A Ablation study of normalization

A.1 For LEHD model

In Table 5, we explore the effects of eliminating normalization from the attention layer in our LEHD model. We train three LEHD models with the same training scheme and training budget, differing solely in the attention layer: one with batch normalization (BN), one with instance normalization (IN), and one without normalization (w/o). Our experimental results demonstrate that the LEHD model without normalization in the attention layer significantly outperforms the other two models with normalization.

Table 5: Effect of normalization for LEHD model.

	TSP100	TSP200	TSP500	TSP1000
BN	0.775%	1.312%	3.808%	12.209%
IN	0.640%	1.197%	34.391%	222.730%
w/o	0.577%	0.859%	1.560%	3.168%

A.2 For POMO model

We also compare the performance of POMO [30] models with different types of normalization: one with batch normalization (BN), one with instance normalization (IN), and one without normalization (w/o) in Table 6. We train all three POMO models with the same reinforcement learning method with POMO strategy and training budget (1000 epochs). The results show that different types of normalization have few effects on the POMO model.

Table 6: Effect of normalization for POMO model.

	TSP100	TSP200	TSP500	TSP1000
BN	1.325%	5.502%	27.616%	41.631%
IN	1.449%	5.602%	27.454%	41.748%
w/o	1.321%	4.990%	28.598%	45.747%

The results in Table 6 show that removing normalization from attention layer has little impact on the model with a heavy encoder and a light decoder. However, the results in Table 5 show that removing normalization from attention layer has a positive impact on the performance of the LEHD model, but it is not the critical factor for the LEHD model's strong generalization ability since the LEHD model with batch normalization still performs significantly better than POMO and SGBS in the case of TSP1000. Instead, we can conclude that the underlying reason for the model's strong generalization ability lies in the heavy decoder structure.

B Implementation details for TSP

B.1 Problem setup

The task of solving a TSP instance with n nodes involves finding the shortest loop that visits each node exactly once and eventually returns to the first visited node . We generate TSP instances following the approach in [28], where the coordinates of n nodes are sampled uniformly at random from the unit square.

B.2 Implementation details

For a TSP instance $\mathbf{S}$, the node features $(\mathbf{s}_1, \dots, \mathbf{s}_n)$ are the 2-dimentional coordinates of the n nodes in the graph.

Similar to AM, the encoder produces the node embedding $\mathbf{h}_i$ for $i = 1, \dots, n$.

In the original AM decoder, irrelevant nodes are masked during each construction step. In our model, we remove the embeddings of irrelevant nodes from the decoder input. This removal serves the same purpose as masking them in every decoder attention layer but also saves computational resources since the decoder is not required to perform computations related to irrelevant nodes. Consequently, for each construction step, the input node embeddings for the decoder consist of the starting node embedding, the destination node embedding, and the avaliable node embeddings.

Here is an extended explanation of Equation 2 in the case of TSP. After L attention layers, $H^{(0)}$ is transformed to $H^{(L)} = \{\mathbf{h}_i^{(L)}, i \notin \{x_{2:t-2}\}\}$, and each vector $\mathbf{h}_i^{(L)} \in \mathbb{R}^d$ is transformed into a scalar o_i by applying the linear projection $W_O \in \mathbb{R}^{d \times 1}$, i.e. $o_i = W_O \mathbf{h}_i^{(L)}$. When calculating the probability $P_t = \mathrm{softmax}(O)$, the scalars $o_i, i = \{x_1, x_{t-1}\}$ corresponding to the starting node and destination node are masked.

C Implementation details for CVRP

C.1 Problem setup

A CVRP instance involves n customer nodes and one depot node, with each customer node i having a specific demand δ_i that must be fulfilled. We aim to determine a set of sub-tours starting and ending at the depot such that the sum of demand satisfied by each sub-tour is within the capacity constraint D of the vehicle. Given the capacity constraint D, the objective is to minimize the total distance of the set of sub-tours. Similarly, following [28], we generate CVRP instances where the coordinates of custormer nodes and depot nodes are sampled uniformly from the unit square. The demand δ_i is sampled uniformly form $\{1, \ldots, 9\}$. And the vehicle capacity $D = 50, 80, 100, 250$ for $N = 100, 200, 500, 1000$, respectively.

Following [12, 29], we define the formation of a feasible solution for CVRP. Rather than treating a visit to the depot as a separate step, we use binary variables to indicate whether a customer node is reached via the depot or another customer node. Specifically, in a feasible solution, a node is assigned a value of 1 if it is reached via the depot and a value of 0 if it is reached through another customer node.

For example, a feasible CVRP solution $\{0, 1, 2, 3, 0, 4, 5, 0, 6, 7, 0\}$ where 0 represents the depot, can be denoted as follows:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{5}$$

In this notation, the first row represents the sequence of visited nodes in the solution, and the second row indicates whether each node is reached via the depot or another customer node.

The purpose of using this notation is to ensure solution alignment. In CVRP instances, solutions with the same number of customer nodes may have varying numbers of sub-tours, leading to potential misalignment. By employing this notation, we can avoid such issues.

C.2 Implementation details

For CVRP, the node feature $\mathbf{s}_i$ is represented as a 3-dimensional vector, comprising the 2-dimensional coordinates and the demand of node i. The demand of the depot is assigned a value of 0. Without loss of generality, we normalize the vehicle capacity D to $\hat{D} = 1$, and the demand δ_i to $\hat{\delta}_i = \frac{\delta_i}{D}$ [28].

In the decoder, the dynamically changing remaining capacity is added to both the starting node and destination node embeddings, resembling the approach employed in [30]. Similar to TSP, the irrelevant node embeddings are excluded from the decoder input.

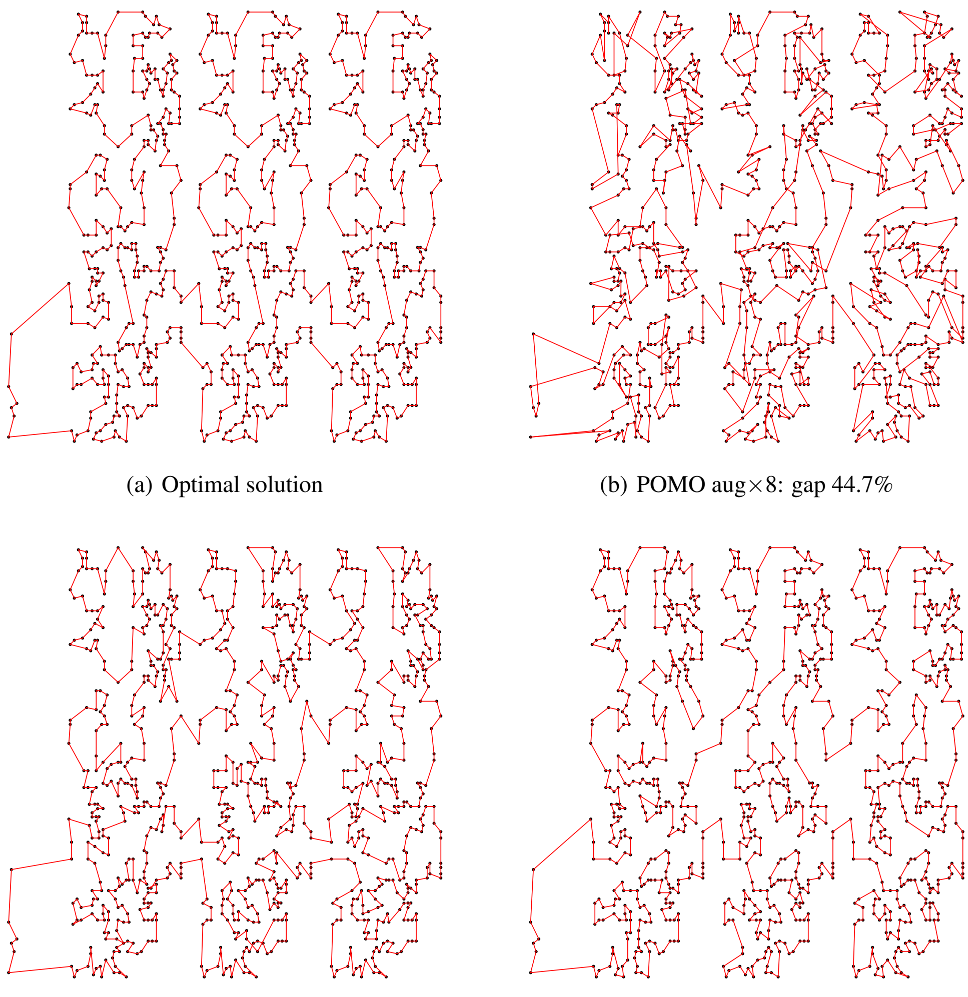
Here is an extended explanation of Equation 2 in the case of CVRP. After L attention layers, $H^{(0)}$ is transformed to $H^{(L)} = \{\mathbf{h}_i^{(L)}, i \notin \{x_{2:t-2}\}\}$. Each vector $\mathbf{h}_i^{(L)} \in \mathbb{R}^d$ is projected to a 2-dimensional vector o_i using the linear projection $W_O \in \mathbb{R}^{d \times 2}$, i.e. $o_i = W_O \mathbf{h}_i^{(L)}$. Each o_i corresponds to two actions associated with the node i: either being reached via the depot or another customer node. This relation corresponds to the notation mentioned in equation 5. Subsequently, resembling the approach employed in [12], O is flattened, and the softmax is utilized to compute the probability associated with each possible action. The actions associated with the starting node and destination node are masked.

D Solution Visualizations

Table 2 shows the test results on TSPLib and CVRPLib instances with different sizes and distributions. For TSPLib, we report the results on 2D Euclidean TSP instances with size smaller than 5000 (up to 4461). For CVRPLib, we report the results on the instances without additional constraints such as time windows.

Figures 3, 4 show the solutions of two instances in TSPLib. Figures 5, 6 show the solutions of two instances in CVRPLib. For each figure, panel (a) shows the optimal solution, panel (b), (c), and (d) show the solution generated by POMO, BQ, and LEHD, respectively.

D.1 Solution visualizations of two TSPLib instances



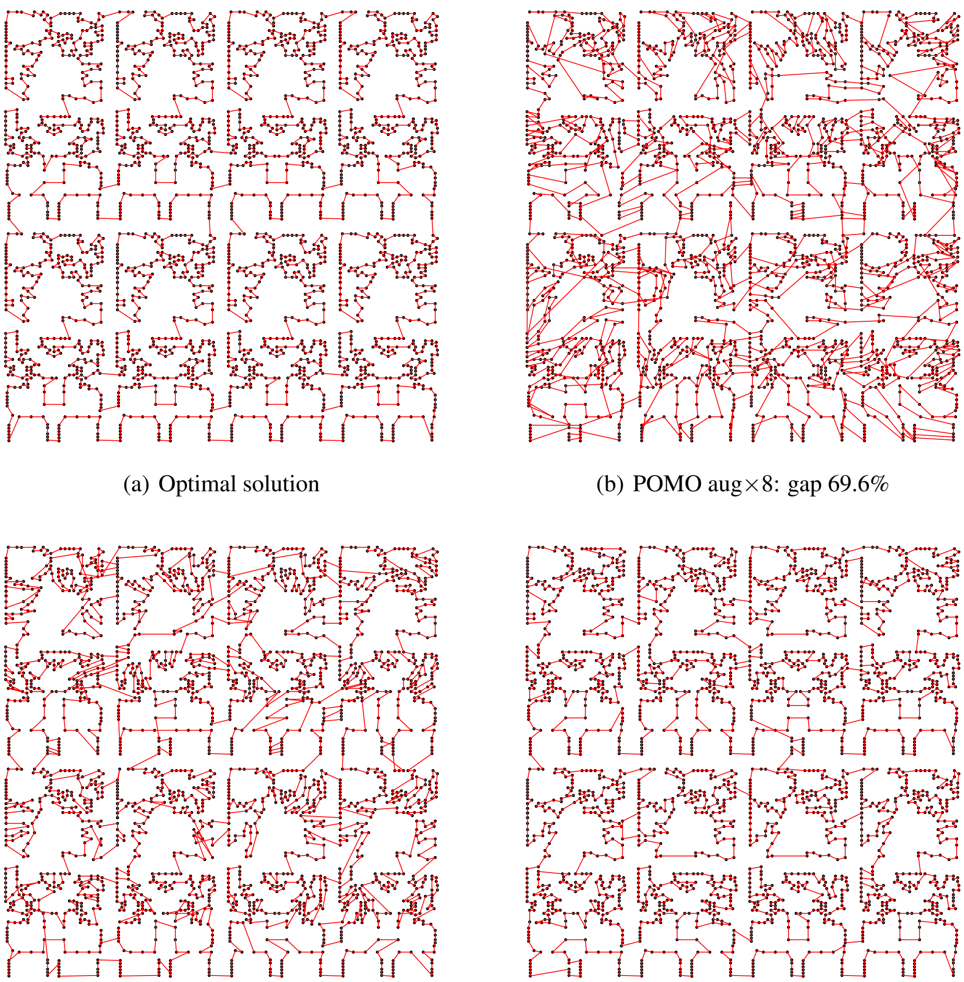
(a) Optimal solution

(b) POMO aug×8: gap 44.7%

(c) BQ bs16: gap 7.4%

(d) LEHD RRC 1000: gap 1.6%

Figure 3: Instance pr1002 with 1002 nodes.



(a) Optimal solution

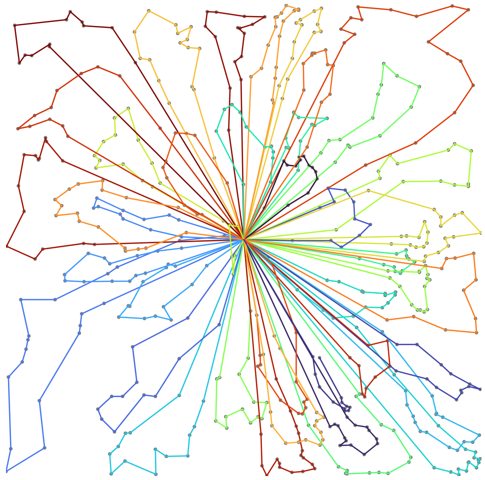
(b) POMO aug×8: gap 69.6%

(c) BQ bs16: gap 29.4%

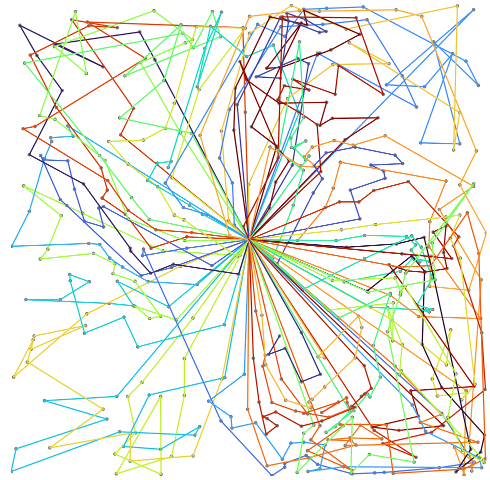
(d) LEHD RRC 1000: 5.5%

Figure 4: Instance pr2392 with 2392 nodes.

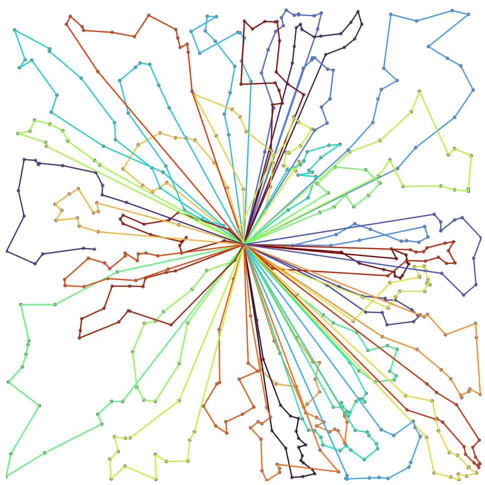
D.2 Solution visualizations of two CVRPLib instances



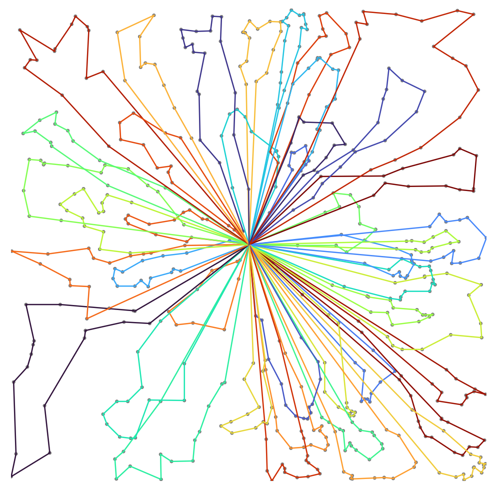
(a) Optimal solution



(b) POMO aug×8: gap 65.6%

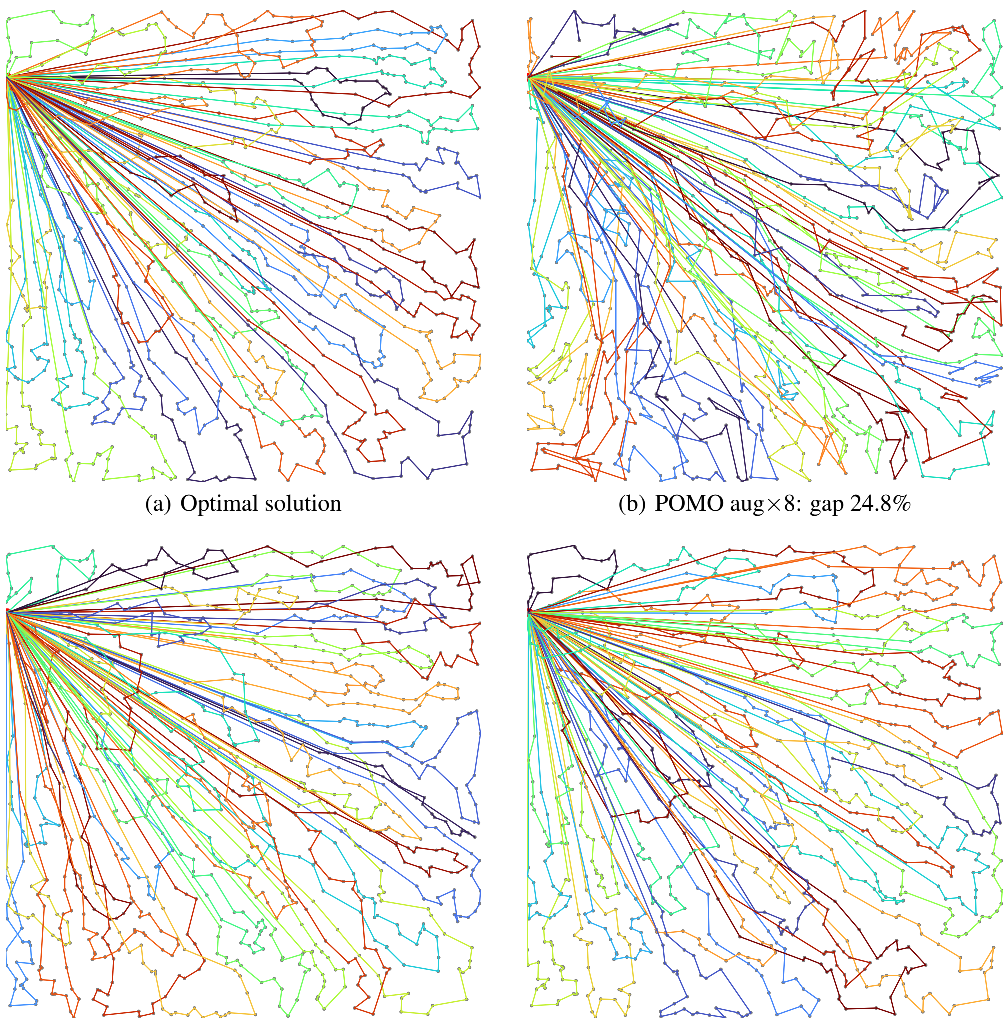


(c) BQ bs16: gap 4.7%



(d) LEHD RRC 1000: gap 0.8%

Figure 5: Instance X-n561-k42 with 560 customer nodes.



(a) Optimal solution

(b) POMO aug×8: gap 24.8%

(c) BQ bs16: gap 6.9%

(d) LEHD RRC 1000: gap 3.1%

Figure 6: Instance X-n1001-k43 with 1000 customer nodes

E Licenses

Table 7: List of licenses for the codes and datasets we used in this work

Resource	Type	Link	License
OR-Tools [43]	Code	https://github.com/google/or-tools	Apache License 2.0
LKH3 [16]	Code	http://webhotel4.ruc.dk/ keld/research/LKH-3/	Available for academic research use
HGS [50]	Code	https://github.com/chkwon/PyHygese	MIT License
Concorde [2]	Code	https://github.com/jvkersch/pyconcorde	BSD 3-Clause License
POMO [30]	Code	https://github.com/yd-kwon/POMO	MIT License
Att-GCN+MCTS [13]	Code	https://github.com/SaneLYX/TSP_Att-GCRN-MCTS	MIT License
EAS [20]	Code	https://github.com/ahottung/EAS	Available online
SGBS [8]	Code	https://github.com/yd-kwon/SGBS	MIT License
TSPLib	Dataset	http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/	Available for any non-commercial use
CVRPLib	Dataset	http://vrp.galgos.inf.puc-rio.br/index.php/en/	Available for academic research use

The licenses for the codes and the datasets used in this work are listed in Table 7.