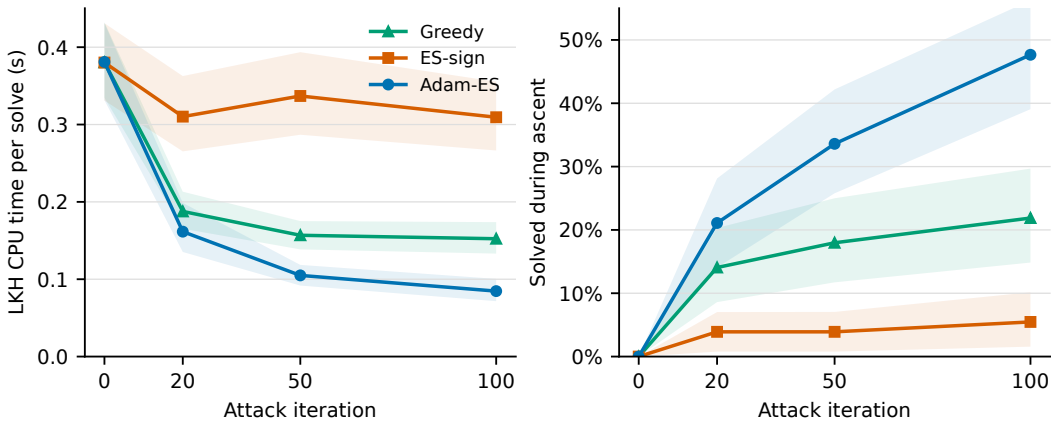


Figure C2: LKH work on frozen POMO attack trajectories. Shading gives 95% bootstrap intervals over 128 original instances; each instance has two timing replays.

Table C1: Reference-length-matched model-blind and Adam-ES attacks.

		LKH tour		Neural tour		Gap (%)	
Model	Pairs	Blind	Adam-ES	Blind	Adam-ES	Blind	Adam-ES
AM	110	7.359	7.360	7.755	8.598	5.39	16.85
POMO	90	7.445	7.448	7.462	7.837	0.23	5.22
Pointerformer	89	7.456	7.456	7.486	7.976	0.39	6.99

To test whether larger gaps merely reflect shorter reference tours, we generate 128 model-blind trajectories under the same 100-iteration candidate budget and the constraints in Appendix B.1. At each iteration, we map 32 paired uniform-noise candidates into $\mathcal{A}_\epsilon(X^0)$ and accept the candidate with the shortest LKH tour only if it shortens the current reference; no neural model is queried during generation. For each original instance, we select the saved blind iteration whose reference length is closest to the Adam-ES endpoint and retain pairs within 0.5% relative length. Table C1 compares the two coordinate sets using the same model and reference solver. Despite closely matched reference lengths, Adam-ES yields longer neural tours and larger gaps on all three backbones; the targeted gap is higher in 110/110 AM, 90/90 POMO, and 86/89 Pointerformer pairs. Reference shortening alone therefore does not account for the observed degradation. All 12,928 saved blind instance–iteration pairs satisfy the four local constraints.

We also remove the changing denominator from the candidate objective in Eq. (5), replacing its gap term with $100[L_{\theta,c}(X') - C_{\mathrm{ref}}(X')]/C_{\mathrm{ref}}(X^0)$, where $C_{\mathrm{ref}}(X^0)$ is the original instance's reference length. Candidate generation, regularization, geometric constraints, and evaluation with the ordinary gap remain unchanged. Table C2 compares the two search objectives on 32 originals per backbone. Both scores produce similar evaluated gaps at iteration 100.

For 432 distinct clean and perturbed coordinates in these diagnostics, Concorde certifies that LKH attains the exact optimum under the same rounded EUC_2D edge costs in every case. Thus the reduced reference time in these diagnostics reflects less solver work, rather than a change in the reference configuration or tour validity.

C.3 Reference Error and Cache Updates

We examine whether the cached reference in Eq. (9) preserves useful candidate scores, and whether refreshing the cache improves the attack.

Figure C3 illustrates reuse with one cached LKH tour on a TSP20 candidate drawn with $\sigma = 0.01$. Moving the nodes preserves the visiting order's feasibility, while its length changes from 3.53105 to 3.56542. A fresh LKH solve returns 3.56357 on the same coordinates: the reused route is 0.00185

Table C2: Gap at iteration 100 with current- and fixed-reference denominators.

	Gap at iteration 100 (%)	
Model	Ordinary score	Fixed denominator
AM	16.90	16.36
POMO	5.30	4.77
Pointerformer	6.54	6.47

longer, a relative difference of 0.052%. Thus the same visiting order can supply a nearby candidate's reference without rerunning LKH.

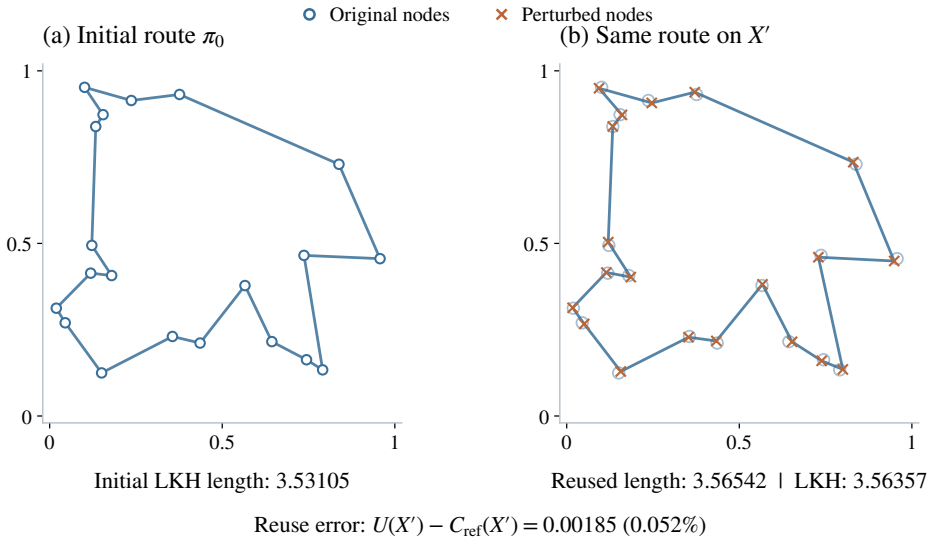


Figure C3: Reusing an initial TSP20 tour after coordinate perturbation.

We compare RRA scores with fresh LKH scores on the same 32 candidates at iterations 1, 10, 31, 50, and 99 of 16-instance attacks. Diagnostic solves do not feed back into the search. For percentage-gap scores $\widehat{G}_j = \widehat{G}_t(X_{tj})$ and $G_j = G_{\theta,c}(X_{tj})$, we measure

$$\text{MAE} = \frac{1}{M}\sum_j |\widehat{G}_j - G_j|, \qquad r_s = \text{corr}(\text{rank}(\widehat{G}), \text{rank}(G)). \tag{C1}$$

MAE is mean absolute error in percentage points; r_s is Spearman rank correlation. With the same regularizer and smoothing, Eq. (B3) gives directions $\widehat{D}_{\text{reuse}}$ and $\widehat{D}_{\text{LKH}}$ from cached and fresh references. Their cosine similarity is

$$s_D = \frac{\langle \widehat{D}_{\text{reuse}}, \widehat{D}_{\text{LKH}} \rangle_F}{\|\widehat{D}_{\text{reuse}}\|_F \|\widehat{D}_{\text{LKH}}\|_F}. \tag{C2}$$

The Frobenius inner product treats the coordinate arrays as vectors. We average all 80 instance–iteration pairs per model; AM uses single-start decoding. Table C3 reports their means. The reference approximation changes gap values by 0.33–0.44 percentage points while retaining positive ranking and direction agreement.

To test the need for cache updates, we compare an initial-tour-only cache with caches of the most recent one or 32 tours. All arms still solve LKH at each recorded update point. Table C4 shows that retaining 32 recent tours increases AAG relative to a frozen cache for all three backbones. With 32 tours, Pointerformer has a lower median B_{100} but a higher mean AAG, consistent with its heavy error tail.

Table C3: Accuracy of cached-reference scores and update directions.

Model	MAE (pp) ↓	Rank correlation ↑	Direction cosine ↑
AM	0.4386	0.9726	0.9834
POMO	0.3282	0.8315	0.8278
Pointerformer	0.4104	0.8908	0.8992

Table C4: Effect of reference-cache updates. Times cover 16 instances.

Backbone	Reference set	AAG_{100} ↑	Median B_{100} ↑	Time/s ↓
AM	Frozen	15.2017	17.8621	530.90
AM	Recent 1	16.1300	18.9789	507.94
AM	Recent 32	16.9881	20.6972	548.50
POMO	Frozen	3.3379	5.2957	620.98
POMO	Recent 1	3.6611	4.9120	611.57
POMO	Recent 32	3.8740	5.7425	613.24
Pointerformer	Frozen	37.4212	8.1765	583.94
Pointerformer	Recent 1	34.5706	9.1552	558.79
Pointerformer	Recent 32	56.4949	7.4377	575.10

C.4 Coordinate Constraints and Perturbation Budgets

We check whether the attack gains occur within the feasible neighborhood $\mathcal{A}_\epsilon(X^0)$ defined in Appendix B.1, measuring displacement, centroid error, marginal distance (Eq. (B1)), and local spacing (Eq. (B2)). Table C5 checks all four local constraints on the 16 POMO instances; none of the returned coordinates violate them. Table C6 extends the displacement and marginal-distance checks to 128 instances per backbone, with single-start AM. The mean W_{avg} is at most 0.0113 and the maximum is 0.0143. Figure C4 compares coordinate changes on the same original instance attacked against POMO; colors identify nodes and axes are shared. The HAC panel shows its original update, whereas HAC-local applies our perturbation constraints.

Table C5: Local-constraint checks on POMO.

Attack	Displacement	Mean error	W_{avg}	NN change	Violations
Greedy	0.0300	4.79×10^{-16}	0.0095	0.0834	0/16
Adam-ES	0.0300	4.86×10^{-16}	0.0141	0.0834	0/16
RRA	0.0300	4.49×10^{-16}	0.0131	0.0967	0/16

Each numeric column reports the largest value over returned instances. Displacement is the largest absolute coordinate change; mean error is the largest per-axis mean change; NN change is $|d_{\mathrm{NN}}(X')/d_{\mathrm{NN}}(X) - 1|$. Violations count instances exceeding any constraint in Appendix B.1.

Table C7 varies ϵ for POMO on 16 instances with initial gap 0.2401%. The best-found gap increases up to approximately $\epsilon = 0.03$ and then plateaus while the additional geometric constraints remain active.

On the same 16 POMO instances, at both $\epsilon = 0.01$ and 0.03, RRA exceeds the greedy attack in AAG_{100} with 1,616 rather than 52,816 reference calls (Table C8). Adam-ES attains the largest equal-iteration gap at the higher reference cost. Gaps are percentages; times are seconds throughout this appendix unless stated otherwise.

C.5 Attack Parameters and Problem Size

To assess sensitivity around the defaults in Table B1, Table C9 changes one parameter at a time on the same 16 POMO instances. The candidate count has the largest effect within the tested ranges:

Table C6: Marginal drift of RRA instances.

Model	Max. displacement	Mean W_{avg}	Max. W_{avg}	Violations
AM	0.0300	0.0113	0.0143	0/128
POMO	0.0300	0.0112	0.0139	0/128
Pointerformer	0.0300	0.0112	0.0133	0/128

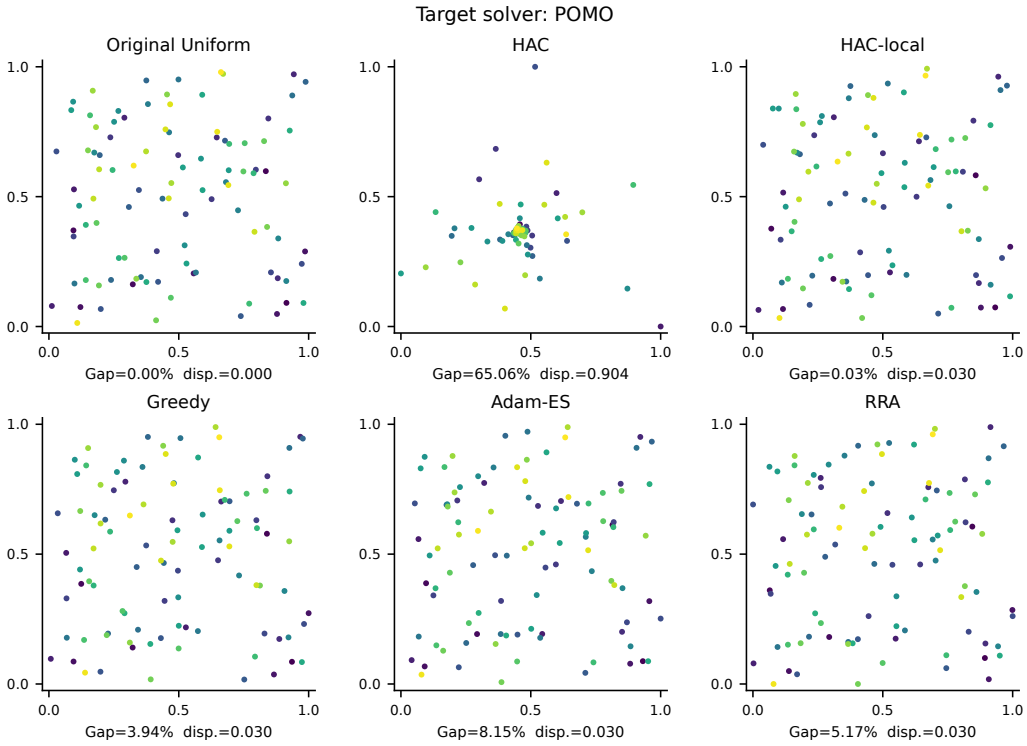


Figure C4: Coordinate changes under different attack rules.

increasing it from 16 to 64 raises AAG from 3.6840% to 4.9429% and runtime from 350.39 to 1,117.91 seconds. Learning rate and regularization weight have smaller effects.

To test whether the TSP100 comparison extends to smaller problems, Table C10 repeats the four attacks on 128 instances each of TSP20 and TSP50 using size-matched checkpoints; PF abbreviates Pointerformer. Initial gaps for AM/POMO/Pointerformer are 0.3582/0.0300/0.0422% at $n = 20$ and 1.7463/0.1043/0.1306% at $n = 50$. Attacks reach smaller absolute gaps on smaller instances, while the reference-call reduction remains unchanged. At TSP20, Greedy exceeds RRA at equal queries on all three backbones. The multiplier κ_{100} is the ratio of mean best-found to mean initial gap, as in

Table C7: Sensitivity to the coordinate budget. Times cover 16 instances.

Budget ϵ	AAG_{100} ↑	B_{100} ↑	Time/s ↓
0.01	3.0377	3.9879	630.34
0.02	3.1290	4.3203	607.35
0.03	3.8740	5.4419	637.89
0.05	4.0801	5.5629	607.58
0.07	3.9677	5.5025	605.41
0.10	3.8514	5.5167	623.88

Table C8: Attack comparison at two perturbation budgets.

Budget ϵ	Attack	$\text{AAG}_{100} \uparrow$	$B_{100} \uparrow$	LKH $\downarrow$
0.01	Greedy	2.4865	2.8805	52,816
	Adam-ES	**3.3868**	**4.1813**	52,816
	RRA	3.0377	3.9879	**1,616**
0.03	Greedy	3.3424	3.8670	52,816
	Adam-ES	**4.6915**	**6.1487**	52,816
	RRA	3.8740	5.4419	**1,616**

Table C9: RRA parameter sensitivity on POMO. Times cover 16 instances.

Parameter	Value	$\text{AAG}_{100} \uparrow$	$B_{100} \uparrow$	Time/s $\downarrow$
Candidates M	16	3.6840	4.9078	350.39
Candidates M	32	3.8740	5.4419	637.89
Candidates M	64	4.9429	6.6455	1117.91
Step size η	0.005	4.0572	5.8184	582.60
Step size η	0.01	3.8740	5.4419	637.89
Step size η	0.02	3.8278	5.0293	621.25
Noise σ	0.005	3.5574	4.7468	619.38
Noise σ	0.01	3.8740	5.4419	637.89
Noise σ	0.02	3.7127	5.1766	588.51
Penalty λ	0	3.7369	4.9956	602.21
Penalty λ	100	3.8877	5.4333	597.52
Penalty λ	200	3.8740	5.4419	637.89
Penalty λ	400	3.9499	5.3124	604.80

Table 2. Time is the average wall-clock cost per instance, as in Table 1. Figure C5 plots the same attacks against elapsed time per 16-instance block, averaging the eight blocks as in Figure 3.

Table C10: Attack strength and cost on TSP20 and TSP50.

		TSP20				TSP50			
Model	Attack	$\text{AAG}_{100} \uparrow$	$B_{100} \uparrow$	κ_{100}	Time/s $\downarrow$	$\text{AAG}_{100} \uparrow$	$B_{100} \uparrow$	κ_{100}	Time/s $\downarrow$
AM	Greedy	4.156	**4.901**	13.7	14.31	10.581	12.518	7.2	32.47
	ES-sign	3.147	3.874	10.8	14.68	8.817	10.548	6.0	37.19
	Adam-ES	3.734	4.650	13.0	14.37	10.705	13.186	7.6	30.21
	RRA	3.365	4.311	12.0	**7.27**	10.693	**13.367**	7.7	**16.16**
POMO	Greedy	1.042	1.347	44.9	16.18	2.709	3.320	31.8	42.78
	ES-sign	0.782	1.038	34.6	16.32	2.127	2.760	26.5	55.14
	Adam-ES	1.124	**1.541**	51.4	16.36	2.996	**4.051**	38.9	43.27
	RRA	0.603	0.871	29.0	**9.10**	2.139	3.187	30.6	**19.27**
PF	Greedy	0.961	1.203	28.5	14.42	3.323	4.024	30.8	40.46
	ES-sign	0.697	0.930	22.0	14.75	3.501	**7.741**	59.3	50.16
	Adam-ES	0.954	**1.306**	30.9	14.89	4.294	6.909	52.9	38.12
	RRA	0.475	0.715	16.9	**7.45**	3.693	6.170	47.2	**17.70**

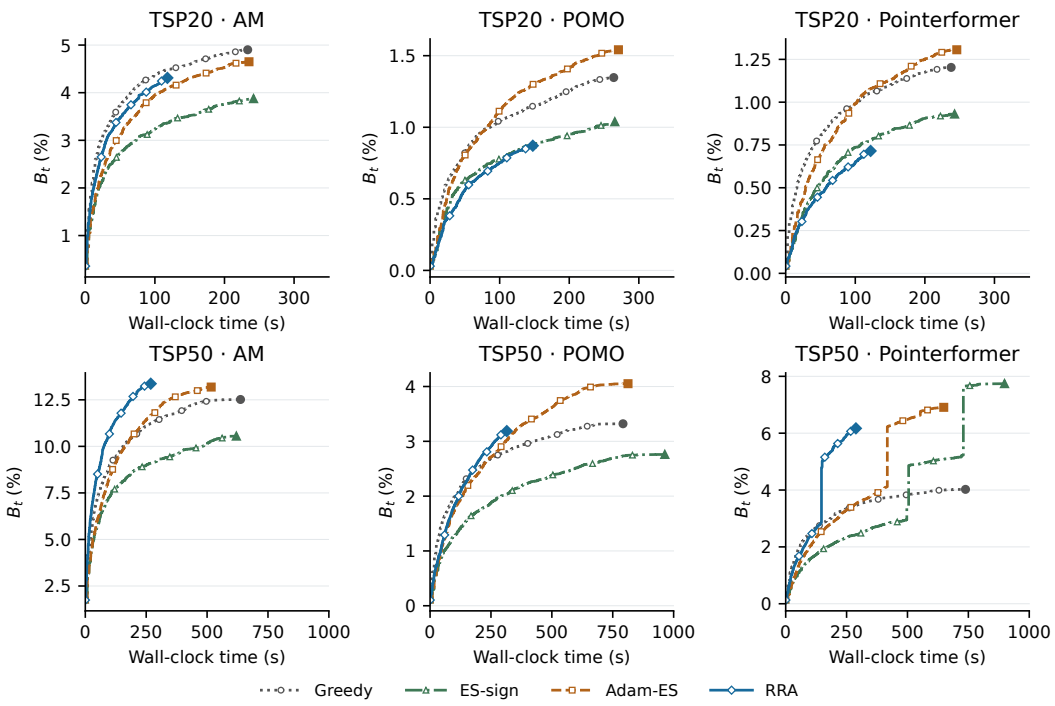


Figure C5: Attack progress on TSP20 and TSP50.

C.6 Gradient-Based Hardness Generation

We compare direct gap optimization with a gradient-based hardness objective. HAC uses input gradients to generate harder training instances (Zhang et al., 2022). We adapt its released greedy-generation surrogate to $\mathcal{A}_\epsilon(X^0)$ from Appendix B.1, denoting this local comparison by HAC-local. Let π_{sel} be the shortest greedy tour across starting nodes and $p_\theta(\pi_{\text{sel}} \mid X)$ its policy probability. With the discrete tour selection held fixed, the surrogate and update are

$$\begin{aligned} J(X) &= C(\pi_{\text{sel}}, X) \log p_\theta(\pi_{\text{sel}} \mid X), \\ X_{t+1} &= \mathcal{P}_{X^0}\left(X_t + \eta_H \frac{\nabla_X J(X_t)}{\|\nabla_X J(X_t)\|_\infty}\right), \qquad \eta_H = 0.01. \end{aligned} \tag{C3}$$

Both cost and log-probability are differentiated. This greedy surrogate differs from HAC's expected-hardness objective. On 16 common POMO instances over 100 updates, HAC-local reaches $B_{100} = 0.3264\%$, versus 5.4419% for RRA and 6.1487% for Adam-ES (Table C11). Direct gap optimization outperforms the tested HAC-local surrogate under this protocol.

Table C11: HAC-local and black-box attacks on POMO.

Attack	Access	AAG_{100} ↑	B_{100} ↑	Operation count
Greedy	black-box	3.3424	3.8670	52,816 LKH
Adam-ES	black-box	4.6915	**6.1487**	52,816 LKH
RRA	black-box	3.8740	5.4419	1,616 LKH
HAC-local	white-box	0.3255	0.3264	1,600 grad

C.7 Adversarial Tours and Transfer

The same original coordinates can lead to different adversarial instances for different models. Figure C7 shows independently attacked instances and their reference and neural tours. AM, POMO,

and Pointerformer start at gaps of 3.84%, 0.58%, and 1.28% on this original instance and reach 20.45%, 5.71%, and 7.08%, respectively. Figure C6 illustrates the larger errors in Pointerformer's tail.

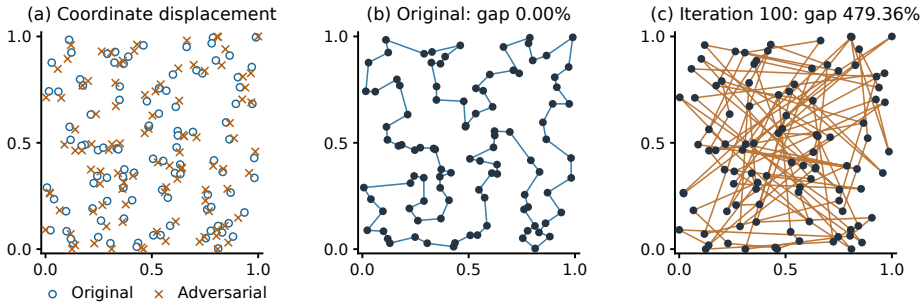


Figure C6: Pointerformer before and after an RRA attack on TSP100.

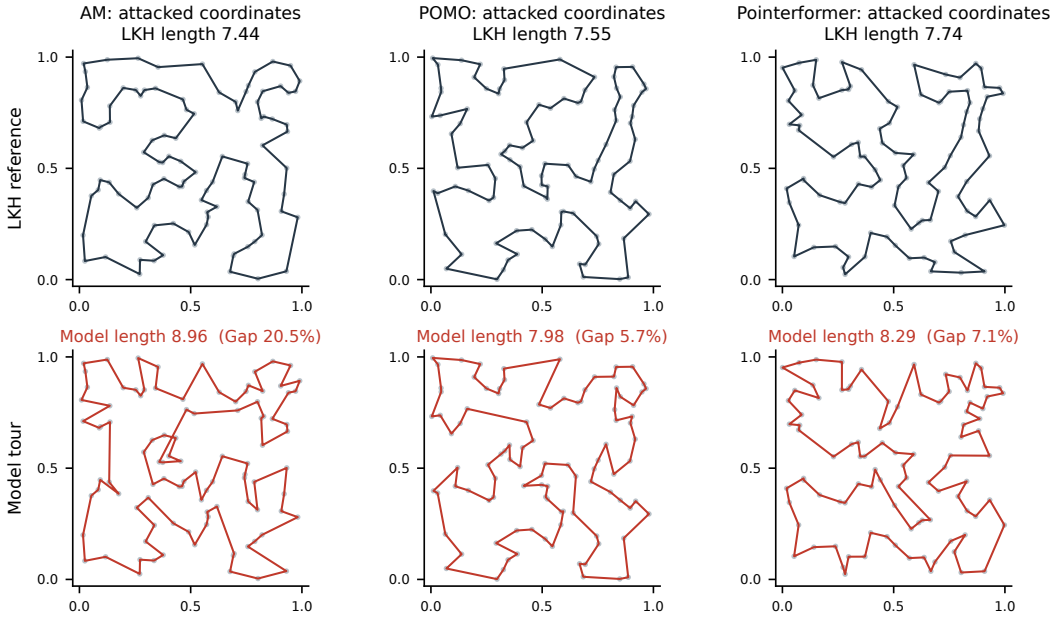


Figure C7: Reference and neural tours after independently attacking three solvers on the same original instance.

The 2-opt comparison in Table 3 uses best-improvement exchanges on the complete datasets specified in Section 5.5. Mean and median gaps include every instance; the OOD median pools all 10,240 inputs. Synthetic instances use Euclidean distances and TSPLIB uses official integer edge weights for both exchanges and scoring.

To determine whether an adversarial instance is difficult for solvers other than its target, we transfer RRA's best-found coordinates among the three backbones on 128 common originals (Table C12). Rows identify the attacked source; columns identify the solver evaluated on those fixed coordinates. Each entry is the mean increase over that solver's clean gap, in percentage points. The large diagonal entries and uneven off-diagonal gains show that the attacks are largely model-specific and transfer asymmetrically. Because selection and updates use only AM's feedback, AM-targeted perturbations can worsen AM's tours while improving those of POMO and Pointerformer.

Table C12: Attack transfer: increase over each target's clean gap (percentage points).

Attacked source	AM target	POMO target	Pointerformer target
AM	15.6278	−0.1862	−0.3092
POMO	2.6477	5.0932	3.3270
Pointerformer	2.7113	0.2105	23.4944

Figure C8 overlays original and RRA-perturbed coordinates for POMO on TSP20 and TSP50. At each size, we select the largest adversarial gap among all 128 instances. Their gaps rise from 0.00% to 5.33% and from 0.00% to 6.10%, respectively. Circles and crosses identify the same nodes before and after attack.

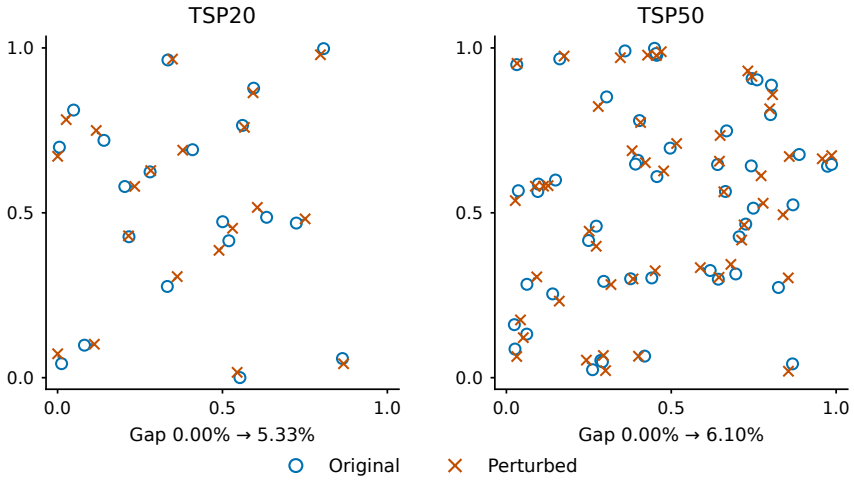


Figure C8: Coordinate displacements in maximum-gap RRA examples on TSP20 and TSP50.

C.8 Distribution of Attack Gaps

To distinguish broad degradation from a few extreme errors, Figure C9 shows the per-instance B_{100} values underlying Table 1: 128 instances per TSP solver and 16 per CVRP solver. Boxes span the interquartile range, with a line at the median; whiskers reach the most extreme observations within 1.5 interquartile ranges. Dots show every instance, and diamonds mark means. Gap axes are logarithmic and scaled separately for each panel. Pointerformer's upper tail separates its mean from its median, whereas POMO's errors are more concentrated.

C.9 Greedy Acceptance Rule

We test whether Greedy's acceptance before feasibility mapping disadvantages it in Table 1. Write $q(Y)$ for the regularized score in Eq. (5). The original rule selects the highest-scoring candidate Y_t and moves to $\mathcal{P}_{X^0}(Y_t)$ if $q(Y_t) > q(X_t)$. The map enforces the constraints in Appendix B.1, but may reduce the score below $q(X_t)$. Our control additionally requires $q(\mathcal{P}_{X^0}(Y_t)) > q(X_t)$; otherwise it retains X_t. In this control, the mapped selected proposal enters the best-found history even when rejected as the next search state; raw candidate probes do not. The rules use identical noise streams on 16 TSP100 originals for 100 iterations, with 52,816 queries to each solver. The extra check needs no new query.

Table C13 reports attacks under both rules and, for the original rule, the frequency and magnitude of these score decreases. The extra check slightly increases POMO's final gap and decreases those of AM and Pointerformer; paired 95% bootstrap intervals for all three B_{100} differences span zero. Both Greedy variants remain below Adam-ES and RRA in mean B_{100} on this subset.

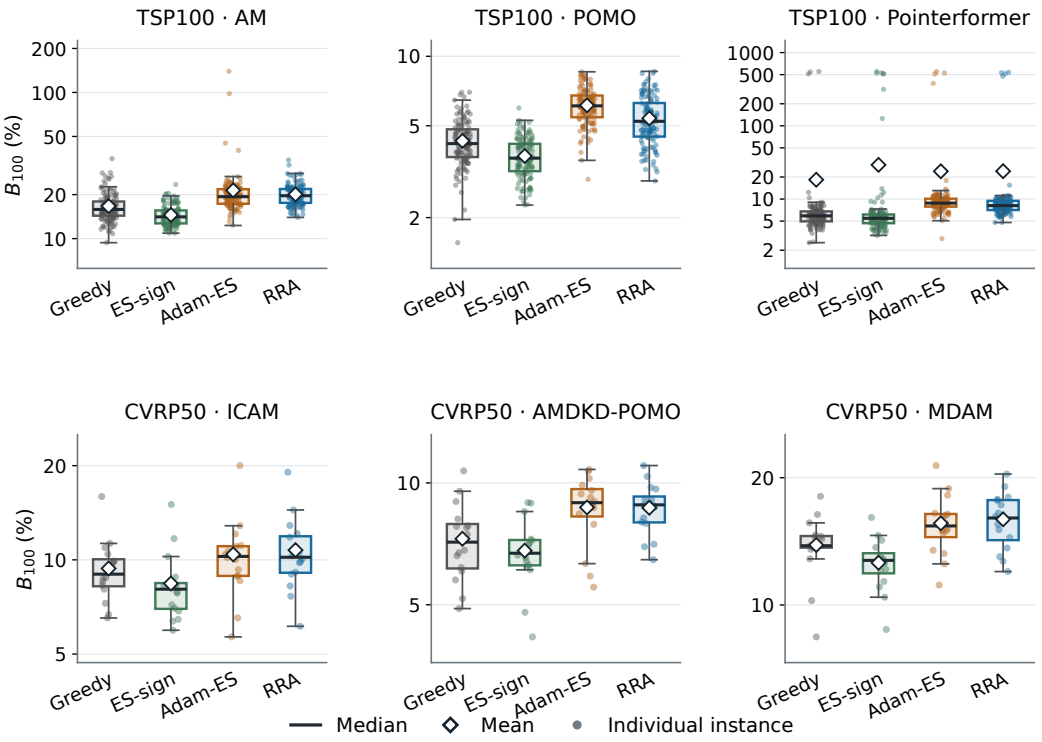


Figure C9: Per-instance best-found gaps after 100 attack iterations.

Table C13: Greedy with and without post-map acceptance on 16 TSP100 instances.

	B_{100} (%)		AAG_{100} (%)	
Model	Original	Post-map	Original	Post-map
AM	17.4577	17.0191	14.8922	14.6058
POMO	3.8670	3.9356	3.3424	3.3562
Pointerformer	6.2357	5.9688	5.0925	4.9395

	Original-rule score decreases		Drop (percentage points)	
Model	Count / pre-accepted	Rate (%)	Mean	Maximum
AM	12 / 234	5.13	2.0552	5.8775
POMO	7 / 236	2.97	1.3528	5.3199
Pointerformer	40 / 330	12.12	0.5408	3.5042

C.10 Update Rules with a Shared Route Cache

To separate coordinate-update gains from reference reuse, we compare Greedy, ES-sign, and Adam-ES with fresh candidate references and with the same cache of 32 recent unique tours. All six configurations use the 16 TSP100 originals from Table 2, 100 iterations, 32 candidates, and the settings in Table B1. Cached Greedy compares candidates and the current instance using the same cache and retains its pre-map acceptance rule. Every configuration solves a fresh reference after each update to record the gap in Eq. (3).

Table C14 reports gaps in percent and block wall-clock time divided by 16, excluding final verification. Bold marks the highest gaps and lowest time within each model and reference scheme. Each attack uses 52,816 neural queries; reuse reduces search-time reference solves from 52,816 to 1,616. Adam-ES with the cache is RRA.

Table C14: Update rules with fresh and reused candidate references on TSP100.

		Fresh references			Route cache		
Model	Update	B_{100}	AAG_{100}	Time/s	B_{100}	AAG_{100}	Time/s
AM	Greedy	17.4577	14.8922	93.09	15.5529	13.6242	22.70
	ES-sign	14.5547	12.4991	105.38	14.2536	12.6740	**21.75**
	Adam-ES	**19.2338**	**16.0369**	**82.98**	**20.2442**	**16.9881**	33.60
POMO	Greedy	3.8670	3.3424	113.51	3.0680	2.6026	**25.49**
	ES-sign	4.0416	3.1575	150.88	2.9952	2.3039	28.67
	Adam-ES	**6.1487**	**4.6915**	**92.99**	**5.4419**	**3.8740**	38.31
Pointerformer	Greedy	6.2357	5.0925	102.77	3.8671	3.2070	**20.10**
	ES-sign	**70.3630**	**62.0687**	131.70	**68.7241**	**62.4623**	23.09
	Adam-ES	40.3788	30.6232	**89.19**	68.5093	56.4949	35.93

Reuse speeds up every update rule but lowers Greedy's B_{100} on all three solvers. With the same cache, Adam-ES achieves the highest B_{100} and AAG on AM and POMO; ES-sign reaches a similar B_{100} but higher AAG on Pointerformer. The preferred update rule therefore depends on the solver, even under identical reference-reuse settings.

D Adversarial Data for Training

Table 5 tests whether RRA's lower generation cost translates into useful training data. Matching library sizes compares the two attacks' data quality; fixing generation time tests the value of RRA's greater throughput.

D.1 Data Generation

We generate data in blocks of 16 originals within 5,400 seconds, retaining only blocks completed within the budget. Adam-ES completes 48 originals; RRA completes 128. A count-matched RRA library uses the same first 48 originals as Adam-ES. Each attack contributes its best-found coordinates at iterations 20, 40, 60, 80, and 100, giving five training slots per original, with repeated coordinates when the best point does not change. Selection follows a fixed input order. To separate attack-guided selection from perturbation alone, the random-noise control creates five independent perturbations per original, drawn from $\mathcal{U}[-0.03, 0.03]$ and passed through $\mathcal{P}_{X^0}$ (Appendix B.1) without score-based selection. This control uses the same 128 originals, independently of the fresh clean stream used during training. To control for extra training on the original inputs, the unperturbed control fills the same library slots with the corresponding original coordinates, retaining the same augmentation and update schedule.

D.2 Training and Evaluation

We train one model per data condition using seed 42 and 400 Adam updates, with learning rate 5×10^{-5}, weight decay 10^{-6}, and gradient clipping at 10. Each batch contains 45 fresh Uniform instances and 19 instances sampled from the selected library: adversarial examples for attack-generated libraries, and the corresponding unperturbed or random-noise inputs for controls. Original instances and saved iterations are sampled uniformly within the library. The loss is

$$\mathcal{L} = (1-\omega)\frac{1}{|\mathcal{D}_{\text{clean}}|}\sum_{X\in\mathcal{D}_{\text{clean}}}\ell(X) + \omega\frac{1}{|\mathcal{D}_{\text{lib}}|}\sum_{X\in\mathcal{D}_{\text{lib}}}\ell(X), \quad \omega = 0.3, \tag{D1}$$

where $\mathcal{D}_{\text{clean}}$ and $\mathcal{D}_{\text{lib}}$ denote the fresh Uniform and library groups. The POMO policy-gradient loss (Kwon et al., 2020) uses the mean cost across starting nodes as its baseline:

$$\ell(X) = \frac{1}{n}\sum_{i=1}^{n}\big(C(\pi^{(i)}, X) - \bar{C}(X)\big)\log p_\theta(\pi^{(i)} \mid X, i). \tag{D2}$$

Here $\pi^{(i)}$ is a tour sampled from starting node i, $\bar{C}(X) = n^{-1}\sum_i C(\pi^{(i)}, X)$, and $p_\theta(\pi^{(i)} \mid X, i)$ is its policy probability conditional on that starting node. The cost difference is held constant when

differentiating this policy-gradient loss. Library inputs receive random square-symmetry augmentation; fresh Uniform inputs are used as sampled. We evaluate the final weights and independently re-attack each model on the same 128 held-out originals using RRA and the settings in Table B1.

Table D1 gives paired differences for RRA (128), with 95% percentile intervals from 10,000 bootstrap resamples of the 128 test instances. Both B_{100} and AAG decrease relative to the untrained and non-adversarial controls. Differences among the adversarial libraries are small, with intervals spanning zero. The intervals describe variation across test instances for the fixed trained models.

Table D1: Re-attack differences for RRA (128) minus each control (percentage points).

Control	B_{100} difference [95% interval]	AAG_{100} difference [95% interval]
No training	-0.3546 $[-0.5683, -0.1420]$	-0.2967 $[-0.4778, -0.1198]$
Unperturbed (128)	-0.6695 $[-0.9015, -0.4418]$	-0.5770 $[-0.7620, -0.3885]$
Random noise (128)	-0.7403 $[-0.9718, -0.5061]$	-0.5840 $[-0.7757, -0.3938]$
Adam-ES (48)	$+0.0230$ $[-0.1615, 0.2124]$	-0.0375 $[-0.1835, 0.1103]$
RRA (48)	$+0.0048$ $[-0.1639, 0.1751]$	-0.0624 $[-0.2023, 0.0742]$

E Inference Budget and Refinement

The recovery results in Section 5.5 raise two questions: how much can extra inference improve fixed adversarial inputs, and can RRA attack a solver that combines several outputs? We first vary views and refinement budgets, then attack solver ensembles directly.

E.1 Two-View Inference and Candidate Scoring

Inference on fixed inputs. To test recovery through symmetry alone, we use the 128 originals and the best-found coordinates from each backbone's RRA run in Table 1. The two-view ensemble solves (x, y) and $(1 - x, y)$, then retains the shorter tour after evaluation in the original coordinates. With frozen weights, this doubles the number of single-view evaluations: AM produces two greedy tours, while POMO and Pointerformer use all 100 starting nodes in each view.

Table E1 uses batch size 16 and each saved input's reference. Times cover all 128 inputs, excluding loading and warm-up.

Table E1: Recovery with two-view inference.

Model	Views	Clean gap	Adv. gap	Adv. time (s)
AM	1	4.4505	20.0784	1.5655
	2	3.3800	7.0016	3.0446
POMO	1	0.2917	5.3849	1.1626
	2	0.2048	2.3274	1.4415
Pointerformer	1	0.5239	24.0182	0.8680
	2	0.3729	2.5525	0.9103

Two-view inference reduces gaps on attacks found against single-view deployments for all three backbones. The attacked inputs remain harder than clean instances under the same two-view budget.

Candidate scoring. Does scoring several views also strengthen an attack on a single-view solver? We replace the neural cost in Eq. (9) with its mean over two views, each using a uniformly sampled square symmetry and node permutation. The eight symmetries include the identity. A fixed seed assigns transformations to batch slots, reused across iterations. Both variants use cached LKH references and are evaluated on the ordinary, untransformed solver output. AM uses one start per view; POMO and Pointerformer use all starting nodes.

On eight common instances, we compare one- and two-view scoring after 100 iterations, at 26,408 model queries, and at shared time limits of 86.63, 171.39, and 106.43 seconds for AM, POMO, and

Pointerformer. Table E2 distinguishes the scoring views from the single-view output being attacked. Figure E1 shows the corresponding trajectories by iterations, model queries, and elapsed time.

Table E2: One- and two-view candidate scoring.

Model	Scoring views	B_{100}	AAG_{100}	Query-matched B_t	Time-matched B_t
AM	1	21.5738	17.4523	21.5738	13.9845
	2	12.5370	10.6633	10.8921	12.5370
POMO	1	5.9443	4.5550	5.9443	4.3522
	2	5.2804	4.0289	4.1573	5.2804
Pointerformer	1	8.8515	6.8348	8.8515	4.8665
	2	7.3317	5.7923	6.1474	7.3317

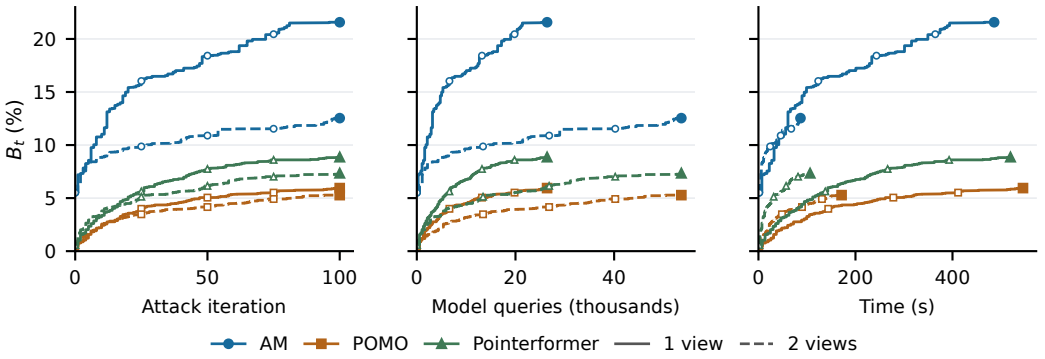


Figure E1: Attack trajectories with one- and two-view candidate scoring.

One-view scoring finds larger gaps at the same query budget on all three models. Its time-matched advantage holds only for AM, reflecting different batching efficiencies. Using multiple views to recover a route and using them to score an attack candidate therefore have different effects.

E.2 Reconstruction on Fixed Adversarial Inputs

To distinguish recovery on known inputs from robustness to a new attack, Table 4 uses two sets of adversarial inputs per solver, found against its base and -10 configurations on the same 16 originals. We then evaluate refinement budgets of 0, 5, and 10 without updating weights. LEHD and SIL reconstruct partial tours; L2C-Insert repairs tours through insertion. No classical local search is added. Budget 0 means construction only.

Table 4 reports mean percentage gaps and synchronized inference time on all 16 -10-targeted inputs, excluding loading and warm-up. For L2C-10, saved per-instance maximizers are reassembled into one fixed batch, so its re-evaluated gap differs from the maxima observed during joint-batch search (Appendix B.3). At ten refinements, LEHD and SIL retain larger gaps on -10-targeted inputs than on base-targeted inputs; refinement alone therefore does not remove the vulnerability of the refined solver.

E.3 Diffusion and Rewrite Budgets

We next test whether recovery also occurs in a diffusion-based solver. FastT2T exposes two inference controls: diffusion steps d and guided rewrites r. We vary both on the same 16 clean and adversarial instances, reporting total time for the batch (Table E3). These inputs come from the one-step, one-rewrite configuration in Table 2.

Table E3: FastT2T under diffusion and rewrite budgets.

d	r	Clean gap (%)	Adv. gap (%)	Adv. time (s)
1	1	0.157	12.748	1.7
1	3	0.036	6.031	6.2
5	1	0.061	3.249	3.5
5	3	0.034	1.812	9.8
50	1	0.011	0.199	26.2
50	3	0.011	0.175	55.4

At one diffusion step, three rewrites reduce the attacked gap from 12.748% to 6.031%. At 50 steps, the same increase in rewrites yields a much smaller change, from 0.199% to 0.175%, while roughly doubling time. As in Section E.2, additional inference computation improves solutions on the fixed coordinates found against the lower-budget configuration.

E.4 Attacking Solver Ensembles

The limited transfer in Table C12 motivates attacking several solvers jointly. For an ensemble of K fixed solvers, let $G_m(X')$ be member m's gap on the same perturbed instance, with a shared reference cost. We use either of two attack objectives:

$$G_{\text{mean}}(X') = \frac{1}{K}\sum_{m=1}^{K} G_m(X'), \qquad G_{\min}(X') = \min_{1\le m\le K} G_m(X'). \tag{E1}$$

The first targets average member performance; the second targets a solver that returns the shortest tour among all members. RRA substitutes the chosen objective into its candidate score (Eqs. (5) and (9)), using one shared reference cache and the same geometric penalty. We attack four ensembles on the same 16 TSP100 originals for 100 iterations with 32 candidates and $\epsilon = 0.03$. Decoding follows Appendix B.3, without eightfold square-symmetry augmentation; FastT2T resets sampling seed 12345 per instance in both joint and individual attacks; Table 2 uses seed 42, hence its different initial and attacked gaps. PF denotes Pointerformer.

Tables E4 and E5 use the same return rule for both attack objectives: retain the iterate with the highest $G_{\min}$ for each original instance. Thus B_t is the best-found shortest-tour gap through iteration t, and AAG follows Eq. (4). Member gaps in Table E5 come from that same iterate. Each ensemble score averages the per-instance minimum, not the minimum of member averages. The single-model attacks use the same originals, checkpoints, and per-member query budget.

Table E4: RRA attacks on shortest-tour ensembles.

		Mean-gap attack			Min-gap attack		
Ensemble	B_0	B_{100}	AAG_{100}	Time	B_{100}	AAG_{100}	Time
AM + POMO + PF	0.232	3.245	2.172	16.6	**5.096**	**3.721**	16.9
AM + POMO + PF + ICAM	0.226	3.841	2.653	22.9	**4.597**	**3.073**	22.4
POMO + FastT2T	0.045	1.002	**0.659**	89.1	**1.149**	0.604	88.1
AM + POMO + LEHD-10	0.037	1.212	0.795	182.7	**2.702**	**1.738**	180.7

Gaps and AAG are percentages; time is minutes for the full 16-instance attack. Bold marks the stronger attack in each gap column. Each run uses 1,616 fresh reference solves and 52,816 queries per member; ensemble size therefore changes total model cost.

Individual attacks maximize each member's own gap on separate inputs. Bold rows report the shortest-tour ensemble evaluated on the jointly attacked inputs.

Min-gap attacks yield larger final ensemble gaps than mean-gap attacks at similar runtime in all four combinations, although their AAG is slightly lower for POMO + FastT2T. Thus, even an ensemble that selects the shortest member tour remains vulnerable to joint local attacks.

Table E5: Member gaps under individual and joint attacks (%).

Ensemble	Member	Clean	Individual	Joint attack	
				Mean-gap	Min-gap
AM + POMO + PF	AM	4.825	20.244	13.090	7.401
	POMO	0.240	5.442	3.286	5.185
	PF	0.596	68.509	3.542	5.329
	Best-of-members	**0.232**	–	**3.245**	**5.096**
AM + POMO + PF + ICAM	AM	4.825	20.244	11.747	7.458
	POMO	0.240	5.442	3.870	4.656
	PF	0.596	68.509	4.458	4.806
	ICAM	0.968	11.542	4.444	5.094
	Best-of-members	**0.226**	–	**3.841**	**4.597**
POMO + FastT2T	POMO	0.240	5.442	1.708	1.236
	FastT2T	0.333	11.971	3.617	3.828
	Best-of-members	**0.045**	–	**1.002**	**1.149**
AM + POMO + LEHD-10	AM	4.825	20.244	11.425	5.824
	POMO	0.240	5.442	1.327	2.870
	LEHD-10	0.102	4.526	1.625	2.795
	Best-of-members	**0.037**	–	**1.212**	**2.702**

F Extension to Capacitated Vehicle Routing

F.1 Coordinate-Only Attacks

CVRP tests whether RRA's route-reuse principle extends beyond TSP when solutions contain multiple capacity-constrained routes. In CVRP, each customer has a demand, and each vehicle starts and ends at a common depot. A solution partitions customers into routes whose total demands do not exceed capacity. We perturb only customer coordinates, keeping the depot, demands, and capacity fixed. The complete vehicle routes therefore remain feasible. RRA caches these route sets, including depot boundaries, and re-evaluates their Euclidean lengths at candidate coordinates. The coordinate update and constraints follow the TSP procedure in Appendix B.1.

We use 16 common CVRP50 instances, with depot and customer coordinates sampled uniformly from $[0, 1]^2$, demands drawn uniformly from $\{1, \ldots, 9\}$, and capacity 40. All four attacks use the defaults in Table B1 with seed 42; the geometric constraints apply to customer coordinates, and the cache holds complete route sets.

We compare four attacks on ICAM, AMDKD-POMO, and MDAM (Zhou et al., 2026a; Bi et al., 2022; Xin et al., 2021). All use greedy decoding without eightfold augmentation; AMDKD-POMO and ICAM use one start per customer; MDAM chooses the shortest output from its five decoders. AMDKD-POMO and MDAM use size-matched weights, whereas ICAM uses released weights trained on sizes 100–500.

HGS supplies heuristic references (Vidal, 2022), using 500 consecutive non-improving iterations, seed 42, and 12 workers for independent solves. Returned routes are scored with unrounded Euclidean distances. The clean and saved best-found inputs are also evaluated with a 2,000-iteration HGS budget; their reference costs agree. These endpoint checks do not guide the attack.

Attack times include endpoint checks but exclude model loading. Each attack makes 52,816 search-time model queries. RRA reduces search-time HGS calls from 52,816 to 1,616.

F.2 Robustness across Backbones

To assess whether the three-model result extends to other architectures, we also attack AM, Sym-NCO, LEHD, L2C-Insert, DGL, ELG, and L2R (Kool et al., 2019; Kim et al., 2022; Luo et al., 2023; 2025b; Xiao et al., 2025; Gao et al., 2024; Zhou et al., 2026b). AM and L2R use one start; DGL uses 15 fixed starts and ELG uses 50. Sym-NCO uses all customer starts; LEHD and L2C-Insert use greedy decoding without reconstruction.

We also evaluate CaR-POMO, RouteFinder-TE, MVMoE-4E, PolyNet, and INViT-3V (Bi et al., 2026; Berto et al., 2025; Zhou et al., 2024a; Hottung et al., 2025; Fang et al., 2024). CaR-POMO constructs 50 initial solutions and refines the best two for 20 iterations. Refinement actions are greedy. RouteFinder-TE and MVMoE-4E use 50 starts. PolyNet uses 64 fixed strategy codes with greedy decoding; INViT-3V uses its three nested encoder views and one start. No model uses eightfold augmentation or test-time fine-tuning.

Each solver is attacked independently on the same 16 originals under the settings above. Table F1 reports the training size of each checkpoint, original and adversarial gaps, and the complete 16-instance attack time.

Table F1: Gaps found by RRA on 16 CVRP50 instances (%).

Model	Train n	B_0	B_{100}	AAG_{100}	κ_{100}	Time/s
AM	50	6.6675	19.0904	16.4633	2.9	101.92
ICAM	100–500	2.3983	10.7411	8.9189	4.5	106.53
AMDKD-POMO	50	2.0244	8.6828	7.3763	4.3	210.71
MDAM	50	5.3426	15.9456	13.8558	3.0	168.70
Sym-NCO	100	3.9211	17.9116	14.0687	4.6	133.17
LEHD	100	6.8314	20.9646	18.1909	3.1	1321.39
L2C-Insert	100	6.1787	21.1236	18.5020	3.4	485.54
DGL	100	8.5581	19.6600	17.3528	2.3	589.86
ELG	100	3.3617	12.1133	10.3594	3.6	235.34
L2R	100	3.7848	15.6775	13.6588	4.1	393.27
RouteFinder-TE	50	2.8075	10.4500	8.7839	3.7	258.40
MVMoE-4E	50	2.2193	8.7955	7.5626	4.0	172.81
PolyNet	100	4.7668	19.6874	15.6680	4.1	139.02
INViT-3V	100	8.0232	22.4180	19.4160	2.8	798.46
CaR-POMO	50	1.1118	6.1665	4.9342	5.5	1986.69

The upper group uses construction only; CaR-POMO adds neural refinement.

Across the fifteen configurations, RRA raises the mean best-found gap to 2.3–5.5 times its initial value. CaR-POMO has the smallest original and adversarial gaps, yet its gap grows from 1.1118% to 6.1665%. Figure F1 shows the full trajectories, grouped for readability on shared linear axes. The gap grows under each tested deployment, including multi-task, nested-view, insertion-based, and construction–refinement solvers.

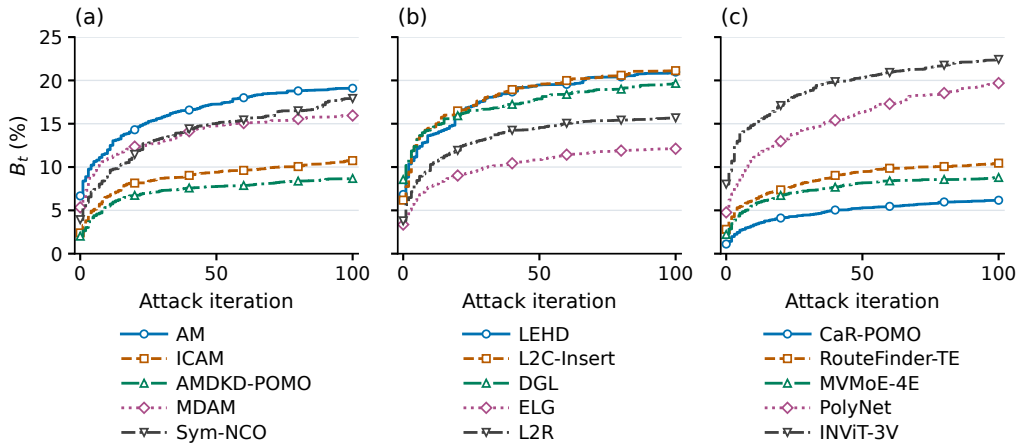


Figure F1: CVRP50 best-found gaps over 100 RRA iterations.

F.3 Coordinates and Vehicle Routes

To relate the measured gap to coordinate and route changes, Figure F2 shows the largest adversarial gap among the 16 AMDKD-POMO instances. The original and adversarial solutions both use six capacity-feasible routes, but the gap rises from 1.78% to 11.04%. The overlay pairs original circles with perturbed crosses, while the route panels show how these small coordinate changes alter visiting order.

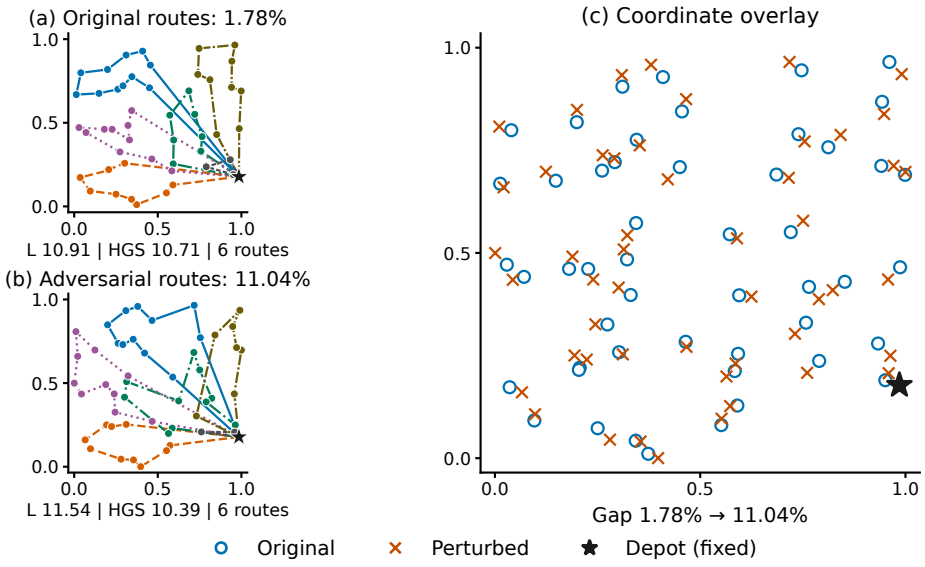


Figure F2: Coordinate changes and vehicle routes under RRA.