Structure As Search: Unsupervised Permutation Learning for Combinatorial Optimization

Yimeng Min
Department of Computer Science
Cornell University
Ithaca, NY, USA
min@cs.cornell.edu

Carla P. Gomes
Department of Computer Science
Cornell University
Ithaca, NY, USA
gomes@cs.cornell.edu

Abstract

We propose a non-autoregressive framework for the Travelling Salesman Problem where solutions emerge directly from learned permutations, without requiring explicit search. By applying a similarity transformation to Hamiltonian cycles, the model learns to approximate permutation matrices via continuous relaxations. Our unsupervised approach achieves competitive performance against classical heuristics, demonstrating that the inherent structure of the problem can effectively guide combinatorial optimization without sequential decision-making.

1 Introduction

The Travelling Salesman Problem (TSP) is a classic NP-hard problem: given a set of cities and pairwise distances, the goal is to find the shortest tour visiting each city once and returning to the start. Exact solvers like Concorde [Applegate et al., 2003] compute optimal solutions for moderate-sized instances but become computationally expensive as problem size grows. For larger instances, heuristics such as Lin–Kernighan–Helsgaun (LKH) [Lin and Kernighan, 1973, Helsgaun, 2000] remain effective, leveraging local search. While traditional methods rely on handcrafted rules, neural approaches aim to learn solutions from data. Early work like the Hopfield–Tank model [Hopfield and Tank, 1985] framed TSP as neural energy minimization but lacked scalability. Modern learning-based methods fall into two categories: reinforcement learning (RL) [Khalil et al., 2017, Deudon et al., 2018, Kool et al., 2018], which constructs tours via learned policies, and supervised learning (SL) [Joshi et al., 2019], where networks generate local preferences followed by search. RL suffers from sparse rewards and high variance, while SL requires costly ground-truth solutions.

In both RL and SL methods, some form of search is involved—via learned policies or explicit heuristics. Most RL models are also autoregressive, generating tours sequentially. But many combinatorial problems, including TSP, have rich structure: the search for the shortest Hamiltonian cycle naturally constrains the solution space. Recent work [Min et al., 2023] shows that this structure can be exploited using unsupervised, non-autoregressive (NAR) learning. This raises a key question: Can we learn good TSP solutions without supervision, search, or autoregressive decoding?

In this work, we propose a different perspective on learning combinatorial structures. Rather than treating structure as the output of a post-hoc search, we explore the idea that *structural inductive bias can replace explicit search*. Building on the unsupervised learning for TSP (UTSP) framework [Min et al., 2023, Min and Gomes, 2023], we formulate the TSP as a permutation learning problem: the model directly generates a Hamiltonian cycle using a permutation matrix. Our fully unsupervised, non-autoregressive method requires no optimal training data, search or rollouts. Instead, we train the model using a Gumbel-Sinkhorn relaxation of permutation matrices, followed by hard decoding via the Hungarian algorithm at inference time. This enables solutions to emerge directly from learned structure alone.

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

We demonstrate that our model consistently outperforms classical baselines, including the nearest neighbor algorithm and farthest insertion, across a range of instance sizes. The structural inductive bias encoded in our model alone generates high-quality solutions without explicit search procedures. Our findings suggest that in combinatorial optimization, appropriately designed structural constraints can serve as effective computational mechanisms, offering a complementary paradigm to conventional search approaches. This indicates that structure itself may be sufficient to guide optimization in certain combinatorial problems.

2 Background: Unsupervised Learning for the TSP

The TSP seeks the shortest tour visiting each of n cities (with coordinates $x \in \mathbb{R}^{n\times 2}$) exactly once and returning to the start. It finds a permutation $\sigma \in S_n$ minimizing: $\min_{\sigma\in S_n} \sum_{i=1}^{n} \|x_{\sigma(i)} - x_{\sigma(i+1)}\|_2$, with $\sigma(n+1) := \sigma(1)$. To enable optimization over Hamiltonian cycles using neural networks, we first introduce a matrix-based representation of permutations. We begin by defining the cyclic shift matrix $\mathbb{V} \in \{0,1\}^{n\times n}$ for $n \geq 3$ as

$$\mathbb{V}_{i,j} = \begin{cases} 1 & \text{if } j \equiv (i+1) \pmod{n} \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

for $i, j \in \{0, 1, \ldots, n-1\}$. This matrix has the explicit form:

$$\mathbb{V} = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{2}$$

The matrix $\mathbb{V}$ represents the canonical Hamiltonian cycle $1 \to 2 \to 3 \to \cdots \to n \to 1$, where each row has exactly one entry equal to unity, indicating the next city in the sequence. More generally, a matrix $\mathcal{H} \in \{0,1\}^{n\times n}$ represents a Hamiltonian cycle if it is a permutation matrix whose corresponding directed graph forms a single cycle of length n.

The key insight is that any Hamiltonian cycle can be generated from the canonical cycle $\mathbb{V}$ through a similarity transformation [Min and Gomes, 2023]. Specifically, if $\mathbf{P} \in S_n$ is a permutation matrix, then $\mathbf{P}\mathbb{V}\mathbf{P}^\top$ represents a Hamiltonian cycle obtained by reordering the nodes according to permutation $\mathbf{P}$. Given a distance matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$, where $\mathbf{D}_{ij}$ represents the distance between cities i and j, the TSP objective becomes finding the optimal permutation matrix that minimizes:

$$\min_{\mathbf{P}\in S_n} \langle \mathbf{D}, \mathbf{P}\mathbb{V}\mathbf{P}^\top \rangle, \tag{3}$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \text{tr}(\mathbf{A}^\top \mathbf{B})$ denotes the matrix inner product. The inner product $\langle \mathbf{D}, \mathbf{P}\mathbb{V}\mathbf{P}^\top \rangle$ is the distance of the Hamiltonian cycle represented by $\mathbf{P}\mathbb{V}\mathbf{P}^\top$.

Since finding the optimal discrete permutation matrix is NP-hard and backpropagating through discrete variables is non-differentiable, we relax the problem by replacing the hard permutation matrix $\mathbf{P}$ with a soft permutation matrix $\mathbb{T} \in \mathbb{R}^{n\times n}$. Following [Min et al., 2023, Min and Gomes, 2023], we use a Graph Neural Network (GNN) to construct $\mathbb{T}$ and optimize the loss function:

$$\mathcal{L}_{\text{TSP}} = \langle \mathbf{D}, \mathbb{T}\mathbb{V}\mathbb{T}^\top \rangle. \tag{4}$$

The soft permutation matrix $\mathbb{T}$ approximates a hard permutation matrix. Here, the Hamiltonian cycle constraint is implicitly enforced through the structure $\mathbb{T}\mathbb{V}\mathbb{T}^\top$. This enables gradient-based optimization to find good approximate solutions, while naturally incorporating both the shortest path objective and the Hamiltonian cycle constraint. The GNN learns to generate a soft permutation matrix $\mathbb{T}$ that, when used in the transformation $\mathbb{T}\mathbb{V}\mathbb{T}^\top$, yields a soft adjacency matrix representing a Hamiltonian cycle. This approach provides a non-autoregressive, unsupervised learning (UL) framework without sequential decision-making or ground truth supervision, enabling efficient end-to-end training directly from the combinatorial optimization objective.

3 From Soft Permutation $\mathbb{T}$ to Hard Permutation $\mathbf{P}$

To obtain a hard permutation matrix $\mathbf{P}$ from the GNN output, we follow the method proposed in [Min and Gomes, 2025]. We use the Gumbel-Sinkhorn operator [Mena et al., 2018], which provides a differentiable approximation to permutation matrices during training. At inference time, we extract a discrete permutation via the Hungarian algorithm. Following the UTSP model [Min et al., 2023], the GNN processes geometric node features $f_0 \in \mathbb{R}^{n\times 2}$ (city coordinates) along with an adjacency matrix $A \in \mathbb{R}^{n\times n}$ defined by:

$$A = e^{-\mathbf{D}/s}, \tag{5}$$

where $\mathbf{D}$ is the Euclidean distance matrix and s is a scaling parameter. The GNN generates logits $\mathcal{F} \in \mathbb{R}^{n\times n}$ that are passed through a scaled hyperbolic tangent activation:

$$\mathcal{F} = \alpha \tanh(f_{\text{GNN}}(f_0, A)), \tag{6}$$

where α is a scaling parameter, and $f_{\text{GNN}} : \mathbb{R}^{n\times 2} \times \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is a GNN that operates on node features f_0 and the adjacency matrix A. The scaled logits are passed through the Gumbel-Sinkhorn operator to produce a differentiable approximation of a permutation matrix:

$$\mathbb{T} = \text{GS}\left(\frac{\mathcal{F} + \gamma\epsilon}{\tau}, l\right), \tag{7}$$

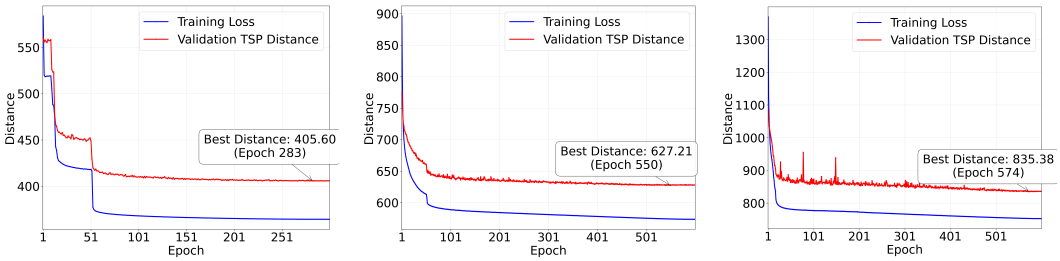
where ϵ is i.i.d. Gumbel noise, γ is the noise magnitude, τ is the temperature parameter which controls relaxation sharpness, and l is the number of Sinkhorn iterations. Lower values of τ yield sharper, near-permutation matrices. At inference, we derive a hard permutation using the Hungarian algorithm:

$$\mathbf{P} = \text{Hungarian}\left(-\frac{\mathcal{F} + \gamma\epsilon}{\tau}\right). \tag{8}$$

The final Hamiltonian cycle is reconstructed as $\mathbf{P}\mathbb{V}\mathbf{P}^\top$, yielding a discrete TSP tour.

Training and Inference Our training objective minimizes the loss function in Equation 4, incorporating a structural inductive bias: the structure $\mathbb{T}\mathbb{V}\mathbb{T}^\top$ implicitly encodes the Hamiltonian cycle constraint, guiding the model toward effective solutions.

During inference, we decode the hard permutation matrix $\mathbf{P}$ using the Hungarian algorithm as previously described in Equation 8. The final tour is obtained directly through $\mathbf{P}\mathbb{V}\mathbf{P}^\top$, which always generates a valid Hamiltonian cycle by construction. Unlike conventional TSP heuristics requiring local search, our solutions naturally emerge from the learned permutation matrices. The key advantage of this framework lies in its structural guarantee: regardless of the quality of the learned permutation matrix $\mathbf{P}$, the transformation $\mathbf{P}\mathbb{V}\mathbf{P}^\top$ will always yield a feasible TSP solution. The optimization process thus focuses entirely on finding the permutation that minimizes tour length, while the constraint is automatically satisfied.[1]



TSP-20 TSP-50 TSP-100

Figure 1: Training history across TSP sizes. Distances are scaled by a factor of 100.

[1]While we use the Hungarian algorithm to obtain hard permutations from each model's soft output, this step is deterministic and not part of any heuristic or tree-based search procedure. We use the term *search* in the sense of explicit exploration or rollout over solution candidates, as in beam search or reinforcement learning.

4 Experiment

Our training loss with respect to the validation distance is shown in Figure 1. The training curves consistently converge across all problem sizes, with the training loss (blue) steadily decreasing and stabilizing over epochs. Notably, there is a strong correlation between training loss reduction and validation TSP distance improvement (red), indicating effective learning without overfitting. On 20-node instances, the model achieves the best validation distance of 405.60 at epoch 283; on 50-node instances, our model achieves its best distance of 627.21 at epoch 550; on 100-node instances, we achieve the best validation distance of 835.38 at epoch 574. Across all scales, the validation performance closely tracks the training loss trajectory, confirming that the model generalizes well and that minimizing the objective in Equation 4 consistently leads to improved TSP solution quality.

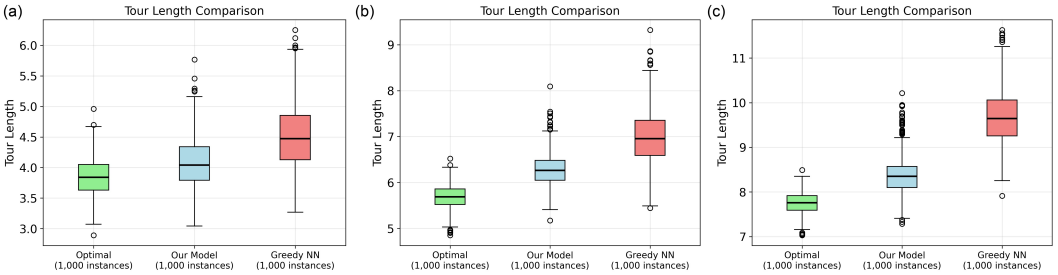


Figure 2: Tour length comparison across TSP instance sizes. (a) TSP-20, (b) TSP-50, (c) TSP-100. Box plots show the distribution of tour lengths over 1,000 test instances for each method: Optimal (Concorde), Our Model, and Greedy Nearest Neighbor (NN).

Length Distribution Figures 2 shows the tour length distributions on the test set for 20-, 50-, and 100-node instances, using the model with the lowest validation length across all hyperparameters. Our model consistently outperforms the Greedy Nearest Neighbor (NN) baseline—which constructs tours by iteratively selecting the nearest unvisited node—achieving substantial gains across all problem sizes. The distribution histograms reveal that our model produces more concentratedly distributed, shorter tour lengths with mean values of $\mu = 4.06$, 6.28, and 8.37 compared to Greedy NN's $\mu = 4.51$, 6.99, and 9.67.

Table 1: Comparison of tour quality across different heuristics on TSP instances of varying sizes.

Method	Type	TSP-20		TSP-50		TSP-100	
		Tour Len.	Gap	Tour Len.	Gap	Tour Len.	Gap
Concorde	Solver	3.83	0.00%	5.69	0.00%	7.75	0.00%
Beam search (w=1280)	Search	4.06	6.00%	6.83	20.0%	9.89	27.6%
Greedy NN	G	4.51	17.8%	6.99	22.8%	9.67	24.8%
Our method	UL, NAR	4.06	6.00%	6.28	10.4%	8.37	8.00%

We evaluate the inference efficiency of our approach by measuring the average time per instance, including the GNN forward pass and construction of the hard permutation as defined in Equation 8. On an NVIDIA H100 GPU with batch size 256, the model achieves inference times of 0.17 ms for TSP-20, 0.15 ms for TSP-50, and 0.40 ms for TSP-100 .

Optimality Gap Calculation The optimality gap is computed as: $\frac{l_\theta - l_O}{l_O}$, where l_θ denotes the tour length obtained by our method, and l_O is the optimal tour length. This measures how far a method's tour length deviates from the optimal solution, with smaller gaps indicating better performance.

Our unsupervised, search-free approach demonstrates competitive performance across TSP instances of varying sizes, achieving optimality gaps of 6.00%, 10.4%, and 8.00% on TSP-20, TSP-50, and TSP-100 respectively (Table 1). Notably, our method matches beam search performance on TSP-20 while significantly outperforming it on larger instances (10.4% vs 20.0% gap on TSP-50, and 8.00% vs 27.6% gap on TSP-100). Our approach outperforms the Greedy NN baseline across

all sizes, with greater gains on larger instances. These results suggest that our model effectively captures global tour structure and long-range city dependencies, enabling better solutions compared to methods that rely primarily on local, greedy decisions or limited search strategies. This indicates that structural inductive biases alone can enable the model to discover competitive combinatorial solutions without supervision or explicit search.

5 Hamiltonian Cycle Ensemble

Examining the results in Figure 2, we observe a long tail distribution where our model yields notably suboptimal solutions on some instances. This suggests that while the model generally performs well, it sometimes generates significantly suboptimal solutions.

To overcome this limitation, we modify the training objective (Equation 4) by introducing an ensemble over powers of the cyclic shift matrix $\mathbb{V}^k$, where $\gcd(k, n) = 1$ ensures each $\mathbb{V}^k$ defines a valid Hamiltonian cycle. We train separate models for each of the $\varphi(n)$ variants, where $\varphi(n)$ is *Euler's totient function*.

Theorem 5.1 ($\mathbb{V}^k$ Hamiltonian Cycle Characterization (Appendix C))**.** *Let $\mathbb{V}$ be the $n \times n$ cyclic shift matrix and $k \in \mathbb{Z}^+$. Then $\mathbb{V}^k$ represents a Hamiltonian cycle if and only if* $\gcd(k, n) = 1$.

Corollary 5.2 (Euler's Totient Function Connection)**.** *The number of distinct Hamiltonian cycle matrices of the form $\mathbb{V}^k$ is exactly $\varphi(n)$, where φ is Euler's totient function.*

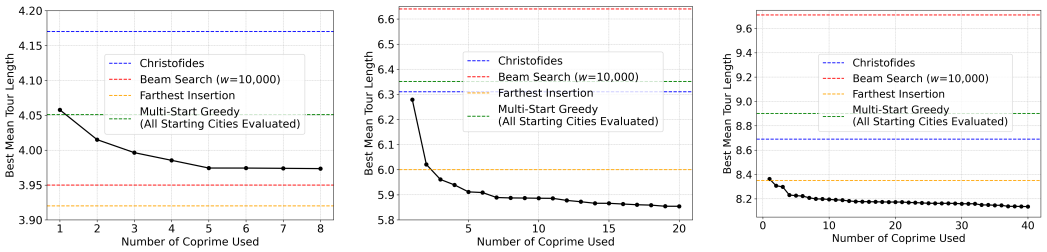


Figure 3: Mean tour length on TSP instances of varying sizes (TSP-20, TSP-50, TSP-100) as the number of coprime shifts increases. Utilizing more shift combinations significantly reduces tour length, Christofides and Beam Search bounds are included for reference.

Table 2: Detailed performance comparison of learning-based TSP solvers across different instance sizes (TSP-20/50/100). Metrics include average tour length and optimality gap (%). Results for baseline methods are taken from [Joshi et al., 2019]. While all methods use uniformly generated TSP instances, test sets vary slightly across works. Note that many more recent models exist, we select a subset for comparison.

Method	Type	TSP-20		TSP-50		TSP-100	
		Tour Len.	Gap	Tour Len.	Gap	Tour Len.	Gap
PtrNet Vinyals et al. [2015]	SL, G	3.88	1.15%	7.66	34.48%	-	-
PtrNet Bello et al. [2016]	RL, G	3.89	1.42%	5.95	4.46%	8.30	6.90%
S2V Khalil et al. [2017]	RL, G	3.89	1.42%	5.99	5.16%	8.31	7.03%
GAT Deudon et al. [2018]	RL, G, 2-OPT	3.85	0.42%	5.85	2.77%	8.17	5.21%
GAT Kool et al. [2018]	RL, G	3.85	0.34%	5.80	1.76%	8.12	4.53%
GCN Joshi et al. [2019]	SL, G	3.86	0.60%	5.87	3.10%	8.41	8.38%
POMO Kwon et al. [2020]	RL,G	3.83	0.12%	5.73	0.64%	7.84	1.07%
Concorde	Solver	3.83	0.00%	5.69	0.00%	7.75	0.00%
Greedy NN (all start cities)	G	4.05	5.74%	6.35	11.6%	8.90	14.8%
Beam search (w=5,000)	Search	3.98	3.92%	6.71	17.9%	9.77	26.1%
Beam search (w=10,000)	Search	3.95	3.13%	6.64	17.0%	9.71	25.3%
Farthest insertion	Heuristics	3.92	2.35%	6.00	5.45%	8.35	7.74%
Christofides	Heuristics	4.17	8.88%	6.31	10.9%	8.69	12.1%
Hamiltonian cycle ensemble	UL, NAR	3.97	3.52%	5.85	2.81%	8.14	5.03%

Ensemble Training and Inference Our ensemble strategy trains separate models for each valid $\mathbb{V}^k$ matrix. Specifically, we construct $\varphi(n)$ models, each optimizing the modified objective:

$$\mathcal{L}_{\text{TSP}}(k) = \langle \mathbf{D}, \mathbb{T}_{(k)} \mathbb{V}^k \mathbb{T}_{(k)}^\top \rangle, \tag{9}$$

where $\gcd(k, n) = 1$ and $\mathbb{T}_{(k)}$ represents the learned soft permutation matrix corresponding to $\mathbb{V}^k$. For each k, we train identical models differing only in the underlying Hamiltonian cycle $\mathbb{V}^k$. At inference, we evaluate all $\varphi(n)$ models per test instance. Each model decodes a hard permutation $\mathbf{P}_{(k)}$ from its soft output $\mathbb{T}_{(k)}$ using the Hungarian algorithm (Equation 8), producing a valid tour via $\mathbf{P}_{(k)} \mathbb{V}^k \mathbf{P}_{(k)}^\top$. We select the shortest among all ensemble tours: $\text{Tour}_{\text{final}} = \mathbf{P}_{(k^*)} \mathbb{V}^{k^*} \mathbf{P}_{(k^*)}^\top$, where $k^* = \arg\min_{k:\gcd(k,n)=1} \langle \mathbf{D}, \mathbf{P}_{(k)} \mathbb{V}^k \mathbf{P}_{(k)}^\top \rangle$. This instance-wise selection ensures that each test problem is solved using the most suitable cycle structure from the ensemble.

Figure 3 demonstrates the effectiveness of this ensemble approach across 20, 50, and 100-node problems respectively. As the number of coprime shifts increases, the mean tour length decreases substantially, with dramatic improvements observed initially that gradually plateau. For TSP-20, using all $\varphi(20) = 8$ coprime shifts reduces mean tour length from 4.06 to 3.97, surpassing both multi-start greedy and beam search performance. Similar trends are observed for TSP-50 and TSP-100, where our ensemble approach achieves mean tour lengths of 5.85 and 8.14 respectively when using all available coprime shifts. This demonstrates that leveraging multiple Hamiltonian cycle structures effectively mitigates the long tail problem while consistently improving solution quality. Here, each permutation learner in our framework is trained with respect to a fixed initial Hamiltonian cycle $\mathbb{V}^k$, which serves as a structural prior guiding solution formation. This initialization anchors the training process, focusing learning on the permutation of the initial Hamiltonian cycle $\mathbb{V}^k$, effectively biasing the model. Since different initial cycles encode distinct structural priors, using an ensemble of models with $\mathbb{V}^k$ promotes diversity and improves overall solution quality.

Despite not using supervision or autoregressive decoding, our method achieves competitive results across all TSP sizes. Our Hamiltonian cycle ensemble approach significantly outperforms classical baselines, achieving optimality gaps of 3.52%, 2.81%, and 5.03% on TSP-20, TSP-50, and TSP-100 respectively, compared to greedy NN (all start cities) search's 5.74%, 11.6%, and 14.8%. We also improve upon beam search variants as the problem size grows, with beam search achieving gaps of 3.13–3.92% on TSP-20, 17.0–17.9% on TSP-50, and 25.3–26.1% on TSP-100.

Among learning-based methods, our approach demonstrates competitive performance. We achieve comparable results to Pointer Networks and S2V. Our method also performs competitively with supervised method [Joshi et al., 2019], achieving 5.03% optimality gap versus 8.38% on TSP-100. Our performance is comparable to the RL-based approaches. Notably, we are competitive with the GAT model of [Deudon et al., 2018] even when it is augmented with 2-OPT local search, a strong post-hoc refinement step. While models such as the attention-based approach by [Kool et al., 2018] leverage RL and autoregressive decoding, our unsupervised, non-autoregressive framework attains similar optimality gaps without requiring either RL training or explicit search procedures. However, we do not yet match the RL model such as [Kwon et al., 2020], which benefits from exploiting multiple equivalent solutions through parallel rollouts. Our results are also competitive with classical heuristics such as farthest insertion, while offering a fundamentally different approach grounded in structural inductive bias. Overall, our results highlight that non-autoregressive, unsupervised methods can effectively tackle combinatorial optimization problems without sequential decoding. Detailed comparisons are provided in Table 2.

6 Conclusion

We present a fully unsupervised, non-autoregressive framework for solving the TSP without relying on explicit search or supervision. By framing the problem as learning permutation matrices that satisfy Hamiltonian cycle constraints via similarity transformations, our approach incorporates structural constraints as inductive biases into the learning process. This formulation enables the model to generate valid tours without sequential decision-making. Our method achieves competitive results and we further demonstrate that ensembles over different Hamiltonian cycles enhance robustness and improve average solution quality, especially on larger problem instances. These results suggest that learned structural biases provide a promising alternative to traditional heuristic search methods by integrating problem structure as an inductive bias in combinatorial optimization.

References

David Applegate, Robert Bixby, Vasek Chvátal, and William Cook. Concorde tsp solver. `http://www.math.uwaterloo.ca/tsp/concorde.html`, 2003.

Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

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

John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15*, pages 170–181. Springer, 2018.

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

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.

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.

Yimeng Min and Carla Gomes. Unsupervised learning permutations for tsp using gumbel-sinkhorn operator. In *NeurIPS 2023 Workshop Optimal Transport and Machine Learning*, 2023.

Yimeng Min and Carla P Gomes. Unsupervised ordering for maximum clique. *arXiv preprint arXiv:2503.21814*, 2025.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018.

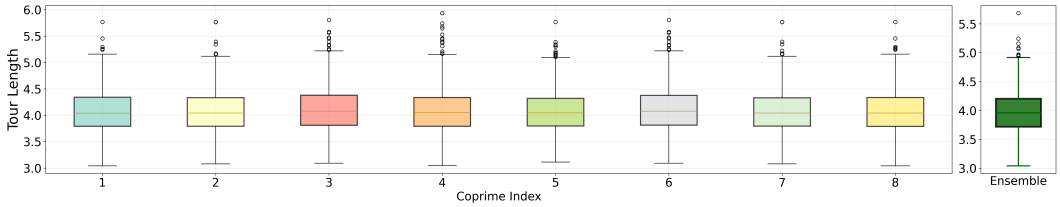
Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

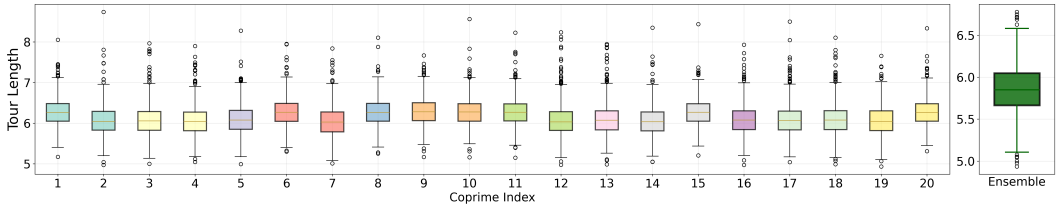
Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Yimeng Min, Frederik Wenkel, Michael Perlmutter, and Guy Wolf. Can hybrid geometric scattering networks help solve the maximum clique problem? *Advances in Neural Information Processing Systems*, 35:22713–22724, 2022.

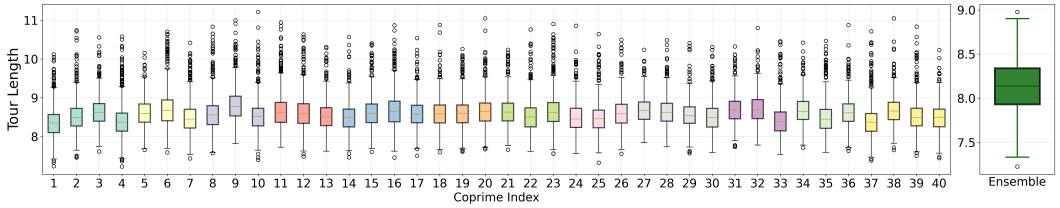
Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.



TSP-20: Ensemble vs. 8 individual coprime models



TSP-50: Ensemble vs. 20 individual coprime models



TSP-100: Ensemble vs. 40 individual coprime models

Figure 4: Tour length distributions across individual models and ensemble output for various TSP sizes. Each index corresponds to a model trained using a different coprime shift matrix $\mathbb{V}^k$, where $\gcd(k, n) = 1$. The ensemble result (rightmost box in green) selects the minimum-length tour across all coprime-specific models for each instance.

A Appendix

Effectiveness of the Hamiltonian Cycle Ensemble Figure 4 shows the tour length distributions produced by models trained with different coprime shifts $\mathbb{V}^k$ for TSP instances of size 20, 50, and 100. Each colored boxplot represents the output distribution from a single model trained on a specific cyclic structure, while the green box on the right shows the ensemble result obtained by selecting the shortest tour across all models for each instance. Notably, while individual models exhibit varying performance and often display long-tail distributions with significant outliers, the ensemble output consistently achieves shorter average tour lengths with reduced variance. This demonstrates that the ensemble strategy effectively mitigates the long-tail failure cases seen in individual models by leveraging structural diversity. Consequently, the ensemble approach leads to more robust and consistent solutions across problem instances.

Randomness by Hardware Perturbation Inference We introduce Hardware Perturbation Inference (HPI), a simple yet effective technique that leverages the inherent non-determinism of low-level numerical operations to generate diverse inference outcomes without modifying the model or introducing explicit stochasticity. Even when using the same GPU architecture (e.g., NVIDIA H100), small numerical discrepancies can arise from differences in fused multiply–add (FMA) kernel execution and TensorFloat-32 (TF32) rounding modes. These subtle perturbations may propagate through the computation, leading to slightly different outputs. HPI exploits this phenomenon to produce multiple candidate solutions for the same problem instance, which can then be ensembled to improve robustness and solution quality—all without requiring changes to the model parameters or training procedure.

In our experiments, we apply HPI on NVIDIA H100 GPUs by toggling the use of fused multiply–add (FMA) operations under TensorFloat-32 (TF32) precision. Specifically, we compare inference with TF32+FMA enabled versus disabled, which yields distinct perturbations in the numerical pathways and consequently different solutions for the same input instance $\mathbb{V}^k$. By combining these outputs in an ensemble, we observe further improvements in solution quality: on the TSP-100 benchmark, the ensemble reduces the optimality gap to 8.10.

While hardware-level perturbations provide a simple mechanism for generating diversity, there are many other ways to introduce randomness to further enhance ensemble performance. We leave a broader discussion of such strategies for future work.

B Training Details

B.1 Experimental Setup

Our experiments span three distinct problem sizes: 20-node, 50-node, and 100-node TSP instances. For each problem size, we perform hyperparameter sweeps to identify the optimal configurations. Training data consist of uniformly distributed TSP instances generated for each problem size, with 100,000 training instances for 20-node, 500,000 training instances on 50-node instances, and 1,500,000 training instances for 100-node instances. All problem sizes use 1,000 instances each for validation and test.

B.2 Hyperparameter Configuration

We conduct parameter exploration through grid search to evaluate our approach. The 20-node instances are trained across all combinations of temperature $\tau \in \{2.0, 3.0, 4.0, 5.0\}$ and noise scale $\gamma \in \{0.005, 0.01, 0.05, 0.1, 0.2, 0.3\}$, resulting in 24 configurations for each size; the 50-node instances are trained across all combinations of temperature $\tau = 5.0$ and noise scale $\gamma \in \{0.005, 0.01, 0.05, 0.1, 0.2, 0.3\}$, while the 100-node instances use only one temperature value $\tau = 5.0$ with an expanded noise scale $\gamma \in \{0.1, 0.2, 0.3\}$, totaling 3 configurations. We employ $\ell = 60$ Sinkhorn iterations for 20-node instances and $\ell = 80$ for 50- and 100-node instances, with training conducted over 300 epochs for 20-node instances and extended to 600 epochs for 50- and 100-node instances to ensure sufficient convergence. For evaluation, tour distances are computed using hard permutations $\mathbf{P}$ obtained via the Hungarian algorithm as described in Equation 8, in contrast to the soft permutation $\mathbb{T}$ used during training.

B.3 Network Architecture and Training Details

Following the UTSP model [Min et al., 2023], we employ Scattering Attention Graph Neural Networks (SAGs) with 128 hidden dimensions and 2 layers for 20-node instances, 256 hidden dimensions and 6 layers for 50-node instances, and 512 hidden dimensions and 8 layers for 100-node instances [Min et al., 2022]. For TSP-20, we use SAGs with 6 scattering channels and 2 low-pass channels; for TSP-50 and TSP-100, we use SAGs with 4 scattering channels and 2 low-pass channels. We train the networks using the Adam optimizer [Kingma and Ba, 2014] with weight decay regularization $\lambda = 1 \times 10^{-4}$. Learning rates are set to 1×10^{-3} for 20-node instances and 2×10^{-3} for 50- and 100-node instances. To ensure training stability, we implement several regularization techniques: (i) learning rate scheduling with a 15-epoch warmup period, (ii) early stopping with patience of 50 epochs, and (iii) adaptive gradient clipping to maintain stable gradients throughout the optimization process.

For each problem size, we select the best performing model based on validation performance across all hyperparameter combinations of τ and noise scale γ. The model configuration that achieves the lowest validation distance is then evaluated on the corresponding test set.

C Proof of Main Theorem

Proof. We prove both directions of the equivalence.

Necessity ($\Rightarrow$): Suppose $\mathbb{V}^k$ represents a Hamiltonian cycle. Let $d = \gcd(k, n)$. The matrix $\mathbb{V}^k$ corresponds to the mapping $\sigma^k : i \mapsto (i + k) \bmod n$ on the vertex set $\{0, 1, \ldots, n-1\}$.

Consider the orbit of vertex 0 under this mapping:

$$\mathcal{O}_0 = \{0, k \bmod n, 2k \bmod n, \ldots, (m-1)k \bmod n\}, \tag{10}$$

where m is the smallest positive integer such that $mk \equiv 0 \pmod{n}$.

Since $d = \gcd(k, n)$, we can write $k = dk'$ and $n = dn'$ where $\gcd(k', n') = 1$. Then:

$$mk \equiv 0 \pmod{n} \iff dn' \mid mdk' \tag{11}$$

$$\iff n' \mid mk' \tag{12}$$

$$\iff n' \mid m \quad (\text{since } \gcd(k', n') = 1). \tag{13}$$

Therefore, the smallest such m is $m = n' = \frac{n}{d}$, so $|\mathcal{O}_0| = \frac{n}{d}$.

If $\mathbb{V}^k$ represents a Hamiltonian cycle, then all n vertices must lie in a single orbit, which requires $|\mathcal{O}_0| = n$. This implies $\frac{n}{d} = n$, hence $d = 1$, i.e., $\gcd(k, n) = 1$.

Sufficiency ($\Leftarrow$): Suppose $\gcd(k, n) = 1$. Then by the argument above, the orbit of vertex 0 has size $\frac{n}{1} = n$. This means the sequence $\{0, k, 2k, \ldots, (n-1)k\}$ modulo n contains all distinct elements of $\{0, 1, \ldots, n-1\}$.

Therefore, $\mathbb{V}^k$ represents a permutation that cycles through all n vertices exactly once, forming a single Hamiltonian cycle. □

Corollary C.1 (Euler's Totient Function Connection)**.** *The number of distinct Hamiltonian cycle matrices of the form $\mathbb{V}^k$ is exactly $\varphi(n)$, where φ is Euler's totient function.*

Proof. By Theorem 5.1, $\mathbb{V}^k$ represents a Hamiltonian cycle if and only if $\gcd(k, n) = 1$. The number of integers $k \in \{1, 2, \ldots, n\}$ such that $\gcd(k, n) = 1$ is precisely $\varphi(n)$. □

Remark C.2 (Directed vs Undirected Cycles). Note that different values of k may yield distinct *directed* Hamiltonian cycles that correspond to the same *undirected* cycle traversed in opposite directions. For instance, when n is even, $\mathbb{V}^1$ and $\mathbb{V}^{n-1}$ represent the same undirected cycle with opposite orientations. However, each $\mathbb{V}^k$ with $\gcd(k, n) = 1$ defines a unique directed cycle, which is the relevant structure for our ensemble method.

D Quadratic upper bound on the optimality gap

Theorem D.1 (Quadratic upper bound on the optimality gap)**.** *Let $C(\mathbf{P}) := \langle \mathbf{D}, \mathbf{P}\mathbb{V}\mathbf{P}^\top \rangle$ for a cost matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$, a cyclic shift matrix $\mathbb{V} \in \mathbb{R}^{n \times n}$, and a permutation matrix $\mathbf{P} \in \Pi_n$. Let the set of optimal permutations be*

$$\mathcal{O} := \arg\min_{\mathbf{P} \in \Pi_n} C(\mathbf{P}), \qquad C^\star := \min_{\mathbf{P} \in \Pi_n} C(\mathbf{P}). \tag{14}$$

Given a (soft) doubly-stochastic matrix $\mathbb{T}$ produced by the model and a hard permutation $\widehat{\mathbf{P}}$ obtained from $\mathbb{T}$ at inference, define

$$\delta_* := \min_{\mathbf{P} \in \mathcal{O}} \|\mathbb{T} - \mathbf{P}\|_F, \qquad \varepsilon := \|\widehat{\mathbf{P}} - \mathbb{T}\|_F. \tag{15}$$

If $\|\mathbb{V}\|_2 \leq 1$ and $\|\mathbb{T}\|_2 \leq 1$, then

$$C(\widehat{\mathbf{P}}) - C^\star \;\leq\; \|\mathbf{D}\|_F \big(2\delta_* + \delta_*^2 + 2\varepsilon + \varepsilon^2\big). \tag{16}$$

Proof. Since Π_n is finite, there exists $\mathbf{P}^\dagger \in \mathcal{O}$ such that $\delta_* = \|\mathbb{T} - \mathbf{P}^\dagger\|_F$. We decompose the gap into a "soft" part and a "rounding" part:

$$C(\widehat{\mathbf{P}}) - C^\star = \underbrace{\big(C(\mathbb{T}) - C(\mathbf{P}^\dagger)\big)}_{\text{soft approximation}} + \underbrace{\big(C(\widehat{\mathbf{P}}) - C(\mathbb{T})\big)}_{\text{rounding}}. \tag{17}$$

Soft term. Let $E := \mathbb{T} - \mathbf{P}^\dagger$. Expanding,

$$\mathbb{T}\mathbb{V}\mathbb{T}^\top - \mathbf{P}^\dagger\mathbb{V}\mathbf{P}^{\dagger\top} = E\mathbb{V}\mathbf{P}^{\dagger\top} + \mathbf{P}^\dagger\mathbb{V}E^\top + E\mathbb{V}E^\top. \tag{18}$$

Using the submultiplicative bounds

$$\|AXB\|_F \le \|A\|_F\|X\|_2\|B\|_2, \qquad \|AXB\|_F \le \|A\|_2\|X\|_F\|B\|_2, \tag{19}$$

together with $\|\mathbf{P}^\dagger\|_2 = 1$, $\|\mathbb{V}\|_2 \le 1$, and $\|E\|_2 \le \|E\|_F$, we obtain

$$\|E\mathbb{V}\mathbf{P}^{\dagger\top}\|_F \le \delta_*, \quad \|\mathbf{P}^\dagger\mathbb{V}E^\top\|_F \le \delta_*, \quad \|E\mathbb{V}E^\top\|_F \le \delta_*^2. \tag{20}$$

Thus

$$\|\mathbb{T}\mathbb{V}\mathbb{T}^\top - \mathbf{P}^\dagger\mathbb{V}\mathbf{P}^{\dagger\top}\|_F \le 2\delta_* + \delta_*^2. \tag{21}$$

By Cauchy–Schwarz,

$$|C(\mathbb{T}) - C(\mathbf{P}^\dagger)| \;\le\; \|\mathbf{D}\|_F(2\delta_* + \delta_*^2). \tag{22}$$

Rounding term. Let $\Delta := \widehat{\mathbf{P}} - \mathbb{T}$, so $\|\Delta\|_F = \varepsilon$. Expanding,

$$\widehat{\mathbf{P}}\mathbb{V}\widehat{\mathbf{P}}^\top - \mathbb{T}\mathbb{V}\mathbb{T}^\top = \Delta\mathbb{V}\mathbb{T}^\top + \mathbb{T}\mathbb{V}\Delta^\top + \Delta\mathbb{V}\Delta^\top. \tag{23}$$

Using the same bounds and $\|\mathbb{T}\|_2 \le 1$, $\|\mathbb{V}\|_2 \le 1$,

$$\|\Delta\mathbb{V}\mathbb{T}^\top\|_F \le \varepsilon, \quad \|\mathbb{T}\mathbb{V}\Delta^\top\|_F \le \varepsilon, \quad \|\Delta\mathbb{V}\Delta^\top\|_F \le \varepsilon^2, \tag{24}$$

hence

$$\|\widehat{\mathbf{P}}\mathbb{V}\widehat{\mathbf{P}}^\top - \mathbb{T}\mathbb{V}\mathbb{T}^\top\|_F \le 2\varepsilon + \varepsilon^2, \tag{25}$$

and by Cauchy–Schwarz,

$$|C(\widehat{\mathbf{P}}) - C(\mathbb{T})| \;\le\; \|\mathbf{D}\|_F(2\varepsilon + \varepsilon^2). \tag{26}$$

Combine. By the triangle inequality,

$$C(\widehat{\mathbf{P}}) - C^\star \;\le\; \|\mathbf{D}\|_F\big(2\delta_* + \delta_*^2 + 2\varepsilon + \varepsilon^2\big). \tag{27}$$

□

Lemma D.2 (Spectral norm of $\mathbb{V}$)**.** *Let $\mathbb{V} \in \{0,1\}^{n\times n}$ be the cyclic shift matrix defined in Equation 2. Then*

$$\|\mathbb{V}\|_2 = 1. \tag{28}$$

Proof. The matrix $\mathbb{V}$ is a permutation matrix corresponding to a cyclic shift. Permutation matrices are orthogonal, i.e. $\mathbb{V}^\top\mathbb{V} = I$. Hence, all eigenvalues of V have absolute value 1, and

$$\|\mathbb{V}\|_2 = \sqrt{\lambda_{\max}(\mathbb{V}^\top\mathbb{V})} = \sqrt{\lambda_{\max}(I)} = 1. \tag{29}$$

Equivalently, $\mathbb{V}$ is diagonalizable by the discrete Fourier transform, with eigenvalues $\{e^{2\pi ik/n} : k = 0, \dots, n-1\}$, all lying on the unit circle. Thus the spectral norm of $\mathbb{V}$ is exactly 1. □

Lemma D.3 (Spectral norm of $\mathbb{T}$)**.** *If $\mathbb{T}$ is doubly-stochastic, then $\|\mathbb{T}\|_2 \le 1$.*

Proof. By the Birkhoff–von Neumann theorem, any doubly-stochastic $\mathbb{T}$ can be written as a convex combination of permutation matrices: $\mathbb{T} = \sum_k \alpha_k\mathbf{P}_k$, with $\alpha_k \ge 0$ and $\sum_k \alpha_k = 1$. The spectral norm is convex, hence

$$\|\mathbb{T}\|_2 = \Big\|\sum_k \alpha_k\mathbf{P}_k\Big\|_2 \;\le\; \sum_k \alpha_k\|\mathbf{P}_k\|_2 \;=\; \sum_k \alpha_k \cdot 1 \;=\; 1, \tag{30}$$

since each permutation matrix $\mathbf{P}_k$ is orthogonal and thus has spectral norm 1. □

Remark D.4 (Interpretation). δ_* measures how close the learned soft matrix $\mathbb{T}$ is to *some* optimal permutation in $\mathcal{O}$, so the bound handles non-uniqueness naturally. When there are symmetries (e.g. reversed cycles, relabelings), δ_* will be the distance to the closest such symmetry, which tightens the bound compared to fixing an arbitrary $\mathbf{P}^\dagger$.

E Zero-Shot Generalization

We propose a zero-shot evaluation strategy inspired by [Min and Gomes, 2025], leveraging *dummy nodes*. As an example, consider testing on a TSP instance with 95 cities using a model trained on TSP-100. To construct such a test case, we randomly select 5 *parent nodes* from the 95 cities and introduce 5 additional dummy nodes, each placed very close to one of the selected parents. This augmentation produces an effective 100-node instance, which we then solve using the TSP-100 model.

If the resulting tour connects each dummy node directly to its parent node, we merge them to recover a valid tour for the original 95-city problem. If this condition is not met, we repeat the process by re-sampling the parent and dummy nodes.

Using this strategy, our model achieves a mean tour length of 8.24 across $1{,}000$ unseen test instances, compared to 9.49 for the greedy baseline. For reference, the optimal mean tour length is 7.57. These results demonstrate that our dummy-node construction enables effective zero-shot transfer across problem sizes while maintaining competitive performance.

F Discussion and Future Work

In this paper, we use 24 configurations of (τ, γ) pairs for TSP-20, 6 configurations for TSP-50, and 3 configurations for TSP-100. This limited yet targeted hyperparameter exploration is sufficient to support our central claim: that *structural inductive bias*, when coupled with a permutation-based formulation, can drive the emergence of high-quality solutions in a fully unsupervised, non-autoregressive setting. We restrict our analysis to minimal hyperparameter settings and adopt the same architecture as UTSP [Min et al., 2023], which is sufficiently expressive to illustrate our main claim. While preliminary evidence indicates that performance can be further improved through extensive hyperparameter tuning or architectural variations (e.g., alternative message passing schemes), such enhancements lie outside the scope of our primary contribution and are left for future work.

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: The abstract and introduction clearly state that the paper proposes an unsupervised method for TSP.

   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: [No]

   Justification: The paper does not explicitly discuss the limitations. However, it includes a discussion on the computational efficiency of the proposed algorithms and how they scale with the size of the dataset.

   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: The paper provides a complete and correct theoretical justification in the appendix.

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 paper provides sufficient details to reproduce the main experimental results. It specifies the model architecture, training procedure, hyperparameters (e.g., learning rate, hidden dimension, Gumbel-Sinkhorn settings), dataset generation.

- 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: [NA]

   Justification: workshop does not have a supplementary materials code file entry.

   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: Refer to the code in supplementary materials.

   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: the paper reports mean performance metrics across multiple graph instances.

   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: The paper specifies the hardware used for experiments.

   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: [NA]

    Justification: no societal impact

    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: no such risks.

    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: [NA]

    Justification: We do not use existing assets.

    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: [NA]

    Justification: Paper does not release new assets.

    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: Paper does not involve crowdsourcing nor research with human subjects.

    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: Paper does not involve crowdsourcing nor research with human subjects.

    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 paper 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.