DECOMPOSING THE TOUR–MST GAP: AN EDGE-RESOLVED SIGNAL FOR CLASSICAL AND NEURAL TSP SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a topological feedback mechanism for the Travelling Salesman Problem (TSP) by analyzing the divergence between a tour and the minimum spanning tree (MST). Our key contribution is a canonical decomposition theorem which establishes a bijection between (s,t)-tour edges and Minimum Spanning Tree edges. This theorem decomposes the classical tour-MST gap into a sum of non-negative, edge-wise topological divergence gaps derived from RTD-Lite barcodes. These gaps precisely quantify how much each edge deviates from the instance's intrinsic multiscale cluster structure. We conduct experiments and show how the proposed topological signal improves SOTA deep learning methods for TSP: 1) fine-optimizing solutions of heatmap-based neural solvers (e.g., DIFUSCO, ATT-GCN) on large-scale instances (up to 10,000 nodes), achieving up to $14\times$ wall-clock speed-up while producing shorter tours, 2) integration into L2C leads to 0.47%–0.56% shorter tours. Moreover, the proposed topological guidance makes LKH, the SOTA classical method, faster by 24.1% on average across 9 hard instances while preserving overall tour quality. The proposed topological guidance also improves classical heuristics - 2-opt/3-opt. Extensive experiments demonstrate the topology-guided optimization outperforms heuristics across random Euclidean instances, TSPLIB instances, instances with non-trivial topology, non-metric TSP, asymmetric TSP, and Hamiltonian cycle problems.

1 INTRODUCTION

The Travelling Salesman Problem (TSP) remains a cornerstone of combinatorial optimization and a proving ground for machine-learning-augmented solvers. Despite impressive progress from neural construction policies (Vinyals et al., 2015; Kool et al., 2019) and reinforcement-learning (RL) improvements (d O Costa et al., 2020), state-of-the-art pipelines still rely on post-hoc local search, such as 2-opt, to reach competitive tour lengths. These moves examine $O(n^2)$ edge pairs without principled guidance and can dominate run-time on medium-size and large instances.

A major bottleneck in test-time improvement is not the lack of possible moves, but the absence of a principled way to restrict them. Even for a single candidate solution, test-time search exposes a large number of candidate edge exchanges or region modifications, most of which are unproductive. This mirrors a broader challenge in learning-guided search: whether in combinatorial optimization, planning, or reasoning, performance often depends on reducing a large search space of possible next steps to a small set of structurally promising candidates. Existing guidance signals for TSP are either static, such as distance or α-nearness, or learned from data, and therefore may depend on the training distribution. We seek a different kind of signal: one computed from the current solution itself, requiring no training, and reducing the improvement-search space by assigning each edge a structural measure of how much it contributes to the current solution's suboptimality.

The Travelling Salesman Problem is routinely framed as a pure cost-minimization task: almost every heuristic, exact solver, and learning approach measures progress solely by the total tour length. This perspective ignores how cities are connected as the tour evolves, even though decades of geometric and clustering work show that good tours respect the instance's intrinsic connectivity before shaving the residual distance (Zahn, 1971; Held & Karp, 1970). Consequently, contemporary neural solvers

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.

may spend substantial computation on local search to repair long-range structural mistakes that were not explicitly detected during learning (Kool et al., 2019; Xia et al., 2024).

Topology as a missing optimization signal. We argue that explicit topological feedback can bridge this gap. Leveraging recent advances in graph topology, we employ the topology divergence framework (Tulchinskii et al., 2025) to compare a candidate tour with the instance's minimum-spanning tree (MST). Our principal theorem constructs a *canonical bijection* between tour edges and MST edges that decomposes the classical tour-MST length gap into *nonnegative, edge-wise topological divergence gaps*. Each gap quantifies *the difference in scale at which MST clusters are connected by the tour*: large bars reveal tour edges that bridge distant vertex clusters much later than the MST does. By inspecting these bars, we know, for every step of a constructive or improvement algorithm, which edge most distorts the natural connectivity structure.

From length optimization to topology-aware optimization. We turn this insight into a single, training-free edge score $p(e)$ computable in $O(n^2)$ time. The score plugs in unchanged at three points of a modern TSP pipeline: as a move-ordering rule for classical local search (2-opt, 3-opt, Lin–Kernighan); to improve, without retraining, the SOTA learned insertion solver L2C (Luo et al., 2025) and five heatmap-based neural solvers on instances up to $10{,}000$ nodes.

Our experiments on various TSP benchmarks demonstrate that adding this single topological signal delivers the first empirically validated link between persistent-homology theory and large-scale combinatorial optimization. We believe that this topological perspective and the practical gains it brings open a promising research direction beyond length-only objectives.

To the best of our knowledge, this is the first work that (i) provides an *edge-wise* topological characterization of tour sub-optimality and (ii) demonstrates its utility in both classical and learning-based TSP solvers.

Our contributions are summarized as follows:

- **Canonical Tour-MST Decomposition**. We prove a theorem establishing a one-to-one mapping between tour edges and MST edges that decomposes the tour-MST length gap into a sum of nonnegative, edge-specific divergence values (persistent bar lengths). This gives a principled, quantitative measure of how much each tour edge contributes to suboptimality.
- **Improving a frozen SOTA neural solver without retraining.** A single-line modification of the test-time decoding loop of L2C (Luo et al., 2025), the current SOTA learned constructive TSP solver, reduces its optimality gap by 0.47%–0.56% on hard non-uniform instances (paired Wilcoxon $p < 10^{-3}$), with the trained policy and all hyperparameters unchanged.
- **Faster and shorter heatmap-based solvers up to $n{=}10^4$.** Replacing 2-opt with 2-opt+RTDL as the post-processor of five heatmap-based neural solvers (DIFUSCO, ATT-GCN, DIMES, UTSP, SoftDist) yields shorter tours and up to $14\times$ wall-clock speed-up on TSP-10000 (e.g., DIMES: $112.7\text{h} \rightarrow 8.0\text{h}$);
- **Improving LKH.** The proposed topological guidance makes LKH, the SOTA classical method, faster by 24.1% on average across 9 hard instances while preserving overall tour quality.
- **Solver-agnostic gains across classical heuristics and CO problems.** The same penalties improve 2-opt, 3-opt, and Lin–Kernighan on random Euclidean, TSPLIB, non-metric, and non-trivial-topology instances; extend to ATSP and HCP via the standard symmetric reduction ($7\times$ more exact HCP solutions at $n{=}200$).

In summary, our work bridges topology and large-scale combinatorial optimization, providing a novel optimization signal that goes beyond traditional distance-based objectives. We believe this topological perspective opens a promising avenue for developing more structured and efficient search strategies in TSP and related problems.

2 Related Work

Classical TSP heuristics. Local improvement schemes such as 2-opt, 3-opt, and Lin–Kernighan (LK) remain mainstays of high-quality TSP solvers (Croes, 1958; Lin & Kernighan, 1973; Helsgaun, 2000). LK and LKH rely on clever neighbourhood pruning to reduce the search space. The MST and

1-tree bounds have long provided global cost estimates (Held & Karp, 1970). A comparison of first vs. best improvement in 2-opt was studied in Hansen & Mladenović (2006); Aloise et al. (2025).

Neural and RL approaches. Pointer networks (Vinyals et al., 2015), attention models (Kool et al., 2019), curriculum Zhang et al. (2023), population RL (Grinsztajn et al., 2023), and diffusion-based methods (Sanokowski et al., 2024) produce near-optimal tours but delegate the final polishing to the classical local search, which accounts for most wall-time (Xia et al., 2024). Attempts to learn local moves directly (d O Costa et al., 2020) still lack a principled signal for *which* edge to change next.

Topological data analysis (TDA) for graphs. Persistence barcodes capture multiscale connectivity; RTD (Barannikov et al., 2022) and its efficient variant RTD-Lite compare two weighted graphs in $O(n^2)$ time (Tulchinskii et al., 2025). Prior work used RTD as a scalar loss to preserve topology in neural representations; we establish the theorem that turns multiscale tour-vs-MST comparison into an exact, per-edge combinatorial objective, yielding a solver-agnostic edge score that drives both classical local search and learned constructive solvers.

Topology-aware optimization. Graph clustering via MST edge removal dates back to (Zahn, 1971). Recent "heat-map+MCTS" pipelines (Xia et al., 2024; Sun & Yang, 2023; Fu et al., 2021; Qiu et al., 2022; Min et al., 2023; Pan et al., 2026) acknowledge structural cues but still lack a rigorous measure of per-edge mis-alignment. We fill this gap by providing an exact, computable divergence for each edge and showing how it drives both local search and RL.

3 Approach

Consider an undirected weighted complete graph $G = (V, E)$ with weight $w(e)$ for an edge e. Let T_{tour} be a candidate TSP tour. For now, removing the longest tour edge e_{max} consider T_{tour} as a Hamiltonian path T from a start city s to an end city t, sometimes called an (s,t)-tour. Let T_{mst} be a minimum spanning tree (MST) of G. Here and further we use the following notation. For a graph C which is either a spanning tree or a tour: let $L(C)$ denote the sum of edge weights of C. For a graph G let $G^{w\leq a}(G^{w<a})$ denote a subgraph with edges having weights not exceeding (less than) a. For $e \in E(T_{mst})$, let $A, B \in V$ denote the endpoints of e. Define the map $\phi : E(T_{mst}) \to E(T)$ via

$$\phi(e) \coloneqq \underset{\tilde{e}\in E(T)}{\arg\min} \left\{ w(\tilde{e}) \ : \ \exists \text{ path } P \subseteq T^{\leq w(\tilde{e})} \cup T_{\text{mst}}^{<w(e)} \text{ connecting } A \text{ and } B \right\}.$$

Theorem 3.1. *The map ϕ is a one-to-one correspondence between edges of T and T_{mst} so that the standard (TSP tour)-MST gap $L_{(s,t)-tour} - L_{mst}$ is decomposed as the natural sum of edge-wise nonnegative topology divergence gaps:*

$$L_{(s,t)-tour} - L_{mst} = \sum_{e\in E(T_{mst})} w(\phi(e)) - w(e) \tag{1}$$

$$w(\phi(e)) - w(e) \geq 0, \text{ for any } e \in E(T_{mst}). \tag{2}$$

In other words, every MST edge e is paired with a unique tour edge $\phi(e)$ whose weight is greater than or equal to $w(e)$, and these nonnegative weight differences exactly sum up to the gap between the (s,t)-tour length and the MST length.

Proof. For simplicity, assume that all edges in the union $E(T_{mst}) \cup E(T)$ have distinct weights. If some weights are the same, then choose an order on edges with the same weight, the edge penalties do not depend on this choice. Notice that by the MST minimality property, the edge e is the smallest weight edge of the graph G that joins the connected components of A and B in $G^{w<w(e)}$. Therefore, $w(\phi(e)) \geq w(e)$. Conversely, given an edge $\tilde{e} \in E(T)$, let $\tilde{A}, \tilde{B}$, be its endpoints, then $\psi(\tilde{e}) \coloneqq \arg\min_{e'\in E(T_{\text{mst}})} \left\{ w(e') \ : \exists \text{ path } P' \subseteq T^{<w(\tilde{e})} \cup T_{\text{mst}}^{\leq w(e')} \text{ connecting } \tilde{A} \text{ and } \tilde{B} \right\}$. Then the correspondence $\psi : \tilde{e} \mapsto e'$ is the inverse to ϕ, see Appendix C. It follows that $\phi : E(T_{mst}) \to E(T)$ is a one-to-one correspondence with the property (2). □

A step-by-step illustration of the pairing ϕ is provided in Figure 5.

The edge gaps as clustering discrepancies between T and G. The collection of intervals

$$\{(w(e), w(\phi(e))) \,|\, e \in E(T_{mst})\} \tag{3}$$

represents misalignments in multiscale cluster structure of the two weighted graphs T, G in the bottom-up hierarchical clusterization. Here, $w(e)$ is the level at which a pair of clusters C_1, C_2 in G merge into one (i.e., the weight of an edge e from T_{mst}, the minimal spanning tree of G). The right end of the interval $w(\phi(e))$ is the minimal level at which C_1 and C_2 are connected in the graph $T \cup T_{mst}^{w<w(e)}$. Keeping only edges of length not exceeding $\alpha > 0$, the difference in the number of clusters in the tour T and the minimal spanning tree T_{mst}, is always equal to the number of intervals in the set (3) that contain α.

Proposition 3.2. *a) The set of intervals (3) can be identified with the RTDL-Barcode from (Tulchinskii et al., 2025) applied to the weighted graphs (T_{tour}, G); b) the set (3) is stable with respect to variations in weights w; see Appendix D, F.*

Proposition 3.3. $p(e) = 0 \iff e \in E(T_{mst})$. *See Appendix Y.*

Application to Hamiltonian cycles. Let T_{tour} be a Hamiltonian cycle. The hierarchical bottom-up clusterization of the weighted graph T_{tour} is controlled by its MST subgraph $T_{\text{tour}} \setminus \{e_{max}\}$, where e_{max} is the edge of T_{tour} that has the largest weight. Our method incorporates topological information into TSP solvers by computing the penalties

$$p(e) = w(e) - w(\psi(e)) \geq 0 \tag{4}$$

for each $e \in E(T_{\text{tour}}) \setminus \{e_{max}\}$, ψ is the bijection between $E(T_{\text{tour}}) \setminus \{e_{max}\}$ and $E(T_{mst})$ implied by Theorem 3.1. A visualization of the described mapping between MST and tour edges is provided in Figure 1. For each $e \in E(T_{\text{tour}})$ the penalty $p(e)$ is a measure of its "badness", that is, how longer it is than the minimum spanning tree edge. The penalty for e_{max} is not defined by this procedure.

Note on duplicate weights. Any two admissible orderings of equal-weight edges differ by transpositions of consecutive equal-weight pairs, each such transposition is an infinitesimally small weight perturbation, so by stability of the RTDL-Barcode (Proposition 3.2b, Appendix F) the penalties $p(e)$ are preserved. Hence $p(e)$ is uniquely defined regardless of tie-breaking of weights.

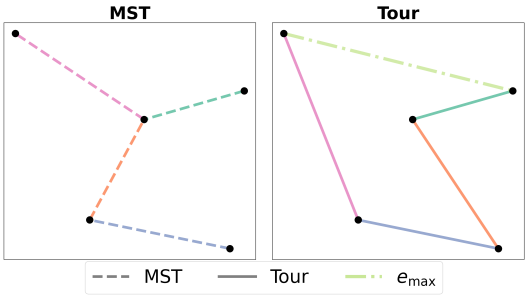


Figure 1: Visualization of the mapping between edges of the Hamiltonian cycle T_{tour} and the minimum spanning tree T_{mst}. Each tour edge (except $e_{\max}$) is bijectively matched to an MST edge.

The following proposition gives a nonnegative topological gap for e_{max} together with a novel lower bound for TSP tour lengths.

Proposition 3.4. *Let $e_{mst}^{max} \in E(T_{mst})$ be the maximum weight edge in the tree T_{mst}. Removing this edge separates the set of all vertices into two clusters C_1 and C_2. Let $\hat{e}$ be the edge that also connects the clusters C_1 and C_2 and is the second-lowest, after e_{mst}^{max}. Then a) $w(e_{max}) \geq w(\hat{e})$; b) $L(T_{tour}) \geq L(T_{mst}) + w(\hat{e})$; see Appendix E.*

We set $p(e_{max}) = w(e_{max}) - w(\hat{e})$. For computational reasons, in most of the experiments in setups with large number of cities, we use an approximation $p(e_{max}) = \min_{e \in \{x \in E(T_{tour}) \,|\, p(x) > 0\}} p(e)$[1].

The α-nearness can be expressed as an RTDL bar. α-nearness is used for edge selection in the LKH-3 algorithm (Helsgaun, 2000) for TSP and VRP. Let G be a weighted graph with weights $w_{i,j}$, $G_1 \subset G$ a subgraph without an arbitrary vertex "1" and incident edges. For an edge (i, j):

$$\alpha(i, j) = L(R^+(i, j)) - L(R), \tag{5}$$

where R is the minimal 1-tree of G, and $R^+(i, j)$ is the the minimal 1-tree of G with a restriction to contain the edge (i, j). We establish connection between α-nearness and RTDL barcode in the following propositions.

Proposition 3.5. *For $i, j \neq 1$, $\alpha(i, j)$ equals to the length of a bar in the RTDL-Barcode$(G_1^{i,j}, G_1)$, corresponding to an edge (i, j). Both G_1, $G_1^{i,j}$ are graphs having all the vertices of G without the vertex 1. G_1 is defined above, and $G_1^{i,j}$ has only one edge (i, j) with a weight $w(i, j)$.*

[1] See Appendix N for an ablation study of penalty for e_{max} and Appendix O for a study of proposed lower bound for tour length.

Table 1: 2D Euclidean TSP. Improvements of heuristic algorithms from RTDL barcodes.

Method	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
2-opt	8.244	1.21	11.465	11.27	13.940	42.87
2-opt+RTDL	8.226	1.07	11.356	9.17	13.798	34.73
2-opt+RTDL+opt(D)	**8.178**	0.72	**11.28**	6.54	**13.620**	26.40
3-opt	8.137	13.61	11.284	179.50	13.747	701.86
3-opt+RTDL	8.046	11.42	11.157	136.50	13.594	616.90
3-opt+RTDL+opt(D)	**8.016**	9.47	**11.119**	106.93	**13.512**	450.48
LK (variable-opt)	8.005	0.223	11.176	0.567	13.672	1.048
LK+RTDL (variable-opt)	**7.963**	0.400	**11.118**	1.144	**13.653**	2.248

The proof is provided in Appendix G. The main difference between our edge penalty $p(e)$ and the α-nearness is that our penalty is tour-dependent. Also, contrary to the α-nearness, our penalties decompose the standard tour length-MST gap into the sum of nonnegative gaps for the tour edges (Thm. 3.1).

Proposition 3.6. *For every edge $e \in T$, we have $p(e) \geq \alpha(e)$. See Appendix G for a proof.*

4 Experiments

4.1 RTDL barcodes and optimal tours

How is the value of an edge's RTDL bar related to the TSP solution? To study this dependency, we generated 100 random 2D Euclidean TSP instances and computed 1) non-optimal tours by running 2-opt procedure with varying number of iterations 2) optimal tours by running Concorde[2]. Then, for each non-optimal tour we ordered edges in an ascending order of their RTDL bars and estimated an empirical probability for an edge to belong to an optimal tour. Figure 2 shows that edges with a low bar length tend to belong to an optimal tour with high probability. See Appendix A for full results with varying number of iterations in non-optimal tours.

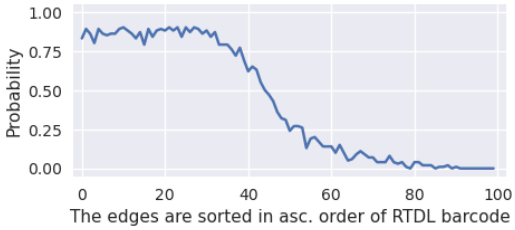


Figure 2: Probability of belonging to an optimal tour vs. RTDL bar (penalty) of an edge.

4.2 Improvements to heuristics for TSP

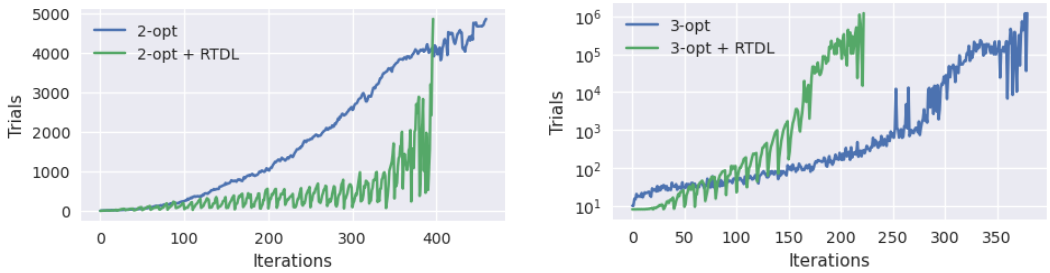
We apply topological guidance to improve 2-opt, 3-opt and LK algorithms. In this simplified setting, we isolate the influence of RTDL-derived edge penalties, avoiding interfering factors such as complex training pipelines. The length of RTDL bar corresponding to a tour's edge defines its "badness". To approximate the optimal tour, a natural heuristic is to prioritize the removal of such edges. In the vanilla 2-opt (Algorithm 1 in Appendix) the edges are checked for removal in the order they are sequentially selected during the route traversal. We modify the 2-opt algorithm by picking edges in a descending order by length of their RTDL bars (Algorithm 2 in Appendix). Similar modifications were done for 3-opt and LK[3]. The modifications are denoted by "+RTDL" below.

Euclidean TSP. Table 1 presents the experimental results for 2D Euclidean TSP (cities are sampled uniformly in a unit square). We report average tour length and average optimization time over 100 trials[4]. For all the settings, an ordering by RTDL bars leads to a shorter tour with a cost of slightly longer time. For the 3-opt algorithm, the ordering by RTDL bars improves both the tour length and the optimization time.

[2] `https://www.math.uwaterloo.ca/tsp/concorde/`

[3] We used a Python implementation `https://github.com/kikocastroneto/lk_heuristic`.

[4] Tables 1–4 report averages over distinct TSP instances, not repeated samples of the same random variable. Statistical significance is established via the Wilcoxon signed-rank test, see Tables 21 and 20.



(a) 2-opt vs. 2-opt+RTDL.

(b) 3-opt vs. 3-opt+RTDL.

Figure 3: Avg. number of trials per iteration.

Results can be further improved by applying weights transformation. Held & Karp (1970) noted that given scalars associated with vertices π_i one can modify the weights $w_{i,j} \leftarrow w_{i,j} + \pi_i + \pi_j$. The modification changes the total lengths of tours by the same constant $2\sum_i \pi_i$. Thus, the optimal tour does not change. The π_i values are optimized by a subgradient optimization procedure pushing the nodes of 1-tree to have degree 2, that is, being more similar to a tour. Obviously, steps of standard 2-opt/3-opt do not change after such modification. However, penalties $p(e)$ and steps of 2-opt/3-opt+RTDL do change. In Table 1 this heuristic is denoted by *opt(D)* and it improves the results even further. All comparisons (heuristic vs. heuristic + RTDL, and their opt(D) variants) yield p-value < 0.01 by Wilcoxon signed-rank test.

We conclude that the ordering by RTDL bars[5] always brings an improvement in objective. Figure 3a shows that for 2-opt ordering by RTDL leads to less number of trials per iteration – number of checked pairs of edges before tour improvement. For 3-opt, ordering by RTDL bars results in faster convergence, see Figure 3b.

Improving LKH. The proposed topological guidance makes LKH, the SOTA classical method, faster by 24.1% on average across 9 hard instances while preserving overall tour quality. See Appendix U.

TSP instances with non-trivial topology. We study the following nodes distribution: clustered, explosion (points from a random circle are removed), implosion (points from a random circle are attracted to it's center). For these instances, the usage of RTDL penalties improves both average length (by $\sim$1%) and optimization time (up to 33%), see Appendix Q.

Non-metric TSP. We have carried out experiments with non-metric TSP (least weight Hamiltonian cycle problem), where edges' weights are sampled from $(0, 1)$. Again, an addition of RTDL penalties always brings a noticeable improvement (13%-25% for 2-opt) in avg. length, see Table 6 in Appendix.

TSPLib instances. For each task, optimization was run 100 times, see Table 5. For certain tasks, the plain 2-opt algorithm struggled to reach the optimal solution and occasionally exceeded its iteration limit, whereas 2-opt+RTDL never encountered such issues under the same initial conditions. See Appendix I for more details. Pooled over all TSPLib instances, 2-opt+RTDL wins on 53.6% of paired runs with Wilcoxon signed-rank test $p = 5.75 \times 10^{-6}$.

4.3 Application to other CO problems

We provide evaluations for other combinatorial optimization problems – ATSP (Asymmetric TSP), HCP (Hamilton Cycle Problem, can be seen as a specific case of ATSP problem). We utilize the fact that non-symmetric TSP can be reduced to symmetric TSP by duplicating the number of vertices and transforming the cost matrix (see Jonker & Volgenant (1983)). For fair comparison, we apply both 2-opt and 2-opt + RTDL to ATSP, reduced to TSP. We utilize the ATSP and HCP problems from the benchmark (Ma et al., 2025). In Table 2 we report the average tour length and the average time. The obtained results confirm that 2-opt + RTDL provides tours with smaller average length. We provide these results as a proof of concept that 2-opt + RTDL can be applied to other combinatorial optimization problems. The current implementation results in computational overhead compared to

[5]Edge ordering by RTDL bars outperforms ordering by distance and α-nearness (TSP-200: 11.356 vs. 11.780 vs. 11.750), see Table 12 in Appendix M. The key reason is tour-dependence: RTDL measures edge "badness" relative to the current tour, unlike static scores.

Table 2: Average length and average running time of 2-opt and 2-opt + RTDL algorithms applied to ATSP, HCP. For each HCP instance, we report the percentage of exact solutions found ("Gap=0") and the percentage of solutions within the 1-gap from the optimal solution ("Gap≤1").

ATSP Type	Optimal Length	2-opt				2-opt + RTDL			
		Length	Time(s)	Gap = 0	Gap ≤ 1	Length	Time(s)	Gap = 0	Gap ≤ 1
Uniform-50	1.55	2.08	0.01	–	–	**1.93**	0.14	–	–
Uniform-100	1.57	2.11	0.06	–	–	**1.98**	0.63	–	–
Uniform-200	1.56	2.16	0.35	–	–	**1.90**	4.01	–	–
Uniform-500	1.57	2.22	3.7	–	–	**2.05**	38.4	–	–
HCP-50	0.0	4.27	0.01	0.32%	3.08%	**2.89**	0.1	2.72%	17.76%
HCP-100	0.0	4.01	0.05	0.36%	4.52%	**2.44**	0.43	4.72%	28.2%
HCP-200	0.0	3.37	0.27	1.0%	4.0%	**1.69**	2.7	7.0%	53.0%
HCP-500	0.0	3.4	2.43	0.0%	5.0%	**1.57**	9.44	13.0%	53.0%

2-opt and we leave its improvement to further work. Remarkably, 2-opt + RTDL compared to 2-opt found the exact solutions in a much bigger number of cases, e.g. 7-fold increase in ATSP HCP-200 and 0 versus 13 in ATSP HCP-500 ("Gap=0" in Table 2).

4.4 Improving the SOTA learned solver L2C without retraining

A single-line modification of the test-time decoding loop of L2C (Luo et al., 2025), the current state-of-the-art learned constructive TSP solver, reduces its optimality gap by 0.47–0.56 percentage points on hard non-uniform instances, with paired Wilcoxon $p < 10^{-3}$ at n=500. L2C constructs a tour by repeatedly sampling a reconstruction centre c, removing its k-nearest tour vertices, and re-inserting them with the learned policy. We replace uniform sampling with RTDL-weighted sampling: we score each candidate centre by aggregating $p(e)$ over tour edges incident to its k-NN cluster, and sample proportionally to the softmaxed scores. No retraining is performed; all other L2C hyperparameters take the authors' best values.

We evaluate on *explosion* instances, whose holey, non-uniform density is the regime where distance-only reasoning is weakest (Table 22). Improving a frozen SOTA model with a training-free, $O(n^2)$ structural signal, without retraining and without altering its trained policy provides evidence that $p(e)$ carries information complementary to the strongest publicly released learned solver. A second, independently designed integration, banning high-$p(e)$ edges and regenerating with INViT (Fang et al., 2024), yields qualitatively the same conclusion (Appendix K).

Table 3: RTDL-guided inference for L2C on *explosion* TSP instances. Δ is L2C gap minus L2C+RTDL gap (positive = improvement). p-values from Wilcoxon signed-rank test.

n	L2C gap (%)	L2C+RTDL gap (%)	Δ / win-rate	p
500	3.364	**2.805**	0.559 / 70%	7.2×10^{-4}
2000	6.891	**6.420**	0.471 / 60%	2.8×10^{-2}

4.5 Improving heatmap-based neural solvers

There is a separate line of neural solvers that generate a heatmap instead of directly predicting the optimal tour. Each entry $\Phi_{i,j}$ of the $n \times n$ heatmap Φ reflects the relevance of the edge (i, j) to the optimal solution. In order to obtain a feasible solution given a heatmap, specialized decoding strategies are used (Sun & Yang, 2023). We follow Sun & Yang (2023) and further improve the initial tour obtained through greedy decoding of generated heatmaps with the 2-opt and 2-opt + RTDL algorithms. With this experiment, we aim to verify whether RTDL-based edge traversal further improves 2-opt when initial tour is already a good approximation of the optimal solution.

Following the work Xia et al. (2024), we use the heatmaps learned by the DIFUSCO (Sun & Yang, 2023), ATT-GCN Fu et al. (2021), DIMES (Qiu et al., 2022), UTSP (Min et al., 2023), and SoftDist (Xia et al., 2024) approaches. We use 128 problems of size 500 and 1000 and 16 problems of size 10000. As shown in Table 4, in most cases, the proposed 2-opt + RTDL leads to solutions with a

Table 4: Average length and total running time of 2-opt and 2-opt + RTDL algorithms applied to initial tours obtained through heatmap greedy decoding. For heatmap-based methods, we separately report time corresponding to heatmap generation and greedy decoding and 2-opt (+ RTDL) runtimes. LKH-3 faces difficulties with instances having non-trivial topology, see Appendix U.

Method	TSP-500		TSP-1000		TSP-10000	
	Length	Time	Length	Time	Length	Time
Baselines						
Concorde	16.55	37.66m	23.12	6.65h	N/A	N/A
LKH-3	16.55	46.28m	23.12	2.57h	71.77	8.8h
DIFUSCO						
2-opt	16.98	5.78m + 0.8m	24.01	21.98m + 12.33m	75.87	3.21h + 31.23h
2-opt + RTDL	**16.88**	5.78m + 0.5m	**23.60**	21.98m + 3.75m	**74.28**	3.21h + 3.85h
ATT-GCN						
2-opt	17.63	3.2m + 2.33m	24.72	10.5m + 26.75m	77.36	2.86h + 78.08h
2-opt + RTDL	**17.3**	3.2m + 1.18m	**24.2**	10.5m + 6.5m	**75.54**	2.86h + 6.97h
DIMES						
2-opt	17.74	4.04m + 3.55m	24.84	12.88m + 34.25m	77.68	2.97h + 112.73h
2-opt + RTDL	**17.35**	4.04m + 1.55m	**24.31**	12.88m + 7.6m	**75.3**	2.97h + 7.99h
UTSP						
2-opt	17.54	3.62m + 1.98m	24.57	12.37m + 21.75m	–	–
2-opt + RTDL	**17.25**	3.62m + 1.2m	**24.12**	12.37m + 6.28m	–	–
SoftDist						
2-opt	17.41	2.18m + 1.6m	24.31	6.00m + 13.72m	75.41	1.61h + 23.4h
2-opt + RTDL	**17.24**	2.18m + 1.05m	**24.09**	6.00m + 5.65m	**74.65**	1.61h + 6.89h

shorter average tour length and a more efficient computation. Heatmap-based methods are slightly worse than LKH-3 and Concorde which reflects the current performance of this family of methods. Figure 4 further confirms that 2-opt + RTDL provides faster convergence for all models. For an additional examination, we also provide the average length of the initial tours through heatmaps' greedy decoding in Table 7. For visual analysis, we provide the initial tours along with 2-opt and 2-opt + RTDL optimized tours for the same TSP-500 problem and all models in Figure 7. We find that RTDL updates every 5 (5, 100) iteration and $N = 500$ ($N = 1000$, $N = 10000$) for TSP-500 (TSP-1000, TSP-10000) works universally well for all models.

Sensitivity analysis. The proposed 2-opt+RTDL algorithm has 2 hyperparameters: RTDL evaluation frequency and update granularity (N). The sensitivity analysis is presented in Figure 8 in Appendix. We can note that 2-opt+RTDL with Freq=1 (i.e., update on each iteration) typically leads to higher computation time, for all values of granularity. In the majority of cases, the average tour length for Freq=1 can be matched or even improved with Freq=5, 10 with appropriate granularity with sufficiently smaller computation time. Overall, for Freq>1, more frequent updates lead to smaller average tour length. In general, the larger the granularity, the slower the computation, although this effect is less pronounced for higher frequencies. Up to some threshold frequency, larger granularity leads to higher average tour length. Above the threshold frequency, larger granularity leads to a decrease in the average tour length. This threshold freq. depends on the model and the problem size.

Statistical significance. Table 19 reports Wilcoxon signed-rank test results for the paired comparison 2-opt vs. 2-opt+RTDL across all heatmap models. In every case the improvement is statistically significant ($p < 10^{-8}$).

MCTS for non-trivial topology. In section S, we demonstrate that RTDL can sufficiently improve heatmap-guided MCTS when TSP has non-trivial topology with clear cluster structure with no additional computational cost.

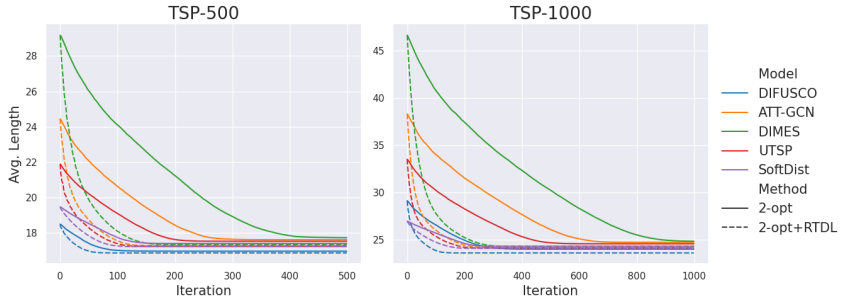


Figure 4: Evolution of average length during tour optimization for heatmap-based neural solvers: 2-opt and 2-opt + RTDL.

Table 5: Experimental results of 2-opt, 2-opt+RTDL, and 2-opt+RTDL-Full compared against Concorde on TSPlib tasks. The GAP (%) is relative to the best solution obtained by Concorde. See details of the algorithms in Appendix G.

Problem	2-opt			2-opt+RTDL-Full			2-opt+RTDL		
	Length	Time	Gap	Length	Time	Gap	Length	Time	Gap
ulysses16	74.54 ± 0.48	0.02s	0.6%	74.51 ± 0.41	0.06s	0.54%	**74.49 ± 0.46**	0.06s	**0.52%**
ulysses22	76.28 ± 0.43	0.05s	0.8%	**76.18 ± 0.37**	0.11s	**0.64%**	76.25 ± 0.51	0.13s	0.78%
att48	34842.07 ± 648.93	0.20s	3.9%	**34632.78 ± 608.78**	0.79s	**3.3%**	34848.44 ± 609.44	1s	4.0%
eil51	451.66 ± 31.26	0.23s	4.9%	450.84 ± 7.54	0.44s	4.9%	**446.68 ± 7.82**	0.58s	**3.9%**
berlin52	8150.23 ± 240.14	0.27s	8.0%	8133.56 ± 217.95	0.53s	7.0%	**7957.48 ± 237.26**	0.69s	**5.5%**
st70	714.90 ± 46.18	0.63s	5.5%	**705.53 ± 10.45**	1.02s	**4.1%**	709.45 ± 11.09	0.62s	4.7%
eil76	604.50 ± 37.62	1.12s	11.0%	**583.92 ± 11.66**	1.01s	**7.2%**	586.20 ± 11.36	1.83s	7.6%
pr76	113227.56 ± 12746.41	0.81s	4.7%	113392.47 ± 2581.35	1.1s	4.8%	**112657.0 ± 2221.49**	1.56s	**4.2%**
gr96	541.12 ± 12.91	1.56s	5.6%	543.32 ± 12.71	1.66s	6.1%	**538.54 ± 12.01**	2.6s	**5.1%**
kroC100	22009.93 ± 553.36	1.95s	6.1%	22020.69 ± 528.93	1.7s	6.1%	**21815.72 ± 598.33**	2.79s	**5.1%**

4.6 Reward shaping for constructive RL

The same penalties also serve as a potential-based reward shaping (Ng et al., 1999) for a constructive DQN: setting $r(e) = -w(e) - \lambda p(e)$ with $\lambda \in \{0, 1\}$ leaves the optimal policy invariant by Theorem 3.1 (since $\sum_{e \in T} p(e) = L_T - L_{\text{mst}}$ is constant once the cycle closes). Empirically, the unshaped baseline plateaus and diverges after $\sim$300 episodes, while DQN+RTDL stays stable and reduces TSP-100 mean tour length from 18.66 to 11.75, full protocol, learning curves and per-size results are in Appendix J (Figure 10, Table 9).

5 Conclusion

In this work, we have introduced a novel topological perspective on the TSP by decomposing the classical tour-MST total gap into edge-wise divergence contributions. We successfully used topological signals to construct topology-guided 2-opt/3-opt/LK heuristics leading to shorter tour lengths for random Euclidean, TSPLib, non-metric, and non-trivial topology instances. Moreover, topology-guided 2-opt provided improvements for fine-optimization of tours generated by heatmap-based neural solvers, up to 10000 nodes. Using this insight, we modified the test-time decoding loop of L2C (Luo et al., 2025), the current SOTA learned constructive TSP solver, that reduced its optimality gap by 0.47%–0.56% on hard non-uniform instances, with the trained policy and all hyperparameters unchanged. The proposed topological guidance made LKH, the SOTA classical method, faster by 24.1% on average across 9 hard instances while preserving overall tour quality. Furthermore, we established an equivalence between the α-nearness from the LKH-3 algorithm and a specific variant of the RTDL barcode, strengthening the link between our topological methods and established heuristic techniques. By bridging persistent homology with deep learning, our work opens a new avenue for incorporating global graph structure into deep reinforcement learning for combinatorial optimization.

AI USE STATEMENT

In this work, we used generative AI tools for polish writing, proof-reading and implementing some methods (RTDL speedup, topological guidance of LKH-3) and have not used generative AI for the rest of tasks. LLM-generated code was verified by running tests and a manual check. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We release our source code for reproducibility.

REFERENCES

Daniel Aloise, Robin Moine, Celso C Ribeiro, and Jonathan Jalbert. First-improvement or best-improvement? an in-depth local search computational study to elucidate a dominance claim. *European Journal of Operational Research*, 2025.

Anonymous. Comparative connectivity barcodes in near-linear time. Manuscript under review, 2026.

Serguei Barannikov, Ilya Trofimov, Nikita Balabin, and Evgeny Burnaev. Representation topology divergence: A method for comparing neural network representations. *International Conference on Machine Learning, ICML*, 2022.

George A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 1958.

Paulo R d O Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian conference on machine learning*, pp. 465–480. PMLR, 2020.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. Invit: A generalizable routing problem solver with invariant nested view transformer. In *International Conference on Machine Learning*, pp. 12973–12992. PMLR, 2024.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7474–7482, 2021.

David Gale. Optimal assignments in an ordered set: an application of matroid theory. *Journal of Combinatorial Theory*, 4(2):176–180, 1968.

Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Tom Barrett. Winner takes it all: Training performant rl populations for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:48485–48509, 2023.

Pierre Hansen and Nenad Mladenović. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5):802–817, 2006.

Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations research*, 18(6):1138–1162, 1970.

Keld Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

Roy Jonker and Ton Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2:161–163, 1983. URL https://api.semanticscholar.org/CorpusID:123004903.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://github.com/wouterkool/attention-learn-to-route.

Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 1973.

Fu Luo, Xi Lin, Mengyuan Zhong, Fei Liu, Zhenkun Wang, Jianyong Sun, and Qingfu Zhang. Learning to insert for constructive neural vehicle routing solver. *NeurIPS 2025*, 2025.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. Ml4co-bench-101: Benchmark machine learning for classic combinatorial problems on graphs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=ye4ntB1Kzi.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. *Advances in Neural Information Processing Systems*, 36:47264–47278, 2023.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pp. 278–287. Morgan Kaufmann, 1999.

Xuanhao Pan, Chenguang Wang, Chaolong Ying, Ye Xue, and Tianshu Yu. Beyond the heatmap: A rigorous evaluation of component impact in mcts-based tsp solvers. In *International Conference on Learning Representations*, volume 2026, pp. 140561–140589, 2026.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35:25531–25546, 2022.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization (DiffUCO). In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, Vienna, Austria, 2024. PMLR. URL https://proceedings.mlr.press/v238/sanokowski24a.html.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

Ilya Trofimov, Daniil Cherniavskii, Eduard Tulchinskii, Nikita Balabin, Evgeny Burnaev, and Serguei Barannikov. Learning topology-preserving data representations. In *ICLR 2023 International Conference on Learning Representations*, volume 2023, 2023.

Eduard Tulchinskii, Daria Voronkova, Ilya Trofimov, Evgeny Burnaev, and Serguei Barannikov. RTD-Lite: Scalable topological analysis for comparing weighted graphs in learning tasks. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 258 of *Proceedings of Machine Learning Research*. PMLR, 2025. URL https://proceedings.mlr.press/v258/tulchinskii25a.html.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, 2015.

Yifan Xia, Xianliang Yang, Zichuan Liu, Zhihao Liu, Lei Song, and Jiang Bian. Position: Rethinking post-hoc search-based neural approaches for solving large-scale traveling salesman problems. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54178–54190. PMLR / OpenReview.net, 2024.

Charles T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 1971.

Dongxiang Zhang, Ziyang Xiao, Yuan Wang, Mingli Song, and Gang Chen. Neural tsp solver with progressive distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12147–12154, 2023.

A ADDITIONAL RESULTS

Figure 6 shows probabilities for an edge to belong to an optimal tour, edges are taken from 2-opt optimization with varying number of iterations. Table 6 shows results for of 2-opt and 3-opt for non-metric TSP problems.

B LIMITATIONS

RTDL evaluation runs in $\mathcal{O}(n^2)$ time (see also Appendix Z for $\mathcal{O}(n \log n)$ acceleration), matching the distance-matrix cost, but the constant factor adds overhead on small instances. On large instances (TSP-1000 and above), the net effect is typically a wall-clock speedup due to fewer swap attempts (Table 4, Figure 4). The 2-opt+RTDL algorithm introduces two hyperparameters: update frequency and granularity. While the sensitivity analysis (Figure 8) shows robust performance across a range of settings, automatic tuning is an open direction. Our experiments cover symmetric and asymmetric TSP, HCP, and heatmap-based solvers. Extension to richer routing problems (CVRP, VRPTW) are natural next steps.

C DETAILS OF THEOREM 3.1 PROOF

Denote by γ^e the path that connects the edge $e \in E(T_{mst})$ endpoints A and B and consist of smaller than $w(e)$ weight $T_{mst}-$edges and smaller or equal $w(\tilde{e})$ weight $T-$edges, with the smallest weight $w(\tilde{e})$. Denote by $\tilde{\gamma}^{\tilde{e}}$ the path that connects the edge $\tilde{e} \in T$ endpoints $\tilde{A}$ and $\tilde{B}$ and consist of smaller than $w(\tilde{e})$ weight $T-$edges and smaller or equal $w(e')$ weight $T_{mst}-$edges, with the smallest weight $w(e')$. If for such a path $w(e') < w(e)$ then the union $\tilde{\gamma}^{\tilde{e}} \cup \gamma^e \setminus \tilde{e}$ would be a path connecting the vertices A, B while consisting of smaller than $w(e)$ weight $T_{mst}-$edges and strictly ***smaller*** than $w(\tilde{e})$ weight $T-$edges, which contradicts the definition of $\tilde{e}$. Hence, the path $e \cup \gamma^e \setminus \tilde{e}$ satisfies the conditions for path $\tilde{\gamma}^{\tilde{e}}$ with $w(e') = w(e)$, and e' coinciding with e. Hence $\psi(\phi(e)) = e$, and the maps ϕ, ψ are bijective.

D THE MAP ϕ AND RTDL-BARCODE

Proof of Proposition 3.2a. The RTDL-Barcode (Tulchinskii et al., 2025) is defined for two complete weighted graphs having the same set of vertices. To apply it to the pair of weighted graphs (T_{tour}, G) we first construct the complete weighted graph T^c_{tour} by setting the edge weights of the absent in T_{tour} edges to $w = +\infty$. Then the edgewise minimum weighted graph of (T^c_{tour}, G) coincides with G and the MST of this edge-wise minimum graph is T_{mst}. The MST of the complete weighted graph T^c_{tour} is $T = T_{tour} \setminus e_{max}$. Then for such two MSTs the construction of the map ϕ from the proof of Theorem 3.1 can be seen as unravelling of Algorithm 1 from loc.cit. for RTDL-Barcode in the case of (T^c_{tour}, G). And hence the collection of intervals (3) coincides the RTDL-Barcode of the pair (T_{tour}, G).

Proposition 3.2a and Section 4 in loc.cit. imply that the collection of intervals (3) is the barcode that describes $\operatorname{Ker} H_0(T^{w<a}_{tour}) \to H_0(T^{w<a}_{mst})$ for all a, comparing the multiscale connected components structures of T_{tour} and T_{mst}.

E PROOF OF THE TOPOLOGICAL GAP NONNEGATIVITY FOR e_{max}

Let $e^{max}_{mst} \in E(T_{mst})$ be the maximum weight edge in the tree T_{mst}. Removing this edge separates the set of all graph vertices into two clusters C_1 and C_2. The edge e^{max}_{mst} is the lowest weight edge in G that connects C_1 and C_2. Then $\phi(e^{max}_{mst}) = e'$, $e' \in E(T_{tour} \setminus e_{max})$, is the lowest weight edge of the tour that connects C_1 and C_2. Let $\hat{e} \in G$ be the edge in G that connects the clusters C_1 and C_2 and is the second-lowest. There is another edge $e'' \in T_{tour}, e'' \neq e'$ connecting C_1 and C_2, since each time the tour goes from the cluster C_1 to C_2 it must then return to C_1. Then $w(e_{max}) \geq w(e'') \geq w(\hat{e})$ and the topological gap $w(e_{max}) - w(\hat{e})$ is always nonnegative.

Since $L(T_{tour}) = L(T) + w(e_{max})$, for any tour $L(T_{tour}) \geq L(T_{mst}) + w(\hat{e})$ is a lower bound.

F STABILITY OF EDGE-WISE GAPS

The stability of RTDL-Barcode (T_{tour}, G)=RTDL-Barcode (T, G) is analogous to the stability of RTD-Barcode(T, G) proven in (Trofimov et al., 2023). Similar to RTD-Barcode, a filtered complex computing the barcode of $\mathrm{Ker}\, H_0(T_{tour}^{w<a}) \to H_0(T_{mst}^{w<a})$ can be written, then the stability of the set of edge penalties follows from the stability of the filtered complex barcode. Alternatively, one can use the stability for barcodes of kernels of maps of filtered complexes.

G RELATION OF RTDL BARS AND α-NEARNESS

Let G be a weighted graph with weights $w_{i,j}$. Consider a subgraph $G_1 \subset G$ without an arbitrary vertex "1" and incident edges.

Definition. The 1-tree is a spanning tree of G_1 with the vertex "1" and two edges connecting the vertex "1" and G_1.

α-nearness is used for edge selection in the state-of-the-art LKH algorithm Helsgaun (2000) for TSP and VRP problems. α-nearness is a measure of edge's "badness". Edges with low α-nearness tend to belong to the optimal tour with high probability. For an edge (i, j):

$$\alpha(i, j) = L(R^+(i, j)) - L(R),$$

where R is the minimal 1-tree of G, and $R^+(i, j)$ is the the minimal 1-tree of G with a restriction to contain the edge (i, j), $L(\cdot)$ is the total length function. Let $G_1^{i,j}$ be a graph having the same vertices as G_1 and only one edge (i, j) with a weight $w(i, j)$. Consider

$$\text{RTDL-Barcode}(G_1^{i,j}, G_1) = \{(w(e_1), w(i, j)), (w(e_2), +\infty), \ldots, (w(e_n), +\infty)\},$$

where $e_i \in E(G_1)$. Since $G_1^{i,j}$ has only one edge (i, j), this barcode has only one interval which death time is distinct from infinity.

Proposition. For $i, j \neq 1$, $\alpha(i, j) = w(i, j) - w(e_1)$.

Proof. Consider an algorithm for calculating RTDL-Barcode(A, B) (Tulchinskii et al., 2025) and a special case when the graph A has only one edge e_a. We assume that $A = G_1^{i,j}, B = G_1$. A precondition of RTDL evaluation is a bijection between vertices of weighted graphs A, B. The auxiliary graph C contains an edge e_c iff corresponding vertices are connected by an edge in any of graphs A, B. The weights $w(e_c)$ are defined as minimum values of corresponding edges of A, B. When an edge is missing, its weight is considered $+\infty$. Then, the Kruskal's algorithm for finding the MST of the graph C is executed. Due to the Algorithm 1 in (Tulchinskii et al., 2025), $e_a \in E(A)$ is paired with some $e_c \in E(C)$. In the Kruskal's algorithm, the edge e_c connects two connected components C_1, C_2. In the Algorithm 1 in (Tulchinskii et al., 2025), edges from the MST of A are added in an increasing order by a weight, in our special case it's one edge e_a. After adding e_a, a path connecting C_1 and C_2 (any of their vertices) appears. The pair (e_c, e_a) forms a bar in the RTDL-Barcode. Note, that always $w(e_c) \leq w(e_a)$. If $e_a \neq e_c$, then after adding e_a to $C_1 \cup C_2$ a cycle appears. As shown in (Helsgaun, 2000), the 1-tree $R^+(i, j)$ (if $i, j > 1$) can be obtained from the $T_{mst} \cup e_a$ by removing the longest edge in the aforementioned cycle. Obviously, such edge is e_c. Thus, $\alpha(i, j) = w(e_a) - w(e_c) = w(i, j) - w(e_1)$. □

Proposition. For every edge $e \in T$, we have $p(e) \geq \alpha(e)$.

Proof. We assume that all edge weights are distinct. Let $e \in T$ be an edge from a (s, t)-tour. If e is incident to the selected vertex "1", $\alpha(e) = 0$ and $p(e) \geq \alpha(e)$. By the RTDL algorithm, e is paired with an edge $\psi(e) \in T_{\text{mst}}$, and the penalty is defined as $p(e) = w(e) - w(\psi(e))$, where $w(e)$ is the minimal level at which some clusters C_1 and C_2 become connected in $T^{w \leq w(e)} \cup T_{\text{mst}}^{w < w(\psi(e))}$, and $w(\psi(e))$ is the minimal level at which C_1 and C_2 are connected in $G^{w \leq w(\psi(e))}$ (or $T_{\text{mst}}^{w \leq w(\psi(e))}$).

The α-nearness is computed as follows: one adds e to T_{mst}, forming a fundamental cycle if $e \notin T_{\text{mst}}$. Let f_e denote largest weight edge on that cycle excluding e. Then $\alpha(e) = w(e) - w(f_e)$.

We consider three cases:

Case 1: $e = \psi(e)$. Then $p(e) = \alpha(e) = 0$.

Case 2: $e \neq \psi(e)$ and e connects C_1 and C_2. In this case, $C_1 \cup C_2 \cup \{e, \psi(e)\}$ contains a fundamental cycle, so $f_e = \psi(e)$, which yields $p(e) = \alpha(e) > 0$.

Case 3: $e \neq \psi(e)$ and e does not connect C_1 and C_2. Thus, for the edge f_e in the fundamental cycle of $T_{\text{mst}} \cup e$ we have $w(f_e) > w(\psi(e))$ (otherwise we get case 2). Consequently,

$$\alpha(e) = w(e) - w(f_e) < w(e) - w(\psi(e)) = p(e).$$

In all cases, $p(e) \geq \alpha(e)$, which completes the proof. □

H VARIANTS OF 2-OPT ALGORITHM

Algorithm 1 presents the vanilla 2-opt algorithm. Algorithm 2 presents the 2-opt+RTDL algorithm which has 2 hyperparameters: 1) batch_size defining the granularity of updates and 2) RTDL evaluation period defining the frequency of RTDL evaluation. Between evaluations an RTDL barcode is cached. Depending on values of hyperparameters, 2-opt+RTDL was benchmarked in two settings:

- **2-opt+RTDL-Full (w/o batching)**: batch_size = Len(tour);
- **2-opt+RTDL (w/ batching)**: batch_size is 10 by default, except sensitivity study in Sec. 4.5.

Algorithm 1: 2-opt algorithm

```
Initialize tour;
repeat
    START:
    improved ← false;
    foreach e1 in tour do
        foreach e2 in tour do
            tour_new ← tour;
            remove e1, e2 from tour_new and rewire it;
            if Len(tour_new) < Len(tour) then
                tour ← tour_new ;
                improved ← true;
                goto START:
            end
        end
    end
until improved;
return tour
```

I ROBUSTNESS ANALYSIS OF 2-OPT AND RTDL VARIANTS

In this section, we conduct extensive experiments with 2-opt, 2-opt+RTDL, and 2-opt+RTDL-Full to evaluate their ability to converge from different initial tours.

We observe that plain 2-opt sometimes fails to converge to the global optimum from certain initial tours, whereas 2-opt+RTDL consistently avoids such cases. Figure 9 shows the distributions of convergence times and resulting tour lengths for the 76-city Christofides/Eilon instance, while Table 8 summarizes the frequency of extreme cases across a broader set of problems.

In particular, we report two types of failures: (i) runs where the tour length exceeds 10% above the global optimum ("GAP $> 10\%$"), and (ii) runs that exceed the time budget of 20 seconds ("Time limit"). The plain **2-opt** baseline shows the highest rate of such failures, while **2-opt+RTDL-Full**

Algorithm 2: 2-opt+RTDL algorithm

Input: batch_size (default 10), period (default 10).

Initialize `tour`;
$i \leftarrow 0$;
$N \leftarrow \min(\text{batch_size}, |\texttt{tour}|)$
repeat
  START: improved $\leftarrow$ false ;
  if *i % period = 0* **then**
    Evaluate RTDL barcode;
  end
  $E \leftarrow$ edges from `tour`[0:N] sorted by RTDL bar lengths in desc. order;
  foreach e_1 *in E* **do**
    foreach e_2 *in E* **do**
      `tour_new` $\leftarrow$ `tour`;
      remove e_1, e_2 from `tour_new` and rewire it;
      if *Len(`tour_new`) < Len(`tour`)* **then**
        `tour` $\leftarrow$ `tour_new` ;
        improved $\leftarrow$ true ;
        goto START:
      end
    end
  end
  if $N < |tour|$ **then**
    $N \leftarrow \min(N + \text{batch_size}, |\texttt{tour}|)$;
    improved $\leftarrow$ true ;
  end
  $i \leftarrow i + 1$;
until *improved*;
return `tour`

Table 6: Non-metric TSP. Improvements of 2-opt and 3-opt with RTDL barcodes.

Method	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
2-opt	1.871	0.21	2.340	1.60	2.732	4.02
2-opt+RTDL	1.631	2.31	1.866	9.65	2.061	32.90
2-opt+RTDL+opt(D)	**1.581**	4.16	**1.832**	18.53	**2.037**	50.65
3-opt	1.161	28.05	1.234	346.36	1.304	1482.76
3-opt+RTDL	1.156	29.00	1.210	329.49	1.282	1602.79
3-opt+RTDL+opt(D)	**1.144**	22.03	**1.203**	310.40	**1.278**	1327.23

Table 7: Average length of initial tours obtained via heatmap greedy decoding.

Model	TSP-500	TSP-1000	TSP-10000
DIFUSCO	18.51	29.12	113.74
ATT-GCN	24.45	38.29	205.84
DIMES	29.17	46.61	331.73
UTSP	21.89	33.51	–
SoftDist	19.48	26.97	84.20

Table 8: Comparison of failure cases for different TSP solvers on TSPLib instances. For each instance we report the percentage of runs where the constructed tour length exceeds 10% above the global optimum ("Len. $> 10\%$ opt.") and the percentage of runs that failed to converge within the allocated time budget ("Time limit") in 20 seconds.

Name	2-opt		2-opt+RTDL-Full		2-opt+RTDL	
	GAP $> 10\% \downarrow$	Time limit (%) $\downarrow$	GAP $> 10\% \downarrow$	Time limit (%) $\downarrow$	GAP $> 10\% \downarrow$	Time limit (%) $\downarrow$
ulysses16	0.0	0.0	0.0	0.0	0.0	0.0
ulysses22	0.0	0.0	0.0	0.0	0.0	0.0
att48	0.6	0.0	0.7	0.0	**0.4**	0.0
eil51	0.7	0.0	0.9	0.0	**0.5**	0.0
berlin52	27.1	0.0	23.5	0.0	**20.7**	0.0
st70	5.6	4.0	5.9	**0.0**	**4.0**	0.0
eil76	7.7	2.7	9.7	**0.2**	**2.3**	1.5
pr76	1.6	0.0	2.7	0.0	**1.2**	0.0
gr96	4.9	0.0	6.4	0.0	**3.0**	0.0
kroC100	9.2	0.0	11.8	0.0	**7.8**	0.0
rd100	12.25	0.0	19.0	0.0	**9.0**	0.0
kroD100	6.75	0.0	6.75	0.0	**3.75**	0.0
eil101	45.25	39.0	4.75	**1.5**	**7.25**	7.75
lin105	12.25	0.5	19.0	**0.25**	**12.0**	0.5
ch130	9.0	0.0	7.5	0.0	**2.0**	0.0
ch150	11.2	0.0	19.2	0.0	**5.0**	0.0

significantly reduces them. The **2-opt+RTDL** variant achieves the most robust performance overall, with consistently lower failure rates across different TSPLib instances.

J Details of experiments with DQN

J.1 RTDL-derived rewards for Deep Q-Learning

To further evaluate the potential of incorporating topological insights into the Traveling Salesman Problem (TSP), we design a series of experiments based on a popular class of reinforcement learning (RL) methods: Deep Q-Networks (DQN). Our aim is to assess both a baseline learning strategy and a modified approach that takes advantage of additional topological information through reward tuning.

For these experiments, we randomly generated 2D Euclidean TSP instances with different numbers of cities and then asked our models to iteratively construct tours multiple times for each instance independently. Each model is given 600 episodes to build a tour from scratch, and we record their best results.

In the DQN framework, the central idea is to learn a Q-function, $Q(s, a)$, which approximates the expected cumulative reward of taking action a in state s and then following the current policy thereafter. In the context of TSP, the "state" corresponds to a partially constructed tour, while the "action" represents choosing the next city to visit. By updating the Q-function through repeated interactions, the agent gradually learns to prefer actions that are more likely to lead to shorter tours, thereby improving its decision-making process across episodes. Details of the algorithm are provided in Appendix J.

In our setup, the reward is dense: at each step the agent receives a negative edge weight corresponding to the cost of the chosen edge. This ensures that the cumulative reward over an episode is exactly the negative tour length. We incorporate topological information by reward shaping $r(e) = -w(e) - \lambda p(e)$. Following Theorem 3.1, $\sum_{e \in T} p(e) = L(T) - L(T_{mst})$ and our modification shifts total reward by a constant.

All models are trained on identical city configurations with fixed starting and ending points, and are evaluated under equivalent conditions. Additionally, we benchmark the learned solutions against the global optimum computed using the Concorde TSP solver. The experiments, summarized in Figure 10, demonstrate that RTDL-based reward shaping significantly stabilizes and improves the training of DQN agents. The baseline DQN quickly plateaus and even diverges after about 300 episodes, whereas DQN+RTDL remains stable and continues to reduce the tour length. Overall,

DQN+RTDL consistently produces shorter tours and exhibits lower variance across seeds compared to the baseline. The numerical results in Table 9 corroborate these findings across different TSP sizes.

Table 9: Mean tour lengths for different TSP sizes comparing the baseline DQN and the RTDL-enhanced variant.

Method	**TSP-50**	**TSP-70**	**TSP-100**
DQN (base)	8.76	10.31	18.66
DQN + RTDL	**7.65**	**8.46**	**11.75**

J.2 DQN ALGORITHM

In this section, we present the standard Deep Q-Learning (DQN) algorithm 3 applied to the TSP. For all experiments, the Q-function was parameterized by a fully connected multilayer perceptron (MLP) consisting of three hidden layers with 128 units each, ReLU activations, and Layer Normalization.

Algorithm 3: Deep Q-Learning (DQN) for TSP

Initialize replay buffer `D`;
Initialize update frequency `C`;
Initialize Q-network with random weights θ;
Initialize target network with weights $\theta_{\text{target}} \leftarrow \theta$;
Initialize environment with fixed city coordinates `env`;
Initialize `best_tour` and `best_length` $\leftarrow \infty$;

for *episode* $= 1$ **to** M **do**
  $state \leftarrow$ `env.reset()`;
  for $t = 1$ **to** T **do**
    With probability ε select random action a, otherwise $a \leftarrow \arg\max_a Q(state, a; \theta)$;
    `state'`, `reward`, `done` $\leftarrow$ `env.step`(a);
    Store (`state`, `a`, `reward`, `state'`, `done`) in `D`;
    if $|D| \geq$ *batch_size* **then**
      Sample random minibatch of transitions from `D`;
      $y_j \leftarrow \texttt{reward}_j + \gamma \cdot (1 - \texttt{done}_j) \cdot \max_{a'} Q(state'_j, a'; \theta_{\text{target}})$;
      $L(\theta) \leftarrow \frac{1}{B} \sum_j \left(Q(state_j, a_j; \theta) - y_j\right)^2$;
      Update $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta L(\theta)$;
    end
    if $t \bmod C = 0$ **then**
      $\theta_{\text{target}} \leftarrow \theta$;
    end
    if *done* **then**
      if *tour length* $<$ *best_length* **then**
        Update `best_tour` and `best_length`;
      end
      break;
    end
    $state \leftarrow state'$;
  end
end
return *best_tour*

K IMPROVING INViT WITH RTDL

In this section we show how to improve INViT Fang et al. (2024) using RTDL. Sinse we can compute the MST gap for each edge in the tour, we can ban the edges with the biggest gap and regenerate the

Table 10: Result of the experiment of improving INViT with banning edges based on gap score

Method	Avg. gap (%)	Win rate (%)	W-test sign.
Vanilla INViT	4.862	-	-
5 edges banned	4.648	47	2.397e-9
10 edges banned	4.598	50	7.557e-10
20 edges banned	4.566	53	2.386e-10

tour without the edges we banned. We call one such generation a trial. Experiment details: we used the checkpoint provided by the authors of Fang et al. (2024). The problem size is 100 points. The gaps are computed against LKH-3 Helsgaun (2000). For inference we used mixed augmentations with 20 samples per problem (See Fang et al. (2024) for details). The augmentations provided to the original INViT and INViT with our correction is the same. We used fixed starting point 0 and deterministic next hop choice strategy for each problem. The results are presented in Table 10. As we can see, using RTDL we can improve generation and this improvement is statistically significant.

L Improving L2C with RTDL-Guided Inference

In this section we show that RTDL can also be used as an inference-time control signal for a SOTA learned insertion-based TSP solver. We start from the L2C framework of Learning to Construct with Insertion-based Paradigm, where the model is trained to insert vertices into a partial tour. During inference, L2C first produces an initial tour with a greedy rollout and then applies a local reconstruction cycle to improve this solution. We do not change the training procedure or the model itself. Instead, we modify only the inference-time rule used to choose the next region to reconstruct.

At each local reconstruction step, the baseline L2C inference first samples a reconstruction center at random and then removes the n nearest vertices around this center. Our variant keeps the same reconstruction mechanism, but replaces the random choice of the center with an RTDL-guided one. For each candidate center, we build its geometric k-nearest-neighbor cluster and aggregate the RTDL penalties of tour edges incident to vertices from this cluster. Thus, we score candidate centers by a topological property of the tour fragment that would be removed, and then sample the center from the resulting score distribution.

We evaluate this modification on synthetic explosion instances. These instances are obtained by sampling cities uniformly in the plane, then selecting several explosion centers and pushing cities iteratively inside each sampled radius away from its center before renormalizing to the unit square. The resulting benchmarks contain sparse holes and dense surrounding regions, making them a natural testbed for topology-aware guidance. All standard L2C inference hyperparameters are kept at the best values recommended in the original paper.

Results

Table 11 summarizes the comparison of baseline inference and RTDL-guided inference variant for each problem size. The reported quantity $\Delta\%$ is defined as `baseline_gap` $-$ `advanced_gap`, so positive values indicate that the RTDL-guided inference is closer to Concorde than the original baseline. On both 500-city and 2000-city explosion instances, the proposed inference-time heuristic improves the final solution quality and wins on a majority of test instances.

Table 11: RTDL-guided inference for L2C on explosion instances. Reported gaps (%) are averaged over multiple generated instances and computed w.r.t. Concorde. $\Delta\%$ denotes the average improvement over the baseline (positive is better). RTDL consistently reduces the gap and wins on a majority of instances.

size	baseline gap (%)	advanced gap (%)	$\Delta\%$ / win rate
500	3.364	**2.805**	0.559 / 70.0%
2000	6.891	**6.420**	0.471 / 60.0%

Table 12: 2D Euclidean TSP. Ablation study of edge penalties.

Method	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
2-opt+dist	8.474	0.72	11.780	6.32	14.291	26.64
2-opt+alpha	8.471	0.76	11.750	6.70	14.31	29.65
2-opt+RTDL	**8.226**	2.84	**11.356**	18.08	**13.798**	62.30

Table 13: 2D Euclidean TSP. Ablation study of maximum edge's penalty.

Penalty $p(e_{max})$	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
$w(e_{max})$	8.244	2.31	11.395	17.21	13.806	51.24
$\min_{e\in\{x\in E(T_{tour})\mid p(x)>0\}}$	8.243	2.38	**11.355**	17.20	13.797	51.97
$w(e_{max}) - w(e_{\text{next mst}})$	**8.235**	2.91	11.360	19.59	13.806	74.97
$w(e_{max}) - w(e_{\text{next cluster}})$	8.241	2.36	11.360	17.04	**13.785**	66.09

M Ablation study of edge penalties

In ablation study, we compare sorting edges by RTDL bars with sorting by distances and α-nearness in a descending order, see Table 12. Edge ordering by RTDL bars outperforms alternatives.

N Ablation study of a maximum edge's penalty

We carry out an ablation study of a maximum edges' penalty $p(e_{max})$, see Table 13. Results are averaged over 400 trials. In the Table 13 we use the following definitions:

- $e_{\text{next mst}} \in E(G)$ denotes the edge that is next in length after the maximum edge in the MST.
- consider a bottom-up clusterisation of the graph G via the Kruskal's algorithm. At the last step, the maximum edge in the MST connects two clusters into one. $e_{\text{next cluster}} \in E(G)$ denotes the edge that is next in length after the maximum edge in the MST among edges which connects these two clusters.

O Evaluation of a proposed lower bound

We compare the proposed lower bound (MST + $e_{\text{next cluster}}$) to tour's length with the classical lower bound implying minimal 1-tree Helsgaun (2000). We optimize scalars associated with vertices π_i by subgradient optimization (see Section 4) and update weights of edges $w_{i,j} \leftarrow w_{i,j} + \pi_i + \pi_j$. The desirable lower bound is $\sum_{e\in T} w(e) - 2\sum_i \pi_i$, where $T = T_{mst} \cup e_{\text{next cluster}}$ or T a minimal 1-tree. Table 14 present experimental results. The proposed variant MST + $e_{\text{next cluster}}$ is slightly better for the number of cities less than 100. Such a small improvement is because the difference between two approximations is in one edge only, albeit leading to different optimization steps.

Table 14: 2D Euclidean TSP. Lower bounds to TSP tour length.

Tour's approximation	TSP-10	TSP-25	TSP-50	TSP-100	TSP-200
minimal 1-tree	2.8486	4.2050	5.6598	7.7057	10.6091
MST + $e_{\text{next cluster}}$	**2.8494**	**4.2072**	**5.6606**	7.7057	10.6091

P Detailed Construction of ϕ

Figure 5 illustrates the canonical bijection $\phi\colon E(T_{\text{mst}}) \to E(T)$ from the proof of Theorem 3.1, applied to the 5-point Euclidean instance of Figure 1. **(a)** The minimum spanning tree of G. **(b)** The

Table 15: Results of TSP solution with non-trivial instance topology.

Method	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
Clusters						
2-opt	5.532	**3.106**	7.442	29.766	8.815	111.651
2-opt+RTDL	**5.504**	3.112	**7.365**	**24.740**	**8.740**	**78.951**
Explosion						
2-opt	6.838	3.376	9.166	29.738	10.785	116.366
2-opt+RTDL	**6.790**	**2.757**	**9.155**	**22.626**	**10.772**	**77.866**
Implosion						
2-opt	7.650	2.595	10.260	25.222	12.309	104.972
2-opt+RTDL	**7.602**	**2.502**	**10.229**	**22.792**	**12.178**	**81.178**

tour path $T = T_{\text{tour}} \setminus e_{\max}$. Steps 1–4. MST edges are processed in increasing weight order. At each step, the blue dashed edge is the current MST edge e; green dashed edges form the subgraph $T_{\text{mst}}^{w<w(e)}$ of lighter MST edges already in place; faint orange edges are tour edges of weight $\geq w(e)$ that were tried but did not yet connect the endpoints of e; the purple solid edge is $\phi(e)$, the lightest tour edge that achieves connectivity. Steps 1 and 3 yield zero gap because the tour and MST share edges $(2, 3)$ and $(4, 5)$. Steps 2 and 4 produce the nonzero bars $[w(e),\, w(\phi(e)))$ of the RTD-Lite barcode. The sum of all gaps equals $L_T - L_{\text{MST}} \approx 2.62$, verifying the decomposition of Eq. (1).

Q TSP INSTANCES WITH NON-TRIVIAL TOPOLOGY

We study solution of TSP instances with non-trivial topology, besides uniform distribution in unit square, see examples at Figure 12. We used the following algorithms for instance generation from Fang et al. (2024):

- **clusters**: uniformly i.i.d. coordinates of virtual cluster centers Gaussian coordinates of data points w.r.t. virtual cluster centers;
- **explosion**: first generate uniformly i.i.d. coordinates of data points select an explosion center an expel all data points in a range;
- **implosion**: first generate uniformly i.i.d. coordinates of data points select an implosion center and attracts all data points in a range;

2-opt algorithm. Experimental results are shown in Table 15. The usage of RTDL-penalties improves 2-opt both in terms of average length and time, despite of computational overhead due to RTDL evaluation. The speedup comes from doing less attempts of edges' swap.

LK algorithm. Table 16 show empirical comparison of LK vs. LK+RTDL for instances with non-trivial topology. Usage of RTDL-based topological signal always improves tour length at the cost of longer convergence. All improvements are significant with p-value < 0.05 from Wilcoxon signed-rank test.

R ABLATION STUDY OF BATCHING IN 2-OPT

Table 17 presents a study of batching effect in the vanilla 2-opt (Algorithm 1). We conclude that batching is not efficient for vanilla 2-opt and makes sense only for 2-opt+RTDL. Using batching is a default for 2-opt+RTDL.

Table 16: Improvements of LK from RTDL barcodes on instances with non-trivial topology.

Method	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
Explosion						
LK (variable-opt)	6.808	0.525	9.029	0,777	10.865	1.497
LK+RTDL (variable-opt)	**6.783**	1.254	**8.992**	3.681	**10.781**	6.918
Implosion						
LK (variable-opt)	7.457	0.391	10.260	0.776	12.557	1.241
LK+RTDL (variable-opt)	**7.427**	1.119	**10.173**	3.648	**12.457**	7.583
Clusters						
LK (variable-opt)	5.564	0.416	7.519	0.775	9.052	1.267
LK+RTDL (variable-opt)	**5.539**	1.178	**7.478**	3.533	**9.010**	6.922

Table 17: 2D Euclidean TSP. Ablation study of batching in 2-opt.

Method	TSP-100		TSP-200		TSP-300	
	Length	Time(s)	Length	Time(s)	Length	Time(s)
2-opt w/o batching	8.243	1.813	11.465	11.566	13.940	44.306
2-opt w/ batching	8.298	1.658	11.517	17.957	13.992	76.533
2-opt+RTDL w/o batching	8.284	1.473	11.552	7.826	14.071	17.205
2-opt+RTDL w/ batching	8.226	2.84	11.356	18.08	13.798	62.30

S Improving heatmap-guided MCTS for TSP with non-trivial topology

In this section, we estimate the effect of RTDL on the pretrained heatmap-guided neural solvers for TSP with non-trivial topology. For this purpose, we first generate 128 problems of size 500 and 1000 located in $[0; 1]^2$ unit square possessing clear cluster structure (see Figure 13 for an example). We use DIFUSCO (Sun & Yang, 2023) and SoftDist (Xia et al., 2024) as top-performing approaches and apply two post-processing techniques – greedy decoding with 2-opt (+ RTDL) and MCTS (+ RTDL) over generated heatmaps. MCTS for TSP typically includes 2-opt for a local search, and by "MCTS + RTDL" we denote MCTS with RTDL-enhanced 2-opt (i.e., 2-opt + RTDL). In this implementation, we compute RTDL over all vertices on each iteration after tour refinement, thus, introducing no additional hyperparameters. We use pre-trained model checkpoints and MCTS implementation from the corresponding repositories without additional finetuning or hyperparameter search.

Table 18 demonstrates the results. Although being popular post-processing approach, classic MCTS works sufficiently worse for the problems with non-trivial topology: the average tour length is bigger than that of greedy decoding with 2-opt (+ RTDL). We hypothesize this fault is caused by the local structure of 2-opt search restricted to swap edges among the nearest neighbors. In contrast, MCTS with 2-opt + RTDL results in sufficient improvement of the tour length, reducing the gap to the Concorde solver performance. Moreover, while RTDL provides consistent improvement for both 2-opt and MCTS, it does not lead to computational overhead.

T Details on statistical significance

See Tables 19, 20, 21.

U Improving LKH

Across nine hard instances with 20 matched seeds each, **LKH+RTDL was faster in all 180/180 paired runs**, while preserving essentially the same tour quality. Mean CPU reductions ranged from 13.36% to 34.72%, with a 21.4% aggregate reduction. The n=5000 experiments use an *optimized*

Table 18: Average length and total running time of 2-opt (+ RTDL) applied to initial tours obtained via greedy decoding and MCTS (+ RTDL) applied to heatmaps for TSP problem with non-trivial topology. For MCTS-based methods, we report separately heatmap generation and tour optimization times. For 2-opt-based methods, we report separately heatmap generation with greedy decoding and tour optimization times.

Method	TSP-500		TSP-1000	
	Length	Time(m)	Length	Time(m)
Baseline				
Concorde	9.02	22.56	11.4	68.18
DIFUSCO				
2-opt	9.65	$7 + 2.28$	12.19	$22.32 + 18.02$
2-opt + RTDL	9.46	$7 + 1.07$	11.96	$22.32 + 9.28$
MCTS	12.34	$4.78 + 22.32$	15.3	$16.25 + 44.08$
MCTS + RTDL	9.21	$4.78 + 22.12$	11.73	$16.25 + 45.02$
SoftDist				
2-opt	9.61	$1.33 + 1.4$	12.12	$3.16 + 10.97$
2-opt + RTDL	9.48	$1.33 + 0.93$	11.97	$3.16 + 5.23$
MCTS	13.16	$0.13 + 22.25$	15.51	$0.18 + 44.53$
MCTS + RTDL	9.19	$0.13 + 22.15$	11.78	$0.18 + 44.28$

Table 19: Wilcoxon signed-rank test for heatmap-guided 2-opt vs. 2-opt+RTDL with alternative: 2-opt tour length $>$ 2-opt+RTDL tour length.

Model	TSP-500		TSP-1000	
	Win%	p-value	Win%	p-value
DIFUSCO	70.3	1.1×10^{-8}	98.4	5.4×10^{-23}
ATT-GCN	94.5	3.8×10^{-22}	97.7	5.7×10^{-23}
DIMES	95.3	1.1×10^{-22}	99.2	5.1×10^{-23}
UTSP	89.8	1.1×10^{-20}	98.4	6.0×10^{-23}
SoftDist	79.7	9.9×10^{-16}	87.5	2.4×10^{-19}

$\mathcal{O}(n \log n)$ *barcode-pairing implementation*, improving further net end-to-end speedups to 13.36% and 18.58% on the largest instances while preserving tour quality. The per-instance results are shown in Table 23.

Mean tour quality improves on 6/9 instances; on the remaining three, the differences are only +0.065%, +0.089%, and +0.103%. Averaged across the nine instances, the **mean tour-length change is -0.107%**, i.e. LKH+RTDL is slightly better in mean quality overall. RTDL should not be viewed as a universally stronger replacement for α-nearness. Rather, it provides complementary, tour-dependent guidance: on these hard n=1000–5000 instances, **adding RTDL makes LKH substantially more efficient while preserving tour quality**. Hard TSP instances are show in Fig. 14.

U.1 RTDL PENALTY AS A REPLACEMENT OF α-NEARNESS

To address this, we embedded a new edge-ranking into LKH and compared it head-to-head against unmodified LKH under identical candidate sets and seeds, differing only in the deletion-stage prior: LKH's native α-nearness (keep an edge if it is in the best tour or the minimum 1-tree) versus our gap prior (keep an edge if it deviates little from the minimum spanning tree). On standard TSPLIB instances the two are statistically indistinguishable, so the gap prior is competitive but not a wholesale improvement. Its value emerges on clustered topology, where α-nearness is structurally blind: we built an instance of 18 points in 3 tight clusters (see coordinates below) whose exact optimum (3196) requires a long inter-cluster bridge that is the 13th-nearest neighbor of its endpoints, so any α/nearest-neighbor ranking excludes it. With such a restricted candidate set and a single restart, stock LKH reaches the optimum in only 9 of 15 seeds (trapped at 3201), because α-nearness protects that

Table 20: Wilcoxon signed-rank p-values (alternative: tour length without RTDL is greater). All tests are paired across instances.

Setting	Win-rate	p-value
LK+RTDL vs. LK (4 topologies, $n \leq 300$)	54.2%	$\leq 2.5\times10^{-2}$
2-opt+RTDL vs. 2-opt (TSPLib pooled)	53.6%	5.8×10^{-6}
2-opt+RTDL vs. 2-opt (heatmap, TSP-500)	70.3–95.3%	$\leq 1.1\times10^{-8}$
2-opt+RTDL vs. 2-opt (heatmap, TSP-1000)	87.5–99.2%	$\leq 6.0\times10^{-19}$
L2C+RTDL vs. L2C (TSP-500, explosion)	70%	7.2×10^{-4}
L2C+RTDL vs. L2C (TSP-2000, explosion)	60%	2.8×10^{-2}

Table 21: Wilcoxon signed rank test p-values for random Euclidean TSP instances.

Method	TSP-100	TSP-200	TSP-300
2-opt+RTDL vs. 2-opt	2.13×10^{-06}	1.10×10^{-05}	7.28×10^{-07}
2-opt+RTDL+opt(D) vs. 2-opt	1.92×10^{-03}	4.93×10^{-08}	4.98×10^{-15}
3-opt+RTDL vs. 3-opt	3.57×10^{-05}	2.29×10^{-08}	2.32×10^{-10}
3-opt+RTDL+opt(D) vs. 3-opt	3.52×10^{-09}	2.49×10^{-10}	3.46×10^{-12}

long bridge; the gap-biased version reaches it in 12 of 15 (26 of 30 at larger sample) by leaving the deviant bridge breakable. This confirms the gap prior is a genuinely new structural signal, not a re-derivation of α-nearness.

Cluster A — centre $(-550, 0)$:

$$\begin{array}{lll} 1\text{: } (-560,-5) & 2\text{: } (-550,-5) & 3\text{: } (-540,-5) \\ 4\text{: } (-560,5) & 5\text{: } (-550,5) & 6\text{: } (-540,5) \end{array}$$

Cluster B — centre $(0, 835.16)$:

$$\begin{array}{lll} 7\text{: } (-10,830.16) & 8\text{: } (0,830.16) & 9\text{: } (10,830.16) \\ 10\text{: } (-10,840.16) & 11\text{: } (0,840.16) & 12\text{: } (10,840.16) \end{array}$$

Cluster C — centre $(550, 0)$:

$$\begin{array}{lll} 13\text{: } (540,-5) & 14\text{: } (550,-5) & 15\text{: } (560,-5) \\ 16\text{: } (540,5) & 17\text{: } (550,5) & 18\text{: } (560,5) \end{array}$$

V On extension to CVRP

The MST reference stays meaningful wherever the objective is dominated by spatial connectivity, and degrades as feasibility rather than distance becomes binding. Concretely:

CVRP. A solution is a set of routes sharing a depot; the natural reference is a minimum spanning forest on each route's customer set plus the depot, giving a per-route barcode, together with a global MST barcode that scores inter-route edges. Capacity does not distort the geometry, so we expect the signal to transfer, with the caveat that a high-p edge may be structurally forced by a capacity split; the score must therefore be used as a move-ordering prior filtered by a feasibility check, not as a removal rule.

VRPTW / pickup-delivery. Here the effective cost is no longer a metric on positions — two spatially adjacent customers can be temporally incompatible — so an MST on Euclidean distance is the wrong reference. The principled fix is to build the reference tree on a feasibility-aware weight (e.g. w_{ij} + waiting/violation penalty), which keeps the decomposition valid because Theorem 3.1 requires only a weighted graph, not a metric — our non-metric TSP experiments (Table 7) are evidence that the machinery is not metric-dependent. Precedence constraints are the hardest case: the tour edge set is no longer symmetric-reducible without doubling, and we flag this as the main open obstacle.

Table 22: RTDL-guided inference for L2C on non-uniform and uniform TSP instances. Δ is L2C gap minus L2C+RTDL gap (positive = improvement). p-values from Wilcoxon signed-rank test (one-sided).

layout	n	L2C gap (%)	L2C+RTDL gap (%)	Δ / win-rate	p
explosion	500	3.364	**2.805**	0.559 / 70%	7.2×10^{-4}
implosion	500	4.854	**4.190**	0.664 / 74%	2.3×10^{-5}
clustered	500	1.773	**1.751**	0.022 / 46%	0.45
uniform	500	1.290	**1.126**	0.164 / 66%	1.4×10^{-2}
explosion	2000	6.891	**6.420**	0.471 / 60%	2.8×10^{-2}
implosion	2000	6.227	**5.955**	0.271 / 73%	5.3×10^{-2}
clustered	2000	7.194	7.594	-0.400 / 27%	0.98
uniform	2000	1.572	**1.522**	0.050 / 47%	0.26

Table 23: Performance of LKH+RTDL.

Instance	**Mean tour-length change**	**Mean CPU reduction**
H1000	-0.00006%	30.01%
H2000a	-0.248%	29.83%
H2000b	-0.236%	15.22%
H2000c	-0.004%	28.82%
H2000d	+0.065%	17.57%
H3000a	+0.089%	34.72%
H3000b	-0.571%	18.80%
H5000a	-0.157%	13.36%
H5000b	+0.103%	18.58%

W LICENSES OF ASSETS

- TSPLib: Non-Commercial Software License;
- `https://github.com/kikocastroneto/lk_heuristic`: MIT License;
- L2C: MIT License, INViT: license not specified, Ml4co-bench-101: license not specified;
- checkpoints of heatmap-based neural solvers Xia et al. (2024): license not specified;

X ON COMPARISON WITH GALE (1968)

Gale (1968) yields sorted-order domination $w_i(T_{\text{mst}}) \le w_i(T)$ based on the matroid theory, hence the existence of some dominating bijection, the rank matching φ_{rank}. It does not produce ϕ: the two maps already differ on the 5-point instance of Fig. 5, where rank matching assigns penalty **0.68** to the tour edge $(4,5) \in E(T_{\text{mst}}) \cap E(T)$ — an edge that cannot contribute to suboptimality — while underscoring the genuinely misplaced edge $(3,5)$ (**0.34** vs. the true **1.01**), see the detailed comparion on the two matchings below. Indeed, a careful reading of the construction of ϕ, ψ shows they obey the law $p(\tilde{e}) = 0 \iff \tilde{e} \in E(T_{\text{mst}})$. Rank and exchange bijections generically violate this law: a signal that penalizes provably correct edges is worse than no signal, which is why no matroid-theoretic bijection could power any experiment in the paper. Moreover, dominating bijections are highly non-unique, whereas ϕ is canonical: it coincides with the persistence pairing of $\text{Ker}(H_0(T_{\text{tour}}^{w<a}) \to H_0(G^{w<a}))$, unique by the structure theorem for persistence barcodes, so $w(\phi(e)) - w(e)$ is exactly the interval of scales over which the two clusters merged by e remain disconnected in the tour. This canonicity is what yields stability and tie-independence (Appendix D), the identification of LKH's α-nearness as a degenerate RTDL bar (Appendix G), and the empirical per-edge signal (Fig. 2, Table 12). None of these follows from matroid theory.

A comparison with rank-order bijection (Gale, 1968) on a specific graph.

We compare our RTDL-derived bijection with Gale (1968). The two maps already differ on the 5-point instance of Fig. 5.

MST edges: $e_1 = (2,3), w = 4.03; e_2 = (3,4), w = 4.13; e_3 = (4,5), w = 4.81; e_4 = (1,3), w = 6.40$; $L_{\text{MST}} \approx 19.37$.

Tour path $T = T_{\text{tour}} \setminus e_{\max}$: $\tilde{e}_1 = (2,3), 4.03; \tilde{e}_2 = (4,5), 4.81; \tilde{e}_3 = (3,5), 5.15; \tilde{e}_4 = (1,4), 8.01$; $L_T \approx 21.99$; gap ≈ 2.62.

(i) Gale (1968) / rank-order bijection (sorted weights matched by rank):

$$e_1 \mapsto (2,3) : 0, \qquad e_2 \mapsto (4,5) : 0.68, \qquad e_3 \mapsto (3,5) : 0.34, \qquad e_4 \mapsto (1,4) : 1.61.$$

Induced tour-edge penalties: $p(2,3) = 0,\ p(4,5) = 0.68,\ p(3,5) = 0.34,\ p(1,4) = 1.61$.

(ii) Canonical (kernel-barcode) bijection (Thm. 3.1 / Fig. 5):

$$e_1 \mapsto (2,3) : 0, \qquad e_2 \mapsto (3,5) : 1.01, \qquad e_3 \mapsto (4,5) : 0, \qquad e_4 \mapsto (1,4) : 1.60.$$

Induced penalties: $p(2,3) = 0,\ p(4,5) = 0,\ p(3,5) = 1.01,\ p(1,4) = 1.60$.

Both sum to ≈ 2.62, as Eq. (1) forces for any bijection. But the rank pairing (i) assigns strictly positive badness to $(4,5) \in E(T_{\text{mst}}) \cap E(T)$, an edge that cannot contribute to suboptimality relative to the MST, and (ii) under-scores $(3,5)$, the edge that actually bridges the clusters 2, 3 and 4, 5 a full scale-interval later than the MST does. A descending-penalty 2-opt (Alg. 2) driven by rank gaps therefore attempts to remove a correct edge before the misplaced one. Only the connectivity-defined pairing has the property that $p(\tilde{e}) = 0$ whenever $\tilde{e} \in E(T_{\text{mst}})$. And more generally that $(w(e), w(\varphi(e)))$ is the exact interval of scales a at which the clusters merged by e in $G^{w<a}$ remain disconnected in $T^{w<a}$ — i.e., a bar of $\text{Ker}(H_0(T^{w<a}) \to H_0(G^{w<a}))$. This per-edge attribution, not the existence of a dominating bijection, is what all algorithms in the paper consume.

Y Proof of Proposition: Zero-Penalty Edges

Proposition. $p(e) = 0 \iff e \in E(T_{\text{mst}})$.

Proof. Consider a case of distinct edge weights. Since $p(e) = w(e) - w(\psi(e)) \geq 0$, $p(e) = 0 \iff e = \psi(e)$, and $e = \psi(e) \in T_{\text{mst}}$ by definition of ψ. □

Remark. This proposition establishes a clear structural interpretation: the proposed penalty $p(\tilde{e})$ perfectly isolates the topological redundancy introduced by the tour. Edges that belong to the instance's MST—the natural backbone of the graph's connectivity—incur zero penalty. Any positive penalty strictly reflects the excess scale at which a tour edge bridges clusters compared to the optimal MST connectivity.

Z Experiments compute resources

Experiments were carried out at server with NVIDIA TITAN RTX, 80 cpu cores.

On Complexity. Each iteration of RTDL Tulchinskii et al. (2025) requires $\mathcal{O}(n^2)$, yielding a total cost of $\mathcal{O}(Tn^2)$ over T iterations. We note that a recent preprint (Anonymous, 2026) proposes a modification that reduces the per-iteration cost to $\mathcal{O}(n \log n)$. Our method is compatible with this variant without modification. We report experiments with the original, peer-reviewed version of RTDL and leave a thorough empirical evaluation of the accelerated variant to future work.

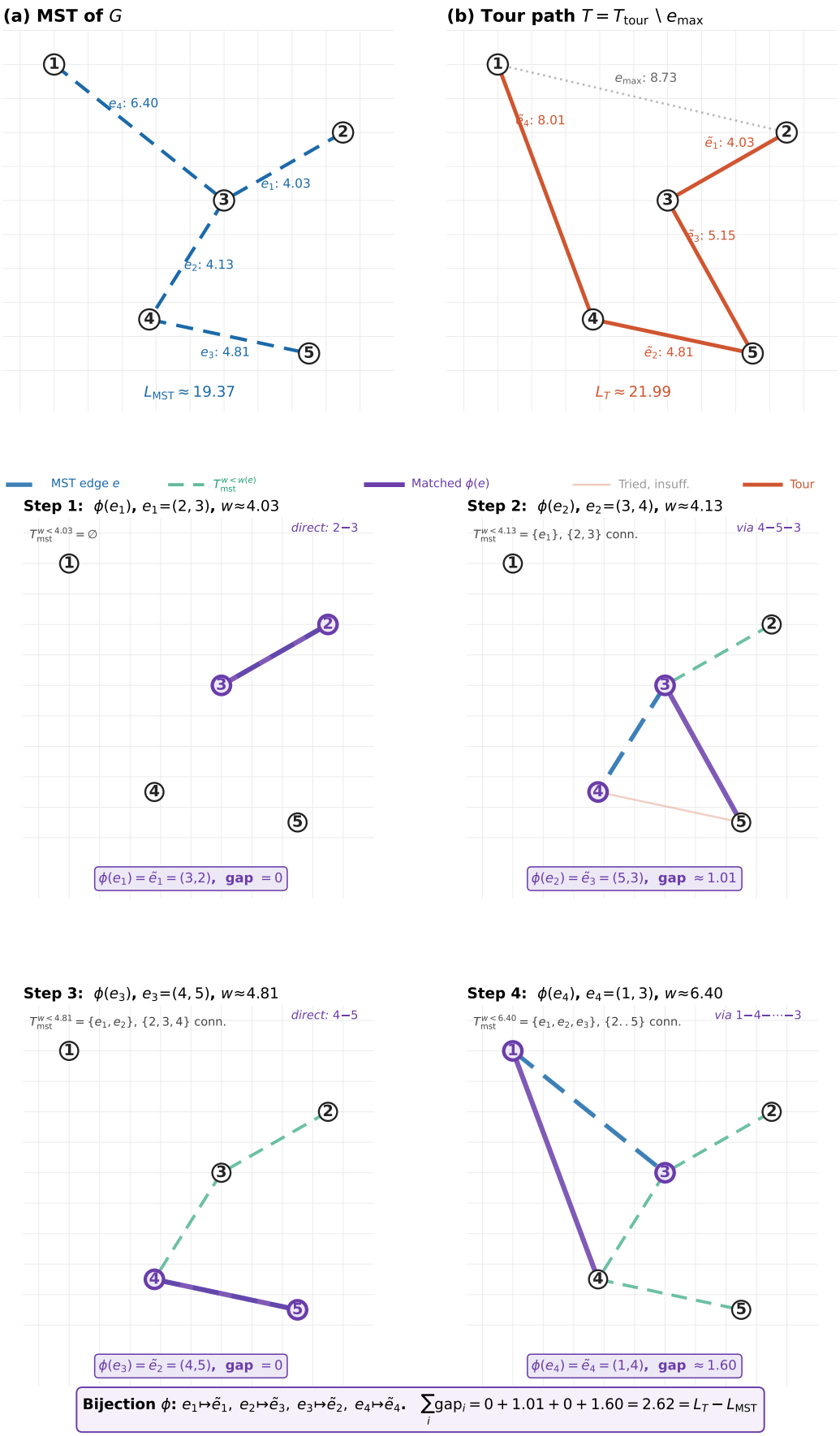
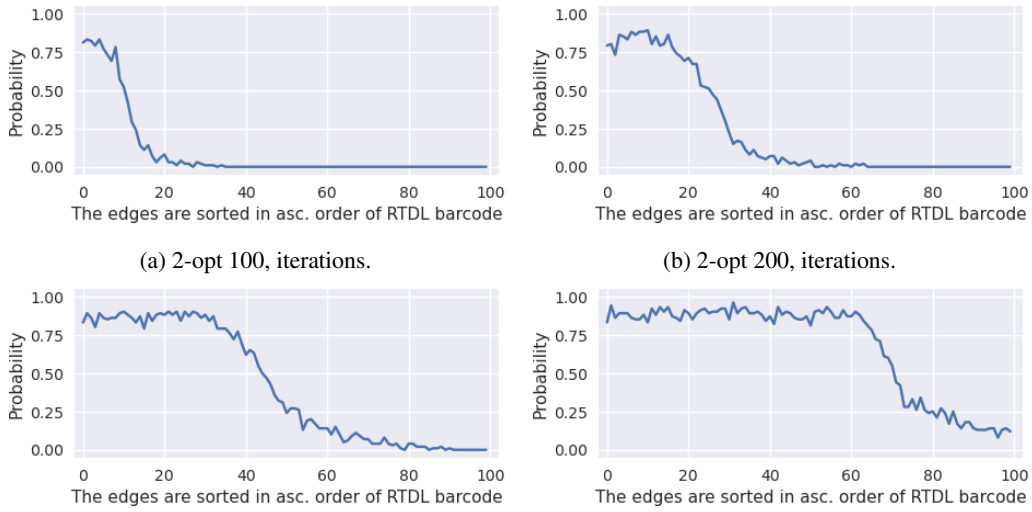


Figure 5: Illustration of the canonical bijection $\phi\colon E(T_{\mathrm{mst}}) \to E(T)$ constructed in Theorem 3.1.



(a) 2-opt 100, iterations.

(b) 2-opt 200, iterations.

(c) 2-opt 300, iterations.

(d) 2-opt, converged.

Figure 6: Probability of belonging to an optimal tour vs. RTDL bar of an edge.

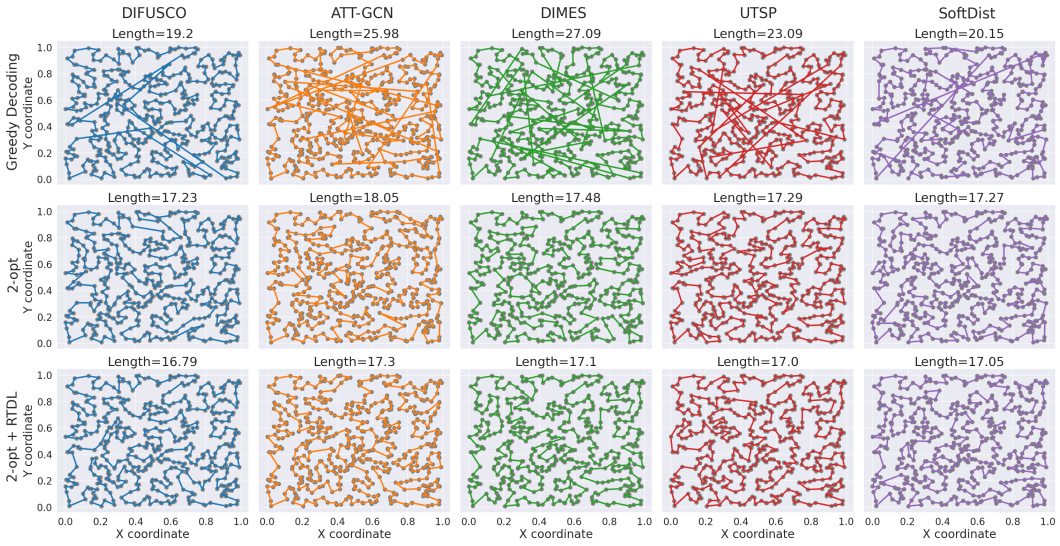


Figure 7: Example of solutions for the same TSP-500 problem through different models and optimization methods. For each model, we provide initial tour obtained through heatmap greedy decoding, 2-opt and 2-opt + RTDL optimized tours.

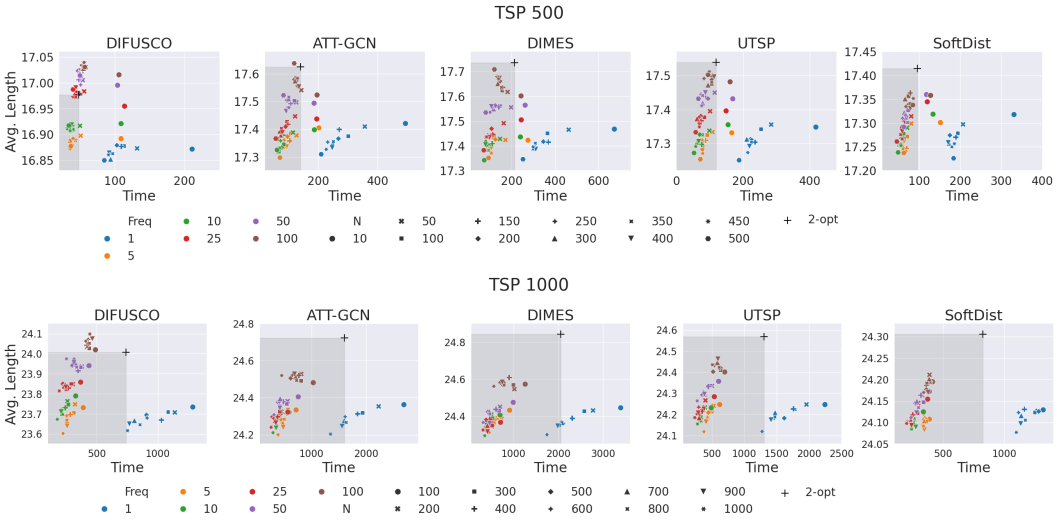


Figure 8: Sensitivity analysis for heatmap-guided 2-opt + RTDL. *Freq* refers to RTDL update frequency, *N* refers to number of vertices to optimize. Grey area indicates solutions better than 2-opt solution, i.e. shorter average length and faster computation.

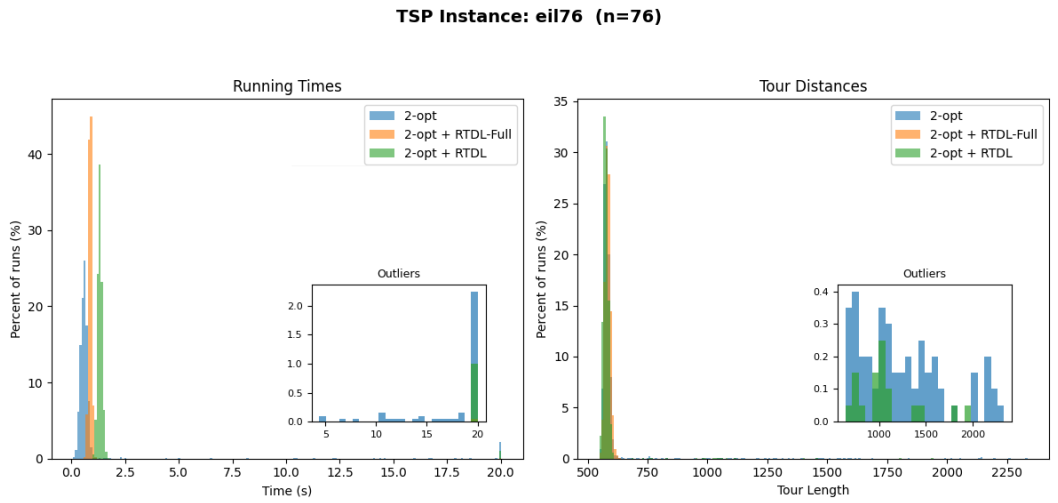
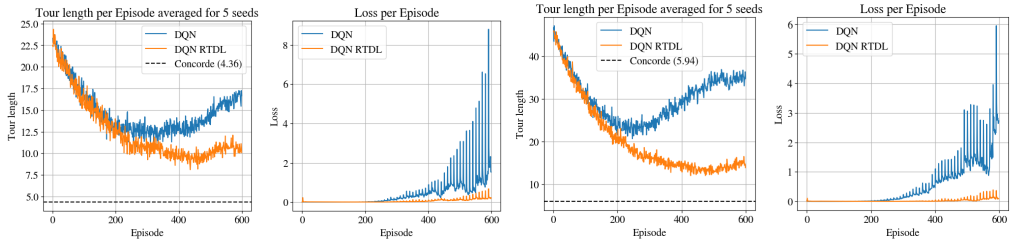


Figure 9: The distributions of calculation times and resulting tour lengths for different versions of 2-opt on a given TSP instance.



(a) 50 cities (b) 100 cities

Figure 10: Learning curves of the baseline DQN and DQN+RTDL on TSP instances with 50 and 100 cities. Each curve is averaged over five random initializations.

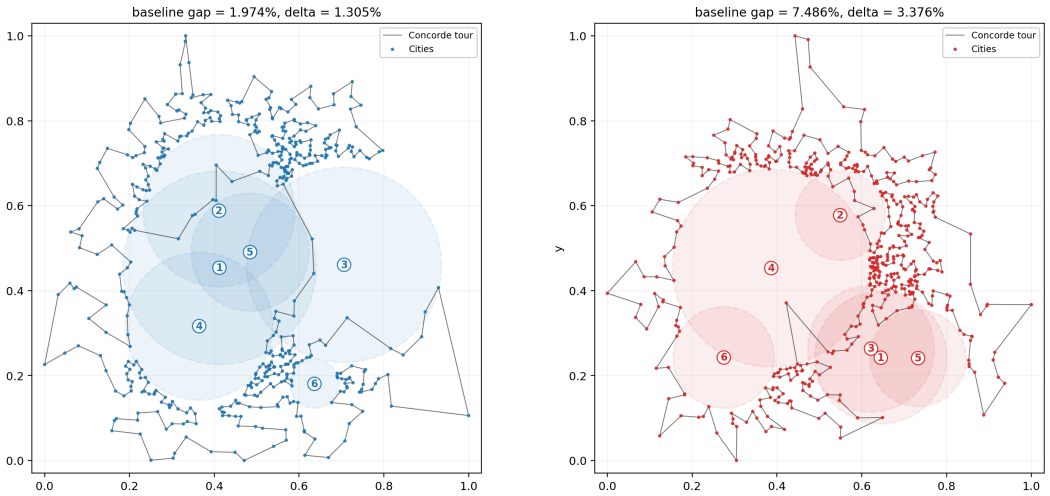
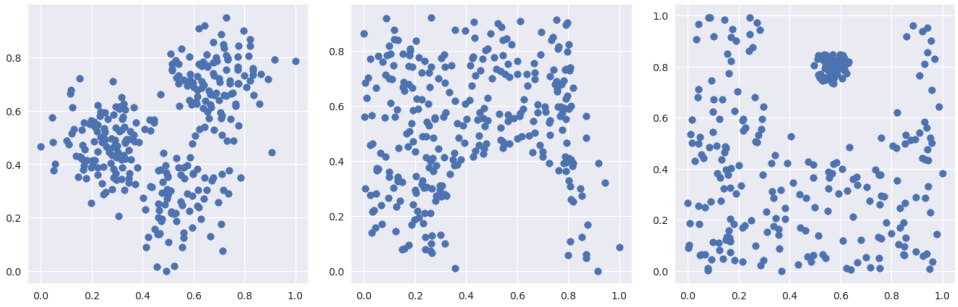


Figure 11: Representative explosion instances with Concorde tours. The left panel shows an instance that is relatively easy for the baseline solver, while the right panel illustrates a harder case where structured holes create more room for improvement.



(a) Clusters. (b) Explosion. (c) Implosion.

Figure 12: Examples of TSP instances with non-trivial topology.

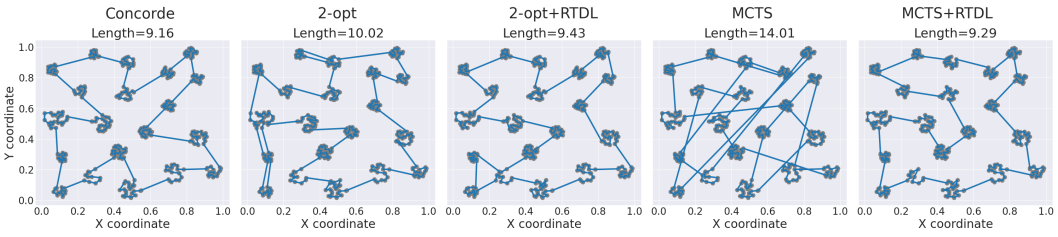


Figure 13: Example of solutions for the same TSP-500 problem and SoftDist heatmaps through different refinement methods. We provide optimized tours given by Concorde, 2-opt (+ RTDL) and MCTS (+ RTDL)

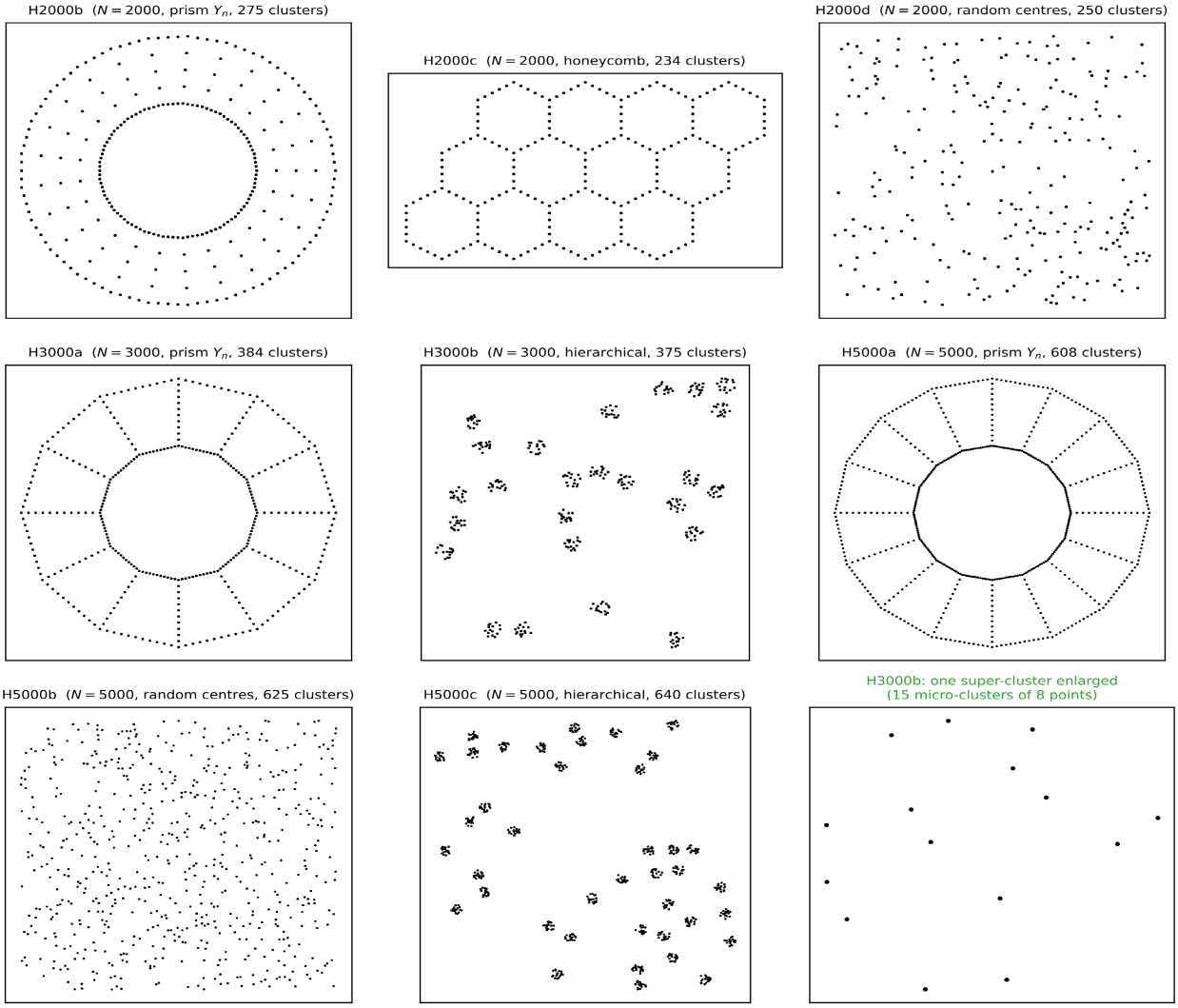


Figure 14: Hard instances for LKH.