# Unsupervised Learning Permutations for TSP using Gumbel-Sinkhorn Operator

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

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

## Abstract

The Machine Learning community has recently shown a growing interest in Optimal Transport (OT). Methods that leverage entropy regularization based on OT have proven especially effective for various tasks, including ranking, sorting, and solving jigsaw puzzles Mena et al. [2018], Adams and Zemel [2011], Cuturi [2013]. In our study, we broaden the application of entropy regularization methods to address the NP-hard Travelling Salesman Problem (TSP). We first formulate TSP as identifying the permutation of a Hamiltonian Cycle with the shortest length. Following this, we establish the permutation representation using the Gumbel-Sinkhorn operator with entropic regularization. Our findings indicate a balance between entropy and generalization. We further discuss how to generalize across different hardnesses.

## 1 Introduction

Optimal Transport (OT) offers a structured approach to efficiently move one probability distribution to another Villani et al. [2009]. Sinkhorn algorithm provides a differentiable approximation of the discrete OT problem, making it suitable for gradient-based optimization techniques commonly used in machine learning Sinkhorn [1964], Adams and Zemel [2011]. The Sinkhorn algorithm has been demonstrated to be effective in the context of entropy regularization for optimal transport.

Recent studies have demonstrated that methods based on the Sinkhorn algorithm can effectively learn to approximate permutations in a differentiable manner Mena et al. [2018]. The Sinkhorn operator enhances the smooth optimization of the entropy regularization term, leading to robust and efficient solutions for various optimal transport challenges. Given the pivotal role of permutations in numerous applications, the Sinkhorn operator offers promising potential in addressing a variety of problems. For instance, Linderman et al. [2018] introduced a rounding method employing the Sinkhorn operator to draw samples from matrices close to the Birkhoff polytope. , Wang et al. [2019] constructed a fully differentiable deep network architecture to learn affinity in graph matching, employing the Sinkhorn Network Adams and Zemel [2011]. Meanwhile, Pang et al. [2020] illustrated that inferring the permutation matrix through Sinkhorn iterations outperforms the traditional classifier approach of Jigsaw self-supervision.

In our study, we broaden the application of entropy regularization methods to build a heuristic for the NP-hard Travelling Salesman Problem (TSP). TSP is a classic optimization problem that holds fundamental importance in operations research and combinatorial optimization. The problem poses the question: With a given list of cities and the distances between them, how can you determine the shortest route that passes through each city only once and then returns to the starting city? From a graph perspective, this can be rephrased as: *In a complete weighted graph (where cities are vertices, and roads are edges), identify the Hamiltonian Cycle with the shortest distance.*

Accepted to the Optimal Transport and Machine Learning Workshop at NeurIPS 2023.

A variety of methods have been put forth to address the TSP. These include the exact solver Concorde [1] and the Lin-Kernighan-Helsgaun (LKH) heuristic Helsgaun [2000, 2017]. Recently, there has been a surge in interest and research efforts focused on developing data-driven heuristics for solving the TSP Bengio et al. [2021]. Researchers have been actively exploring various data-driven methodologies, including reinforcement learning (RL), supervised learning (SL), and unsupervised learning (UL), to build heuristics for the TSP.

In reinforcement learning (RL), the agent interacts with the environment by iteratively selecting actions (e.g., visiting cities in a specific order or performing 2-opt) to optimize the total tour length. The agent receives feedback in the form of rewards or penalties based on the quality of the tours it constructs Kool et al. [2018]. However, RL methods often come with certain challenges. One primary issue is the sparse reward problem. The agent receives minimal or no immediate feedback on the quality of its actions during exploration, making it difficult to learn from the rewards. As a result, RL algorithms may struggle to converge to optimal or near-optimal solutions, especially in high-dimensional and large-scale instances of the TSP.

In contrast, supervised learning (SL) models, often leveraging Graph Neural Networks (GNNs), learn from labelled data to eliminate the need for extensive exploration. These GNN-based models typically generate heat maps indicating edge probabilities in TSP tours, later converted to discrete decisions via methods like greedy or beam search Joshi et al. [2019], Fu et al. [2021]. Yet, SL demands vast labelled training sets, requiring optimal tour calculations for numerous TSP instances. For instance, Fu et al. [2021] used 1 million solved TSP instances for TSP 100, while Sun and Yang [2023] used 1.5 million. Given TSP's NP-hard nature, solving large instances is computationally expensive due to exponentially growing possible solutions.

UL enables models to discover hidden structures in TSP without labelled training instances. In a recent study by Min et al. [2023], the authors introduced an unsupervised learning method known as UTSP. In UTSP, the training process incorporates a surrogate loss function to effectively train the Graph Neural Network (GNN). This surrogate loss function consists of two components: one encourages the model to discover the shortest path, while the other acts as a surrogate for the constraint that the route must form a Hamiltonian Cycle. As mentioned, UL training does not depend on any labelled training data, making it suitable for scenarios where obtaining labelled instances is difficult or impractical. Additionally, UL training offers faster convergence compared to RL methods due to the absence of the sparse reward problem that RL often encounters. As a result, UTSP presents a promising solution for efficiently tackling the TSP without the need for extensive labelled data or encountering convergence issues associated with RL Min et al. [2023].

In UL, for $n$ city coordinates $x \in \mathbb{R}^{n\times 2}$, the GNN generates a transition matrix $\mathbb{T}$ for the TSP. From $\mathbb{T}$, we derive a heat map $\mathcal{H}$, where each element in $\mathcal{H}$ represents the likelihood of an edge being part of the optimal solution. After building the heat map $\mathcal{H}$, we then apply local search to find the TSP solution. However, Min et al. [2023] focuses on generalization within a single distribution, meaning both training and test data originate from the same distribution, we delve into broader scenarios where training and test data might arise from different distributions.

**Our contributions** Here, we reformulate the TSP challenge as *Identifying a permutation matrix that reorders the nodes to yield the shortest Hamiltonian Cycle*. Subsequently, we employ a method based on the Gumbel-Sinkhorn operator to learn this permutation matrix. We train our model in an unsupervised learning fashion, and our findings indicate that (1) there exists a **trade-off** between entropy and generalization; (2) by adjusting the parameters of the Gumbel-Sinkhorn operator, we can achieve better generalization; (3) we discuss a more general scenario where training and test data might arise from different distributions, building upon the phase transition phenomenon observed in Gent and Walsh [1996], we illustrate that the model learned from harder cases can be effectively extended to easier instances.

## 2 Unsupervised Learning TSP

In UTSP, the authors leverage a GNN to generate a doubly stochastic matrix $\mathbb{T} \in \mathbb{R}^{n\times n}$ (a square matrix of nonnegative real numbers, each of whose rows and columns sums to 1). Given TSP instance with coordinates $x \in \mathbb{R}^{n\times 2}$, the authors first feed the coordinates into a GNN and the model outputs

[1]http://www.math.uwaterloo.ca/tsp/concorde/index.html

$\mathbf{h} \in \mathbb{R}^{n\times n}$ Min et al. [2023]. Subsequently, the authors apply a column-wise Softmax activation to the matrix $\mathbf{h}$ and generate $\mathbb{T}$. They then construct a heat map $\mathcal{H}$ based on $\mathbb{T}$:

$$\mathcal{H} = \mathbb{T}\mathbb{V}\mathbb{T}^T, \tag{1}$$

where

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

is the Sylvester shift matrix, $\mathbb{V} \in \mathbb{R}^{n\times n}$. $\mathcal{H} \in \mathbb{R}^{n\times n}$ and $\mathcal{H}_{ij}$ estimates the probability of edge $(i \to j)$ belonging to the optimal solution. During the training phase, the model is optimized using the following loss function:

$$\mathcal{L} = \lambda_1 \underbrace{\sum_{i=1}^{n}(\sum_{j=1}^{n} \mathbb{T}_{i,j} - 1)^2}_{\text{Row-wise constraint}} + \underbrace{\lambda_2 \mathrm{tr}(\mathcal{H})}_{\text{No self-loops}} + \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{D}_{i,j}\mathcal{H}_{i,j}}_{\text{Minimize the distance}}, \tag{3}$$

where $\mathcal{H}$ is constructed using Equation 1 and $\mathbf{D} \in \mathbb{R}^{n\times n}$ represents the distance matrix. The combination of column-wise Softmax activation and the row-wise penalty term $\lambda_1 \sum_{i=1}^{n}(\sum_{j=1}^{n} \mathbb{T}_{i,j} - 1)^2$ serves as a surrogate to ensure that $\mathbb{T}$ behaves like a doubly stochastic matrix. Equation 3 can also be expressed in another form: $\mathcal{L} = \lambda_1 \sum_{i=1}^{n}(\sum_{j=1}^{n} \mathbb{T}_{i,j} - 1)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{\mathbf{D}}_{i,j}\mathcal{H}_{i,j}$, where $\tilde{\mathbf{D}} = \mathbf{D} + \lambda_2 \mathcal{I}_n$, $\mathcal{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix, $\lambda_2$ is the distance of self-loops.

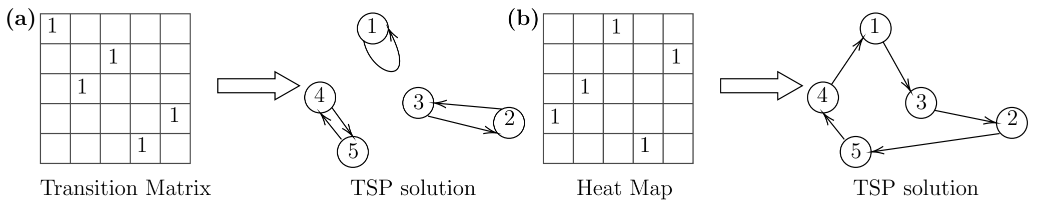


Figure 1: Illustration of $\mathbb{T} \to \mathcal{H}$ transformation.

The intuition behind the $\mathbb{T} \to \mathcal{H}$ transformation is to encode the Hamiltonian Cycle constraint implicitly into the model. When we use $\mathbb{T}$ directly as the heat map, the corresponding route does not satisfy the constraint, an example under binary assumption is shown in Figure 1. Figure 1(a) shows the occasion when directly using $\mathbb{T}$ as the output heatmap: when city 2 and city 3 are close to each other, minimize $\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{D}_{i,j}\mathcal{H}_{i,j}$ can result in a loop between the cities. Since TSP asks us to find the shortest Hamiltonian Cycle on a graph, this loop should be discouraged. Figure 1(b) shows the situation when using $\mathbb{T} \to \mathcal{H}$ transformation. The first row of matrix $\mathcal{H}$ represents the probability distribution of directed edges originating from city 1. Specifically, if the third element in the first row is the only non-zero entry, it indicates the presence of a directed edge $1 \to 3$ in our TSP solution. Similarly, the first column of matrix $\mathcal{H}$ corresponds to the probability distribution of directed edges that terminate at city 1.

## 3 Learning TSP via Permutation

In fact, $\mathbb{T}$ represents an approximation of a general permutation matrix, which can be interpreted as a differentiable relaxation of the a sequence of swapping nodes actions. Let's first consider an elementary permutation matrix $\mathbb{P}(q,t) \in \mathbb{R}^{n\times n}$ which swaps the $q_{th}$ row with the $t_{th}$ row of a matrix:

$$\mathbb{P}(q,t)_{i,j} = \begin{cases} 1, & \text{if } i=j \text{ and } (i \neq q) \text{ and } (i \neq t) \\ 1, & \text{if } i=q \text{ and } j=t \\ 1, & \text{if } i=t \text{ and } j=q \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

When we apply the elementary permutation matrix $\mathbb{P}(q,t)$ on matrix $A \in \mathbb{R}^{n\times n}$, the $q_{\text{th}}$ and $t_{\text{th}}$ rows of matrix $A$ are interchanged while leaving all other elements unchanged. Similarly, if we post-multiply $A$ by $\mathbb{P}(q,t)$, it results in permuting the $q_{\text{th}}$ and $t_{\text{th}}$ columns of $A$ while keeping the remaining elements unaltered. So given $A$, $\mathbb{P}(q,t)A\mathbb{P}(q,t)^T = \mathbb{P}(q,t)A\mathbb{P}(q,t)$ permutes the $q_{\text{th}}$ and $t_{\text{th}}$ columns and rows of $A$.

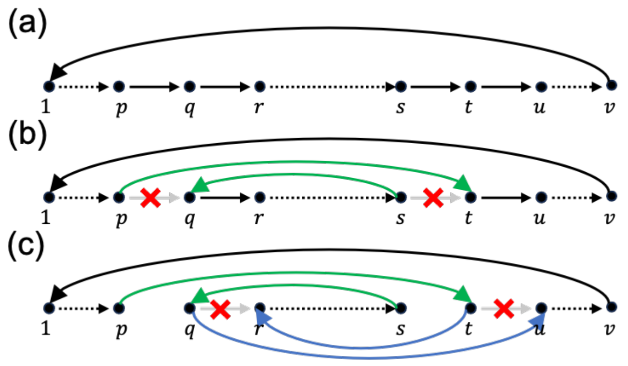


Figure 2: Illustration of an elementary operation on a Hamiltonian Cycle: $1 \to \cdots \to p \to q \to r \to \cdots \to s \to t \to u \to \cdots \to v \to 1$. (a): the original route; (b) the route corresponds to $A\mathbb{P}(q,t)^T$; (c) the route corresponds to $\mathbb{P}(q,t)A\mathbb{P}(q,t)^T$.

Now let's consider when $A$ represents a Hamiltonian Cycle: $1 \to \cdots \to p \to q \to r \to \cdots \to s \to t \to u \to \cdots \to v \to 1$. As mentioned before, the $q_{th}$ or $t_{th}$ row of matrix $A$ represents the probability distribution of directed edges originating from city $q$ or $t$, and the $q_{th}$ or $t_{th}$ column of matrix $A$ corresponds to the probability distribution of directed edges that terminate at city $q$ or $t$. That is to say: when we perform the post-multilpying operation $A\mathbb{P}(q,t)^T$, we swap the parent nodes of city $q$ and $t$, as shown in Figure 2 (b), it results in the removal of two directed edges: $p \to q$ and $s \to t$, additionally, two new directed edges are added: $p \to t$ and $s \to q$. We then apply the elementary permutation matrix $\mathbb{P}(q,t)$ to the new route, which leads to the swapping of child nodes associated with city $q$ and city $t$: we remove the directed edges $q \to r$ and $t \to u$ rom the route, and introduce the directed edges $q \to u$ and $t \to r$ instead, as shown in Figure 2 (c). Overall, the operation $\mathbb{P}(q,t)A\mathbb{P}(q,t)^T$ corresponds to a node swapping operation. It swaps the child and parent nodes of city $q$ and $t$ in the cycle, while preserving the order of the remaining nodes in the cycle. This operation maintains the edge count and doesn't introduce any new sub-loop. Therefore, the new route after swapping nodes actions still satisfies the Hamiltonian Cycle constraint.

As mentioned, the doubly stochastic matrix $\mathbb{T}$ is a differentiable relaxations of permutation matrices Adams and Zemel [2011]. Since any permutation matrix can be decomposed into a sequence of elementary permutation matrices. Thus, $\mathbb{T}$ can be interpreted as a probabilistic relaxation of a sequence of node swapping actions. Now, let's recall Equation 1, if we visualize the Sylvester shift matrix $\mathbb{V}$ in Equation 2 as a heat map, it actually represents the following Hamiltonian Cycle $\mathcal{H}_v : 1 \to 2 \to 3 \to \cdots \to n \to 1$. Hence, the TSP can be formulated as: given $\mathcal{H}_v$, our goal is to find a **permutation matrix that swaps the nodes, resulting in the shortest Hamiltonian Cycle**. The mapping $\mathbb{T} \to \mathcal{H}$ provides a probabilistic representation of the node swapping actions of the Hamiltonian Cycle $\mathcal{H}_v$.

In Min et al. [2023], the authors emphasize the significance of low entropy (non-smooth) representations. When the heat map exhibits high entropy (smooth), then the probability distribution of node-swapping actions is evenly spread, leading to a representation lacking significant meaning. In other words, if the heat map shows a smooth transition of probabilities across different regions, it suggests that all swapping actions are equally possible, making it difficult to distinguish significant swapping actions. On the contrary, by controlling the smoothness of the heat map, we can adjust the probabilities of actions to be more distinct and informative (low entropy). Ideally, we aim to regulate the "entropy" of the heat map, allowing for a clearer depiction of the probabilities associated with actions.

## 4 Build Permutation Matrix using Gumbel-Sinkhorn operator

Here, we use the Gumbel-Sinkhorn operator mentioned earlier to control the entropy of a heat map.

**Sinkhorn operator** The Sinkhorn operator $S(X)$ over $X \in \mathbb{R}^{n \times n}$ can be written as:

$$\begin{aligned} S^0(X) &= \exp(X), \\ S^l(X) &= \mathcal{T}_c\left(\mathcal{T}_r(S^{l-1}(X))\right), \\ S(X) &= \lim_{l \to \infty} S^l(X). \end{aligned} \tag{5}$$

where $\mathcal{T}_r(X) = X \oslash (X \mathbf{1}_n \mathbf{1}_n^\top)$, and $\mathcal{T}_c(X) = X \oslash (\mathbf{1}_n \mathbf{1}_n^\top X)$ as the row and column-wise normalization operators of a matrix, with $\oslash$ denoting the element-wise division and $\mathbf{1}_n$ a column vector of ones Jang et al. [2017], Maddison et al. [2017], Adams and Zemel [2011].

**Building Gumbel-Sinkhorn distribution** Mena et al. [2018] shows that we can build a differentiable approximation of the permutation matrix by introducing Gumbel noise to the Sinkhorn operator. To be specific, they show that we can build a continuous relaxation of a permutation matrix $\mathbb{T}$ by a temperature-dependent Sinkhorn operator:

$$S(\frac{X+\epsilon}{\tau}), \tag{6}$$

where $\epsilon$ is sampled from Gumbel noise, $\tau$ is the temperature and $S$ is the Sinkhorn operator. Overall, there are two parameters controlling the entropy (smoothness) of the approximation: the number of iterations in Equation 5 and the temperature parameter in Equation 6.

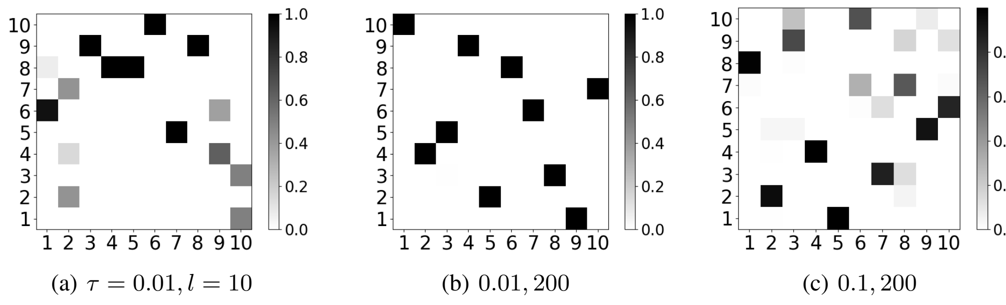

(a) $\tau = 0.01, l = 10$ (b) $0.01, 200$ (c) $0.1, 200$

Figure 3: Illustration on how the temperature parameter $\tau$ and iteration number parameter $l$ control the entropy of the representation. Higher temperature or smaller iteration number can lead to a higher entropy presentation.

Figure 3 shows how a $10 \times 10$ matrix looks like under Gumbel-Sinkhorn operator with different parameters, where $X \in \mathbb{R}^{10 \times 10}$ is a random input matrix. In our model, given a GNN's output $\mathbf{h} \in \mathbb{R}^{n \times n}$, we first construct the transition matrix $\mathbb{T}$ by applying a Gumbel-Sinkhorn Operator $S$ on $\mathbf{h}$. This operation can be denoted as: $\mathbb{T} = S\left(\frac{\mathbf{h}+\epsilon}{\tau}\right)$, we then train the model using:

$$\mathcal{L} = \lambda_1 \Big\{ \underbrace{\sum_{i=1}^{n} (\sum_{j=1}^{n} \mathbb{T}_{i,j} - 1)^2}_{\text{Column-wise constraint}} + \underbrace{\sum_{j=1}^{n} (\sum_{i=1}^{n} \mathbb{T}_{i,j} - 1)^2}_{\text{Row-wise constraint}} \Big\} + \underbrace{\lambda_2 \mathrm{tr}(\mathcal{H})}_{\text{No self-loops}} + \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{D}_{i,j} \mathcal{H}_{i,j}}_{\text{Minimize the distance}}. \tag{7}$$

Here, the entropy of $\mathbb{T}$ actually represents the clarity of the node swapping actions. A low-entropy $\mathbb{T}$ implies that the transitions between different nodes are more distinct and well-defined, which can aid in better decision-making for node swapping during the search. On the other hand, an high-entropy $\mathbb{T}$ might introduce more ambiguity in the node swapping actions, potentially leading to suboptimal solutions. By adjusting the parameters in the Gumbel-Sinkhorn operator, we can control the entropy of $\mathbb{T}$ and, consequently, fine-tune the optimization process to achieve better results for the TSP. Note that the Gumbel-Sinkhorn Operator does not guarantee the matrix to be precisely doubly-stochastic. Therefore, we additionally impose Column-wise and Row-wise constraints.

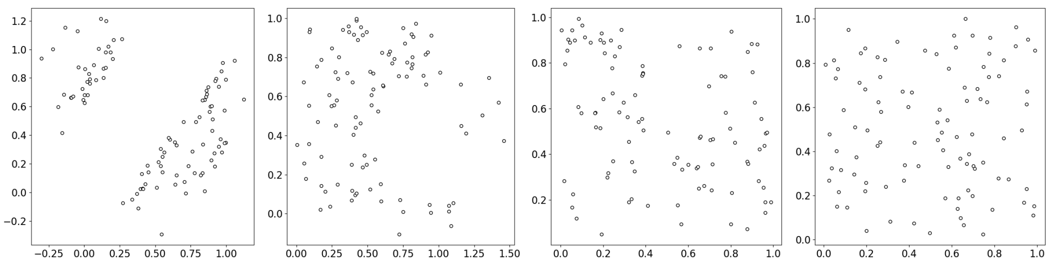

(a) Expansion (b) Explosion (c) Implosion (d) Uniform

Figure 4: Illustration of different distributions on a 2D plane.

**The entropy and generalization trade-off** We first introduce Dirichlet Energy as a metric to measure the entropy of $\mathbb{T}$ to understand how the entropy affects the learned representations. The Dirichlet Energy enables us to quantify the extent of smoothness in a $d$-dimensional representation $f \in \mathbb{R}^{n,d}$. The Dirichlet Energy is:

$$E = \text{Trace}(f^T \Delta f), \tag{8}$$

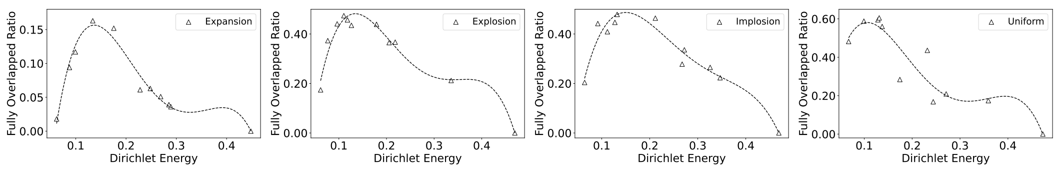
where $\Delta = I_n - D^{-1/2} W D^{-1/2} \in \mathbb{R}^{n \times n}$ is the graph Laplacian, $W \in \mathbb{R}^{n \times n}$ is the adjacency matrix and $D \in \mathbb{R}^{n \times n}$ is the degree matrix of the graph. The Dirichlet Energy $E$ can be interpreted as a measure of how much a function defined on the vertices of the graph deviates from being constant. Given a Laplacian $\Delta$, consider the eigenvectors $\psi_1, \psi_2, ..., \psi_n$ with eigenvalues $\mu_1 < \mu_2 < ... < \mu_n$, $\psi_1$ corresponds to a constant function. Subsequent eigenvectors correspond to increasingly oscillatory patterns. Functions that are "smoother" with respect to the graph structure will have lower Dirichlet Energy $E$ and high entropy.

To delve into how entropy affects the model's performance, we follows Min et al. [2023] and employ overlap coefficient $\eta$ as a descriptor for generalization assessment. Given a heat map $\mathcal{H}$, we implement these steps: 1: We select the $M$ largest elements in each row of $\mathcal{H}$ (excluding diagonal elements) and set the remaining $n - M$ elements to 0; 2: The modified heat map is symmetrized to yield $\mathcal{H}' = \tilde{H} + \tilde{H}^T$. We define $\mathbf{E}_{ij} \in \{0, 1\}$ to indicate the presence of an undirected edge $(i, j)$ in our prediction. Without loss of generality, we assume $0 < i < j \leq n$. The definition of $\mathbf{E}_{ij}$ is as follows:

$$\mathbf{E}_{ij} = \begin{cases} 1, & \text{if } \mathcal{H}'_{ij} = \mathcal{H}'_{ji} > 0 \\ 0, & \text{otherwise} \end{cases}.$$

Let $\Pi$ denote the set of undirected edges in $\mathcal{H}'$ with $\mathbf{E}_{ij} = 1$, and $\Gamma$ represent the ground truth edge set. The overlap coefficient $\eta$ is defined as the ratio of the number of edges in the intersection of $\Gamma$ and $\Pi$ to the total number of edges in $\Gamma$,where $\eta = \frac{|\Gamma \cap \Pi|}{|\Gamma|}$. The value of $\eta$ indicates how well our predicted edge set $\Pi$ covers the ground truth solution $\Gamma$. If $\eta = 1$, it implies that $\Gamma$ is a subset of $\Pi$, demonstrating successful inclusion of all ground truth edges in our prediction.

We then train and test the model on four different distributions: expansion, explosion, implosion, and uniform. The visual representations of these four distributions are depicted in Figure 4. We experiment with various $\tau$ and $l$ values on the TSP 100 dataset, leading to different levels of entropy (smoothness) in the models. Subsequently, we evaluate these models on 4 different distributions. The outcomes are depicted in Figure 5, where, the $x$-axis represents the Dirichlet Energy $E$ and the $y$-axis corresponds to the fully overlapped ratio with the ground truth solutions, we use 1,000 test samples for each distribution. Figure 5 shows how the entropy (Dirichlet Energy) of heat map



(a) Expansion (b) Explosion (c) Implosion (d) Uniform

Figure 5: Generated using different $\tau$,$l$ and learning rate.

affects the generalization performance. We observe a trade-off between entropy and generalization. In other words, the resulting heat map becomes less informative when the Dirichlet Energy $E$ is lower, indicating a high entropy representation with equally possible swapping actions. When the Dirichlet Energy is higher ($> 0.3$), indicating a low entropy condition, the model is no longer a soft relaxation of a sequence of node swapping actions. Instead, it becomes limited to generating only very few swapping actions, which also hurts the generalization. Our findings indicate that both low Dirichlet Energy (high entropy) and high Dirichlet Energy (low entropy) hurt the model's generalization capability. To achieve the best generalization performance, it is crucial to control the smoothness of the representation.

## 5 Hardness Generalization

Figure 5 also shows that various distributions exhibit distinct levels of generalization ability. For instance, the expansion distribution shows a fully overlapped ratio of 0.15, while the explosion distribution demonstrates a higher fully overlapped ratio of 0.50. Here, we aim to borrow a model from one distribution and leverage it to enhance the overlapped ratio in another distribution. In other words, our objective is to address the challenge of generalization across these different distributions.

**TSP Difficulty and the Phase Transition** Before exploring generalization across different distributions, we start by constructing a hardness-based descriptor for each distribution. This descriptor allows us to measure the characteristics of instances for a given distribution to gain insights into which features of an instance make our model perform well or poorly Smith-Miles et al. [2010].

Cheeseman et al. [1991] show that computational difficulty is often associated with this phase transition. The phenomenon of phase transitions in combinatorial problems has been observed in various domains, such as Boolean satisfiability Kirkpatrick and Selman [1994], Mitchell et al. [1992] and graph coloring Cheeseman et al. [1991]. Gent and Walsh [1996] identify a difficulty parameter $\gamma$ for the two-dimensional Euclidean traveling salesman problem. They demonstrate that when dealing with random instances of the problem, there exists a rapid transition between soluble and insoluble TSP instances at a critical value of $\gamma$. The difficult parameter $\gamma$ is defined as:

$$\gamma = (l_{opt}/\sqrt{nA} - 0.78)n^{1/1.5}, \tag{9}$$

where $l_{opt}$ is the optimal tour lengths, $n$ is the number of cities and $A$ represents the area covered by the instance. By testing on various instances, they observe a mean value of $\gamma = 0.58$, which aligns with the phase transition point and represents the most challenging condition, often referred to as the hardest condition. We compute the levels of difficulty for various distributions shown in Figure 4. Our findings indicate that the order from easy to hard is as follows: expansion, explosion, implosion, and uniform, as depicted in Figure 6(a).

We evaluate the generalization performance across these four distributions on TSP 100. As shown in Figure 6(b), where the $x$-axis represents the test dataset and different bars correspond to the training dataset. The $y$ axis is the fully overlap ratio with ground truth. Our results suggest that directly learning on easy instances may result in a lack of generalization power for the GNN when applied to other instances within the same distribution. On the other hand, learning on harder instances can enable the GNN to generalize not only to those difficult instances but also to easier ones. For example, when using an expansion distribution as the test dataset and training the model using expansion TSP, we observe a fully overlapped ratio of 0.133. However, when we train models on harder datasets such as explosion, implosion, and uniform data, we observe ratios of 0.258, 0.233, and 0.179, respectively.

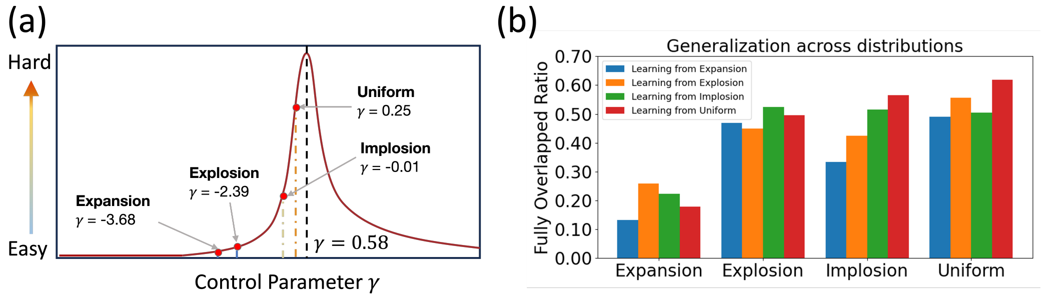


Figure 6: (a): Hardness of different distributions; (b): Generalization across different distributions.

Table 1: Results of using different models + Local Search. Each test dataset consists of 1000 TSP instances, the Search is adapted from UTSP Min et al. [2023].

| Train / Test | Expansion | Explosion | Implosion | Uniform |
|---|---|---|---|---|
| Expansion | 0.002% | -0.007% | -0.005% | -0.006% |
| Explosion | - | 0.005% | -0.001% | -0.003% |
| Implosion | - | - | 0.002% | -0.004% |
| Uniform | - | - | - | -0.001% |

On the other hand, when using the hardest dataset, uniform, as the test dataset, training the model using uniform data yields a ratio of 0.619. In comparison, the models trained on expansion, explosion, and implosion datasets exhibit ratios of 0.49, 0.556, and 0.505, respectively.

We then study how these various models affect the TSP solution quality. Given a learned GNN model, we use the local search method proposed in Min et al. [2023] to decode the final solution. Our results are summarized in Table 1. We observe that models learned on harder instances are able to facilitate the search process on easier instances, resulting in improved solution quality.

Overall, our results suggest that when the GNNs are trained solely on easy instances, the models may not be exposed to the diverse range of patterns and complexities present in more challenging instances. As a result, the learned models are not be capable of effectively capturing and generalizing the underlying structures and relationships in the data. This limited generalization ability can hinder its performance when applied to unseen or more difficult instances. By contrast, training the GNN on harder instances forces the model to learn and adapt to a broader range of scenarios. The increased complexity and variability in these instances provide a richer training environment, allowing the GNN to capture more intricate patterns and relationships during the unsupervised training. As a result, the learned model becomes more robust and capable of generalizing not only to the instances from same distribution but also to easier instances.

## 6 Conclusion

In this work, we apply the Gumbel-Sinkhorn operator to the TSP. We first reformulate the TSP as *Identifying a permutation matrix that reorders the nodes to yield the shortest Hamiltonian Cycle*. Our model is trained unsupervised, and by controlling entropy with the Gumbel-Sinkhorn operator, we observe a trade-off between entropy and the generalization of TSP models. Refining the operator's parameters helps improve the generalization. We further study how to generalize a learned model across different distributions. Our results show that models learned from harder instances exhibit better generalization capability. This highlights the importance of training with harder instances to augment the model's capacity to generalize to unseen instances. We anticipate that our model will motivate researchers to employ machine learning and optimal transport techniques in addressing more NP-hard problems using a data-driven approach.

**Acknowledgement** We appreciate the constructive suggestions provided by the reviewers. This project is partially supported by the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a Schmidt Futures program; the National Science Foundation (NSF) and the National Institute of Food and Agriculture (NIFA); the Air Force Office of Scientific Research (AFOSR); the Department of Energy; and the Toyota Research Institute (TRI).

## References

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=Byt3oJ-0W`.

Ryan Prescott Adams and Richard S. Zemel. Ranking via sinkhorn propagation. *CoRR*, abs/1106.1925, 2011. URL `http://arxiv.org/abs/1106.1925`.

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

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

Scott Linderman, Gonzalo Mena, Hal Cooper, Liam Paninski, and John Cunningham. Reparameterizing the birkhoff polytope for variational permutation inference. In *International Conference on Artificial Intelligence and Statistics*, pages 1618–1627. PMLR, 2018.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3056–3065, 2019.

Kaiyue Pang, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Solving mixed-modal jigsaw puzzle for fine-grained sketch-based image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10347–10355, 2020.

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

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

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour dhorizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

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.

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

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *arXiv preprint arXiv:2302.08224*, 2023.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. *arXiv preprint arXiv:2303.10538*, 2023.

Ian P. Gent and Toby Walsh. The TSP phase transition. *Artif. Intell.*, 88(1-2):349–358, 1996. doi: 10.1016/S0004-3702(96)00030-6. URL `https://doi.org/10.1016/S0004-3702(96)00030-6`.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=rkE3y85ee`.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=S1jE5L5gl`.

Kate Smith-Miles, Jano Van Hemert, and Xin Yu Lim. Understanding tsp difficulty by learning from evolved instances. In *Learning and Intelligent Optimization: 4th International Conference, LION 4, Venice, Italy, January 18-22, 2010. Selected Papers 4*, pages 266–280. Springer, 2010.

Peter C Cheeseman, Bob Kanefsky, William M Taylor, et al. Where the really hard problems are. In *Ijcai*, volume 91, pages 331–337, 1991.

Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264(5163):1297–1301, 1994.

David Mitchell, Bart Selman, Hector Levesque, et al. Hard and easy distributions of sat problems. In *Aaai*, volume 92, pages 459–465, 1992.

## 7 Appendix

### 7.1 Proof

**Theorem:** Any general permutation matrix $\mathbb{T}$ can be decomposed into a sequence of elementary matrices.
**Proof:** Let $\mathbb{T}$ be a general permutation matrix. We will show that $\mathbb{T}$ can be decomposed into a sequence of elementary matrices. We begin by considering an identity matrix $\mathbb{I}_n \in \mathbb{R}^{n\times n}$ and initializing it as the starting point for the decomposition process.

- For each row $i$ from 1 to $n$, we examine the $i_{\text{th}}$ row of $\mathbb{T}$ to identify the index $j$ where the element is 1.
- If $j \neq i$, we perform a row swap to bring the $i_{\text{th}}$ row to the $j_{\text{th}}$ position. We use a sequence of elementary matrices. We apply the elementary matrix $\mathbb{P}(i, j)$ to the current matrix.

By iteratively applying the appropriate elementary matrices for each row, we eventually obtain a matrix in which $\mathbb{T}$ is transformed into an identity matrix, as the rows are permuted to match the original permutation. Thus, we have shown that any general permutation matrix $\mathbb{T}$ can be decomposed into a sequence of elementary matrices, an example can be found in Equation 20. Since every general permutation matrix can be decomposed into a sequence of elementary matrices, $\mathbb{T}$ can be interpreted as a sequence of actions. That is to say, given a $\mathbb{T}$ in Figure 7, we have:

$$\mathbb{P}(2,6)\mathbb{P}(6,5)\mathbb{P}(5,1)\mathbb{P}(2,4)\mathbb{P}(1,3)\mathbb{T} = \mathbb{I}_6, \tag{10}$$

we then have:

$$\mathbb{T} = \mathbb{P}(1,3)\mathbb{P}(2,4)\mathbb{P}(5,1)\mathbb{P}(6,5)\mathbb{P}(2,6), \tag{11}$$

and

$$\mathbb{T}^T = \mathbb{P}(2,6)^T\mathbb{P}(6,5)^T\mathbb{P}(5,1)^T\mathbb{P}(2,4)^T\mathbb{P}(1,3)^T. \tag{12}$$

### 7.2 Example

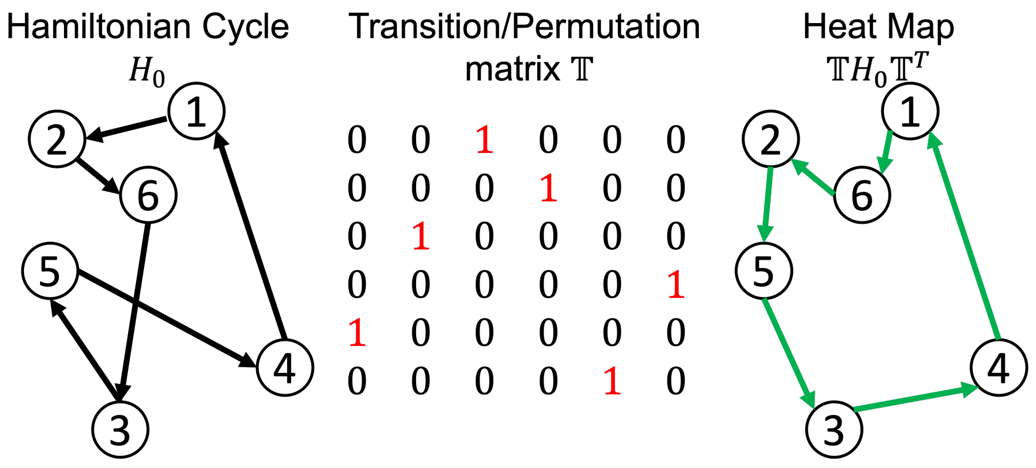


Figure 7: Illustration of applying a general operation on $\mathcal{H}_0$.

A general permutation matrix is a square matrix that represents a permutation of its rows or columns. There is exactly one nonzero entry in each row and each column. It rearranges the elements of an identity matrix based on a specific permutation. Let's consider a general permutation matrix $\mathbb{T} \in \mathbb{R}^{n\times n}$.

Given a Hamiltonian Cycle $\mathcal{H}_0$ in Figure 7, $\mathbb{T}\mathcal{H}_0\mathbb{T}^T$ represents applying a sequence of swap operation on $\mathcal{H}_0$. $\mathbb{T} \rightarrow \mathcal{H}$ transformation corresponds to 5 swap operations: swap city $(2,6)$, $(5,6)$, $(1,5)$, $(2,4)$

$$
\begin{aligned}
\mathbb{T}\mathcal{H}_0\mathbb{T}^T =& \mathbb{P}(1,3)\mathbb{P}(2,4)\mathbb{P}(5,1)\mathbb{P}(6,5)\underbrace{\mathbb{P}(2,6)\mathcal{H}_0\mathbb{P}(2,6)^T}_{\text{swap 2,6}}\mathbb{P}(6,5)^T\mathbb{P}(5,1)^T\mathbb{P}(2,4)^T\mathbb{P}(1,3)^T \\
&= \mathbb{P}(1,3)\mathbb{P}(2,4)\mathbb{P}(5,1)\underbrace{\mathbb{P}(6,5)\mathcal{H}_1\mathbb{P}(6,5)^T}_{\text{swap 5,6}}\mathbb{P}(5,1)^T\mathbb{P}(2,4)^T\mathbb{P}(1,3)^T \\
&= \mathbb{P}(1,3)\mathbb{P}(2,4)\underbrace{\mathbb{P}(5,1)\mathcal{H}_2\mathbb{P}(5,1)^T}_{\text{swap 1,5}}\mathbb{P}(2,4)^T\mathbb{P}(1,3)^T \\
&= \mathbb{P}(1,3)\underbrace{\mathbb{P}(2,4)\mathcal{H}_3\mathbb{P}(2,4)^T}_{\text{swap 2,4}}\mathbb{P}(1,3)^T = \underbrace{\mathbb{P}(1,3)\mathcal{H}_4\mathbb{P}(1,3)^T}_{\text{swap 1,3}}.
\end{aligned}
\tag{13}
$$

and $(1,3)$, as shown in Equation 13. More specifically, these different routes are:

$$1 \to 2 \to 6 \to 3 \to 5 \to 4 \to 1 \;\; \mathcal{H}_0, \text{swap}(2,6); \tag{14}$$
$$1 \to 6 \to 2 \to 3 \to 5 \to 4 \to 1 \;\; \mathcal{H}_1, \text{swap}(5,6); \tag{15}$$
$$1 \to 5 \to 2 \to 3 \to 6 \to 4 \to 1 \;\; \mathcal{H}_2, \text{swap}(1,5); \tag{16}$$
$$5 \to 1 \to 2 \to 3 \to 6 \to 4 \to 5 \;\; \mathcal{H}_3, \text{swap}(2,4); \tag{17}$$
$$5 \to 1 \to 4 \to 3 \to 6 \to 2 \to 5 \;\; \mathcal{H}_4, \text{swap}(1,3); \tag{18}$$
$$5 \to 3 \to 4 \to 1 \to 6 \to 2 \to 5 \;\; \mathbb{T}\mathcal{H}_0\mathbb{T}^T. \tag{19}$$

$$
\begin{aligned}
&\begin{bmatrix} 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&1\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0 \end{bmatrix}
\xrightarrow{\mathbb{P}(1,3)}
\begin{bmatrix} 0&1&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&1\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0 \end{bmatrix}
\xrightarrow{\mathbb{P}(2,4)}
\begin{bmatrix} 0&1&0&0&0&0\\ 0&0&0&0&0&1\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0 \end{bmatrix} \\
&\xrightarrow{\mathbb{P}(5,1)}
\begin{bmatrix} 1&0&0&0&0&0\\ 0&0&0&0&0&1\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&1&0 \end{bmatrix}
\xrightarrow{\mathbb{P}(6,5)}
\begin{bmatrix} 1&0&0&0&0&0\\ 0&0&0&0&0&1\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&1&0&0&0&0 \end{bmatrix}
\xrightarrow{\mathbb{P}(2,6)} \mathbb{I}_6
\end{aligned}
\tag{20}
$$