Equity-Transformer: Solving NP-Hard Min-Max Routing Problems as Sequential Generation with Equity Context

Jiwoo Son*[1,2], Minsu Kim*[1], Sanghyeok Choi[1], Hyeonah Kim[1], Jinkyoo Park[1,2]

[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Omelet
{sonleave25, min-su, sanghyeok.choi, hyeonah_kim, jinkyoo.park}@kaist.ac.kr

Abstract

Min-max routing problems aim to minimize the maximum tour length among multiple agents by having agents conduct tasks in a cooperative manner. These problems include impactful real-world applications but are known as NP-hard. Existing methods are facing challenges, particularly in large-scale problems that require the coordination of numerous agents to cover thousands of cities. This paper proposes Equity-Transformer to solve large-scale min-max routing problems. First, we employ sequential planning approach to address min-max routing problems, allowing us to harness the powerful sequence generators (e.g., Transformer). Second, we propose key inductive biases that ensure equitable workload distribution among agents. The effectiveness of Equity-Transformer is demonstrated through its superior performance in two representative min-max routing tasks: the min-max multi-agent traveling salesman problem (min-max mTSP) and the min-max multi-agent pick-up and delivery problem (min-max mPDP). Notably, our method achieves significant reductions of runtime, approximately 335 times, and cost values of about 53% compared to a competitive heuristic (LKH3) in the case of 100 vehicles with 1,000 cities of mTSP. We provide reproducible source code: https://github.com/kaist-silab/equity-transformer.

Introduction

Routing problems are combinatorial optimization problems that are notoriously difficult to solve. The traveling salesman problem (TSP) and vehicle routing problems (VRPs) are representative problems where the objective is to determine the optimal or shortest tour route(s) for one or multiple agents, such as robots, vehicles, or drones. These problems are classified as NP-hard, posing significant challenges (Papadimitriou 1977). Various approaches have been proposed to solve routing problems, including mathematical programming techniques that aim to achieve provable optimality (Gurobi Optimization, LLC 2023; David Applegate and Cook 2023), task-specific heuristic solvers (Helsgaun 2017; Perron and Furnon 2019), and deep learning-based methods that provide task-agnostic and fast heuristic solvers (Khalil et al. 2017; Kool, van Hoof, and Welling 2019). The deep learning-based methods have shown promising results even for large-scale problems that have more than 2,000 cities (Fu, Qiu, and Zha 2021; Qiu, Sun, and Yang 2022; Sun and Yang 2023; Zhang et al. 2023; Sun et al. 2023).

Min-max routing problems are distinct from standard (min-sum) routing problems in that they focus on minimizing the cost of the most expensive route among multiple agents (i.e., minimizing the total completion time), rather than minimizing the sum of the costs of routes. These problems are particularly relevant in time-critical applications such as disaster management (Cheikhrouhou and Khoufi 2021), where minimizing completion time (or service time) is crucial. However, min-max routing problems are far more challenging than min-sum counterparts because algorithms for min-max routing require coordinated cooperation among multiple agents to ensure an equitable assignment of workload among them. Classical exact algorithms struggle to solve min-max routing problems due to their NP-hardness (França et al. 1995). Additionally, powerful heuristic approaches for min-sum problems are not well generalized to the min-max case (Bertazzi, Golden, and Wang 2015), particularly for large-scale problems (Kim, Park, and Park 2023), owing to the inherent differences between min-max and min-sum problems.

Recently, deep learning methods have been utilized to address min-max routing problems (Hu, Yao, and Lee 2020; Cao, Sun, and Sartoretti 2021; Park, Kwon, and Park 2023) as an alternative to classical approaches. Notably, representative min-max routing techniques such as ScheduleNet (Park, Kwon, and Park 2023) and the decentralized attention network (DAN) (Cao, Sun, and Sartoretti 2021) aim to handle the min-max nature with event-based *parallel planning*. They model a parallel decision-making process among multiple agents in a decentralized way. The parallel planning methods can be directly applied as a real-time dispatcher, which is advantageous in handling dynamic situations where states contain stochastic changes.

However, parallel planning encounters challenges in modeling decentralized decision-making, which necessitates searching for the joint space of agents' actions. These challenges become particularly pronounced when attempting to apply parallel planning to large-scale routing problems (Park, Kwon, and Park 2023). On the contrary, sequential planning presents an alternative approach involving a hierarchical decomposition of action choices among agents. This

*These authors contributed equally.

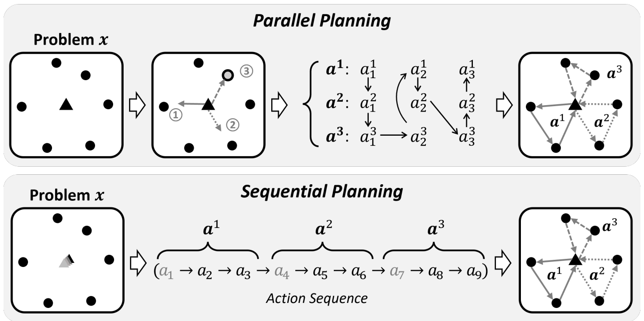

Parallel Planning
Problem x
①
②
③
$\boldsymbol{a}^1$: a_1^1 a_2^1 a_3^1
$\boldsymbol{a}^2$: a_1^2 a_2^2 a_3^2
$\boldsymbol{a}^3$: a_1^3 a_2^3 a_3^3
$\boldsymbol{a}^1$
$\boldsymbol{a}^2$
$\boldsymbol{a}^3$
Sequential Planning
Problem x
$\boldsymbol{a}^1$
$\boldsymbol{a}^2$
$\boldsymbol{a}^3$
$(a_1 \to a_2 \to a_3 \to a_4 \to a_5 \to a_6 \to a_7 \to a_8 \to a_9)$
Action Sequence

Figure 1: Illustration of parallel planning and sequential planning on min-max VRPs. In sequential planning, when an action selects a depot index (gray colored a_t), the planning for currently active agent is terminated to start planning for the new agent that corresponds to the selected depot.

results in a substantial reduction in modeling complexity when compared to parallel planning. However, an important drawback of sequential planning lies in its diminished relational context between agents due to its sequential representation, which might lead to imbalanced tours among agents, i.e., increased tour costs. Figure 1 illustrates the difference between parallel planning and sequential planning.

To achieve equitable assignment and keep leveraging reduced complexity via sequential planning, we propose a novel sequential planning architecture, *Equity-Transformer*. Specifically, we tackle the min-max routing problems by generating one long sequence via Equity-Transformer, where each sub-sequence represents a specific agent's tour. To contextualize relational decision-making and ensure equitable workload assignment among the agents, Equity-Transformer introduces two essential inductive biases as follows:

- **Multi-agent positional encoding for order bias.** We introduce virtual orders on agents to model a parallel decision-making process as a sequence. The homogeneous agents are modeled with the precedence (i.e., order bias among agents), we add positional encoding and inject it into the encoder.
- **Context encoder for equity.** To promote equitable tours for multiple agents, we incorporate an equity context into the sequence generator. Equity context considers the temporal tour length, the target tour length, and the desired number of cities to be visited, which are essential factors for enhancing the fairness of the generated tours.

Our method performs remarkably well at the min-max routing problems, outperforming both existing classical heuristic and learning-based methods. As a highlight, Equity-Transformer achieves 334× speed improvement and 53% reduction of solution cost compared to the representative classical heuristic solver (LKH3) when solving the multi-agent TSP with 1,000 cities. Also, our method achieves 1,217× faster speed and 9% reduction of solution cost than the representative learning-based method (ScheduleNet) with parallel planning.

Problem Formulation

In our work, we focus on tackling min-max routing problems, which involve a scenario where a group of M agents needs to visit N cities with the objective of minimizing the maximum tour length among the individual tours of the agents. In this section, we present the formulation of min-max routing through the lens of sequential planning.

Problem. A routing problem is defined the set of city locations and a depot, represented as $\boldsymbol{x} = \{x_i\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^2$ is the Cartesian coordinates, and x_1 denotes the depot. Since the M agents find tours that start from the depot and return to the depot, we add M dummy depot (each dummy depot assigned to each agent), i.e., $x_{N+1} = \cdots = x_{N+M} = x_1$. Thus, the sequential planning routing is defined as $\boldsymbol{x} = \{x_i\}_{i=1}^{N+M}$. Remarks that we can expand the definition of $\boldsymbol{x}$ so that it can include additional features required for other problems like the capacitated vehicle routing. For simplicity, we represent locations only.

Action. The sequential planning is represented as an action sequence $\boldsymbol{a} = (a_1, \ldots, a_{N+M})$. This action sequence is formed by selecting an index from the set of unvisited nodes at step t, i.e., $a_t \in \{1, \ldots, N+M\} \backslash \{a_1, \ldots, a_{t-1}\}$. The resulting action sequence is partitioned into M subsequences, i.e., agent tours $(\boldsymbol{a}^1, \ldots, \boldsymbol{a}^M)$, by splitting $\boldsymbol{a}$ with depot choosing actions. Thus, each $\boldsymbol{a}^m = (a_1^m, \ldots, a_{L_m}^m)$ starts with a dummy depot index, i.e., $a_1^m \in \{N+1, \ldots, N+M\}$, followed by subsequent city indices. Please refer to Figure 1.

State. The state s_t is defined as the union of the precollected actions $a_1, \ldots, a_{t-1}$ and the problem $\boldsymbol{x}$, i.e., $s_1 = \{\boldsymbol{x}\}$, and $s_t = \{a_1, \ldots, a_{t-1}; \boldsymbol{x}\}$ for $t > 1$.

Cost. The cost is the maximum tour length among all agents' tours of $(\boldsymbol{a}^1, \ldots, \boldsymbol{a}^M)$ of given action sequence $\boldsymbol{a}$, i.e.,

$$\mathcal{L}_{\text{cost}}(\boldsymbol{a}; \boldsymbol{x}) := \max \left\{ \mathcal{L}(\boldsymbol{a}^1; \boldsymbol{x}), ..., \mathcal{L}(\boldsymbol{a}^M; \boldsymbol{x}) \right\}, \text{where}$$

$$\mathcal{L}(\boldsymbol{a}^m; \boldsymbol{x}) := \sum_{t=2}^{L_m} ||x_{a_t^m} - x_{a_{t-1}^m}||_2 + ||x_{a_1^m} - x_{a_{L_m}^m}||_2.$$

Policy. The policy $\pi_\theta(\boldsymbol{a}|\boldsymbol{x})$ is a composition of segment policy $\pi_\theta(a_t|\boldsymbol{s}_t)$, generating action sequences for given problem condition $\boldsymbol{x}$ according to the following expression.

$$\pi_\theta(\boldsymbol{a}|\boldsymbol{x}) = \prod_{t=1}^{N+M} \pi_\theta(a_t|\boldsymbol{s}_t).$$

The θ is the deep neural network parameter of the policy π. The optimal parameter θ^* can be determined by solving the following optimization problem:

$$\theta^* = \arg\min_\theta \mathbb{E}_{P(\boldsymbol{x})} \mathbb{E}_{\pi_\theta(\boldsymbol{a}|\boldsymbol{x})} \mathcal{L}_{\text{cost}}(\boldsymbol{a}; \boldsymbol{x}),$$

where $P(\boldsymbol{x})$ is the distribution of problem $\boldsymbol{x}$.

Methodology

This section presents the architecture of Equity-Transformer $\pi_\theta(\boldsymbol{a}|\boldsymbol{x})$, which generates an action sequence $\boldsymbol{a}$ =

$(a_1, \dots, a_{N+M})$ for a given problem $\boldsymbol{x}$. Our high-level idea is to build a transformer model with *multi-agent positional encoding* and *equity context*.

Our architecture has the following forward propagation:

1. Multi-agent positional encoding for initial node embedding given problem $\boldsymbol{x}$.
2. Employ the encoder of Transformer (Vaswani et al. 2017) to the initial node embedding to obtain $\boldsymbol{H} = [h_1, \dots, h_{N+M}] \in \mathbb{R}^{D \times (N+M)}$, where D is the embedding dimension.
3. Iterative decoding $t = 1, \dots, N+M$ using
   (a) Equity context encoding for $\boldsymbol{c}_t \in \mathbb{R}^D$.
   (b) Decoding to produce $\boldsymbol{a}_t \sim \pi_\theta(\boldsymbol{a}_t | \boldsymbol{H}, \boldsymbol{c}_t)$ by using c_t as attention query of decoder.

The encoding and the iterative decoding procedure process involving contextual queries have been comprehensively addressed in the previous literature (Kool, van Hoof, and Welling 2019; Kwon et al. 2020; Li et al. 2021; Kim, Park, and Park 2022). In this paper, we focus on introducing new elements designed for min-max routing problems, which are multi-agent positional encoding and equity context encoding.

Multi-agent Positional Encoding

We begin with partitioning the problem $\boldsymbol{x}$ into two distinct components: the cities, denoted as $\boldsymbol{x}_{\text{city}}$ with elements $\{x_i\}_{i=1}^{N}$, and the agents, represented by $\boldsymbol{x}_{\text{agent}}$ with elements $\{x_i\}_{i=N+1}^{N+M}$. In order to facilitate sequential relationships between the agents, we employ positional encoding f_{PE} to $\boldsymbol{x}_{\text{agent}}$. This positional encoding incorporates sine and cosine functions with differing frequencies, following the work by Vaswani et al. (2017).

Next, we concatenate the linearly projected vectors of $\boldsymbol{x}_{\text{city}}$ and $f_{\text{PE}}(\boldsymbol{x}_{\text{agent}})$ to form the initial node embedding. The embedding is subsequently fed into the encoder, a structural component akin to the encoder of AM as illustrated in Figure 2. It is worth to note that the original AM architecture itself is not appropriate for addressing multi-agent problems, due to its incapability to consider the multi-agent nature. In contrast, our approach incorporates positional encoding to $\boldsymbol{x}_{\text{agent}}$ with the specific intention of introducing a virtual order bias. Consequently, we can sequentially generate the tour sequence of an agent while considering both preceding and succeeding agents in the assigned order.

Equity Context Encoding

For every decoding step t, we utilize four distinct contexts as ingredients of the equity context, $\boldsymbol{c}_t$. Each of the ingredient contains useful information for equitable decoding, described as follows:

- **Problem context $\boldsymbol{h}^{\text{problem}} \in \mathbb{R}^D$:** The problem context collects the average of representations $\boldsymbol{h}^{\text{problem}} = \frac{1}{M+N} \sum_{i=1}^{M+N} \boldsymbol{h}_i$. This aligns with the context embedding of AM, which is primarily intended to capture the global context of problem $\boldsymbol{x}$ by averaging each city and agent representation.

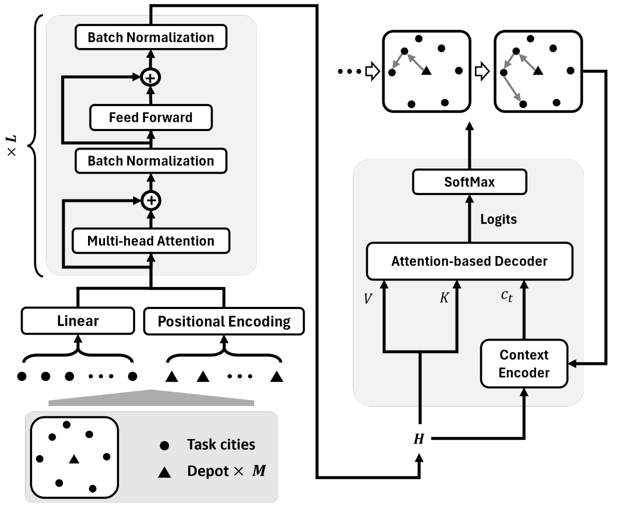


Figure 2: Illustration of Equity-Transformer. The L stands for the number of sequential layers, where we set $L = 3$.

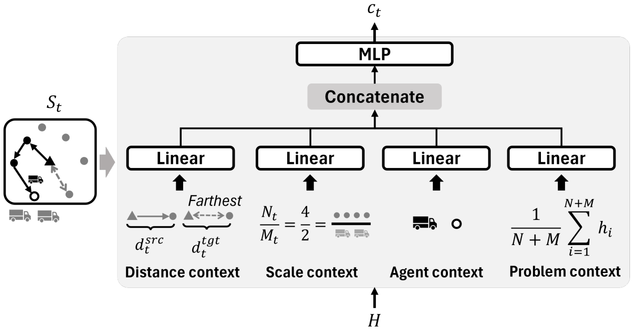


Figure 3: Illustration of equity context encoding

- **Agent context $\boldsymbol{h}_t^{\text{agent}} \in \mathbb{R}^D$:** We set agent context using representations of the returning depot and the last visited city of the current active agent m_t. Precisely, $\boldsymbol{h}_t^{\text{agent}} = g_{\text{agent}}(\boldsymbol{h}_{N+m_t} \oplus \boldsymbol{h}_{a_{t-1}})$, where $g_{\text{agent}} : \mathbb{R}^{2D} \rightarrow \mathbb{R}^D$ represents a linear projection. This context highlights the currently active agents.
- **Scale context $\boldsymbol{h}_t^{\text{scale}} \in \mathbb{R}^D$:** We incorporate the ratio between N_t and M_t as a scaling context, where N_t represents the number of remaining cities and M_t denotes the current number of un-used agents at the depot, i.e., $N_t/M_t \in \mathbb{R}$. We then generate $\boldsymbol{h}_t^{\text{scale}} = g_{\text{scale}}(N_t/M_t)$, where $g_{\text{scale}} : \mathbb{R} \rightarrow \mathbb{R}^D$ represents a linear projection. This context offers valuable insights into the approximate number of cities an agent should visit to achieve equity. Consequently, the scale ratio can provide the effective information to decide whether to continue visiting additional cities or return to the depot.
- **Distance context $\boldsymbol{h}_t^{\text{dist}} \in \mathbb{R}^D$:** We make use of dynamic changes in the agent's tour length and the distance of remaining cities from the depot at the current step t. Firstly, we employ d_t^{source}, which represents the

current tour length of the active agent. Secondly, we utilize d_t^{target}, which denotes the maximum distance between the depot and the remaining unvisited cities. Subsequently, we form $\boldsymbol{h}_t^{\text{dist}} = g_{\text{dist}}(d_t^{\text{source}} \oplus d_t^{\text{target}})$, where $g_{\text{dist}} : \mathbb{R}^2 \rightarrow \mathbb{R}^D$ is a linear projection. This information holds significant importance in terms of the equity of tour length among agents in the min-max routing problem. The context prompts the decoders to consider the agent's current tour length and remaining tasks, aiding in the decision-making process of whether to stop visiting (i.e., return to the depot) or continue the tour while considering the min-max tour length.

The context encoder $f_{\text{CE}} : \mathbb{R}^{4D} \rightarrow \mathbb{R}^D$, which is a multilayer perception (MLP), produces equity context $\boldsymbol{c}_t$ using these four contexts, i.e.,

$$\boldsymbol{c}_t = f_{\text{CE}}(\boldsymbol{h}^{\text{problem}} \oplus \boldsymbol{h}_t^{\text{agent}} \oplus \boldsymbol{h}_t^{\text{scale}} \oplus \boldsymbol{h}_t^{\text{dist}}).$$

We adopt an approach similar to that of Kool, van Hoof, and Welling (2019), where use $\boldsymbol{c}_t$ as a contextual query for the attention-based decoder $\pi_\theta(\boldsymbol{a}_t | \boldsymbol{H}, \boldsymbol{c}_t)$, as shown in Figure 3. This utilization of task-equity information from the equity context enables the decoder to sequentially generate balanced tours.

Training Scheme

The Equity-Transformer is trained with REINFORCE (Williams 1992) with the shared baseline scheme similar to Kwon et al. (2020) and Kim, Park, and Park (2022). The shared baseline with symmetric samples makes symmetric exploration for the combinatorial solution space. The training loss with the symmetric shared baselines is as follows:

$$\mathcal{L}_{\text{train}}(\boldsymbol{\theta}) = \mathbb{E}_{P(\boldsymbol{x})}\mathbb{E}_{\pi_\theta(\boldsymbol{a}|\boldsymbol{x})}\mathcal{L}_{\text{cost}}(\boldsymbol{a}; \boldsymbol{x}),$$

$$\nabla\mathcal{L}_{\text{train}}(\boldsymbol{\theta}) \approx \sum_{i=1}^{B}\sum_{j=1}^{L}\left(\mathcal{L}_{\text{cost}}\left(\boldsymbol{a}^{(i,j)}; \boldsymbol{x}^{(i)}\right) - b_{\text{shared}}\right),$$

where $b_{\text{shared}} := 1/L\sum_{j=1}^{L}\mathcal{L}_{\text{cost}}(\boldsymbol{a}^{(i,j)}; \boldsymbol{x}^{(i)})$. Each $\boldsymbol{a}^{(i,l)}$ is sampled sequence from training solver given L symmetric $\boldsymbol{x}^{(i)}$: $\pi_\theta(\boldsymbol{a}|\mathcal{T}_1(\boldsymbol{x}^{(i)})), ..., \pi_\theta(\boldsymbol{a}|\mathcal{T}_L(\boldsymbol{x}^{(i)}))$, where $\mathcal{T}_1, ..., \mathcal{T}_L$ are symmetric transformation of problem instance $\boldsymbol{x}^{(i)}$. See Kim, Park, and Park (2022) for a detailed training scheme.

Experiments

In this section, we present the experimental results of the Equity-Transformer model on two min-max routing problems: the multi-agent traveling salesman problem (mTSP) and the multi-agent pick-up and delivery problem (mPDP).

Training Setting. we use uniform distribution for the problem distribution $P(\boldsymbol{x})$, following Kool, van Hoof, and Welling (2019). For the training hyperparameters we set exactly the same hyperparameters for every task and experiment; see Appendix B. We train Equity-Transformer on $N = 50$, and finetune it to target distribution of $N = 200, 500$. The training time for the min-max mTSP is approximately one day, while for the min-max mPDP, it takes around four days.

Experiments Metric. It is important to carefully measure the performance comparison between methods, as often there is a trade-off between run time and solution quality. To this end, we present time-performance multi-objective graphs to compare tradeoffs between performance and computation time. In the result tables, we present the average cost achieved within a specific time limit, recognizing that every method has the potential to reach optimality given an unlimited amount of time.

Target Problem Instances. To evaluate the performance of our methods, we report results on randomly generated synthetic instances of min-max mTSP and min-max mPDP at different problem scales of N and M. We generate the 100 problems set with a uniform distribution of node locations per scale. We conduct experiments with $N = 200, 500, 1000, 2000, 5000$ and set M such that $10 \leq N/M \leq 20$ by referring practical setting of min-max routing application (Ma et al. 2021).

Speed Evaluation. All experiments were performed using a single NVIDIA A100 GPU and an AMD EPYC 7542 32-core processor as the CPU. Comparing the speed performance of classical algorithms (CPU-oriented) and learning algorithms (GPU-oriented) poses a significant challenge (Kool, van Hoof, and Welling 2019; Kim, Park, and Kim 2021), given the need for a fair evaluation. While certain approaches exploit the parallelizability of learning algorithms on GPUs, enabling faster solutions to multiple problems than classical algorithms, our method follows a serial approach in line with the prior min-max learning methods (Park, Kwon, and Park 2023; Kim, Park, and Park 2023). Note that when we leverage the parallelizability of our method, our approach can achieve speeds more than 100× faster; refer to Appendix A.

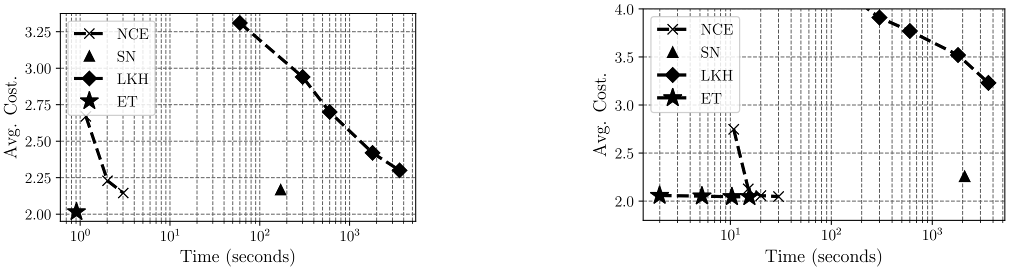
Performance Evaluation on mTSP

Baselines for mTSP. We consider two representative deep learning-based baseline algorithms: the ScheduleNet (Park, Kwon, and Park 2023) and Neuro Cross Exchange (Kim, Park, and Park 2023) for min-max mTSP. For conciseness, We denote them as SN and NCE, respectively. We have also included two classical heuristic methods, namely LKH3 (Helsgaun 2017) and OR-Tools (Perron and Furnon 2019), with respective time limits of 60 seconds and 600 seconds per instance. Specifically, LKH3 utilizes λ-opt improvement iterations to enhance the solution within the given time budget. The time limit directly influences the number of iterations performed (following the approach in Xin et al. (2021b)). Similarly, OR-Tools incorporates an iterative local search procedure for solution improvement, with a time limit governing the iterations of the local search.

Results. The results in Table 1. demonstrate that the Equity-Transformer (denoted as 'ET' in tables and figures) outperforms all baselines with impressive speed. As the problem scale increases, the performance gap between ET and other methods widens further. Specifically, for $N = 1000$ and $M = 100$, ours achieves a cost of 2.05, significantly better than LKH3 (2.92) and NCE (2.16). More-

		Classic-based				Learning-based		
N	M	LKH3 (60)	LKH3 (600)	OR-Tools (60)	OR-Tools (600)	SN	NCE	ET (*ours*)
200	10	2.52 (60)	2.08 (600)	4.97 (60)	2.22 (600)	2.35 (9.70)	2.07 (5.07)	**2.05** (0.36)
	15	2.39 (60)	2.03 (600)	4.82 (60)	2.15 (600)	2.13 (10.52)	1.97 (5.07)	**1.97** (0.37)
	20	2.29 (60)	2.02 (600)	3.74 (60)	2.04 (600)	2.07 (11.40)	1.96 (5.07)	**1.96** (0.37)
500	30	3.31 (60)	2.70 (600)	7.90 (60)	6.44 (600)	2.16 (171)	2.07 (5.20)	**2.02** (0.87)
	40	3.10 (60)	2.55 (600)	7.46 (60)	6.69 (600)	2.12 (276)	2.01 (5.38)	**2.01** (0.90)
	50	2.93 (60)	2.48 (600)	8.50 (60)	7.26 (600)	2.09 (217)	2.01 (5.05)	**2.01** (0.92)
1000	50	4.45 (60)	3.77 (600)	11.65 (60)	9.89 (600)	2.26 (2094)	2.13 (15.05)	**2.06** (1.72)
	75	3.71 (60)	3.26 (600)	13.16 (60)	11.50 (600)	2.17 (1678)	2.07 (15.05)	**2.05** (1.80)
	100	3.23 (60)	2.92 (600)	10.79 (60)	8.93 (600)	2.16 (1588)	2.05 (15.01)	**2.05** (1.79)
2000	100	6.60 (60)	4.61 (600)	20.99 (60)	18.85 (600)	*OB*	2.85 (43.96)	**2.09** (3.49)
	150	5.08 (60)	4.02 (600)	14.00 (60)	13.17 (600)	*OB*	2.83 (44.77)	**2.08** (3.41)
	200	4.13 (60)	3.36 (600)	11.00 (60)	10.41 (600)	*OB*	2.08 (30.30)	**2.08** (3.60)
5000	300	12.30 (60)	7.87 (600)	17.00 (60)	17.00 (60)	*OB*	2.97 (290)	**2.40** (8.78)
	400	8.85 (60)	6.15 (600)	13.00 (60)	13.00 (600)	*OB*	2.92 (204)	**2.21** (8.61)
	500	7.14 (60)	5.37 (600)	11.00 (60)	11.00 (600)	*OB*	2.89 (198)	**2.19** (9.02)

Table 1: Results on min-max mTSP. Every performance is average performance among 100 instances. The bold symbol indicates the best performance. Average running times (in seconds) are provided in brackets.



(a) mTSP with $(500, 30)$

(b) mTSP with $(1000, 50)$

Figure 4: Time-performance trade-off graph for mTSP. The left and bottom indicate the Pareto frontier.

over, our method is $15.01/1.79 \approx 8.39\times$ faster than NCE and about $600/1.79 \approx 335\times$ faster than LKH3. The time-performance trade-off analysis shown in Fig. 4 confirms that our method outperforms every baseline and provides the Pareto frontier on multi-objective of time and cost.

For the large-scale problem of $N = 5000$, the ScheduleNet suffers from the complexity inherent in the parallel planning, failing to produce a solution within 10,000 seconds per problem, making it out-of-budget (OB). While LKH3 and OR-Tools methods can provide solutions within the allotted time, their performance is inadequate due to the inherent difficulty of large-scale problems, requiring a significantly higher number of improvement iterations for low-cost solutions. On the other hand, the NCE surpasses classical approaches, but ours outperforms NCE by a substantial margin of approximately $290/8.78 \approx 33\times$ faster speed and $(2.97 - 2.40)/2.97 \approx 19\%$ reduced cost.

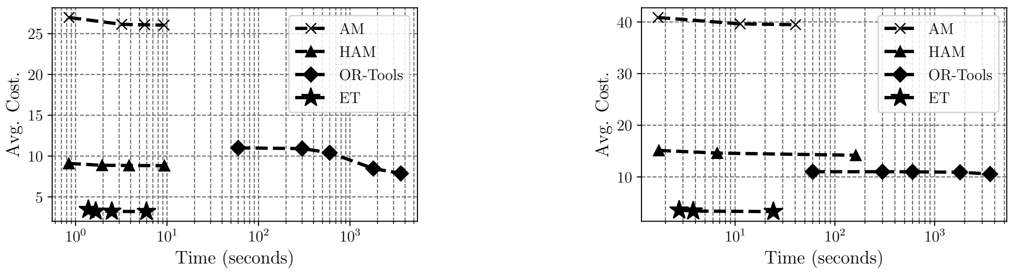
Performance Evaluation on mPDP

Baselines for mPDP. We consider representative two deep learning-based baseline algorithms: AM (Kool, van Hoof, and Welling 2019) and heterogeneous AM (Li et al. 2021), denoted as HAM. We retrain AM, and HAM using min-max objective; the details are provided in Appendix B. We also marked † by giving more trials for inference solutions such as sampling width (Kool, van Hoof, and Welling 2019) and augmentation width (Kwon et al. 2020). We include a heuristic method, OR-Tools (Perron and Furnon 2019), while LKH3 (Helsgaun 2017) for min-max mPDP. We exclude the multi-agent PDP (MAPDP) model (Zong et al. 2022) due to the inaccessible source code.

As shown in Table 2, our methods (i.e., ET and ET†) outperform all other baselines, aligning with the findings from the mTSP experiments. Compared to OR-Tools, ET†

		Classic-based		Learning-based					
N	M	OR-Tools		AM	AM†	HAM	HAM†	ET (*ours*)	ET† (*ours*)
200	10	20.96 (60)	18.76 (600)	15.88 (0.33)	15.65 (0.67)	5.69 (0.33)	5.30 (0.55)	5.03 (0.54)	**4.68** (0.55)
	15	13.96 (60)	8.46 (600)	15.88 (0.33)	15.57 (0.69)	5.21 (0.34)	5.09 (0.57)	3.91 (0.55)	**3.65** (0.56)
	20	10.67 (60)	5.70 (600)	15.88 (0.35)	15.55 (0.71)	5.21 (0.35)	5.09 (0.61)	3.39 (0.56)	**3.18** (0.61)
500	30	16.99 (60)	16.99 (600)	26.98 (0.82)	26.15 (3.10)	9.10 (0.84)	8.86 (1.92)	4.38 (1.33)	**4.11** (1.55)
	40	12.99 (60)	12.65 (600)	26.98 (0.86)	26.14 (3.17)	9.10 (0.84)	8.87 (1.95)	3.75 (1.36)	**3.52** (1.62)
	50	10.99 (60)	10.41 (600)	26.98 (0.85)	26.14 (3.20)	9.10 (0.88)	8.87 (1.95)	3.44 (1.38)	**3.23** (1.66)
1000	50	21.00 (60)	21.00 (600)	40.86 (1.61)	39.63 (11.28)	15.12 (1.63)	14.58 (6.35)	4.91 (2.63)	**4.73** (3.56)
	75	14.00 (60)	14.00 (600)	40.86 (1.68)	39.61 (11.44)	15.12 (1.70)	14.59 (6.41)	3.96 (2.65)	**3.77** (3.63)
	100	11.00 (60)	10.98 (600)	40.86 (1.69)	39.63 (11.24)	15.12 (1.72)	14.61 (6.61)	3.56 (2.75)	**3.38** (3.80)
2000	100	21.00 (60)	21.00 (600)	62.85 (3.24)	61.31 (24.98)	25.68 (3.40)	25.06 (15.17)	5.15 (5.22)	**4.91** (9.22)
	150	14.00 (60)	14.00 (600)	62.85 (3.34)	61.28 (25.43)	25.68 (3.40)	25.04 (16.26)	4.17 (5.31)	**3.97** (9.50)
	200	11.00 (60)	11.00 (600)	62.85 (3.35)	61.33 (26.33)	25.68 (3.50)	25.06 (16.47)	3.79 (5.43)	**3.62** (10.01)
5000	300	17.00 (60)	17.00 (600)	114.73 (8.30)	112.84 (180)	54.07 (34.65)	53.46 (279)	4.81 (52.66)	**4.60** (79.23)
	400	13.00 (60)	13.00 (600)	114.73 (8.31)	112.90 (182)	54.07 (34.44)	53.43 (283)	4.33 (54.86)	**4.11** (82.59)
	500	11.00 (60)	11.00 (600)	114.73 (8.33)	112.83 (186)	54.07 (34.46)	53.45 (286)	4.12 (54.77)	**3.88** (82.87)

Table 2: Results on min-max mPDP. Every performance is average performance among 100 instances. The bold symbol indicates the best performance. Average running times (in seconds) are provided in brackets.



(a) mPDP with $(500, 50)$

(b) mPDP with $(1000, 100)$

Figure 5: Time-performance trade-off graph for mPDP. The left and bottom indicate the Pareto frontier.

exhibits a remarkable speed improvement of $600/0.55 \approx 1901\times$, while reducing the objective cost by approximately $(18.76 - 4.68)/18.76 \approx 75\%$ at $N = 200, M = 10$. Moreover, as shown in Figure 5, ours consistently presents the Pareto frontier compared to others.

Importantly, in certain instances, both AM and HAM produce identical cost values as the number of agents M increases. For instance, when $N = 500$, AM and HAM yield the same scores for $M = 30, 40, 50$. These methods were primarily designed to address min-sum problems (with HAM especially focusing on min-sum mPDP), suffering from considering *equity* among agents unlike ours.

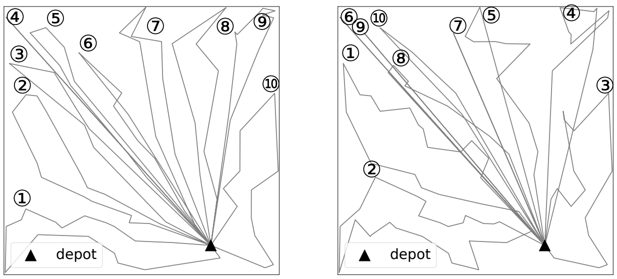
Ablation Study

To assess the influence of each component within our methodology on performance enhancement, we conducted an ablation study. As illustrated in Table 3, both components of our approach yielded significant performance improvements. Notably, the $\emptyset$ configuration, which represents the absence of these components, resulted in the poorest performance, indicating that a straightforward application of sequential planning to the min-max routing problem is not inherently promising. However, when we combined the multi-agent positional encoder (MPE) and the context encoder (CE), we observed substantial performance improvements, particularly in larger-scale scenarios.

Ablation Study for Order Bias. As depicted in Figure 6, MPE contributes to inducing an order bias among agents by generating cyclic sub-tours in the Euclidean space with specific orders. This can be interpreted as successful modeling of tour generation from multiple agents in the sequence space, which is the primary goal of MPE.

N	100			200		
M	5	10	15	10	15	20
$\emptyset$	2.86	2.12	2.12	2.92	2.90	2.90
{MPE}	2.35	1.97	1.96	2.51	2.33	2.80
{CE}	2.52	1.97	1.95	2.28	2.01	1.98
{MPE, CE}	**2.35**	**1.96**	**1.95**	**2.15**	**1.99**	**1.98**

Table 3: Ablation study for the combination of our components. The $\emptyset$ represents the original AM.



(a) With MPE

(b) Without MPE

Figure 6: Ablation study for the multi-agent positional encoding (MPE) on mTSP with $N = 100$ and $M = 10$.

Additional Experiments

We conducted several additional experiments in the Appendix C and Appendix D. First, we validate the performance of the Equity-Transformer in a real-world benchmark dataset (Appendix C). Among the baseline methods, Equity-Transformer demonstrates superior performance in almost all instances. Moreover, we assessed the robustness of Equity-Transformer under various problem distributions (Appendix D.1) and different N/M ratios (Appendix D.2), comparing it with LKH3. These experiments confirmed the robustness of Equity-Transformer to the changes in problem distributions and N/M ratios. Lastly, we compared Equity-Transformer with two competitive two-stage mTSP solvers (Hu, Yao, and Lee 2020; Liang et al. 2023), and ours outperformed them in terms of performance (Appendix D.3).

Related Work

Vehicle Routing Problems

After Vinyals, Fortunato, and Jaitly (2015) suggested the Pointer Networks, which constructively generate permutation sequences as routing solutions, termed *constructive solver*, Bello et al. (2017) turns it into deep reinforcement learning. Kool, van Hoof, and Welling (2019) reinvent the Pointer Network using a transformer, termed attention model (AM), which becomes standard architecture for solving vehicle routing problems. By extending AM into various applications, including those outlined in recent research (Li et al. 2021; Jiang et al. 2022; Ma et al. 2021; Xin et al. 2021a; Ma et al. 2021, 2022), several challenges within the field of vehicle routing are addressed. In recent studies, there has been a notable emphasis on assessing the robustness of neural solvers concerning both scale shift (Hottung, Kwon, and Tierney 2021; Son et al. 2023) and distributional shift (Jiang et al. 2022; Bi et al. 2022; Zhou et al. 2023).

Independent of constructive solution generation, other studies try to solve VRPs by learning to revise the solution iteratively, terms *improvement solver*. Chen and Tian (2019); Li, Yan, and Wu (2021); Kim, Park, and Kim (2021); Wang et al. (2021) leverages local solver to rewrite partial tour to improve solution. Some studies train existing local search solvers such as 2-opt heuristic (da Costa et al. 2020; Wu et al. 2021), large neighborhood search (Hottung and Tierney 2020), iterative dynamic programming (Kool et al. 2021), and LKH (Xin et al. 2021b) using deep learning. Some studies use fine-tuning schemes focused on test-time adaptation in iterative learning (Hottung, Kwon, and Tierney 2021; Choo et al. 2022). While a constructive solver is invaluable for quickly generating an initial feasible solution, an improvement solver plays a crucial role in refining the solution to achieve enhanced optimality. These two approaches are fundamentally distinct and orthogonal in their objectives.

Min-Max Vehicle Routing Problems

Most deep learning-based VRPs studied focus on min-sum routing which focuses on minimizing total tour length among multiple agents. The min-max routing problem focuses on minimizing the maximum tour length among multiple agents, making it highly relevant for time-critical tasks such as disaster management and vaccine delivery. The min-max routing method considers the equity of tours among the multiple agents (França et al. 1995).

In constructive approaches, Cao, Sun, and Sartoretti (2021) and Park, Kwon, and Park (2023) advocate for a constructive solver that models decentralized parallel decisions made by multiple agents. Additionally, in addressing the specific challenge of min-max mTSP with time windows and rejections, Zhang et al. (2022) introduced a constructive solver leveraging a graph neural network in conjunction with meticulous training and inference strategies.

On the other hand, in improvement-based methodologies, Kim, Park, and Park (2023) propose an enhancement solver that learns to optimize tour components through cross-exchanges. Meanwhile, Hu, Yao, and Lee (2020) and Liang et al. (2023) advocate a two-stage solver approach, wherein the initial stage employs a constructive solver, followed by an improvement solver that refines the solution further.

Conclusion

This paper introduced Equity-Transformer, sequential models for min-max routing problems. Our method outperformed the existing classic methods and state-of-the-art neural solvers, achieving a Pareto frontier in balancing cost and runtime on representative tasks like mTSP and mPDP. Our method demonstrates its scalability, handling large-scale cities with up to $N = 5000$ nodes and agent fleets of up to $M = 500$. Equity-Transformer holds potential for broader applications in general min-max vehicle routing problems, which we identify as a promising avenue for future research.

Acknowledgements

We thank anonymous reviewers for providing helpful feedback for preparing our manuscripts. This work was supported by a grant of the KAIST-KT joint research project through AI2XL Laboratory, Institute of convergence Technology, funded by KT [Project No. G01210696, Development of Multi-Agent Reinforcement Learning Algorithm for Efficient Operation of Complex Distributed Systems].

References

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2017. Neural Combinatorial Optimization with Reinforcement Learning. arXiv:1611.09940.

Bertazzi, L.; Golden, B.; and Wang, X. 2015. Min–max vs. min–sum vehicle routing: A worst-case analysis. *European Journal of Operational Research*, 240(2): 372–381.

Bi, J.; Ma, Y.; Wang, J.; Cao, Z.; Chen, J.; Sun, Y.; and Chee, Y. M. 2022. Learning Generalizable Models for Vehicle Routing Problems via Knowledge Distillation. In *Advances in Neural Information Processing Systems*.

Cao, Y.; Sun, Z.; and Sartoretti, G. 2021. DAN: Decentralized Attention-based Neural Network for the MinMax Multiple Traveling Salesman Problem. *arXiv preprint arXiv:2109.04205*.

Cheikhrouhou, O.; and Khoufi, I. 2021. A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. *Computer Science Review*, 40: 100369.

Chen, X.; and Tian, Y. 2019. Learning to Perform Local Rewriting for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*.

Choo, J.; Kwon, Y.-D.; Kim, J.; Jae, J.; Hottung, A.; Tierney, K.; and Gwon, Y. 2022. Simulation-guided beam search for neural combinatorial optimization. *arXiv preprint arXiv:2207.06190*.

da Costa, P. R. d. O.; Rhuggenaath, J.; Zhang, Y.; and Akcay, A. 2020. Learning 2-opt Heuristics for the Traveling Salesman Problem via Deep Reinforcement Learning. In Pan, S. J.; and Sugiyama, M., eds., *Proceedings of The 12th Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, 465–480. Bangkok, Thailand: PMLR.

David Applegate, V. C., Robert Bixby; and Cook, W. 2023. Concorde TSP Solver.

França, P. M.; Gendreau, M.; Laporte, G.; and Müller, F. M. 1995. The m-traveling salesman problem with minmax objective. *Transportation Science*, 29(3): 267–275.

Fu, Z.-H.; Qiu, K.-B.; and Zha, H. 2021. Generalize a small pre-trained model to arbitrarily large TSP instances. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 7474–7482.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. URL https://www.gurobi.com, Last Accesss on January 10, 2024.

Helsgaun, K. 2017. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. *Roskilde: Roskilde University*.

Hottung, A.; Kwon, Y.-D.; and Tierney, K. 2021. Efficient Active Search for Combinatorial Optimization Problems. In *International Conference on Learning Representations*.

Hottung, A.; and Tierney, K. 2020. Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem. In *ECAI 2020*, 443–450. IOS Press.

Hu, Y.; Yao, Y.; and Lee, W. S. 2020. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowledge-Based Systems*, 204: 106244.

Jiang, Y.; Wu, Y.; Cao, Z.; and Zhang, J. 2022. Learning to Solve Routing Problems via Distributionally Robust Optimization. In *36th AAAI Conference on Artificial Intelligence*.

Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; and Song, L. 2017. Learning Combinatorial Optimization Algorithms over Graphs. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30, 6348–6358. Curran Associates, Inc.

Kim, M.; Park, J.; and Kim, J. 2021. Learning Collaborative Policies to Solve NP-hard Routing Problems. In *Advances in Neural Information Processing Systems*.

Kim, M.; Park, J.; and Park, J. 2022. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35: 1936–1949.

Kim, M.; Park, J.; and Park, J. 2023. Learning to CROSS exchange to solve min-max vehicle routing problems. In *The Eleventh International Conference on Learning Representations*.

Kool, W.; van Hoof, H.; Gromicho, J. A. S.; and Welling, M. 2021. Deep Policy Dynamic Programming for Vehicle Routing Problems. *CoRR*, abs/2102.11756.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *International Conference on Learning Representations*.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. POMO: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 21188–21198.

Li, J.; Xin, L.; Cao, Z.; Lim, A.; Song, W.; and Zhang, J. 2021. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(3): 2306–2315.

Li, S.; Yan, Z.; and Wu, C. 2021. Learning to delegate for large-scale vehicle routing. *Advances in Neural Information Processing Systems*, 34.

Liang, H.; Ma, Y.; Cao, Z.; Liu, T.; Ni, F.; Li, Z.; and Hao, J. 2023. SplitNet: a reinforcement learning based sequence splitting method for the MinMax multiple travelling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8720–8727.

Ma, Y.; Hao, X.; Hao, J.; Lu, J.; Liu, X.; Xialiang, T.; Yuan, M.; Li, Z.; Tang, J.; and Meng, Z. 2021. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in Neural Information Processing Systems*, 34: 23609–23620.

Ma, Y.; Li, J.; Cao, Z.; Song, W.; Guo, H.; Gong, Y.; and Chee, Y. M. 2022. Efficient Neural Neighborhood Search for Pickup and Delivery Problems. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 4776–4784. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Papadimitriou, C. H. 1977. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3): 237 – 244.

Park, J.; Kwon, C.; and Park, J. 2023. Learn to Solve the Min-max Multiple Traveling Salesmen Problem with Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 878–886.

Perron, L.; and Furnon, V. 2019. OR-Tools.

Qiu, R.; Sun, Z.; and Yang, Y. 2022. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35: 25531–25546.

Son, J.; Kim, M.; Kim, H.; and Park, J. 2023. Meta-SAGE: Scale Meta-Learning Scheduled Adaptation with Guided Exploration for Mitigating Scale Shift on Combinatorial Optimization. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 32194–32210. PMLR.

Sun, H.; Goshvadi, K.; Nova, A.; Schuurmans, D.; and Dai, H. 2023. Revisiting Sampling for Combinatorial Optimization. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 32859–32874. PMLR.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30, 5998–6008. Curran Associates, Inc.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28, 2692–2700. Curran Associates, Inc.

Wang, H.; Zong, Z.; Xia, T.; Luo, S.; Zheng, M.; Jin, D.; and Li, Y. 2021. Rewriting by Generating: Learn Heuristics for Large-scale Vehicle Routing Problems.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.

Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; and Lim, A. 2021. Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9): 5057–5069.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021a. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of 35th AAAI Conference on Artificial Intelligence*, 12042–12049.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021b. NeuroLKH: Combining Deep Learning Model with Lin-Kernighan-Helsgaun Heuristic for Solving the Traveling Salesman Problem. *Advances in Neural Information Processing Systems*, 34.

Zhang, D.; Xiao, Z.; Wang, Y.; Song, M.; and Chen, G. 2023. Neural TSP solver with progressive distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 12147–12154.

Zhang, R.; Zhang, C.; Cao, Z.; Song, W.; Tan, P. S.; Zhang, J.; Wen, B.; and Dauwels, J. 2022. Learning to solve multiple-TSP with time window and rejections via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 24(1): 1325–1336.

Zhou, J.; Wu, Y.; Song, W.; Cao, Z.; and Zhang, J. 2023. Towards Omni-generalizable Neural Methods for Vehicle Routing Problems. In *International Conference on Machine Learning*.

Zong, Z.; Zheng, M.; Li, Y.; and Jin, D. 2022. Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9980–9988.