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

Jiwoo Son\textsuperscript{1,2}, Minsu Kim\textsuperscript{1}, Sanghyeok Choi\textsuperscript{1}, Hyeonah Kim\textsuperscript{1}, Jinkyoo Park\textsuperscript{1,2}

\textsuperscript{1}Korea Advanced Institute of Science and Technology (KAIST)
\textsuperscript{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 (LKHS) 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 (Helsgau 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 Khoui 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
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 $x = \{x_i\}_{i=1}^N$ 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 $x = \{x_i\}_{i=1}^{N+M}$. Remarks that we can expand the definition of $x$ 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 $a = (a_1, \ldots, a_{N+M})$. This action sequence is formed by selecting an index from the set of unvisited nodes at step $t$, i.e., $a_t \in \{1, \ldots, N+M\} \setminus \{a_1, \ldots, a_{t-1}\}$. The resulting action sequence is partitioned into $M$ subsequences, i.e., agent tours $(a_1^1, \ldots, a_{M}^1)$, by splitting $a$ with depot choosing actions. Thus, each $a^m = (a_1^m, \ldots, a_{L_m}^m)$ 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 $x$, i.e., $s_t = \{x\}$, and $s_t = \{a_1, \ldots, a_{t-1}; x\}$ for $t > 1$.

**Cost.** The cost is the maximum tour length among all agents’ tours of $(a_1^1, \ldots, a^M)$ of given action sequence $a$, i.e.,

$$
L_{\text{cost}}(a; x) := \max \{L(a_1^1; x), \ldots, L(a^M; x)\},
$$

where

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

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

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

The $\theta$ is the deep neural network parameter of the policy $\pi$. The optimal parameter $\theta^*$ can be determined by solving the following optimization problem:

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

where $P(x)$ is the distribution of problem $x$.

**Methodology**

This section presents the architecture of Equity-Transformer $\pi_\theta(a|x)$, which generates an action sequence $a =
(a_1, \ldots, a_{N+M}) for a given problem 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 x.
2. Employ the encoder of Transformer (Vaswani et al. 2017) to the initial node embedding to obtain \( H = [h_1, \ldots, h_{N+M}] \in \mathbb{R}^{D \times (N+M)} \), where \( D \) is the embedding dimension.
3. Iterative decoding \( t = 1, \ldots, N + M \) using
   - (a) Equity context encoding for \( c_t \in \mathbb{R}^D \).
   - (b) Decoding to produce \( a_t \sim \pi_\theta(a_t | H, c_t) \) 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 x into two distinct components: the cities, denoted as \( x_{\text{city}} \), with elements \( \{x_i\}_{i=1}^N \), and the agents, represented by \( x_{\text{agent}} \) with elements \( \{x_j\}_{j=N+1}^{N+M} \). In order to facilitate sequential relationships between the agents, we employ positional encoding \( f_{\text{PE}} \) to \( x_{\text{agent}} \). 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 \( x_{\text{city}} \) and \( f_{\text{PE}}(x_{\text{agent}}) \) 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 \( x_{\text{agent}} \) 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, \( c_t \). Each of the ingredient contains useful information for equitable decoding, described as follows:

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

- **Agent context \( h_{\text{Agent}} \in \mathbb{R}^D \)**: We set agent context using representations of the returning depot and the last visited city of the current active agent \( m_t \). Precisely, \( h_{\text{Agent}} = g_{\text{Agent}}(h_{N+1} \oplus h_{a_{t-1}}) \), where \( g_{\text{Agent}} : \mathbb{R}^{2D} \rightarrow \mathbb{R}^D \) represents a linear projection. This context highlights the currently active agents.

- **Scale context \( h_{\text{Scale}} \in \mathbb{R}^D \)**: 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 \( h_{\text{Scale}} = g_{\text{Scale}}(N_t / M_t) \), 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 \( h_{\text{Dist}} \in \mathbb{R}^D \)**: 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 \( d_{\text{Source}} \), which represents the
current tour length of the active agent. Secondly, we utilize $d_i^{\text{target}}$, which denotes the maximum distance between the depot and the remaining unvisited cities. Subsequently, we form $h_i^{\text{dist}} = g^{\text{dist}}(d_i^{\text{source}} \oplus d_i^{\text{target}})$, 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 $f_{CE} : \mathbb{R}^{4D} \rightarrow \mathbb{R}^D$, which is a multi-layer perception (MLP), produces equity context $e_t$ using these four contexts, i.e.,

$$e_t = f_{CE}(h_{\text{problem}} \oplus h_{\text{agent}} \oplus h_{\text{scale}} \oplus h_{\text{dist}}).$$

We adopt an approach similar to that of Kool, van Hoof, and Welling (2019), where use $e_t$ as a contextual query for the attention-based decoder $\pi_0(a_t | H, e_t)$, 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 transformation of problem instance $x$ a symmetric transformation of problem instance $x$, following Kool, van Hoof, and Welling (2019), where use $c_t$ as a contextual query for the attention-based decoder $\pi_0(a_t | H, e_t)$, as shown in Figure 3. This utilization of task-equity information from the equity context enables the decoder to sequentially generate balanced tours.

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 $P(x)$, 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 $100 \times$ 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 $\lambda$-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-
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.

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

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

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†
Importantly, in certain instances, both AM and HAM produce identical cost values as the number of agents 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.

![Figure 5: Time-performance trade-off graph for mPDP. The left and bottom indicate the Pareto frontier.](image)

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.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$M$</th>
<th>OR-Tools</th>
<th>AM</th>
<th>AM$^\dagger$</th>
<th>HAM</th>
<th>HAM$^\dagger$</th>
<th>ET ($ours$)</th>
<th>ET$^\dagger$ ($ours$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>20.96 (60)</td>
<td>18.76 (600)</td>
<td>15.88 (0.33)</td>
<td>15.65 (0.67)</td>
<td>5.69 (0.33)</td>
<td>5.30 (0.55)</td>
<td>5.03 (0.54)</td>
</tr>
<tr>
<td>200</td>
<td>15</td>
<td>13.96 (80)</td>
<td>8.46 (600)</td>
<td>15.88 (0.33)</td>
<td>15.57 (0.69)</td>
<td>5.21 (0.34)</td>
<td>5.09 (0.57)</td>
<td>3.91 (0.55)</td>
</tr>
<tr>
<td>20</td>
<td>10.67 (60)</td>
<td>5.70 (600)</td>
<td>15.88 (0.35)</td>
<td>15.55 (0.71)</td>
<td>5.21 (0.35)</td>
<td>5.09 (0.61)</td>
<td>3.39 (0.56)</td>
<td>3.18 (0.61)</td>
</tr>
<tr>
<td>50</td>
<td>16.99 (60)</td>
<td>16.99 (600)</td>
<td>26.98 (0.82)</td>
<td>26.15 (3.10)</td>
<td>9.10 (0.84)</td>
<td>8.86 (1.92)</td>
<td>4.38 (1.33)</td>
<td>4.11 (1.55)</td>
</tr>
<tr>
<td>500</td>
<td>12.99 (60)</td>
<td>12.65 (600)</td>
<td>26.98 (0.86)</td>
<td>26.14 (3.17)</td>
<td>9.10 (0.84)</td>
<td>8.87 (1.95)</td>
<td>3.75 (1.36)</td>
<td>3.52 (1.62)</td>
</tr>
<tr>
<td>50</td>
<td>10.99 (60)</td>
<td>10.41 (600)</td>
<td>26.98 (0.85)</td>
<td>26.14 (3.20)</td>
<td>9.10 (0.88)</td>
<td>8.87 (1.95)</td>
<td>3.44 (1.38)</td>
<td>3.23 (1.66)</td>
</tr>
<tr>
<td>1000</td>
<td>75</td>
<td>14.00 (60)</td>
<td>14.00 (600)</td>
<td>40.86 (1.61)</td>
<td>39.63 (11.28)</td>
<td>15.12 (1.63)</td>
<td>14.58 (6.35)</td>
<td>4.91 (2.63)</td>
</tr>
<tr>
<td>100</td>
<td>11.00 (60)</td>
<td>10.98 (600)</td>
<td>40.86 (1.69)</td>
<td>39.63 (11.24)</td>
<td>15.12 (1.72)</td>
<td>14.61 (6.61)</td>
<td>3.56 (2.75)</td>
<td>3.38 (3.80)</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>21.00 (60)</td>
<td>21.00 (600)</td>
<td>62.85 (3.24)</td>
<td>61.31 (24.98)</td>
<td>25.68 (3.40)</td>
<td>25.06 (15.17)</td>
<td>5.15 (5.22)</td>
</tr>
<tr>
<td>2000</td>
<td>150</td>
<td>14.00 (60)</td>
<td>14.00 (600)</td>
<td>62.85 (3.34)</td>
<td>61.28 (25.43)</td>
<td>25.68 (3.40)</td>
<td>25.04 (16.26)</td>
<td>4.17 (5.31)</td>
</tr>
<tr>
<td>200</td>
<td>11.00 (60)</td>
<td>11.00 (600)</td>
<td>62.85 (3.35)</td>
<td>61.33 (26.33)</td>
<td>25.68 (3.50)</td>
<td>25.06 (16.47)</td>
<td>3.79 (5.43)</td>
<td>3.62 (10.01)</td>
</tr>
<tr>
<td>300</td>
<td>17.00 (60)</td>
<td>17.00 (600)</td>
<td>114.73 (8.30)</td>
<td>112.84 (180)</td>
<td>54.07 (34.65)</td>
<td>53.46 (279)</td>
<td>4.81 (52.66)</td>
<td>4.60 (79.23)</td>
</tr>
<tr>
<td>500</td>
<td>400</td>
<td>13.00 (60)</td>
<td>13.00 (600)</td>
<td>114.73 (8.31)</td>
<td>112.90 (182)</td>
<td>54.07 (34.44)</td>
<td>53.43 (283)</td>
<td>4.33 (54.86)</td>
</tr>
<tr>
<td>500</td>
<td>11.00 (60)</td>
<td>11.00 (600)</td>
<td>114.73 (8.33)</td>
<td>112.83 (186)</td>
<td>54.07 (34.46)</td>
<td>53.45 (286)</td>
<td>4.12 (54.77)</td>
<td>3.88 (82.87)</td>
</tr>
</tbody>
</table>
The robustness of Equity-Transformer to the changes in problem comparing it with LKH3. These experiments confirmed the ratios (Appendix D.2), and different of Equity-Transformer under various problem distributions almost all instances. Moreover, we assessed the robustness of Equity-Transformer in a real-world benchmark dataset (Appendix C). Among the baseline methods, Equity-Transformer demonstrates superior performance in most deep learning-based VRPs studied focus 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


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


Liang, H.; Ma, Y.; Cao, Z.; Liu, T.; Ni, F.; Li, Z.; and Hao, J. 2023. SplitNet: a reinforcement learning based sequence


