Solving Asymmetric Traveling Salesman Problem via Trace-Guided Cost Augmentation

Zhen Zhang
[1]Responsible AI Research Centre
Australian Institute for Machine Learning
[2]The University of Adelaide & [3]CSRIO
zhen@zzhang.org

Javen Qingfeng Shi
[1]Responsible AI Research Centre
Australian Institute for Machine Learning
[2]The University of Adelaide
javen.shi@adelaide.edu.au

Wee Sun Lee
School of Computing
National University of Singapore
leews@comp.nus.edu.sg

Abstract

The Asymmetric Traveling Salesman Problem (ATSP) is one of the most fundamental and notoriously challenging problems in combinatorial optimization. We propose a novel continuous relaxation framework for ATSP that leverages differentiable constraints to encourage acyclic structures and valid permutations. Our approach integrates a differentiable trace-based Directed Acyclic Graph (DAG) constraint with a doubly stochastic matrix relaxation of the assignment problem, enabling gradient-based optimization over soft permutations. We further develop a projected exponentiated-gradient method with adaptive step size to minimize tour cost while satisfying the relaxed constraints. To recover high-quality discrete tours, we introduce a greedy post-processing procedure that iteratively eliminates subtours through cost-aware cycle merging. Empirically, our method achieves state-of-the-art performance on standard asymmetric TSP benchmarks and demonstrates strong scalability and accuracy, particularly on large or highly asymmetric instances where heuristic solvers such as LKH-3 often struggle.

1 Introduction

The Asymmetric Traveling Salesman Problem (ATSP) is one of the most fundamental and notoriously challenging problems in combinatorial optimization. Many important combinatorial optimization (CO) problems are theoretically reducible to ATSP. Recent work by Pan et al. [22] has shown that such reductions are not only of theoretical interest but can also be practically feasible for certain small-scale CO tasks, suggesting that advances in ATSP solvers could yield broad benefits across the CO landscape.

Although the symmetric TSP is a special case of ATSP, solvers designed specifically for the asymmetric case often perform poorly on symmetric instances. Conversely, any ATSP instance can be transformed into a symmetric TSP by doubling the number of nodes [9], enabling symmetric solvers to be applied to asymmetric problems. Consequently, much of the CO community has focused on the symmetric TSP [1, 8, 20, 26], leading to the development of increasingly effective handcrafted and learning-based heuristics. Recently, methods such as those proposed by Drakulic et al. [4, 5], Pan et al. [22] have employed reinforcement learning to directly search for optimal ATSP tours. These approaches provide faster inference compared to state-of-the-art heuristics such as LKH-3 [8], but typ-

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

ically yield lower-quality solutions—even though LKH-3 was originally developed for the symmetric TSP.

Assignment-problem (AP) relaxations have long been used to provide strong lower bounds within branch-and-bound and branch-and-cut TSP solvers [17, 19], and they remain effective for large-scale asymmetric instances. Several constructive heuristics, including the Karp–Steele Patching (KSP) algorithm [12, 13] and its variants, attempt to lift assignment solutions into valid tours. However, based on existing open-source implementations, these heuristics often underperform relative to LKH-3. Interestingly, AP relaxations tend to be tighter for asymmetric TSPs than for symmetric ones, motivating the development of stronger heuristics grounded in AP formulations.

A general TSP can be formulated as a linear assignment problem with an additional constraint ensuring that the solution forms exactly one cycle [9]. We observe that this constraint is equivalent to requiring all proper subgraphs of the assignment to be Directed Acyclic Graphs (DAGs). Inspired by recent progress in differentiable DAG learning [25, 28, 29], we propose a novel continuous relaxation of the TSP that incorporates DAG constraints into the assignment formulation. We show that the global minimum of this relaxed problem exactly recovers the true TSP solution.

Although the proposed relaxation is non-convex, we find empirically that local minima obtained via approximate projected gradient descent often yield high-quality solutions. In many cases, our method recovers valid tours with shorter lengths than those produced by LKH-3. For instances where the relaxed optimization does not directly yield a valid tour, we introduce a simple greedy post-processing heuristic to eliminate subtours and construct feasible solutions. Finally, we apply a 2-opt local refinement to further improve tour quality. Together, these components constitute a new heuristic for solving general (asymmetric, non-metric) TSP instances that consistently outperforms LKH-3, particularly on large-scale problems. When applied to small-scale symmetric TSPs, our method remains competitive—achieving slightly lower solution quality than LKH-3 but offering significantly faster runtime performance.

In summary, our main contributions are:

- **Novel TSP formulation:** A novel exact continuous formulation that suppresses subtours via a trace-based acyclicity loss.
- **Soft assignment optimization:** A doubly stochastic matrix relaxation of ATSP solved via projected exponentiated gradient descent.
- **Efficient greedy decoding:** A post-processing algorithm that reconstructs valid tours from fractional assignments with minimal additional cost.
- **Strong empirical performance:** Extensive experiments show our method outperforms LKH-3 on ATSP benchmarks and scales well to large problem sizes.

2 Related Works

Learning-based TSP Solvers Recently, learning-based approaches have been increasingly applied to combinatorial optimization (CO) problems, including the Traveling Salesman Problem (TSP). While many existing works focus on the Euclidean TSP, where inputs are given as point coordinates [11, 14, 15, 18, 20, 23, 24, 31], more recent studies [4, 5, 16, 22] have extended these methods to handle general asymmetric TSPs. Kwon et al. [16] treat distance matrices directly as transformer input tokens, which is computationally inefficient and unsuitable for large graphs. Subsequent methods [4, 5, 22] address this limitation by encoding distance matrices more efficiently, enabling scalability to graphs with thousands of nodes. These approaches [4, 5, 16, 22] generate tours in an auto-regressive manner, formulating the tour-construction process as a Markov Decision Process (MDP). Reinforcement learning is then applied to optimize over the MDP, thereby avoiding reliance on expensive ground-truth optimal solutions.

In contrast to these auto-regressive frameworks, Min et al. [20] proposed a surrogate loss based on assignment-problem relaxations of the TSP. However, their approach is limited to the Euclidean setting and only enforces the absence of length-2 loops in the solution, necessitating a post-processing step to construct valid tours.

Learning-based solvers generally outperform classical heuristics such as LKH-3 in terms of computational efficiency, although they may produce slightly less accurate tours. In reinforcement learning

settings, high-quality solutions from fast heuristics (even if suboptimal) can serve as valuable offline data for off-policy algorithms, potentially accelerating training and improving solution quality.

Beyond methods that directly generate tours from node coordinates or distance matrices, another line of research [27, 30] focuses on learning improved search strategies—such as branch-and-bound or local search policies. These approaches replace handcrafted heuristics in traditional solvers with learned decision mechanisms, leading to more adaptive and data-driven optimization behavior.

Assignment Problem Relaxations for TSP Relaxations of the Traveling Salesman Problem (TSP) based on the classical linear Assignment Problem (AP) have been extensively studied due to their computational efficiency and strong theoretical foundations [9, 17]. These relaxations typically involve solving a linear program in which the tour constraints are relaxed, allowing the solution to contain multiple disconnected subtours. To recover a valid TSP tour, additional strategies such as iterative subtour elimination or branch-and-bound frameworks are employed [19]. Although AP relaxations do not always yield feasible tours directly, they provide tight lower bounds [6] and serve as key components in exact solvers and heuristic algorithms. Several tour-construction heuristics have also been developed based on AP relaxations [12, 13]; however, existing open-source implementations often perform worse than modern LKH-3 solvers. For a more comprehensive discussion, we refer readers to Cirasella et al. [2], Glover et al. [7], and Öncan et al. [21].

3 Exact Continuous Relaxation of TSP

Let $\mathbf{D} \in \mathbb{R}_+^{n\times n}$ be the non-negative distance (or cost) matrix, where its entry d_{ij} denotes the distance or cost of traveling from node i to node j. The general TSP seeks a tour (i.e., a permutation of the nodes forming a single cycle that visits each node exactly once) that minimizes the total travel cost. This can be formulated as:

$$\begin{aligned} \min_{\mathbf{X}\in\{0,1\}^{n\times n}} \quad & \langle \mathbf{D}, \mathbf{X}\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i \in \{1,\dots,n\} \quad \text{(each node exits exactly once)}, \\ & \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } j \in \{1,\dots,n\} \quad \text{(each node is entered exactly once)}, \\ & \mathbf{X} \text{ forms a single Hamiltonian cycle (no subtours).} \end{aligned} \tag{1}$$

Here, $\mathbf{X}$ is a permutation matrix corresponding to a single cycle over the nodes. The final constraint prohibits subtours—i.e., disjoint cycles that do not span all nodes—ensuring that the solution corresponds to a valid TSP tour.

Differentiable DAG Constraints Recently, a family of differentiable constraints for enforcing acyclicity in graphs was proposed in the literature [25, 28, 29]. For any positive adjacency matrix $\mathbf{X}$, acyclicity can be enforced by the following constraint:

$$h(\mathbf{X}) = \sum_{i=1}^{n} \operatorname{tr}(\mathbf{X}^i) = 0. \tag{2}$$

In this constraint, each term $\operatorname{tr}(\mathbf{X}^i)$ is strictly positive when $\mathbf{X}$ contains cycles of length i, since all entries of $\mathbf{X}$ are positive. Therefore, setting the total sum to zero guarantees that there are no cycles of any length, ensuring that $\mathbf{X}$ represents a Directed Acyclic Graph (DAG).

Based on these differentiable DAG constraints, we propose a novel continuous relaxation of the TSP by enforcing that any proper subgraph of the assignment matrix $\mathbf{X}$ forms a DAG. This leads to the following formulation:

$$\min_{\mathbf{X}\in\mathcal{D}^n} \langle \mathbf{D}, \mathbf{X}\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \quad \text{subject to} \quad \sum_{i=1}^{n-1} \operatorname{tr}(\mathbf{X}^i) = 0, \tag{3}$$

where $\mathcal{D}^n = \left\{ \mathbf{X} \in [0,1]^{n \times n} \,\middle|\, \mathbf{X}\mathbf{1} = \mathbf{1},\ \mathbf{1}^\top \mathbf{X} = \mathbf{1}^\top \right\}$ is the set of doubly stochastic matrices of size $n \times n$.

One appealing property of this relaxation is its tightness, as stated below:

Proposition 1 (Exactness of the Continuous Relaxation)**.** *The global minimum of the relaxed problem in* (3) *coincides with the optimal solution of the original TSP.*

Proof. To prove the result, it suffices to show that any fractional matrix $\mathbf{X} \in \mathcal{D}^n$ cannot satisfy the constraint $\sum_{i=1}^{n-1} \operatorname{tr}(\mathbf{X}^i) = 0$. In other words, all feasible solutions must be permutation matrices representing Hamiltonian cycles.

The proof follows from the Birkhoff–von Neumann theorem [10], which states that any doubly stochastic matrix $\mathbf{X} \in \mathcal{D}^n$ can be expressed as a convex combination of permutation matrices:

$$\mathbf{X} = \sum_{i=1}^{k} \theta_i \mathbf{P}_i, \quad \theta_i \geq 0,\ \sum_{i=1}^{k} \theta_i = 1, \tag{4}$$

where $\mathbf{P}_i$ are permutation matrices.

Assume for contradiction that a fractional $\mathbf{X} \in \mathcal{D}^n$ satisfies the constraint in (3). Then its Birkhoff–von Neumann decomposition contains at least two distinct permutation matrices $\mathbf{P}_i$. If any $\mathbf{P}_i$ corresponds to a subtour (i.e., not a single Hamiltonian cycle), then $\operatorname{tr}(\mathbf{P}_i^j) > 0$ for some $j < n$, and hence $\operatorname{tr}(\mathbf{X}^j) > 0$ for some $j < n$ due to linearity and positivity. This violates the constraint $\sum_{j=1}^{n-1} \operatorname{tr}(\mathbf{X}^j) = 0$, making $\mathbf{X}$ infeasible.

Even if all $\mathbf{P}_i$ represent valid Hamiltonian cycles, a convex combination of two or more distinct tours will produce fractional entries in $\mathbf{X}$. These fractional values imply the presence of edges from multiple tours, which results the presence of subtours and nonzero traces in $\mathbf{X}^j$ for some $j < n$. Thus, $\mathbf{X}$ would again be infeasible.

Therefore, the only feasible solutions are permutation matrices representing single Hamiltonian cycles. Among them, the one minimizing $\langle \mathbf{D}, \mathbf{X} \rangle$ is exactly the TSP solution.

□

4 Trace Guided Tour Search

To efficiently optimize the relaxation in Equation (3), we introduce an auxiliary variable $\mathbf{Y} \in \mathbb{R}^{n \times n}$ with elementwise constraints $\mathbf{Y} \leqslant 0$, and define the assignment matrix as $\mathbf{X} = \exp(\mathbf{Y})$, where the exponential is applied elementwise. This reparameterization ensures that $\mathbf{X}$ is strictly positive and facilitates optimization over the interior of the doubly stochastic polytope $\mathcal{D}^n$. Compared to optimizing directly over $\mathbf{X}$, working in the log-domain (i.e., optimizing $\mathbf{Y}$) avoids boundary issues, improves numerical stability, and enables smooth gradients that are more amenable to gradient-based learning methods. The relaxed optimization problem is then formulated as:

$$\min_{\exp(Y) \in \mathcal{D}^n} \langle \mathbf{D}, \exp(\mathbf{Y}) \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \exp(y_{ij}) \text{ subject to } \operatorname{tr} \sum_{i=1}^{n-1} (\exp(\mathbf{Y})^i) = 0, \tag{5}$$

where $\mathbf{D}$ is the cost matrix, and the trace constraint eliminates self-loops and short cycles. By introducing a Lagrange multiplier λ, the constrained problem can be rewritten as the following unconstrained objective:

$$\min_{\exp(\mathbf{Y}) \in \mathcal{D}^n} f(\mathbf{Y}) = \langle \mathbf{D}, \exp(\mathbf{Y}) \rangle + \lambda \operatorname{tr} \sum_{i=1}^{n-1} (\exp(\mathbf{Y})^i). \tag{6}$$

The gradient of this objective with respect to $\mathbf{Y}$ is given by:

$$\nabla_{\mathbf{Y}} f(\mathbf{Y}) = \mathbf{D} \odot \exp(\mathbf{Y}) + \lambda \left[\sum_{i=0}^{n-2} (i+1) \exp(\mathbf{Y})^i \right]^\top \odot \exp(\mathbf{Y}), \tag{7}$$

where $\odot$ denotes elementwise multiplication. This formulation enables efficient and scalable optimization while preserving the structural constraints required by the TSP.

Algorithm 1 Approximate Projected Gradient Descent for Exponentiated TSP Relaxation

Input: Cost matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$, initial step size $\alpha > 0$, scaling factor $\gamma > 0$, max iterations T
Initialize $\mathbf{Y} \leftarrow -\mathbf{D}$
Compute initial assignment $\pi_{\text{prev}} \leftarrow \text{LinearAssignment}(-\mathbf{Y})$
for $t = 1$ to T **do**
    Compute gradient:

$$\nabla_{\mathbf{Y}} f(\mathbf{Y}) = \mathbf{D} \odot \exp(\mathbf{Y}) + \lambda \left[\sum_{i=0}^{n-2} (i+1) \exp(\mathbf{Y})^i \right]^\top \odot \exp(\mathbf{Y})$$

    Update $\mathbf{Y}_{\text{new}} \leftarrow \mathbf{Y} - \alpha \nabla_{\mathbf{Y}} f(\mathbf{Y})$
    Project: $\mathbf{X} = \textbf{Sinkhorn}(\exp(\mathbf{Y}_{\text{new}}))$, $\mathbf{Y}_{\text{new}} \leftarrow \log(\mathbf{X})$ ▷ elementwise projection onto $\mathbf{D}^n$
    Compute new assignment $\pi_{\text{new}} \leftarrow \text{LinearAssignment}(-\mathbf{Y}_{\text{new}})$
    if $\pi_{\text{new}} \neq \pi_{\text{prev}}$ **then**
        $\alpha \leftarrow \alpha / 2$
    else
        $\alpha \leftarrow 2\alpha$
    end if
    $\mathbf{Y} \leftarrow \mathbf{Y}_{\text{new}}$, $\pi_{\text{prev}} \leftarrow \pi_{\text{new}}$
    if π_{new} has only one loop **then**
        break
    end if
end for
Output: $\mathbf{Y}$, permutation π_{new}

Approximate Projected Gradient Descent for Exponentiated TSP Relaxation: To optimize the exponentiated relaxation of the TSP problem, we propose an approximate projected gradient descent algorithm over the log-domain variable $\mathbf{Y}$. The algorithm initializes $\mathbf{Y}$ as the negative of the cost matrix, $\mathbf{Y} \leftarrow -\mathbf{D}$, and iteratively updates it using the gradient of the relaxed objective. At each step, the gradient is computed and used to update $\mathbf{Y}$ with a step size α. To maintain feasibility within the doubly stochastic set, we apply Sinkhorn [3] normalization to the exponentiated matrix $\exp(\mathbf{Y})$ and then project back to the log-domain via $\log(\cdot)$, ensuring that $\exp(\mathbf{Y}) \in \mathcal{D}^n$. The step size α is dynamically adjusted based on whether the solution to the linear assignment problem (using $-\mathbf{Y}$ as the cost) changes after the update: if it does, the step size is reduced; otherwise, it is increased to accelerate convergence. This adaptive strategy helps balance exploration and stability. The process terminates either after a fixed number of iterations or once a valid TSP tour (a single loop) is found in the assignment. The algorithm returns the optimized $\mathbf{Y}$ and the corresponding assignment matrix $\exp(\mathbf{Y})$.

Approximate Gradient Computation: In our gradient expression, the term $\left[\sum_{i=0}^{n-2}(i+1)\exp(\mathbf{Y})^i\right]^\top$ involves computing a weighted sum of matrix powers, which can be computationally expensive. Although efficient approximations such as the truncated Neumann series proposed in Zhang et al. [28] can reduce the number of operations to $O(\log n)$ matrix multiplications, the cost remains significant for large instances due to the $O(n^3)$ complexity of matrix multiplication. To address this, we adopt a lightweight approximation by substituting $\exp(\mathbf{Y})$ with the hard assignment matrix corresponding to the current solution π_{prev}. Since permutation matrices are sparse and binary, their powers are simply cyclic shifts, and the sum can be computed in $O(n^2)$ time. This approximation significantly accelerates training while preserving useful structural signals from the current assignment during optimization.

Greedy Tour Decoding from Invalid Assignments: Although the proposed optimization approach typically converges rapidly for asymmetric TSP instances, there exist hard cases where convergence to a valid permutation (i.e., a single-loop tour) does not occur. To address such situations, we apply a greedy decoding strategy to extract a valid TSP tour from the relaxed assignment matrix. Specifically, we begin by selecting a random starting node and iteratively follow the highest-assigned edge (i.e., the node with the largest assignment probability) to determine the next node in the tour. If the selected

Method	ATSP20			ATSP50		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	1.68	0.00%	0.00%	5.18
LKH-3 (1, 500)	0.02%	0.00%	0.02	1.00%	0.40%	0.060
LKH-3 (1, 10000)	0.0005%	0.00%	0.18	0.28%	0.00%	0.54
MatPOENet	27.36%	27.80%	0.0067	688.67%	675.04%	0.016
GOAL	36.50%	33.59%	0.002	108.29%	109.38%	0.0095
Ours	1.9%	0.08%	0.005	2.6%	0.8%	0.012

Table 1: Comparison of methods on small-scale pure random ATSP (TSP20 and TSP50). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. The performance of learning based approaches is far worse than their reported results as they are trained on ATSP with triangle inequality and they failed to generalize to random ATSP.

Method	ATSP20			ATSP50		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	2.54	0.00%	0.00%	4.51
LKH-3 (1, 500)	0.00%	0.00%	0.03	0.06%	0.00%	0.05
LKH-3 (1, 10000)	0.00%	0.00%	0.27	0.004%	0.00%	0.48
MatPOENet	7.4%	5.3%	0.0063	1.4%	1.1%	0.016
GOAL	9.26%	10.15%	0.002	5.87%	5.83%	0.0095
Ours	0.97%	0.00%	0.006	0.6%	0.00%	0.009

Table 2: Comparison of methods on small-scale random ATSP with triangle inequality problems (TSP20 and TSP50). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances.

edge would result in the formation of a premature loop or revisiting a node, we instead choose the closest unvisited node according to the original cost matrix. This process continues until all nodes have been visited exactly once, ensuring the construction of a valid Hamiltonian cycle. This decoding heuristic is simple, efficient, and empirically effective in recovering feasible tours from imperfect or fractional assignments.

Time Complexity. The dominant computational cost in each iteration of the proposed algorithm lies in solving the linear assignment problem for approximating $\exp(\mathbf{Y})$, which has a worst-case time complexity of $O(n^3)$ when using the Hungarian algorithm. However, in practice, the Hungarian algorithm is typically much faster due to early termination and efficient implementations. Additionally, viewing the assignment step through the lens of primal-dual linear programming reveals an opportunity to reuse the dual variables as warm starts for subsequent iterations, potentially accelerating convergence. Alternatively, the assignment problem can also be solved using the Auction algorithm, which offers better parallelism and is more amenable to GPU implementation. This opens up promising directions for future work, where the entire optimization process could be ported to GPUs to achieve significant speed-ups and scalability for large-scale TSP instances.

Local Search Refinement. To further improve the quality of the obtained tour, our implementation applies a simple 2-opt heuristic as a post-processing step. We observe that for asymmetric TSP instances, the 2-opt heuristic often brings negligible improvements due to the directional nature of the cost structure. In contrast, for symmetric TSPs, the 2-opt refinement can lead to noticeable performance gains by eliminating crossing edges and reducing the overall tour length. While 2-opt is lightweight and easy to implement, it may be suboptimal for larger or more complex instances. As a future direction, one could consider replacing the 2-opt heuristic with more powerful local search methods such as the Lin–Kernighan–Helsgaun (LKH) algorithm [8]. In such a setup, our learned solution could serve as a high-quality initialization for LKH, potentially reducing the number of local search iterations required and improving overall efficiency.

5 Experimental Results

Hardware All experiments were conducted on a workstation equipped with an AMD Ryzen 9 3900X CPU, 128GB DDR4 memory, and an NVIDIA RTX 3090 GPU. The LKH-3 heuristic was implemented in C++ and executed in a single-threaded mode on the CPU. Our proposed method was implemented using Python and SciPy, and it was also executed on a single CPU process without utilizing the GPU. In contrast, all learning-based approaches (e.g., GOAL and MatPOENet) were executed on the GPU to leverage hardware acceleration during inference.

Dataset We evaluate our method on randomly generated Asymmetric TSP (ATSP) instances of sizes 20, 50, 100, 200, 500, and 1000. Following prior works [16], the cost matrices for these instances are generated by sampling each entry independently from a uniform distribution over the interval $[0, 1]$. Kwon et al. [16] add an post-processing procedure to ensure the triangle inequality in enforced in the generated ATSP, while we also consider the cases where the triangle inequality is not satisfied. For solvers that require integer-valued cost matrices, we scale the costs by 10^6 and round them to the nearest integer to preserve precision. Additionally, we evaluate our approach on symmetric TSP instances of sizes 20 and 50. In line with previous settings, we first sample n random points in 2D space from a uniform distribution over $[0, 1]^2$, and use the Euclidean distances between these points to construct the symmetric cost matrices.

Competitors We compare our method against several strong baselines, including both classical heuristics and recent learning-based approaches. Specifically, we consider **LKH-3**, a state-of-the-art heuristic algorithm for solving TSP, **GOAL**[5], a reinforcement learning-based approach using graph neural networks, and **MatPOENet**[22], a transformer-based model for solving combinatorial optimization problems. For GOAL and MatPOENet, we use their publicly released pretrained weights to ensure a fair comparison. For LKH-3, we evaluate three configurations: (i) one run with 500 search trials, (ii) one run with 10,000 search trials, and (iii) ten runs with 10,000 search trials each. These configurations allow us to assess both the performance and robustness of LKH-3 under varying computational budgets. Exact solvers such as Gurobi and Concorde [1] are not compared as they are often too slow for large scale problems.

Method	**ATSP100**			**ATSP200**		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	11.40	0.00%	0.00%	18.67
LKH-3 (1, 500)	5.38%	5.06%	0.13	12.59%	12.72%	0.27
LKH-3 (1, 10000)	1.06%	0.91%	1.19	2.02%	1.90%	2.03
MatPOENet	295.18%	295.62%	0.047	747.56%	750.27%	0.165
GOAL	181.92%	177.47%	0.058	265.03%	261.43%	0.16
Ours	3.39%	1.77%	0.043	0.32%	-1.01%	0.13

Table 3: Comparison of methods on medium-scale pure random ATSP problems (TSP100 and TSP200). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. The performance of learning based approaches is far worse than their reported results as they are trained on ATSP with triangle inequality and they failed to generalize to random ATSP.

5.1 Asymmetric Problems

Small Scale Problems The comparison of running time and solution quality for small-scale problems is presented in Table 1 and Table 2. Our method, while yielding slightly worse average performance than LKH-3, demonstrates significantly faster running times. For the learning-based approaches, which are trained on ATSP witht triangle inequality instances, their performance degrades substantially on purely random ATSP problems—underperforming all other competitors. This indicates that current neural combinatorial solvers may struggle to generalize across different problem distributions. Additionally, MatPOENet performs considerably worse on our test set than reported in its original paper, suggesting a potential distribution shift between the two test scenarios.

Surprisingly, GOAL achieves the best performance on asymmetric TSP20 with triangle inequality, outperforming the state-of-the-art LKH-3 heuristic by a substantial margin. Given that GOAL was

Method	ATSP100			ATSP200		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	7.31	0.00%	0.00%	11.69
LKH-3 (1, 500)	0.25%	0.11%	0.11	0.64%	0.06%	0.23
LKH-3 (1, 10000)	0.09%	0.00%	0.787	0.18%	0.01%	1.34
MatPOENet	14.91%	13.84%	0.047	35.58%	35.72%	0.165
GOAL	3.83%	4.04%	0.058	0.77%	0.74%	0.44
Ours	0.52%	0.04%	0.027	0.16%	0.00%	0.10

Table 4: Comparison of methods on medium-scale random ATSP with triangle inequality problems (TSP100 and TSP200). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. There may potentially be a distribution shift between the training set of MatPOENet and our test set.

Method	ATSP500			ATSP1000		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	29.99	0.00%	0.00%	46.64
LKH-3 (1, 500)	24.47%	24.86%	1.08	51.23%	51.48%	3.81
LKH-3 (1, 10000)	2.50%	2.45%	3.85	3.25%	2.94%	7.91
MatPOENet	6400.51%	6400.50%	1.51	15017.16%	15016.70%	10.24
GOAL	369.69%	369.40%	6.84	305.84%	304.62%	58.53
Ours	**-10.77 %**	**-11.59%**	0.87	**-21.40%**	**-21.86%**	5.22

Table 5: Comparison of methods on large-scale pure random ATSP problems (TSP500 and TSP1000). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. Our approaches outperforms the state-of-the-arts LKH-3 heuristics in terms of solution quality. The performance of learning based approaches is far worse than their reported results as they are trained on ATSP with triangle inequality and they failed to generalize to random ATSP.

trained on 100-node triangle asymmetric instances, this result suggests that training neural solvers on large, structured, and challenging problems may enhance their generalization to smaller problem instances.

Medium and Large Scale Problems The comparison of running time and solution quality for medium- and large-scale problems is presented in Table 3, Table 4, Table 5, and Table 6. Consistent with the small-scale setting, learning-based approaches perform significantly worse than others on purely random asymmetric TSPs. In these settings, our algorithm outperforms LKH-3(1,500) for graphs with more than 100 nodes, and even surpasses LKH-3(1,10000) for graphs with more than 200 nodes. As the graph size increases, our method increasingly outperforms all LKH-3 heuristics, with the performance gap becoming more pronounced for larger instances. On 1000-node problems, our algorithm achieves over **20%** improvement compared to LKH-3(10,10000) and over **70%** compared to LKH-3(1,500). Meanwhile, our runtime on large graphs is comparable to that of LKH-3(1,500) and remains faster than the learning-based approaches. On asymmetric TSPs with triangle inequality, similar trends are observed. Our method continues to outperform LKH-3 on large graphs, although the performance margin is smaller compared to the purely random setting.

In terms of running time, it is notable that, despite our implementation not being carefully optimized, its execution speed is often faster than that of LKH-3. On very large graphs, our method can outperform learning-based approaches, even when those methods are accelerated by GPUs. Furthermore, most operations in our algorithm are amenable to parallel execution and could potentially benefit significantly from GPU acceleration.

5.2 Symmetric Problems

Although our algorithm is not specifically designed for symmetric TSPs and does not exploit any symmetry properties, we evaluated its performance on small-scale symmetric instances. The results are presented in Table 7. The performance trends closely resemble those observed on triangle asymmetric TSPs.

Method	ATSP500			ATSP1000		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	20.88	0.00%	0.00%	36.72
LKH-3 (1, 500)	1.08%	1.00%	1.02	51.23%	51.48%	3.93
LKH-3 (1, 10000)	0.25%	0.24%	2.93	3.25%	2.94%	7.05
MatPOENet	112.48%	112.19%	1.50	156.48%	156.24%	10.21
GOAL	1.07%	1.01%	6.85	1.26%	1.22%	58.48
Ours	**-0.06 %**	**-0.11%**	0.74	**-0.35%**	**-0.36%**	5.10

Table 6: Comparison of methods on large-scale random ATSP with triangle inequality problems (TSP500 and TSP1000). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. Our approaches outperforms the state-of-the-arts LKH-3 heuristics in terms of solution quality.

Method	TSP20			TSP50		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	3.21	0.00%	0.00%	4.51
LKH-3 (1, 500)	0.00%	0.00%	0.03	0.06%	0.00%	0.05
LKH-3 (1, 10000)	0.00%	0.00%	0.33	0.004%	0.00%	0.48
MatPOENet	1.50%	1.25%	0.0064	0.63%	0.58%	0.015
GOAL	14.06%	15.14%	0.002	14.34%	15.18%	0.0095
Ours	0.04%	0.00%	0.013	0.86%	0.66%	0.094

Table 7: Comparison of methods on small-scale random symmetric TSP problems (TSP20 and TSP50). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances.

6 Conclusions and Future Works

In this work, we introduced a novel continuous relaxation framework for the Asymmetric Traveling Salesman Problem (ATSP) that combines differentiable acyclicity constraints with doubly stochastic matrix relaxations. Our key innovation lies in the formulation of ATSP as a constrained optimization problem where trace-based DAG constraints naturally eliminate subtours while maintaining differentiability for gradient-based optimization. The proposed projected exponentiated gradient method with adaptive step sizes demonstrates efficient convergence properties, and our greedy post-processing procedure effectively recovers valid tours from fractional assignments.

Through extensive experiments on standard benchmarks, we have shown that our method achieves state-of-the-art performance on asymmetric TSP instances, particularly excelling on large-scale problems where traditional heuristics like LKH-3 struggle. The results reveal several important insights: (1) our continuous relaxation provides a tight formulation that frequently converges to valid tours without post-processing, (2) the approach scales favorably with problem size, maintaining both computational efficiency and solution quality, and (3) the method remains competitive even on symmetric TSP instances despite not being specifically designed for them.

The success of our framework suggests several promising directions for future research. First, the differentiable nature of our formulation makes it potentially amenable to integration with learning-based approaches, where the solver could be trained to predict good initializations or adaptive penalty parameters. Second, the trace-based constraints may generalize to other combinatorial optimization problems that require acyclic substructures. Finally, investigating more sophisticated optimization techniques within our framework could lead to further improvements in both solution quality and computational efficiency.

This work bridges an important gap between continuous relaxations and discrete combinatorial optimization, offering a new perspective on solving challenging problems like ATSP through differentiable programming. The demonstrated effectiveness of our approach on large-scale instances suggests that similar techniques could prove valuable for other NP-hard optimization problems where traditional methods face scalability limitations.

Acknowledgments and Disclosure of Funding

This research is supported by Responsible AI Research Center (RAIR) and the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (A-8001814-00-00).

References

[1] David L Applegate, Robert E Bixby, Vašek Chvátal, William Cook, Daniel G Espinoza, Marcos Goycoolea, and Keld Helsgaun. Certification of an optimal tsp tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15, 2009.

[2] Jill Cirasella, David S Johnson, Lyle A McGeoch, and Weixiong Zhang. The asymmetric traveling salesman problem: Algorithms, instance generators, and tests. In *Workshop on algorithm engineering and experimentation*, pages 32–59. Springer, 2001.

[3] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[4] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:77416–77429, 2023.

[5] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. In *The Thirteenth International Conference on Learning Representations*, 2024.

[6] Alan Frieze and Gregory B Sorkin. The probabilistic relationship between the assignment and asymmetric traveling salesman problems. *SIAM Journal on Computing*, 36(5):1435–1452, 2007.

[7] Fred Glover, Gregory Gutin, Anders Yeo, and Alexey Zverovich. Construction heuristics for the asymmetric tsp. *European Journal of Operational Research*, 129(3):555–568, 2001.

[8] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980, 2017.

[9] Karla L Hoffman, Manfred Padberg, Giovanni Rinaldi, et al. Traveling salesman problem. *Encyclopedia of operations research and management science*, 1:1573–1578, 2013.

[10] GLENN Hurlbert. A short proof of the birkhoff-von neumann theorem. *preprint (unpublished)*, 2008.

[11] 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.

[12] Richard M Karp. A patching algorithm for the nonsymmetric traveling-salesman problem. *SIAM Journal on Computing*, 8(4):561–573, 1979.

[13] Richard M Karp and J Michael Steele. Probabilistic analysis of heuristics. *The traveling salesman problem*, pages 181–205, 1985.

[14] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[15] 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.

[16] 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.

[17] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.

[18] Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:50020–50040, 2023.

[19] Donald L Miller and Joseph F Pekny. Exact solution of large asymmetric traveling salesman problems. *Science*, 251(4995):754–761, 1991.

[20] 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.

[21] Temel Öncan, I Kuban Altınel, and Gilbert Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

[22] 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.

[23] 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.

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

[25] Dennis Wei, Tian Gao, and Yue Yu. Dags with No Fears: A closer look at continuous optimization for learning bayesian networks. *Advances in Neural Information Processing Systems*, 33:3895–3906, 2020.

[26] 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. *arXiv preprint arXiv:2406.03503*, 2024.

[27] Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20284–20292, 2024.

[28] Zhen Zhang, Ignavier Ng, Dong Gong, Yuhang Liu, Ehsan Abbasnejad, Mingming Gong, Kun Zhang, and Javen Qinfeng Shi. Truncated matrix power iteration for differentiable dag learning. *Advances in Neural Information Processing Systems*, 35:18390–18402, 2022.

[29] Zhen Zhang, Ignavier Ng, Dong Gong, Yuhang Liu, Mingming Gong, Biwei Huang, Kun Zhang, Anton van den Hengel, and Javen Qinfeng Shi. Analytic dag constraints for differentiable dag learning. *arXiv preprint arXiv:2503.19218*, 2025.

[30] 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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 12445–12452, 2021.

[31] Zhi Zheng, Changliang Zhou, Tong Xialiang, Mingxuan Yuan, and Zhenkun Wang. Udc: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. *arXiv preprint arXiv:2407.00312*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Justification: The main contributions of our work are emphasized throughout the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We clearly stated that our approaches are for asymmetric TSPs and may not have good performance in symmetric cases.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We proved that our continuous relaxation is exact.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The pseudo code of the algorithm is given, and an implementation based on that would work.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code will be made publication after publication.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The construction of the dataset is clear stated in the experimental section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We not only provide average results, but also provide median results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: It is clearly stated in the experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not have such problem.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party assets used in this paper—such as code, datasets, and models—are properly credited, with licenses and terms of use explicitly acknowledged.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The dataset generation process is clearly stated.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not have any human experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not have any human experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix: Solving Asymmetric Traveling Salesman Problem via Trace-Guided Cost Augmentation

A Linear Assignment Problem

The linear assignment problem aims to determine a permutation matrix that minimizes the total assignment cost given by a cost matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$, whose entries are denoted by d_{ij}. Let $\mathbf{X} \in \mathbb{R}^{n \times n}$ be the assignment matrix, where $x_{ij} = 1$ indicates that row i is assigned to column j, and zero otherwise. The problem can be formulated as the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i = 1, \ldots, n, \\ & \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } j = 1, \ldots, n, \\ & x_{ij} \geq 0 \quad \text{for all } i, j. \end{aligned} \tag{8}$$

The dual problem introduces variables u_i and v_j corresponding to the row and column constraints, respectively, and is given by:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \\ \text{subject to} \quad & u_i + v_j \leq d_{ij} \quad \text{for all } i, j. \end{aligned} \tag{9}$$

Strong duality holds for this problem, guaranteeing that the optimal values of the primal and dual problems are equal when a feasible solution exists.

The Hungarian Algorithm is, in fact, a primal-dual method for solving the Linear Assignment Problem (LAP). This insight allows us to exploit the structure of Algorithm 1, where multiple LAPs are solved iteratively. Specifically, we can retain the previously optimal dual variables $\mathbf{u}$ and $\mathbf{v}$ and use them as the initialization for subsequent runs of the Hungarian Algorithm. This warm-start strategy significantly accelerates convergence, as the optimal dual variables often remain close across successive LAP instances.

B Further Refining Performance by Branch-and-Bound

We further refined our implementation by incorporating two key strategies: (1) reusing the dual variables $\mathbf{u}$ and $\mathbf{v}$, as described in the previous section, and (2) introducing a simple branch-and-bound heuristic. In our branch-and-bound strategy, we first solve the Linear Assignment Problem (LAP) to obtain a lower bound for the ATSP instance. We then identify the edge with the highest cost in the LAP solution and branch on it. For each branch, we apply Algorithm 1 to compute a potentially global upper bound, while the LAP provides a local lower bound. Our heuristic allows up to five branch-and-bound iterations.

With these enhancements, the results on large-scale ATSP instances satisfying the triangle inequality are presented in Table 8. As shown, reusing the dual variables accelerates our algorithm by up to a factor of five compared to the original implementation. Additionally, the branch-and-bound strategy offers further performance improvements. Similar gains are observed on purely random ATSP instances, as shown in Table 9.

Method	TSP500			TSP1000		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	20.88	0.00%	0.00%	36.72
LKH-3 (1, 500)	1.08%	1.00%	1.02	51.23%	51.48%	3.93
LKH-3 (1, 10000)	0.25%	0.24%	2.93	3.25%	2.94%	7.05
MatPOENet	112.48%	112.19%	1.50	156.48%	156.24%	10.21
GOAL	1.07%	1.01%	6.85	1.26%	1.22%	58.48
Ours	-0.06 %	-0.11%	0.74	-0.35%	-0.36%	5.10
Ours-reuse$[\mathbf{u}, \mathbf{v}]$	-0.04%	-0.09%	0.19	-0.35%	-0.36%	0.97
Ours-reuse$[\mathbf{u}, \mathbf{v}]$-BnB	**-0.11%**	**-0.12%**	0.70	**-0.38%**	**-0.38%**	2.91

Table 8: Comparison of methods on large-scale random ATSP with triangle inequality problems (TSP500 and TSP1000). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. Our approaches outperforms the state-of-the-arts LKH-3 heuristics in terms of solution quality.

Method	ATSP500			ATSP1000		
	Avg. GAP	Median GAP	Runtime (s)	Avg. GAP	Median GAP	Runtime (s)
LKH-3 (10, 10000)	0.00%	0.00%	29.99	0.00%	0.00%	46.64
LKH-3 (1, 500)	24.47%	24.86%	1.08	51.23%	51.48%	3.81
LKH-3 (1, 10000)	2.50%	2.45%	3.85	3.25%	2.94%	7.91
MatPOENet	6400.51%	6400.50%	1.51	15017.16%	15016.70%	10.24
GOAL	369.69%	369.40%	6.84	305.84%	304.62%	58.53
Ours	-10.77 %	-11.59%	0.87	-21.40%	-21.86%	5.22
Ours-reuse$[\mathbf{u}, \mathbf{v}]$	-10.89%	-11.67%	0.33	-21.49%	-21.98%	1.54
Ours-reuse$[\mathbf{u}, \mathbf{v}]$-BnB	**-11.88%**	**-12.02%**	0.84	**22.31%**	**-22.39%**	3.63

Table 9: Comparison of methods on large-scale pure random ATSP problems (TSP500 and TSP1000). GAP is measured relative to LKH-3 (10, 10000). The results are average from 128 instances. Our approaches outperforms the state-of-the-arts LKH-3 heuristics in terms of solution quality. The performance of learning based approaches is far worse than their reported results as they are trained on ATSP with triangle inequality and they failed to generalize to random ATSP.

C Experimental Results on TSPLIB instances

We evaluated our algorithm on large-scale ATSP benchmark instances from TSPLIB, particularly those derived from stacker crane problems. Our method consistently outperforms LKH-3 (1–100) in terms of solution quality while also requiring less computational time. These results demonstrate the practical effectiveness of our approach on real-world structured benchmarks, further validating its scalability and competitiveness beyond synthetic instances.

D Comparison with ILP Solver

We conducted a preliminary comparison between our method and an exact ILP solver on both symmetric and asymmetric TSP instances. Specifically, we tested the solver on 100-node symmetric problems and 500-node asymmetric problems and found that the ILP solver was able to return optimal solutions within a reasonable time budget.

Due to time constraints, we evaluated both methods on 10 randomly sampled instances for each case. The results are summarized below. While the ILP solver guarantees optimality, our method achieves competitive solution quality with significantly faster runtimes, particularly for larger asymmetric instances—highlighting its scalability and practical utility for large-scale problems.

D.1 Performance comparison on 100-node symmetric TSP problems

D.2 Performance comparison on 500-node asymmetric TSP problems

Table 10: Comparison of algorithms on asymmetric TSP benchmark instances.

Instance	Algorithm	Best Known Result	Running Time (s)	Result
rbg323	LKH(1–100)	1326	1.46	1388
	LKH(1–10000)		3.38	1346
	LKH(10–10000)		20.32	1346
	Ours		0.42	1360
	Ours (no 2-opt)		0.20	1365
rbg358	LKH(1–100)	1163	1.67	1294
	LKH(1–10000)		4.03	1175
	LKH(10–10000)		25.16	1175
	Ours		0.09	1180
	Ours (no 2-opt)		0.04	1180
rbg403	LKH(1–100)	2465	6.27	2536
	LKH(1–10000)		26.99	2498
	LKH(10–10000)		9.06	2498
	Ours		1.53	2473
	Ours (no 2-opt)		1.28	2473
rbg443	LKH(1–100)	2720	5.44	2813
	LKH(1–10000)		7.87	2762
	LKH(10–10000)		30.06	2756
	Ours		0.52	2760
	Ours (no 2-opt)		0.27	2760

Table 11: Performance comparison on 100-node symmetric TSP problems.

Algorithm	Running Time (s)	Ratio to Optimal (Mean)	Ratio to Optimal (Median)
LKH(1–100)	0.07	0.61%	0.17%
LKH(1–10000)	1.24	0.12%	0.00%
LKH(10–10000)	12.14	0.00%	0.00%
Gurobi	2.06	0.00%	0.00%
Ours (no local search)	0.03	15.2%	14.4%
Ours (local search)	0.05	1.47%	1.35%

Table 12: Performance comparison on 500-node asymmetric TSP problems.

Algorithm	Running Time (s)	Ratio to Optimal (Mean)	Ratio to Optimal (Median)
LKH(1–100)	0.87	2.22%	0.17%
LKH(1–10000)	2.82	0.45%	0.00%
LKH(10–10000)	20.23	0.18%	0.00%
Gurobi	6.36	0.00%	0.00%
Ours (no local search)	0.17	0.005%	0.001%
Ours (local search)	0.30	0.005%	0.001%