UNDERSTANDING HEATMAP-BASED NEURAL TSP SOLVERS AS THE DISTANCE-SHIFTER

Anonymous authors
Paper under double-blind review

ABSTRACT

Heatmap-based neural solvers for the Traveling Salesman Problem (TSP) typically generate probability heatmaps and then extract tours with post-hoc search heuristics. In this paper, we show that this pipeline is fundamentally a *distance shifter*: any probability heatmap can be converted into a surrogate distance matrix through a simple logarithmic transformation, and searching for the maximum-probability tour is therefore equivalent to solving another TSP. This observation has two important implications. First, heatmap generation does not reduce the NP-hardness of TSP, and conventional post-hoc search algorithms such as MCTS, beam search, and k-opt are often suboptimal compared to specialized TSP solvers. Second, instead of learning heatmaps via costly reinforcement learning or classification, we can directly learn the distance-shifting transformation through supervised regression. Building on these insights, we propose a unified distance-shifting paradigm for neural TSP solvers, which includes (i) a rule-based method that converts pretrained heatmaps into surrogate distances, and (ii) learnable distance-shifting architectures, including SVSNet—an ultra-compact model based on singular value shrinkage with only four learnable parameters—and DTNet, a bidirectional cross-attention model in three sizes (0.02M/0.14M/1.09M parameters). Extensive experiments on general asymmetric/symmetric non-metric/Euclidean TSP across four distance distributions show that our methods surpass the SOTA heuristic LKH on in-distribution instances and remain comparable to it under out-of-distribution shifts, while consistently outperforming existing DL-based neural solvers. Moreover, trained on only 1,000 small instances, our models transfer directly from ATSP-100 to ATSP-1000 and generalize across distributions far more robustly than other DL-based solvers, all with extreme parameter efficiency, offering a promising lightweight direction for neural combinatorial optimization. Data and code will be released upon publication.

1 INTRODUCTION

The Traveling Salesman Problem (TSP) (Applegate et al., 2007) is a foundational NP-hard combinatorial optimization task. While often studied in its 2D Euclidean form (Kool et al., 2018; Kwon et al., 2020; Li et al., 2023; Qiu et al., 2022), real-world scenarios—such as logistics affected by traffic and speed limits—require solving the general TSP, where costs are frequently non-Euclidean and asymmetric (Kwon et al., 2021; Pan et al., 2025).

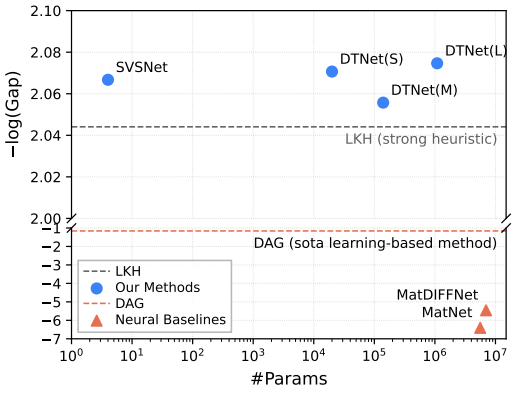


Figure 1: (Highlight) Model performance w.r.t. model scale on uniform ATSP-100. Compared to current DL-based neural heatmap methods, we achieve much better results with less than 1/100,000 model parameters.

Current learning-based neural methods (Sanokowski et al., 2024; Kwon et al., 2021; Pan et al., 2025) typically produce a heatmap encoding edge probabilities, followed by search heuristics such as Monte Carlo Tree Search (MCTS) to extract a tour from this learned heatmap. In this work, we demonstrate that this pipeline effectively acts as a "distance-shifter",

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.

which maps the original TSP instance into a surrogate TSP problem. This observation bears two key implications:

1. Searching over the heatmap remains NP-hard. While the learned heatmap provides probabilistic estimates for edges appearing in the optimal tour, identifying the tour with the overall maximum probability is computationally intractable. Consequently, existing post-hoc search methods such as MCTS (Sanokowski et al., 2024), k-opt (Pan et al., 2025), and beam search (Joshi et al., 2019) are often insufficient for heatmap searching.
2. Instead of constructing sophisticated training pipelines, e.g., reinforcement learning-based training (Kwon et al., 2021) and iterative sampling procedures in diffusion models (Sanokowski et al., 2024), which incur substantial training and inference costs for heatmap generation, we can directly model this distance-shifting transformation between the distance matrices of the original TSP and its surrogate using (weak) supervisory signals.

In this paper, our contributions are as follows:

i) We reveal that searching tours in a heatmap is equivalent to solving another TSP. Taking the negative logarithm of the edge probabilities maps any heatmap $\mathbf{H}$ to a surrogate distance matrix $\widetilde{\mathbf{D}} = -\log \mathbf{H}$ (Proposition 1), showing that heatmap generation does not reduce the NP-hardness of TSP and that heatmap-based neural solvers fundamentally act as *distance shifters* mapping the original distance matrix $\mathbf{D}$ to a surrogate $\widehat{\mathbf{D}}$. Furthermore, we demonstrate that, given the same heatmap, different search algorithms and TSP solvers exhibit substantial performance gaps when searching for the tour with the maximum probability (Fig. 2). Building on this observation, we innovatively propose replacing traditional post-hoc search algorithms (e.g., MCTS, beam search, 2-opt) with specialized TSP solvers (e.g., LKH) for heatmap searching.

ii) We reformulate heatmap learning as a distance regression problem. Instead of training the neural heatmap via reinforcement learning or edge-wise classification (as in previous heatmap-based neural solvers), we innovatively cast the whole pipeline as a regression task: training a neural distance shifter NN_{DS} that directly maps $\mathbf{D}$ to a surrogate matrix $\widehat{\mathbf{D}}$, supervised by contracting edges along high-quality LKH tours. This end-to-end supervised training avoids the instability and cost of RL and diffusion-based sampling, while allowing weak supervisory signals to effectively bias the optimization landscape. Theoretically, Proposition 2 shows that appropriately contracting the edges of an optimal tour preserves the original optimal tour, thus yielding an optimal surrogate TSP.

iii) We develop a family of lightweight distance-shifting models with consistent improvements over the SOTA heuristic, with highlighted results presented in Fig. 1. Under the distance-shifter paradigm, we propose two architectures: **SVSNet**, an algebraic spectral module based on singular value shrinkage with only **4 learnable parameters**, and **DTNet-S/M/L**, bidirectional cross-attention models with **0.02M/0.14M/1.09M parameters**. All of them outperform the current ATSP SOTA learning-based methods with consistent improvements across distributions (Uniform, Normal, Exponential, and 2D), problem scales (ATSP-100 and ATSP-1000, trained only on 1000 ATSP-100 instances), and search budgets (max_trials from 10 to 10,000). Notably, our methods also outperforms the SOTA heuristic LKH in in-distribution and even some out-of-distribution cases under the same computational budget. As an extension, we also further show the framework's promising applicability beyond TSP on ACVRP.

Related Works. We put related works in Appendix C where we detailedly analyze the differences between our work and others and summarize the main points in Table 9.

2 Background

Traveling Salesman Problem. Given a node set $\mathcal{V}$ ($\mathcal{V} = \{1, 2, \cdots, N\}$) along with a $N \times N$ distance matrix $\mathbf{D}_{ij}$ where the entry $\mathbf{D}_{ij}$ is the distance from node i to j, the problem is to find the tour (a Hamiltonian cycle that visits all the nodes exactly once and returns to the starting node) $\tau = (\tau_1, \cdots, \tau_N, \tau_1)$ to minimize the tour length $L_{\mathbf{D}}(\tau) = \sum_{i=1}^{N-1} \mathbf{D}_{\tau_i \tau_{i+1}} + \mathbf{D}_{\tau_N \tau_1}$. Without losing generality or changing the hard level of the problem, we assume that $\mathbf{D}_{ij} > 0$ for $i \neq j$, and $\mathbf{D}_{ij} = +\infty$ for $i = j$. In this paper, both theoretically and experimentally, we discuss TSP in the general cases:

- **Asymmetric:** $\mathbf{D}_{ij} = \mathbf{D}_{ji}$ does NOT have to hold.

- **Non-Metric:** The triangle inequality, i.e. $\mathbf{D}_{ij} + \mathbf{D}_{jk} \geq \mathbf{D}_{ik}$, does NOT have to hold.

Heatmap-based Neural Solver with Post-hoc Search. We formulate the methods that adopt the pipeline of neural heatmap generation followed by a searching heuristic on the heatmap as:

$$\mathbf{D} \xrightarrow{NN(\cdot)} \mathbf{H} \xrightarrow{\mathcal{S}(\cdot)} \tau, \tag{1}$$

where $\mathbf{H} \in [0,1]^{N \times N}$ is the generated heatmap and $\mathcal{S}(\cdot)$ is the searching heuristic. $\mathbf{H}_{ij}$ represents the probability that edge (i,j) belongs to the optimal tour, which can also be understood as the transition probability from node i to node j, $p(j|i)$. $\mathbf{H}_{ij}$ should satisfy the probability normalization rule, i.e. $\sum_{j=1}^{N} \mathbf{H}_{ij} = 1$ for all i.

Tour Probability. We start with a path[1] $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$ of M nodes, then according to Markov chain rule we obtain the probability of π:

$$p(\pi) = p(\pi_1) \prod_{i=2}^{M} p(\pi_i|\pi_{<i}) = p(\pi_1) \prod_{i=2}^{M} p(\pi_i|\pi_{i-1}), \tag{2}$$

where the term $p(\pi_1)$ can be set a constant equally for different choices of π_1, e.g. $1/N$, indicating that we randomly choose a node as the starting node. When it comes to the tour in TSP, we can simply replace the path π in Eq. 2 as a tour τ and obtain:

$$p_{\mathbf{H}}(\tau) = p(\tau_1)p(\tau_1|\tau_N) \prod_{i=1}^{N-1} p(\tau_{i+1}|\tau_i) = p(\tau_1)\mathbf{H}_{\tau_N\tau_1} \prod_{i=1}^{N-1} \mathbf{H}_{\tau_i\tau_{i+1}}. \tag{3}$$

3 Searching Tours in Heatmap is Another TSP

Applying the simple logarithm on $p_{\mathbf{H}}(\tau)$ in Eq. 3, we get

$$\log p_{\mathbf{H}}(\tau) = \log p(\tau_1) + \log \mathbf{H}_{\tau_N\tau_1} + \sum_{i=1}^{N-1} \log \mathbf{H}_{\tau_i\tau_{i+1}}. \tag{4}$$

Omitting the constant term $\log p(\tau_1)$, we can find that the problem of finding the tour of the highest probability from the heatmap is in fact another TSP, whose distances are $\mathbf{D}_{ij} = -\log \mathbf{H}_{ij}$. It means that the learned heatmap does not decrease the complexity of TSP in the sense that they are both NP-hard in general. We provide a formal proposition on such a equivalence as follows.

Proposition 1 (Equivalence of Searching over Heatmaps and a TSP)**.** *Given an $N \times N$ heatmap $\mathbf{H}$, we define the mapping as Eq. 5 to map the heatmap $\mathbf{H}$ to a distance matrix $\widetilde{\mathbf{D}}$:*

$$f_{\mathbf{H}\to\widetilde{\mathbf{D}}}\colon \widetilde{\mathbf{D}}_{ij} = -\log(\mathbf{H}_{ij}). \tag{5}$$

The mapping preserves the total ordering of the solution space $\mathcal{T}$. That is, the ranking of any tour τ by length $L_{\widetilde{\mathbf{D}}}(\tau)$ (ascending, denoted as $R^{\uparrow}_{\widetilde{\mathbf{D}}}(\tau)$) is equivalent to its ranking by probability $p_{\mathbf{H}}(\tau)$ (descending, denoted as $R^{\downarrow}_{\mathbf{H}}(\tau)$).

Proof. We have $L_{\widetilde{\mathbf{D}}}(\tau) = \log p(\tau_1) - \log p_{\mathbf{H}}(\tau)$, so $L_{\widetilde{\mathbf{D}}}(\tau)$ strictly monotonically decreases w.r.t. $p_{\mathbf{H}}(\tau)$. Consequently, for any two tours $\tau_1, \tau_2 \in \mathcal{T}$:

$$L_{\widetilde{\mathbf{D}}}(\tau_1) < L_{\widetilde{\mathbf{D}}}(\tau_2) \iff p_{\mathbf{H}}(\tau_1) > p_{\mathbf{H}}(\tau_2). \tag{6}$$

Thus, the rankings $R^{\uparrow}_{\widetilde{\mathbf{D}}}(\tau)$ and $R^{\downarrow}_{\mathbf{H}}(\tau)$ are identical for all $\tau \in \mathcal{T}$. Q.E.D.

3.1 Sub-optimality of the Searching Algorithms

Existing approaches adopt MCTS (Sanokowski et al., 2024), k-opt (Pan et al., 2025), beam search (Joshi et al., 2019) and other algorithms to perform post-hoc searching over the learned heatmap. Nevertheless, these methods overlook the inherent NP-hardness of the underlying search problem, and thus can scarcely retrieve the maximum-probability tour from the heatmap.

[1] A tour is a special path where the starting node is also the last node and other nodes show exactly once.

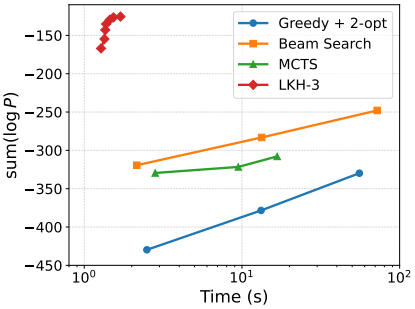


Figure 2: Efficiency comparison of the searching algorithms (MCTS, 2-opt, beam search) and the strong TSP heuristic solver (LKH) for searching the Hamiltonian tours maximizing $p_{\mathbf{H}}(\tau)$ over randomly generated heatmaps.

Empirical Demonstration. We examine heatmaps produced by diverse methods and test various search strategies proposed in prior literature. In Fig. 2, we present the average $\log p_{\mathbf{H}}(\tau)$ with respect to the searching runtime. As a baseline, we employ LKH[2] for searching over $\widetilde{\mathbf{D}}$, which is mapped from $\mathbf{H}$ following Proposition 1. Our results reveal a substantial gap between existing post-hoc search strategies and the maximum-probability tour across randomly generated heatmaps. In particular, MCTS, beam search, and 2-opt require considerably more search time to achieve comparable tour probabilities, while their final solutions remain noticeably inferior to those obtained by LKH. In contrast, LKH rapidly identifies high-probability tours and continues to improve the solution with only a small increase in runtime, demonstrating a clear advantage in both solution quality and search efficiency. These results suggest that directly exploiting the distance structure induced by the heatmap can be substantially more effective than applying generic post-hoc search strategies in the heatmap space.

4 Improve TSP Solvers from the Distance-Shifting Perspective

As demonstrated in the previous sections, the heatmap generation pipeline of existing neural TSP solvers essentially acts as a **"Distance Shifter"**. It transforms the distance matrix $\mathbf{D}$ of the original TSP instance into a surrogate distance matrix $\widehat{\mathbf{D}}$, on which the subsequent tour search is actually performed. Based on this key observation, we propose two viable directions to advance the performance of neural TSP solvers:

- **Search Optimization (Sec. 4.1)**. We formulate the heatmap-based search process as a standard TSP task and adopt dedicated TSP solvers for path exploration. This paradigm enables more efficient and accurate retrieval of high-probability tours.
- **Mapping Paradigm Optimization (Sec. 4.2)**. Instead of learning heatmap representations via a classification-based framework, we reformulate the distance shifting procedure as a regression task and optimize it in an end-to-end manner.

4.1 Pretrained Neural Heatmap as a Distance Shifter

Method overview. The work flow can be described as

$$\mathbf{D} \xrightarrow{\mathrm{NN}_{pre}(\cdot)} \mathbf{H} \xrightarrow{f(\cdot)} \widehat{\mathbf{D}} \xrightarrow{\mathcal{S}(\cdot)} \tau, \tag{7}$$

where NN_{pre} is the pretrained neural network for heatmap generation, and $\widehat{\mathbf{D}}$ is the distance matrix of the surrogate TSP mapped from $\mathbf{H}$ via function f. Our effort is on the designing of f, with prior knowledge from Proposition 1. We investigate MatNet (Kwon et al., 2021) and MatDIFFNet (Pan et al., 2025) as the base neural heatmap generators (NN_{pre}), and integrate them into our distance shifting framework by converting their output heatmaps into surrogate distance matrices via $f(\cdot)$ before solving with $\mathcal{S}(\cdot)$.

Heatmap construction. The procedure for constructing the edge probability heatmap $\mathbf{H}$ depends on the underlying backbone model (NN_{pre}). For MatNet, which relies on autoregressive decoding, we follow the multi-start principle of POMO (Kwon et al., 2020) to generate n probability matrices $\mathbf{H}^{(i)}$ by initiating the rollout from all n possible starting nodes, and obtain the final heatmap by averaging them: $\mathbf{H} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{H}^{(i)}$. In contrast, MatDIFFNet directly predicts a complete heatmap $\mathbf{H}$, which can be readily used without requiring multi-start averaging.

Distance Fusion. Given a probability heatmap $\mathbf{H}$, we first map it to the distance domain via $\widetilde{\mathbf{D}} = -\log(\mathbf{H})$. Then we linearly scale it to the interval $[d_{\min}, d_{\max}]$ where $d_{\min} = 1$ and $d_{\max} = 10^5$ to

[2]Lin-Kernighan-Helsgaun algorithm (Helsgaun, 2000; 2017), one of the most prominent near-optimal heuristic solvers for TSP.

match the scale of the original distance matrix $\mathbf{D}$:

$$\widetilde{\mathbf{D}} \leftarrow d_{\min} + \left(\widetilde{\mathbf{D}} - \min(\widetilde{\mathbf{D}})\right) \cdot \frac{d_{\max} - d_{\min}}{\max(\widetilde{\mathbf{D}}) - \min(\widetilde{\mathbf{D}})}. \tag{8}$$

Finally, the surrogate distance matrix $\widehat{\mathbf{D}}$ is derived by fusing $\widetilde{\mathbf{D}}$ with the original geometric distance matrix $\mathbf{D}$ via an arithmetic average:

$$\widehat{\mathbf{D}} = (\mathbf{D} + \widetilde{\mathbf{D}})/2. \tag{9}$$

This fusion ensures that the final distance metric incorporates both the structural constraints of the original and the learned preferences encoded within the heatmap.

4.2 Train a Distance Shifter End-to-End

Method Overview. Different from relying on a pretrained neural heatmap generator, we can directly train a learnable distance shifter network end-to-end to transform the original distance matrix. Under this paradigm, the workflow simplifies to:

$$\mathbf{D} \xrightarrow{\mathrm{NN}_{DS}(\cdot)} \widehat{\mathbf{D}} \xrightarrow{\mathcal{S}(\cdot)} \tau, \tag{10}$$

where $\mathrm{NN}_{DS}(\cdot)$ denotes the learnable distance shifter network, $\widehat{\mathbf{D}}$ is the transformed distance matrix, and $\mathcal{S}(\cdot)$ represents the downstream TSP solver that yields the final tour τ. The core effort lies in designing an effective distance shifter network NN_{DS} and its corresponding training pipeline. To this end, we propose two distinct architectures as concrete instantiations of NN_{DS}: an algebraic spectral-based module named **Singular Value Shrinkage Net (SVSNet)** and a deep learning-based architecture named **Distance Transformation Net (DTNet)**.

4.2.1 Singular Value Shrinkage Net

We propose a simple Singular Value Shrinkage Net (SVSNet) module as $\mathrm{NN}_{DS}(\cdot)$ module. Given a distance matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ with singular value decomposition $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, SVSNet is defined as: $\mathrm{SVSNet}_\theta(\mathbf{D}) = \mathbf{U} f_\theta(\mathbf{\Sigma})\mathbf{V}^T$, where $f_\theta(\cdot)$ is the singular value shrinkage function, a low-order polynomial function parameterized by θ and element-wisely applied to the singular values. In practice, we apply a 3-order f (a cubic polynomial, i.e. $f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$), which is competent enough to achieve better performance than vanilla methods. We initialize θ as $\theta = [0, 1, 0, 0]^\top$ and add Gaussian noise sampled from $\mathcal{N}(0, 0.01)$. Then the filtered distance matrix $\widehat{\mathbf{D}}$ can be defined by a weighted sum of $\mathbf{D}$ and $\mathrm{SVSNet}_\theta(\mathbf{D})$ parameterized by a hyper-parameter α (0.1 as default):

$$\widehat{\mathbf{D}} = (1 - \alpha)\mathbf{D} + \alpha \mathrm{SVSNet}_\theta(\mathbf{D}). \tag{11}$$

Remarks. Despite of its naivety, there are some highlighted features of SVSNet, including but not limited to: **i) Node order invariance.** The SVD core naturally gifts SVSNet with the invariance to node order permutation, which keeps challenging the generalizability of many neural methods (Kool et al., 2018; Qiu et al., 2022; Kwon et al., 2021). **ii) Crossing-scale applicability.** Since f_θ is an element-wise function, SVSNet trained on small-scale data can be directly applied on data at a larger scale. **iii) Training efficiency.** The lightweight and shallowness of SVSNet also bring convenience to its training and convergence, without a huge amount of training data as most neural solvers.

4.2.2 Distance Transformation Net

We also design a lightweight DL-based Distance Transformation Net (DTNet) as another $\mathrm{NN}_{DS}(\cdot)$ module, using bidirectional cross-attention to capture the global structural information of the input distance matrix. The model first extracts statistical features from the rows and columns of the distance matrix and maps them into separate row and column embeddings. Bidirectional cross-attention is then employed to model the global interactions between the row and column representations. Based on these updated embeddings, an edge-level MLP takes the corresponding row embedding, column embedding, and original edge weight as input to predict a distance correction. The transformed distance matrix is finally obtained in a residual form, allowing the model to adaptively refine the original distances while preserving their underlying structure.

We consider three different model sizes, denoted as DTNet-S, DTNet-M, and DTNet-L, to study the effect of model capacity. They use embedding dimensions of 32, 64, and 128, with 2, 4, and 8 attention heads, and the numbers of attention layers are set to 1, 2, and 4. Accordingly, the three configurations contain approximately 0.02M, 0.14M, and 1.09M trainable parameters, respectively.

Remarks. DTNet possesses several appealing properties: **i) Global contextual expressiveness.** Through bidirectional cross-attention between row and column representations, DTNet integrates information from the entire distance matrix and captures global row-column dependencies before predicting pairwise refinements. **ii) Structure-preserving residual refinement.** Predicting distance corrections in a residual form enables the model to adaptively fine-tune localized metric relationships while largely preserving the underlying geometric structure. **iii) Scalable model capacity.** By varying the embedding dimension, attention heads, and number of layers, DTNet can be instantiated with different model sizes (DTNet-S/M/L), providing a controllable trade-off between representation capacity and computational cost.

4.2.3 Supervised Training on the Distance Shifting

We train SVSNet$_\theta$ and DTNet-S/M/L by supervised learning. The loss is define as the mean squared error loss with diagonal elements masked:

$$\mathcal{L} = \mathrm{MSE}_{non-diag}\big(\Delta\mathbf{D}, -\beta\mathbf{D}\odot\mathbf{T}\big), \quad \Delta\mathbf{D} = \mathrm{NN}_{DS}(\mathbf{D}) - \mathbf{D}, \tag{12}$$

where $\odot$ is the Hadamard product, $\mathbf{T}$ denotes the binary tour adjacency matrix converted from a tour. $\mathbf{T}_{ij} = 1$ indicates that the tour contains an edge from node i to node j. The tour is obtained by LKH with parameters `max_trials` $= 10$, `runs` $= 1$, with the rest parameters set to their default values. β is a hyper-parameter which is empirically set to 0.1 as default. This objective encourages the transformed distances to contract edges along high-quality tours, thereby biasing the optimization landscape in favor of better solutions. Theoretically, the following Proposition 2 (proven in Appendix A) guarantees the reasonability of the loss function Eq. 12.

Proposition 2 (Existence of Optimal Distance Shifts)**.** *Let $\mathbf{D}$ be the distance matrix of a TSP instance with $d_{\min} = \min_{i\neq j}\mathbf{D}_{ij} > 0$, and let $\tau^\star$ be an optimal tour of $\mathbf{D}$ with binary adjacency matrix $\mathbf{T}^\star$. Then for any $\beta \in (0, 1)$, the distance-weighted shift $\Delta\mathbf{D} = -\beta\,\mathbf{D}\odot\mathbf{T}^\star$ makes $\tau^\star$ the unique optimal tour of $\mathbf{D} + \Delta\mathbf{D}$. Moreover, the optimal tour length decreases for at least $2\beta\, d_{\min}$.*

5 Experiments

Dataset Setup. Train data. We train the auxiliary SVSNet on 1000 uniform-distributed ATSP-100 instances. For each instance, tours for supervision are generated by LKH with `max_trials`=10 `runs`=1. **Test data.** We investigate ATSP-100 and ATSP-1000 with distances drawn from the uniform, normal and exponential distributions. For ATSP-100, we generate 10k test instances per distribution, while for ATSP-1000, we generate 128 test instances per distribution. All distance values are scaled to lie within the range $[1, 10^5]$.

Baselines. MatNet (Kwon et al., 2021): Given a predicted edge probability distribution, a tour is constructed by first selecting a starting node and then iteratively choosing the next node with the highest transition probability. This process is repeated by enumerating different starting nodes following the POMO strategy (Kwon et al., 2020), and the best among all roll-outs is selected as the solution. Unless otherwise specified, we adopt the pre-trained MatNet model provided by the official repository. **MatDIFFNet** (Pan et al., 2025): MatDIFFNet utilizes a discrete diffusion framework tailored for general TSP via a novel mix-noised reference map scheme that fuses distance matrices with noised solution states. The model learns a reverse denoising process to progressively reconstruct the optimal solution distribution from random noise, ultimately outputting a probabilistic edge heatmap. Final tours are constructed by decoding this heatmap, where post-hoc local search methods like 2-opt are typically applied to further refine the tour quality. **DAG** (Zhang et al., 2025): A non-learning continuous relaxation solver for ATSP that formulates subtour elimination as a trace-based Directed Acyclic Graph (DAG) constraint over doubly stochastic matrices. It directly optimizes individual cost matrices using projected exponentiated gradient descent and extracts valid discrete tours via greedy subtour decoding followed by 2-opt local search refinement. **LKH3** (Helsgaun, 2017): A well-known near-optimal heuristic solver for TSP. We also consider neural solvers for

Table 1: Performance on TSP-100. Gap $= (L_{\text{method}} - L_{\text{optimal}})/L_{\text{optimal}} \times 100\%$, where L_{method} and L_{optimal} denote the tour lengths obtained by the method and the optimal solution, respectively. For LKH, the value in parentheses denotes the `max-trials`. Time is reported in seconds (s). **Bold**: best in the set. Underline: outperforms LKH (10000). Gray box : outperforms DAG.

Method	Train Data	Uniform		Normal (OOD)		Exponential (OOD)		2D (OOD)	
		Gap	Time	Gap	Time	Gap	Time	Gap	Time
Baselines									
Gurobi (optimal)	–	0.0000	2541.72	0.0000	2525.55	0.0000	2488.89	0.0000	26581.87
LKH (10)	–	6.7277	151.59	0.4354	153.92	7.5742	138.93	1.3029	161.98
LKH (100)	–	0.8548	164.81	0.0696	169.83	0.9702	153.42	0.1540	203.86
LKH (1000)	–	0.2760	285.98	0.0252	271.79	0.2917	281.40	0.0101	561.88
LKH (10000)	–	**0.1295**	1287.77	**0.0102**	1129.61	**0.1326**	1347.49	**0.0009**	3830.02
MatNet (Kwon et al., 2021)	Metric Uniform	606.0314	187.78	33.8119	194.42	1253.8938	192.31	137.7611	193.95
MatDIFFNet (Pan et al., 2025)	Uniform	234.4839	5517.32	29.8846	5580.22	199.6243	5492.49	–	–
DAG (Zhang et al., 2025)	–	3.1914	619.40	0.4616	620.63	3.0700	512.89	4.4365	1625.13
DeepACO (Ye et al., 2024a)	2D	N/A						16.6475	365.46
Fast-T2T (Li et al., 2024)	2D	N/A						2.1722	309.00
Our methods									
MatNet-LKH (10)	Metric Uniform	113.3212	263.41	8.4614	257.89	234.6778	254.67	18.23	272.00
MatDIFFNet-LKH (10)	Uniform	21.7442	5472.78	0.4781	6037.61	41.9951	5840.75	–	–
SVSNet-LKH (10)	Uniform	6.6381	154.55	0.4327	166.76	7.5503	147.28	1.3272	173.88
SVSNet-LKH (100)	Uniform	0.8443	174.27	0.0689	178.32	0.9569	165.87	0.1602	212.71
SVSNet-LKH (1000)	Uniform	0.2708	294.66	0.0248	284.47	0.2911	292.22	0.0105	573.99
SVSNet-LKH (10000)	Uniform	0.1266	1292.12	0.0103	1148.25	**0.1295**	1366.50	0.0009	3836.57
DTNet(S)-LKH (10)	Uniform	6.6137	151.48	0.4389	159.58	7.3792	142.54	1.3192	170.53
DTNet(S)-LKH (100)	Uniform	0.8618	166.95	0.0698	173.91	0.9371	160.83	0.1568	209.29
DTNet(S)-LKH (1000)	Uniform	0.2732	287.96	0.0247	277.16	0.2871	285.33	0.0102	569.93
DTNet(S)-LKH (10000)	Uniform	0.1261	1287.86	**0.0102**	1134.20	0.1301	1343.63	0.0008	3996.22
DTNet(M)-LKH (10)	Uniform	6.6621	160.60	0.4382	162.75	7.5121	143.93	1.3262	173.79
DTNet(M)-LKH (100)	Uniform	0.8467	176.11	0.0693	175.41	0.9641	162.50	0.1568	212.45
DTNet(M)-LKH (1000)	Uniform	0.2738	295.11	0.0244	287.07	0.2915	288.95	0.0107	571.60
DTNet(M)-LKH (10000)	Uniform	0.1280	1386.92	0.0103	1170.44	0.1306	1363.66	**0.0006**	3829.06
DTNet(L)-LKH (10)	Uniform	6.5571	189.28	0.4361	197.53	7.5633	144.22	1.3306	169.14
DTNet(L)-LKH (100)	Uniform	0.8489	211.11	0.0704	211.72	0.9597	163.44	0.1597	208.08
DTNet(L)-LKH (1000)	Uniform	0.2722	356.17	0.0254	343.38	0.2915	288.54	0.0103	567.99
DTNet(L)-LKH (10000)	Uniform	**0.1256**	1601.81	0.0106	1171.80	0.1312	1345.51	0.0007	3817.29

2D TSP although they are limited to instances with 2D coordinates: **DeepACO** (Ye et al., 2024a), **Fast-T2T** (Li et al., 2024).

Hyper-parameters. For MatNet, we follow the default parameters of MatNet. The encoder consists of 5 layers with an embedding dimension of 256. Each layer employs 16 attention heads, where each head utilizes a 16-dimensional mixed-score attention mechanism with an MLP hidden dimension of 16. For decoding, we follow the POMO inference method as described in the original MatNet paper, with a `pomo_size` of 100. For MatDIFFNet, the encoder is a GNN-based architecture consisting of 12 layers with a hidden dimension of 256 and an output channel size of 2. The tour is generated by greedy decoding combined with a 2-opt local search, as in the MatDIFFNet's decoder. The coefficient $\alpha = 0.4$ and $\beta = 0.1$ are selected via grid search. For the LKH solver, we set the candidate set size to 5 and the number of runs to 1, while all other parameters for LKH follow the default configuration.

RQ1: Original Searching over Heatmap $\mathbf{H}$ v.s. TSP-specific solvers over Surrogate Distance $\widehat{\mathbf{D}}$. We compare neural solvers that perform direct search over the generated heatmap $\mathbf{H}$ with our proposed paradigm, which converts $\mathbf{H}$ into a surrogate distance matrix $\widehat{\mathbf{D}}$ via $f(\cdot)$ and subsequently applies a TSP-specific solver $\mathcal{S}(\cdot)$ (e.g., LKH). As shown in Table 1, solving TSP over the transformed surrogate distance matrix $\widehat{\mathbf{D}}$ with LKH consistently achieves substantially shorter tour lengths, particularly under out-of-distribution (OOD) scenarios, while maintaining comparable computational efficiency. This improvement is observed across both MatNet and MatDIFFNet, indicating that converting heatmap guidance into surrogate distances effectively enhances solution quality and significantly alleviates the performance degradation of neural solvers caused by distribution shifts.

RQ2: End-to-End Distance Shifter v.s. Baselines.

i) Performance consistency across distributions. Our distance-shifting models consistently maintain strong performance across Uniform, Normal, Exponential, and 2D distributions. In most settings, SVSNet and DTNet variants achieve lower gaps than the original LKH, demonstrating that the learned distance transformation generalizes well beyond the training distribution.

Table 2: Performance on TSP-1000. **Bold**: best in the set. Underline: outperforms LKH (10000). Gray box : outperforms DAG.

Method	Train Data	Uniform		Normal (OOD)		Exponential (OOD)		2D (OOD)	
		Gap	Time	Gap	Time	Gap	Time	Gap	Time
Baselines									
Gurobi (optimal)	–	0.0000	8029.19	0.0000	7238.46	0.0000	7445.09	Out of time	
LKH (10)	–	53.0720	134.58	0.8894	145.25	61.2901	127.40	2.8910	142.55
LKH (100)	–	21.6170	141.48	0.1382	146.55	30.0391	136.16	1.5551	156.72
LKH (1000)	–	4.3453	186.01	0.0652	181.45	9.5549	191.45	0.6372	298.31
LKH (10000)	–	**1.5507**	603.19	**0.0364**	495.72	**3.1756**	677.02	**0.0000**	1752.85
MatNet (Kwon et al., 2021)		Out of memory							
MatDIFFNet (Pan et al., 2025)		Out of memory							
DAG (Zhang et al., 2025)	–	1.9964	1381.91	0.0731	1389.24	3.9900	1454.07	4.6857	43717.23
Our methods									
SVSNet-LKH (10)	Uniform ATSP-100	51.6094	151.30	0.8454	150.29	61.2734	133.53	2.8872	143.77
SVSNet-LKH (100)	Uniform ATSP-100	22.1872	150.39	0.1399	153.47	32.4422	141.00	1.5412	156.08
SVSNet-LKH (1000)	Uniform ATSP-100	4.6836	190.70	0.0651	181.98	11.9151	198.34	0.7351	286.83
SVSNet-LKH (10000)	Uniform ATSP-100	1.6597	569.35	0.0372	469.95	3.7598	678.66	0.0658	1606.29
DTNet(S)-LKH (10)	Uniform ATSP-100	51.5758	127.66	0.8628	139.99	38.3809	122.94	2.8376	136.25
DTNet(S)-LKH (100)	Uniform ATSP-100	19.9286	133.12	0.1438	143.69	9.4419	126.66	1.5799	147.24
DTNet(S)-LKH (1000)	Uniform ATSP-100	5.1714	174.87	0.0684	174.11	3.4985	143.64	0.6670	276.38
DTNet(S)-LKH (10000)	Uniform ATSP-100	1.6460	556.25	**0.0370**	459.61	**1.6620**	326.90	0.0711	1624.82
DTNet(M)-LKH (10)	Uniform ATSP-100	52.6381	131.37	0.8522	140.37	61.5649	122.24	3.0461	134.48
DTNet(M)-LKH (100)	Uniform ATSP-100	20.8254	136.54	0.1340	143.65	28.8479	128.90	1.6123	146.88
DTNet(M)-LKH (1000)	Uniform ATSP-100	5.3633	176.97	0.0667	171.28	9.3709	177.42	0.6428	277.10
DTNet(M)-LKH (10000)	Uniform ATSP-100	1.6728	549.72	0.0380	459.23	3.1945	600.67	0.0760	1586.51
DTNet(L)-LKH (10)	Uniform ATSP-100	52.3583	136.30	0.8379	148.08	61.3960	130.82	2.9270	147.77
DTNet(L)-LKH (100)	Uniform ATSP-100	21.4549	141.88	0.1445	147.51	27.3151	137.31	1.5289	161.06
DTNet(L)-LKH (1000)	Uniform ATSP-100	4.6749	186.72	0.0657	177.89	7.5584	183.76	0.6773	294.06
DTNet(L)-LKH (10000)	Uniform ATSP-100	**1.4401**	574.44	0.0393	476.42	3.3804	618.25	**0.0579**	1633.10

ii) Efficiency comparison and lightweightness. Compared to conventional neural baselines like MatNet (5.60M parameters) and MatDIFFNet (6.99M parameters), our proposed distance shifter networks offer remarkable parameter efficiency and ultra-fast training speeds while introducing minimal overhead over LKH. Specifically, **SVSNet** contains merely 4 learnable parameters, which is less than 10^{-6} of the baseline parameter counts. It can be trained entirely on a CPU in just 18.75 seconds for 10 epochs (1.88s per epoch) with a modest memory footprint of 741.57 MB. Meanwhile, our **DTNet** variants (DTNet-S/M/L) range from 0.02M to 1.09M parameters—still substantially more compact than neural baselines—and complete 20 training epochs on a GPU in only 13.97s, 22.31s, and 33.26s, respectively. Furthermore, as shown in Table 1 and Table 2, the testing time remains in the same order of magnitude as the original LKH, showing minimal variation and occasionally even decreasing. Overall, these results demonstrate that our end-to-end distance shifting paradigm yields significant performance gains without incurring heavy hardware, training, or inference burdens.

iii) Robustness to searching trials. Across different search budgets from 10 to 10,000 trials, our models exhibit stable performance and consistently follow the improvement trend of LKH. The results indicate that the proposed distance shifting is not sensitive to a particular solver configuration.

iv) Cross-scale transferability and out-of-distribution (OOD) generalizability. Models trained on ATSP-100 can be directly transferred to ATSP-1000 without retraining. This demonstrates that the learned distance transformation is not tightly coupled to a specific problem size. Even trained with only weakly supervised data from ATSP-100, our DTNet(L) outperforms the heuristic SOTA LKH on in-distribution uniform instances. Furthermore, our experiments under normal, exponential, and 2D distributions are all OOD settings. Our method achieves performance comparable to LKH, showing that unlike other neural network approaches, our approach does not suffer from significant performance degradation on out-of-distribution data.

v) Scalability on ATSP-1000. On ATSP-1000, our models remain effective across all tested distributions and search budgets, while introducing only limited additional runtime. In contrast, MatNet and MatDIFFNet cannot be evaluated due to the lack of 1000-node weights. Notably, on the 2D data, DAG consumes more than 20x of the running time than our methods. These results demonstrate the scalability advantage of the proposed framework against other learning-based methods.

RQ3: Ablation Studies on α and β (full results in Appendix B.2). We investigate the sensitivity of SVSNet to its core hyperparameters α and β on TSP-100. Results show that $\alpha = 0.4$ (as we adopt) performs best, suggesting that the surrogate distance matrix provides useful complementary

Table 3: Topological evaluation of distance matrices before and after matrix shift across different model architectures. ΔTCA and ΔTVR represent the relative percentage change with respect to the Raw Baseline.

Model / Variant	TCA	TVR	ΔTCA (%)	ΔTVR (%)
Original	0.387271	0.166884	0.0000	0.0000
SVSNet	0.387263	0.166884	-0.0022	0.0000
DTNet-S	0.387387	0.166914	+0.0300	+0.0179
DTNet-M	0.387331	0.166910	+0.0154	+0.0154
DTNet-L	0.387448	0.166919	+0.0455	+0.0210

information to the original distance matrix. In contrast, SVSNet is largely insensitive to β, with consistent performance observed over a broad range of $\beta \in 0.05, 0.1, 0.15, 0.2$. We therefore simply set $\beta = 0.1$ as the default value for training.

RQ4: Ablation Studies of NN_{DS} (full results in Appendix B.3). We substitute SVSNet with other simple approaches that can be fit into the framework. We investigate **Random-LKH**, which adopts a random $\widehat{\mathbf{D}}$ to guide the searching process of LKH, and **Scaling-LKH**, which adopts z-score and max-scaling for distance shifting. Results show that i) Random-LKH does not work, and ii) compared with the vanilla LKH, Scaling-LKH achieves comparable win-and-lose results, indicating that the simple rescaling approaches fail to introduce additional valuable information.

RQ5: Deep Analysis of Distance Shifter. To quantify the topological behavior of transformed distance matrices, we evaluate two key structural metrics:

- **Travel Cost Asymmetry (TCA):** $TCA = \frac{1}{|E|}\sum_{i\neq j}\frac{|\mathbf{D}_{ij}-\mathbf{D}_{ji}|}{\mathbf{D}_{ij}+\mathbf{D}_{ji}+\epsilon}$, measuring the directional cost imbalance across directed edges ($[0, 1)$).
- **Triangle Inequality Violation Ratio (TVR):** $TVR = \frac{\sum_{i\neq j\neq k}\mathbb{1}[\mathbf{D}_{ij}>\mathbf{D}_{ik}+\mathbf{D}_{kj}]}{N(N-1)(N-2)}$, quantifying the proportion of triplets that violate metric space properties ($[0, 1]$).

We compute TCA and TVR on the test dataset before and after matrix transformation using SVSNet and DTNet-S/M/L. As shown in Table 3, raw matrices exhibit high asymmetry ($\mathrm{TCA} \approx 0.3873$) and non-metricity ($\mathrm{TVR} \approx 0.1669$). Across all model variants, the fluctuations in both metrics remain strictly within $+0.05\%$. We highlight three main insights: **i) Physical Topology Preservation:** The near-zero metric shift ($\Delta\mathrm{TCA} \leq 0.0455\%$, $\Delta\mathrm{TVR} \leq 0.0210\%$) proves that Distance Shifter strictly preserves underlying physical constraints without artificially metricizing or over-smoothing the cost space. **ii)Capacity Invariance:** Increasing model capacity from *Small* to *Large* causes negligible metric drift, confirming the robustness and stability of our residual learning architecture. **iii) Decoupled Performance Gains:** Downstream routing improvements are decoupled from macro-geometric distortions. Distance Shifter enhances routing efficiency via subtle, localized feature refinement in the latent space rather than global topological alterations.

RQ6: Potential Application Possibilities and Advantages for Other Problems Appendix B.5. We apply SVSNet-LKH on ACVRP. Results show that it also works on other routing problems like ACVRP, extending the scope of Distance Shifter from TSP to the more realistic CVRP.

6 Conclusion and Limitations

In this work, we re-evaluate heatmap-based neural solvers through the lens of distance shifting, establishing that probability maximization on heatmaps is equivalent to solving a TSP on a surrogate distance matrix $\widehat{D}$. Within our framework, pre-trained neural models obtain large performance improvements—especially for OOD cases—and our method reaches state-of-the-art performance across diverse problem distributions and scales. The proposed paradigm highlights exceptional parameter efficiency and cross-scale transferability, offering a promising direction for neural combinatorial optimization.

Limitations. The study primarily focuses on TSP and ACVRP, leaving the extension to more complex routing tasks with multi-layered constraints for future exploration.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with translation. We have not used generative AI tools for cleaning and reformatting the dataset, generating synthetic data sets, developing theoretical models, formulating mathematical claims, writing proofs, proposing hypotheses, designing methodology, or interpreting results, and providing critical ingredients for mathematical proofs or supporting qualitative analysis are not applicable to this work. Additionally, we used generative AI tools for editing the research paper to improve readability and formatting references. We have reviewed all AI-assisted work. Textual edits generated by AI were thoroughly reviewed by all authors to ensure academic rigor and preserve original intent. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver, 2006.

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. 2007.

Paulo R d O da Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In Sinno Jialin Pan and Masashi Sugiyama (eds.), *Proceedings of The 12th Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, pp. 465–480. PMLR, 18–20 Nov 2020. URL https://proceedings.mlr.press/v129/costa20a.html.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization, 2023. URL https://arxiv.org/abs/2301.03313.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner, 2024. URL https://arxiv.org/abs/2406.15079.

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*, pp. 7474–7482, 2021.

LLC Gurobi Optimization. Gurobi optimizer reference manual (2020), 2023.

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

Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12, 2017.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. *arXiv preprint arXiv:2110.05291*, 2021.

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

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

Minsu Kim, Sanghyeok Choi, Jiwoo Son, Hyeonah Kim, Jinkyoo Park, and Yoshua Bengio. Ant colony sampling with gflownets for combinatorial optimization. *arXiv preprint arXiv:2403.07041*, 2024.

Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International conference on integration of constraint programming, artificial intelligence, and operations research*, pp. 190–213. Springer, 2022.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. From distribution learning in training to gradient search in testing for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JtF0ugNMv2.

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*, 2024.

Qiang Ma, Suwen Ge, Danyang He, Darshan Thaker, and Iddo Drori. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. *arXiv preprint arXiv:1911.04936*, 2019.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=q1JukwH2yP.

Yimeng Min, Yiwei Bai, and Carla P. Gomes. Unsupervised learning for solving the travelling salesman problem, 2024. URL https://arxiv.org/abs/2303.10538.

Yuichi Nagata and Shigenobu Kobayashi. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS Journal on Computing*, 25(2):346–363, 2013.

Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. Unico: On unified combinatorial optimization via problem reduction to matrix-encoded general tsp. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yEwakMNIex.

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

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Forty-first International Conference on Machine Learning*, 2024.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

IBM ILOG CPLEX Optimization Studio. V20.1: User's manual for cplex. *IBM Corp*, pp. 1–20, 2020.

Jingyan Sui, Shizhe Ding, Rui Tao Liu, Liming Xu, and Dongbo Bu. Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning. In *Asian Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:249038331.

Jingyan Sui, Shizhe Ding, Boyang Xia, Ruizhi Liu, and Dongbo Bu. Neuralgls: learning to guide local search with graph convolutional network for the traveling salesman problem. *Neural Comput. Appl.*, 36(17):9687–9706, October 2023. ISSN 0941-0643. doi: 10.1007/s00521-023-09042-6. URL https://doi.org/10.1007/s00521-023-09042-6.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. Deepaco: neural-enhanced ant systems for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36, 2024a.

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*, 2024b.

Zhen Zhang, Javen Qinfeng Shi, and Wee Sun Lee. Solving the asymmetric traveling salesman problem via trace-guided cost augmentation. In *Advances in Neural Information Processing Systems*, 2025.

Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li. Combining reinforcement learning with lin-kernighan-helsgaun algorithm for the traveling salesman problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12445–12452, May 2021. doi: 10.1609/aaai.v35i14.17476. URL https://ojs.aaai.org/index.php/AAAI/article/view/17476.

Appendix

CONTENTS

1 Introduction **1**

2 Background **2**

3 Searching Tours in Heatmap is Another TSP **3**

3.1 Sub-optimality of the Searching Algorithms 3

4 Improve TSP Solvers from the Distance-Shifting Perspective **4**

4.1 Pretrained Neural Heatmap as a Distance Shifter 4

4.2 Train a Distance Shifter End-to-End 5

4.2.1 Singular Value Shrinkage Net 5

4.2.2 Distance Transformation Net 5

4.2.3 Supervised Training on the Distance Shifting 6

5 Experiments **6**

6 Conclusion and Limitations **9**

A Proof to Proposition 2 **14**

B Supplementary Experiments **15**

B.1 Hardware Setting 15

B.2 Parameter Analysis for Hyper-parameters α and β 15

B.2.1 Setting of α 15

B.2.2 Setting of β 15

B.3 Ablation Studies 16

B.3.1 Comparison to Random-LKH, Table 6 16

B.3.2 Comparison to Scaling-LKH, Table 7 17

B.4 Analysis of Learned θ of SVSNet 18

B.5 Asymmetric Capacitated Vehicle Routing Problem (ACVRP) 19

C Full Related Works **20**

A Proof to Proposition 2

Proposition 3 (Existence of Optimal Distance Shifts)**.** *Let $\mathbf{D}$ be the distance matrix of a TSP instance with $d_{\min} = \min_{i \neq j} \mathbf{D}_{ij} > 0$, and let $\tau^\star$ be an optimal tour of $\mathbf{D}$ with binary adjacency matrix $\mathbf{T}^\star$. Then for any $\beta \in (0, 1)$, the distance-weighted shift $\Delta\mathbf{D} = -\beta\, \mathbf{D} \odot \mathbf{T}^\star$ makes $\tau^\star$ the unique optimal tour of $\mathbf{D} + \Delta\mathbf{D}$. Moreover, the optimal tour length decreases for at least $2\beta\, d_{\min}$.*

Proof. Under the shift, each entry of the shifted matrix is

$$(\mathbf{D} + \Delta\mathbf{D})_{ij} = \mathbf{D}_{ij}\,(1 - \beta\, \mathbf{T}^\star_{ij}) = \begin{cases} (1-\beta)\,\mathbf{D}_{ij}, & (i,j) \in \tau^\star, \\ \mathbf{D}_{ij}, & (i,j) \notin \tau^\star. \end{cases} \tag{13}$$

For any tour τ, let $s(\tau) = |\tau \cap \tau^\star|$ be the number of (directed) edges shared with $\tau^\star$. Its length under the shifted matrix satisfies

$$L_{\mathbf{D}+\Delta\mathbf{D}}(\tau) = L_{\mathbf{D}}(\tau) - \beta \sum_{(i,j) \in \tau \cap \tau^\star} \mathbf{D}_{ij}. \tag{14}$$

Therefore, for any $\tau \neq \tau^\star$,

$$\begin{aligned} L_{\mathbf{D}+\Delta\mathbf{D}}(\tau) - L_{\mathbf{D}+\Delta\mathbf{D}}(\tau^\star) &= \underbrace{\big[L_{\mathbf{D}}(\tau) - L_{\mathbf{D}}(\tau^\star)\big]}_{\geq 0} + \beta \sum_{(i,j) \in \tau^\star \setminus \tau} \mathbf{D}_{ij} \\ &\geq \beta\,\big[n - s(\tau)\big]\, d_{\min}. \end{aligned} \tag{15}$$

The first bracket is non-negative since $\tau^\star$ is an optimal tour of $\mathbf{D}$; the sum runs over edges of $\tau^\star$ not used by τ, and each of the $n - s(\tau)$ such edges contributes at least $d_{\min}$. Two distinct Hamiltonian cycles share at most $n - 2$ edges: removing any single edge from a cycle yields a Hamiltonian path, which can be closed only by the removed edge itself. Hence $n - s(\tau) \geq 2$, and Eq. equation 15 gives

$$L_{\mathbf{D}+\Delta\mathbf{D}}(\tau) - L_{\mathbf{D}+\Delta\mathbf{D}}(\tau^\star) \geq 2\beta\, d_{\min} > 0, \tag{16}$$

i.e., $\tau^\star$ is the unique optimal tour of $\mathbf{D} + \Delta\mathbf{D}$ with an optimality gap of at least $2\beta\, d_{\min}$. □

B SUPPLEMENTARY EXPERIMENTS

B.1 HARDWARE SETTING

Experiments were conducted on a machine equipped with an Intel Core i9-7920X CPU (12 cores, 24 threads, 2.90 GHz) and two NVIDIA GeForce RTX 2080 Ti GPUs with 11 GB VRAM each. Heuristic baselines and SVSNet are run on CPU, and neural baselines and DTNet-S/M/L are run on GPU.

B.2 PARAMETER ANALYSIS FOR HYPER-PARAMETERS α AND β

To investigate the impact of the key hyperparameters (α and β) in SVSNet on the final solving performance, we conducted a analysis on the ATSP-100 dataset. All experiments in this section were performed using LKH (specifically: `max_trials`=10, `runs`=1, and `candidate_num`=5).

B.2.1 SETTING OF α

The parameter α controls the balance between the original distance matrix D and the structural enhancements provided by SVSNet. As shown in Table 4, we observed that the performance gain initially increases with α, peaking at $\alpha = 0.4$ with a maximum improvement of 0.0840% and the consistent improvement across the tested range (0.1 to 0.4) demonstrates that SVSNet successfully injects beneficial structural priors into the distance domain without distorting the underlying problem objective.

Table 4: Hyperparameter analysis for α (best $\alpha = 0.4$, as adopted in the result of our experiment)

α	SVSNet-LKH Avg. Len. ↓	Gain (%) ↑
0 (baseline)	175328.92	0.0000
0.1	175269.79	0.0337
0.2	175226.66	0.0583
0.3	175275.31	0.0306
0.4 (set as default)	**175181.68**	**0.0840**
0.5	175257.30	0.0409
0.6	175374.51	-0.0598

B.2.2 SETTING OF β

The parameter β controls the weight of supervised signals in training. Table 5 shows the performance under different settings of β. The results indicate that SVSNet is largely insensitive to β, with consistent performance observed for $\beta \in 0.05, 0.1, 0.15, 0.2$. The insensitivity of β is consistent with Proposition 2. We set $\beta = 0.1$ as the default value for all experiments.

Table 5: Hyperparameter analysis for β (best $\beta = 0.1$, as used in our experiments)

β	SVSNet-LKH Avg.Len.	Gain (%)
0.1 (set as default)	175181.68	0.0840
0.05	175181.68	0.0840
0.15	175181.68	0.0840
0.2	175181.68	0.0840
0.25	179952.70	-2.6372

B.3 Ablation Studies

To further validate the effectiveness of SVSNet, we conducted additional ablation studies by substituting SVSNet with other possible choices:

- Random matrices $\widehat{\mathbf{D}}$ generated from Uniform(0, 100000).
- z-score: $\widehat{\mathbf{D}} = \frac{\mathbf{D} - mean(\mathbf{D})}{std(\mathbf{D})}$.
- max-scaling: $\widehat{\mathbf{D}} = \frac{\mathbf{D} - min(\mathbf{D})}{max(\mathbf{D}) - min(\mathbf{D})}$.

B.3.1 Comparison to Random-LKH, Table 6

We employ random matrices as the surrogate distance matrix $\widehat{\mathbf{D}}$ for LKH. The results in Table 6 demonstrate that random matrices severely degrade the searching capability of LKH.

Table 6: Comparison of Random-LKH and SVSNet-LKH on ATSP-100. Relative Advantage (RA) $(\%) = (1 - L_{\text{SVSNet-LKH}}/L_{\text{Random-LKH}}) \times 100\%$.

Parameters				Random-LKH	SVSNet-LKH	
Dist.	Max_trials	Runs	# Cand.	Avg. Len. ↓	Avg. Len. ↓	RA (%) ↑
Uniform	10	1	5	948761.71	175181.68	81.5358
Uniform	10	1	10	914359.80	176260.32	80.7231
Uniform	100	1	5	351119.42	165663.79	52.8184
Uniform	100	1	10	346232.48	165851.80	52.0981
Uniform	1000	1	5	238725.10	164721.62	30.9995
Uniform	1000	1	10	237459.43	164757.45	30.6166
Normal	10	1	5	2818490.51	1966641.11	30.2236
Normal	10	1	10	2788588.38	1968611.43	29.4047
Normal	100	1	5	2233165.03	1959466.04	12.2561
Normal	100	1	10	2225076.69	1959907.89	11.9173
Normal	1000	1	5	2046424.39	1958606.79	4.2913
Normal	1000	1	10	2044436.28	1958716.65	4.1928
Expo.	10	1	5	114603.86	18608.74	83.7626
Expo.	10	1	10	109588.18	18709.00	82.9279
Expo.	100	1	5	35284.39	17467.93	50.4939
Expo.	100	1	10	34866.82	17493.45	49.8278
Expo.	1000	1	5	24354.03	17352.73	28.7480
Expo.	1000	1	10	24246.21	17356.22	28.4168

B.3.2 COMPARISON TO SCALING-LKH, TABLE 7

Since LKH only supports integer inputs, both z-score normalization and max-scaling are linearly mapped to the range [1,100000], resulting in identical outcomes for the two methods. We employ them as the surrogate distance matrix $\widehat{\mathbf{D}}$ for LKH. The results in Table 6 indicate that while traditional statistical scaling often leads to search stagnation or performance degradation (negative gain), SVSNet consistently provides effective guidance, achieving superior solution quality across various problem scales and distributions.

Table 7: Comparison of Scaling-LKH and SVSNet-LKH on ATSP-100/1000. Gain$(\%) = (1 - L_{\mathrm{X}}/L_{\mathrm{LKH}}) \times 100\%$.

ATSP-100

Parameters			LKH	SVSNet-LKH		Scaling-LKH	
Dist.	Max_trials	# Cand.	Avg. Len. ↓	Avg. Len. ↓	Gain (%) ↑	Avg. Len. ↓	Gain (%) ↑
uniform	10	5	175328.92	175181.68	0.0840	175396.20	-0.0384
uniform	10	10	176304.20	176260.32	0.0249	176311.68	-0.0042
uniform	100	5	165681.11	165663.79	0.0105	165680.90	0.0001
uniform	100	10	165872.97	165851.80	0.0128	165864.18	0.0053
uniform	1000	5	164730.30	164721.62	0.0053	164727.42	0.0017
uniform	1000	10	164779.08	164757.45	0.0131	164765.98	0.0080
normal	10	5	1966654.42	1966601.97	0.0027	1966653.04	0.0001
normal	10	10	1968912.76	1968745.64	0.0085	1968910.84	0.0001
normal	100	5	1959491.58	1959479.23	0.0006	1959491.21	0.0000
normal	100	10	1959964.42	1959907.89	0.0029	1959965.32	0.0000
normal	1000	5	1958623.20	1958615.27	0.0004	1958623.37	0.0000
normal	1000	10	1958742.10	1958716.65	0.0013	1958741.25	0.0000
expo.	10	5	18612.88	18608.74	0.0222	18612.28	0.0032
expo.	10	10	18709.30	18709.00	0.0016	18732.01	-0.1214
expo.	100	5	17470.23	17467.93	0.0132	17472.32	-0.0120
expo.	100	10	17493.70	17493.44	0.0015	17494.50	-0.0046
expo.	1000	5	17352.84	17352.73	0.0006	17353.01	-0.0010
expo.	1000	10	17357.86	17356.22	0.0094	17357.77	0.0005

ATSP-1000

Parameters			LKH	SVSNet-LKH		Scaling-LKH	
Dist.	Max_trials	# Cand.	Avg. Len. ↓	Avg. Len. ↓	Gain (%) ↑	Avg. Len. ↓	Gain (%) ↑
uniform	10	5	250138.06	249271.30	0.3465	248851.95	0.5142
uniform	10	10	249722.99	249490.40	0.0931	250803.39	-0.4326
uniform	100	5	198736.72	197747.34	0.4978	202240.28	-1.7629
uniform	100	10	209831.13	208580.09	0.5962	211727.23	-0.9036
uniform	1000	5	170512.76	169661.77	0.4991	171055.36	-0.3182
uniform	1000	10	177261.95	174172.36	1.7429	175998.15	0.7130
normal	10	5	18336561.11	18327773.41	0.0479	18336561.11	0.0000
normal	10	10	18468036.03	18460633.33	0.0401	18468036.03	-0.0000
normal	100	5	18200031.56	18198297.05	0.0095	18200031.55	0.0000
normal	100	10	18243946.22	18243751.07	0.0011	18243793.75	0.0008
normal	1000	5	18186747.88	18185332.64	0.0078	18186747.88	-0.0000
normal	1000	10	18197639.88	18196176.45	0.0080	18197355.14	0.0016
expo.	10	5	17735.63	17659.42	0.4297	17566.34	0.9545
expo.	10	10	17637.77	17594.27	0.2467	17490.59	0.8345
expo.	100	5	14299.23	14082.59	1.5151	14250.14	0.3433
expo.	100	10	14889.51	14863.57	0.1742	15089.45	-1.3428
expo.	1000	5	12046.77	12008.78	0.3153	12011.27	0.2947
expo.	1000	10	12561.11	12468.00	0.7412	13005.16	-3.5351

B.4 Analysis of Learned θ of SVSNet

To provide further insights into the structural transformation performed by SVSNet, we report the specific values of the learned coefficients θ obtained from the training on the ATSP-100 dataset.The learned coefficients are $\theta = [2.5347 \times 10^{-3},\ 9.9638 \times 10^{-1},\ 5.3377 \times 10^{-3},\ -5.5315 \times 10^{-4}]^{\top}$.

The numerical distribution of these coefficients reveals the mathematical logic behind the spectral filtering mechanism of SVSNet:

Dominance of the Linear Term θ_1: The first-order coefficient θ_1 (≈ 0.9964) is the most significant component. This indicates that SVSNet treats the original distance matrix as the foundational skeleton. It ensures that the transformed matrix $\widehat{D}$ preserves the essential geometric constraints and physical meaning of the original ATSP instance.

Inhibition Effect of the High-order Term θ_3: The third-order coefficient θ_3 is negative (-5.5315×10^{-4}). In the context of singular value decomposition, this negative correction acts as a high-spectrum suppressor. By applying a subtle down-weighting to extreme singular values, it effectively filters out structural noise that might otherwise mislead the LKH search process or allow specific dimensions to over-dominate the search space.

Positive Contribution of θ_0 and θ_2: Both the constant term θ_0 (2.5347×10^{-3}) and the second-order term θ_2 (5.3377×10^{-3}) provide positive numerical contributions. θ_0 serves as a minimal global offset to ensure numerical positivity and stability during the mapping process. Meanwhile, θ_2 provides a moderate non-linear compensation to the singular value spectrum through quadratic enhancement.

B.5 Asymmetric Capacitated Vehicle Routing Problem (ACVRP)

To evaluate the generalization potential of the distance shifting framework and SVSNet, we extended our evaluation to the Asymmetric Capacitated Vehicle Routing Problem (ACVRP-100). For this experiment, the SVSNet was specifically retrained on a small dataset of 1,000 ACVRP instances to capture the domain-specific structural constraints of vehicle capacities and demands. The results, summarized in Table 8, lead to the following observations:

Generalization across Distributions: SVSNet-LKH mostly matches or outperforms the vanilla LKH baseline across various synthetic distributions. The most significant and stable improvements are observed under the Uniform distribution, with a maximum gain of 0.1436%. This indicates that the structural priors captured by SVSNet from TSP are transferable to more complex VRP variants.

Table 8: Results of ACVRP-100

Parameters				**LKH**	**SVSNet-LKH**	
Dist.	Max_trials	Runs	# Cand.	Avg. Len. ↓	Avg. Len. ↓	Gain (%) ↑
Exponential	10	1	5	70549.3600	70510.2600	0.0554
Exponential	10	1	10	62319.8800	62297.5700	0.0358
Exponential	100	1	5	54756.3400	54787.1300	-0.0562
Exponential	100	1	10	51693.3500	51689.8000	0.0069
Exponential	1000	1	5	46326.2200	46314.4600	0.0254
Exponential	1000	1	10	45191.7500	45190.4100	0.0030
Normal	10	1	5	2681100.7800	2681760.6200	-0.0246
Normal	10	1	10	2654646.1400	2654564.6900	0.0031
Normal	100	1	5	2577153.4400	2577007.5000	0.0057
Normal	100	1	10	2562925.7300	2563049.7300	-0.0048
Normal	1000	1	5	2514449.9800	2514123.1000	0.0130
Normal	1000	1	10	2506840.1100	2506593.9500	0.0098
Uniform	10	1	5	620462.3600	619571.4900	0.1436
Uniform	10	1	10	559746.2700	559466.6700	0.0500
Uniform	100	1	5	493935.0100	493390.3400	0.1103
Uniform	100	1	10	469257.4900	469158.9100	0.0210
Uniform	1000	1	5	423529.0300	422955.4300	0.1354
Uniform	1000	1	10	414183.3500	414040.1900	0.0346

C FULL RELATED WORKS

Exact Solvers. Solvers for linear programming and mixed integer linear programming, e.g., Gurobi and CPLEX, can be used to solve general TSP with exact optimal solutions as output. These methods can be time-consuming in real-world applications where the scale of instances may go large. Concorde is a solver developed for TSP but it fails to deal with general TSP where only distances are available.

Heuristic Solvers. Some trivial heuristic algorithms include nearest neighbor (NN), furthest insertion (FI), etc. After decades of optimizations, strong heuristics such as Lin-Kernighan-Helsgaun (LKH by (Helsgaun, 2017)) and EAX (Nagata & Kobayashi, 2013) have been well developed and are able to give satisfying results with good efficiency and scalability. Beyond classical discrete search, continuous relaxation heuristics have also emerged; for instance, DAG (Zhang et al., 2025) replaces discrete subtour elimination with a trace-based acyclicity constraint over doubly stochastic matrices, enabling gradient-based continuous optimization without any offline training.

Neural Solvers. Compared with the heuristics and exact solvers, neural solvers are believed to be more efficient when the solving procedure is run parallelly on GPUs. Also, the learning-based approaches are believed to have a stronger capability in solving TSP instances from a specific known distribution. Most neural TSP solvers are developed for TSP with coordinates (Kool et al., 2018; Kwon et al., 2020; Li et al., 2023; Qiu et al., 2022; Sun & Yang, 2023; Vinyals et al., 2015; Min et al., 2024). This is due to the current lack of neural networks that can effectively learn in scenarios with **only** pairwise feature information, such as the distances in TSP. MatNet (Kwon et al., 2021) is one of the works that can be applied to BiTSP. It proposes a Transformer-based solver for asymmetric TSP (ATSP) whose input is a distance matrix. However, the heavy-encoder heavy-decoder neural architecture and the DRL-based training put limitations to its scalability and consequently its applicability to real-world problems whose scales are varying. Recently, several works (Drakulic et al., 2023; 2024; Ye et al., 2024b) conducted experiments on ATSP and showed promising results. Yet, such part regarding general TSP solving have not open-sourced.

Learning-enhanced Heuristic Solvers. GNNGLS (Hudson et al., 2021) and NeuralGLS (Sui et al., 2023) hybridize GNNs and Guided Local Search (GLS) for TSP solving. Similar works (da Costa et al., 2020; Sui et al., 2021; Ma et al., 2023) are there to solve COPs by neurally defining 2-, 3-, and k-opt heuristics in a data-driven manner. (Kool et al., 2022; Ye et al., 2024a; Kim et al., 2024) also explore to combine ML techniques with heuristics upon dynamic programming, ant colony, and probabilistic search algorithms. As the NCO community and LKH evolves, developing neural solvers that combine the LKH algorithm and neural networks is also an important line of research. VSR-LKH (Zheng et al., 2021) improves LKH's heuristic strategy with RL-driven policies, which achieves performance improvement on 2D Euclidean TSP compared with vanilla LKH. It is a learning-based solver yet without neural networks. NeuroLKH (Xin et al., 2021), improves LKH with the Sparse Graph Network (SGN) aimed at generating better edge candidates as a substitution for LKH's α-nearest measure. However, SGN relies on node coordinates as input, making NeuroLKH inapplicable to general TSP.

Table 9: Technical comparison of SVSNet-LKH, DTNet-S/M/L and other TSP Solvers.

Solver Type	Method	Distance Type	Description	Open-sourced
Exact solver	Gurobi (Gurobi Optimization, 2023)	General	N/A	Available but not open-sourced
	CPLEX (Studio, 2020)			
	Concorde (Applegate et al., 2006)	Euclidean		
Heuristic solver	LKH (Helsgaun, 2017)	General	See Appendix **??**	✓
	EAX (Nagata & Kobayashi, 2013)	Symmetric	Genetic algorithm	✓
Neural solver	PointerNet (Vinyals et al., 2015)	Euclidean	GPN+RL, pre-trained	✓
	(Ma et al., 2019)			✓
	AM (Kool et al., 2018)		GAT+DRL, pre-trained	✓
	POMO (Kwon et al., 2020)			✓
	SYM-NCO (Kim et al., 2022)			✓
	DIMES (Qiu et al., 2022)		GNN+meta RL, pre-trained	✓
	GCN(Joshi et al., 2019)		GNN+SL, pre-trained	✓
	Att-GCN(Fu et al., 2021)			✓
	DIFUSCO(Sun & Yang, 2023)		Diffusion+Generative, pre-trained	✓
	T2T(Li et al., 2023)			✓
	UTSP (Min et al., 2024)		SAG+UL, pre-trained	✓
	QUBO (Schuetz et al., 2022)		GNN+UL, pre-trained	✓
	BQ-NCO (Drakulic et al., 2023)	**General**	GCN+Transformer+RL, pre-trained	ATSP solver not provided
	GOAL (Drakulic et al., 2024)		GCN+Transformer+RL, pre-trained	✓
	GLOP (Ye et al., 2024b)		Divide-Conquer+RL, pre-trained	No ATSP solver proposed
	MatNet (Kwon et al., 2021)		Transformer+RL, pre-trained	✓
	MatDIFFNet (Pan et al., 2025)		Diffusion+generative, pre-trained	✓
Learning-enhanced heuristic solver	GNNGLS (Hudson et al., 2021)	Euclidean	GNN+GLS, pre-trained	✓
	NeuralGLS (Sui et al., 2023)		GNN+GLS, pre-trained	✗
	DeepACO (Ye et al., 2024a)		GNN+ACO, pre-trained	✓
	NeuroLKH (Xin et al., 2021)		SGN+LKH, pre-trained	✓
	VSR-LKH (Zheng et al., 2021)	**General**	RL+LKH	✓
	DAG (Zhang et al., 2025)		Gradient-based optimization	✓
	• **SVSNet-LKH** (ours)		SVSNet+LKH	✓(Upon publication)
	• **DTNet(S/M/L)-LKH** (ours)		DTNet+LKH	✓(Upon publication)
	• **MatNet-LKH** (ours)		rule-based transformation+LKH	✓(Upon publication)
	• **MatDIFFNet-LKH** (ours)		rule-based transformation+LKH	✓(Upon publication)