Multi-Agent Tsallis Actor–Critic for Autonomous Vehicle Fleet Coordination on Road Graph Networks

Geunje Cheon[1], **Junseok Kim**[2], **Gunmin Lee**[2], **Subin Shin**[3], **Jeongho Park**[2], **Jaewon Lee**[2], **Hyeokjin Kwon**[1], **Songhwai Oh**[1,2,3,*]

[1]Interdisciplinary Program in Artificial Intelligence and ASRI, Seoul National University
{geunje.cheon, hyeokjin.kwon, songhwai}@rllab.snu.ac.kr
[2]Department of Electrical and Computer Engineering and ASRI, Seoul National University
{junseok.kim, gunmin.lee, jaewon.lee, jeongho.park}@rllab.snu.ac.kr
[3]Department of Future Automotive Mobility and ASRI, Seoul National University
subin.shin@rllab.snu.ac.kr
*Corresponding author

Abstract: We introduce the multi-vehicle road graph delivery problem, aimed at solving real-time delivery tasks using autonomous vehicle fleets. Our approach utilizes road graph representations to accurately capture the characteristics of urban road networks. To address this problem efficiently, we propose a multi-agent reinforcement learning (MARL) framework incorporating an attention-based state encoder, which effectively encodes the road network structure and package information. Our modified implementation of a multi-agent Tsallis actor-critic (MATAC) algorithm, combined with the state encoder, is trained to collaboratively minimize delivery makespan using individualized rewards that encourage cooperative vehicle routing behaviors. Experimental results on multiple benchmark maps demonstrate that our algorithm significantly reduces the makespan more than heuristic approaches, while achieving inference times much faster than an exact optimization method with competitive solution quality. These results highlight the applicability of our method for large-scale real-time delivery scenarios involving autonomous vehicles.

Keywords: Multi-Agent Reinforcement Learning, Road Graph Representation

1. INTRODUCTION

Recent advances in autonomous vehicle technology have paved the way for fleets of self-driving cars to revolutionize urban goods delivery [1]. The delivery system needs to devise strategies that enable quick deliveries of ordered packages. While the delivery problem resembles autonomous mobility-on-demand systems (AMoD) [2–5] often explored in intelligent transportation systems community, differences like handling self-moving passengers and post-service rebalancing make AMoD approaches less applicable. Thus, there is a need for research focused on control strategies tailored for delivery systems utilizing autonomous vehicle fleets. Despite limited research on this particular topic, there has been consistent interest in related problems.

In particular, single-vehicle delivery on a customer-based graph has been widely studied in operations research. The customer-based graph is a complete graph where nodes represent customers and edges indicate the travel duration between them. This problem is typically framed as the pickup and delivery problem (PDP), a variant of the vehicle routing problem (VRP) [6–9], where its solution is expressed as an ordered sequence of customer nodes. Recent studies have applied deep reinforcement learning (RL) methods to autoregressive policy models [10–13], such as the work by Li et al. [14], who used RL by leveraging a neural network integrated with a heterogeneous attention mechanism to account for the distinct roles of pickup and delivery nodes.

RL has been utilized to address the multi-agent VRP on the customer-based graph, an extension of the single-vehicle problem to multiple vehicles. Various RL approaches have been proposed, such as decomposing the problem into node assignment and route planning [15], or designing a Markov decision process (MDP) that considers asynchronous decision-making nature of vehicles in the problem [16, 17].

Customer-based graphs present several limitations due to their simplification [18]. These include challenges in representing multiple packages within warehouses and tracking vehicle positions before they arrive at nodes. To address the issues of the customer-based graph, a road graph is used as an alternative. By modeling road connections as edges and treating intersections as nodes, the road graph captures the underlying graphical structure of an urban road network. This graphical representation allows expressing a warehouse that is allocated with multiple packages and represents vehicle positions precisely through discretization of edge weights. In this regard, the road graph serves as a suitable abstraction for solving complex delivery problems.

To tackle the delivery problem, we encode the road graph into an expressive state representation and apply multi-agent reinforcement learning (MARL) [19–22], which has recently gained attention for solving multi-robot tasks. MARL is well-suited for optimizing multi-agent delivery tasks, enabling vehicles to learn cooperative strategies, share traffic and routing information, and ultimately improve delivery efficiency while minimizing redundancies.

In this work, we propose a problem termed the multi-vehicle road graph delivery problem, specifically designed for autonomous delivery fleets. In the proposed problem, autonomous vehicles distributed throughout the road graph aim to achieve fast completion of customer delivery orders. This formulation overcomes the limita-

This work was supported by Korea Evaluation Institute of Industrial Technology(KEIT) grant funded by the Korea government(MOTIE) (No.20024368, Development of Integrated Control System for Multi-mobility and Autonomous Delivery Service Demonstration)

tions of previous methods that relied on customer-based graphs. We introduce a novel encoding method for the road graph, which is used to represent a state. We also introduced a MARL algorithm, multi-agent Tsallis actor-critic (MATAC), which leverages centralized network models and individualized objectives. By integrating this algorithm with the state encoder, MATAC achieves practical inference times while maintaining competitive solution quality for real-world scale problems.

2. PRELIMINARIES AND PROBLEM DESCRIPTION

2.1 Multi-Vehicle Road Graph Delivery Problem

The multi-vehicle road graph delivery problem aims to deliver given packages on a road graph using homogeneous autonomous vehicles in the shortest possible time. The problem is described as follows.

2.1.1 Road Graph

A road graph $\boldsymbol{G}$ is a static representation of a specific region, abstractly simplified to support local delivery service operations. The undirected graph $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{E})$ consists of $\Delta_{\boldsymbol{N}}$ nodes $\boldsymbol{N} = \{n_1, \ldots, n_{\Delta_{\boldsymbol{N}}}\}$ and edges $\boldsymbol{E} = \{(n, n') \mid n, n' \in \boldsymbol{N},\ n \text{ and } n' \text{ are connected}\}$. A node is classified as one of the following: a warehouse where packages are prepared for dispatch, a destination representing the final delivery point for a package, or an intersection that serves as a transition point for vehicles. Each edge in the road graph is assigned a weight, given by the function $w : \boldsymbol{E} \to \mathbb{R}^+$, which represents the effective length of the road by taking into account factors such as traffic volume and vehicle speed.

2.1.2 Packages

$\boldsymbol{P} = \{p_1, \ldots, p_{\Delta_{\boldsymbol{P}}}\}$ is the set of packages over $\Delta_{\boldsymbol{P}}$ packages. Packages are the components to be delivered which have designated warehouse and destination nodes. The association between each package and its warehouse node is defined by the mapping $f_{\text{wh}} : \boldsymbol{P} \to \boldsymbol{N}$, while the link between each package and its destination node is specified by an injective mapping $f_{\text{dst}} : \boldsymbol{P} \to \boldsymbol{N}$.

2.1.3 Vehicles

$\boldsymbol{V} = \{v_1, \ldots, v_{\Delta_{\boldsymbol{V}}}\}$ is the set of vehicles over $\Delta_{\boldsymbol{V}}$ vehicles. Vehicles are the components responsible for performing deliveries. All vehicles are assumed to operate at the same speed and share a container capacity of Δ_C, with container compartment of a vehicle $v \in \boldsymbol{V}$ represented as $\boldsymbol{C}_v = \{c_1, \ldots, c_{\Delta_{\boldsymbol{C}}}\}$. Each package is stored in a single compartment of the container when loaded.

2.1.4 Constraints

The movement of a vehicle is constrained by a road graph. As a vehicle can only travel along roads in the real world, if the vehicle is located at node $n \in \boldsymbol{N}$, it can move from n to any connected node $n' \in \boldsymbol{N}$. In addition, we discretize edge weight $w(n, n')$ to sync the status of the vehicles in the road graph, so that the vehicle takes $\lceil w(n, n') \rceil$ time to traverse the edge. Additionally, a vehicle's load operation is constrained in terms of container capacity. The container capacity constraint makes the vehicle unable to load any packages if the container is full.

2.1.5 Objective

The objective of this problem is

$$J = \min_t \max_{v \in \boldsymbol{V}} (t_{\text{end},v}), \tag{1}$$

where $t_{\text{end},v}$ represents the delivery completion time for each vehicle. The completion time of a vehicle's delivery is defined as the earliest moment when the vehicle does not contain any package, and no packages are left at any warehouse nodes.

2.2 Markov Game

The proposed problem can be formulated as a Markov game (MG) $\mathcal{M} = (\mathcal{V}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \rho_0, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ denotes the state space, $\boldsymbol{\mathcal{A}} = \prod_{v=1}^{\Delta_{\boldsymbol{V}}} \mathcal{A}_v$ represents the joint action space with $\mathcal{A}_v$ denoted as an individual action space of identical size $|\mathcal{A}|$, ρ_0 is the initial state distribution, $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to [0, 1]$ denotes the transition function, $\boldsymbol{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}^{\Delta_V}$ represents the joint reward function with each element $\boldsymbol{R}_v$ means an individual reward function, $\gamma \in [0, 1)$ is the discount factor. Vehicles are treated as agents in the MG. Additionally, the joint policy in the MG, $\boldsymbol{\pi} : \mathcal{S} \to [0, 1]^{\Delta_V \times |\mathcal{A}|}$, is formed by concatenating the individual policies $\pi_v : \mathcal{S} \to [0, 1]^{|\mathcal{A}|}$ for each agent. The objective is to maximize each individual cumulative discounted return

$$J(\pi_v) = \mathbb{E}_{s_{1:T} \sim \rho_0, \mathcal{T}, a_{1:T} \sim \boldsymbol{\pi}} \left[\sum_{t=1}^{T} \gamma^t \boldsymbol{R}_{v,t}(s_t, a_t) \right] \tag{2}$$

for given time horizon T. This definition of the MG is similar to a MG formulation presented by Fu et al. [23].

3. METHOD

To address the problem outlined in Section 2.1 within the MARL framework, it is crucial to define a MG for the multi-vehicle road graph delivery problem. Based on the MG-formulated state representation, we developed a novel state encoder. Integrating this state encoder with MATAC enables agents to collaboratively learn policies that facilitate efficient and fast package deliveries.

3.1 Formulation as a Markov Game

The MG formulation of the multi-vehicle road graph delivery problem also serves as a definition of the MARL environment. Within the simulation environment made from the MG formulation, MARL algorithms can be trained to develop policies that optimize the objective of the problem.

3.1.1 State Space

$\mathcal{S}$ is composed of three components: the information about all agents, the warehouse-package mask, and the node adjacency mask. Each agent's information includes the source node, target node, normalized distance from the source node, and container status. The source node $n_{\text{src}} \in \boldsymbol{N}$ is the agent's most recent departure point, while the target node $n_{\text{tgt}} \in \boldsymbol{N}$ is the intended destination. The normalized distance from the source node d_{src} is defined

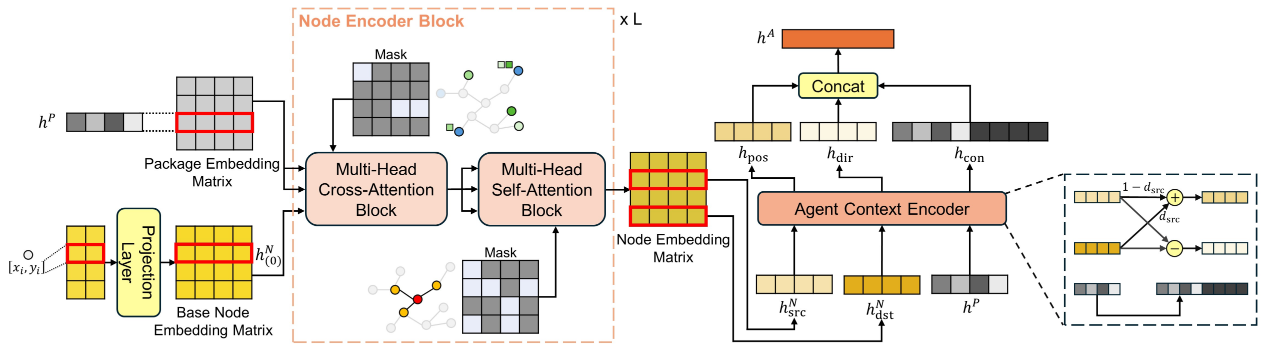

Fig. 1.: The overall structure of the proposed state encoder. The base embedding (left), the node encoder (mid-left), the agent context encoder (mid-right), and the process of creating agent context embedding are illustrated using an example scenario.

as a ratio of the distance between the source node and the agent $\tilde{d}_{\text{src}}$ to the weight of the edge $w(n_{\text{src}}, n_{\text{tgt}})$. In addition, the stored packages of a vehicle $v \in \boldsymbol{V}$ can be represented by a mapping $f_{\text{con},v} : \boldsymbol{C}_v \to \boldsymbol{P} \cup \{p_{\text{empty}}\}$, where p_{empty} denotes a fictitious empty package. Therefore, the container status of an agent can be expressed as $[f_{\text{con},v}(c_1), \ldots, f_{\text{con},v}(c_{\Delta_{\boldsymbol{C}}})]$.

A warehouse-package mask $M^{\boldsymbol{NP}}$ and a node adjacency mask $M^{\boldsymbol{NN}}$ are matrices of size $\Delta_{\boldsymbol{N}} \times \Delta_{\boldsymbol{N}}$ designed to ignore unrelated pairs during attention. The warehouse-package mask has a value of 0 in position $(f_{\text{wh}}(p), f_{\text{dst}}(p))$ for a package $p \in \boldsymbol{P}$ stored in $f_{\text{wh}}(p)$ at given time, and $-\infty$ otherwise. Similarly, the node adjacency mask has a value of 0 at positions $(n, n') \in \boldsymbol{E}$, and $-\infty$ otherwise, representing the edge connections in the road graph.

3.1.2 Action Space

An individual action space $\mathcal{A}_v$ of vehicle $v \in \boldsymbol{V}$ consists of three types of actions: no action, move action, and load action. Since unloading is automatically performed on a package's destination, it is excluded from the action space. The *no action* corresponds to maintaining the current state without performing a move or load operation and is represented as 0. The *move action* is for moving to an adjacent target node n_{tgt} and is represented as $1 + n_{\text{tgt}}$. The *load action* is used to load a package p from a warehouse node to the agent container and is represented as $1 + \Delta_{\boldsymbol{N}} + f_{\text{dst}}(p)$. Note that $|\mathcal{A}| = 1 + 2\Delta_N$.

3.1.3 Transition Function

The transition function $\mathcal{T}$ is defined to be deterministic and strictly follows the movement and loading constraints presented in Section 2.1. Note that invalid actions, such as moving to non-adjacent nodes or attempting to load unavailable packages, are prohibited. The episode ends when all packages have reached their assigned destinations.

3.1.4 Reward Function

An individualized reward $\boldsymbol{R}_v$ of vehicle $v \in \boldsymbol{V}$ is calculated as a weighted sum of the time penalty reward and the two-stage delivery reward. The first reward

$$r_{v,p} = -1/T_{\max}, \tag{3}$$

where $T_{\max}$ denotes the maximum allowed time, motivates agents to complete deliveries more quickly through cooperation. If an agent behaves greedily and delays the deliveries of others, it incurs greater time penalties.

The second delivery reward

$$r_{v,d} = \begin{cases} 0.5 & \text{if } v \text{ loads or unloads a package} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

promotes intermediate objectives such as loading and unloading packages, enhancing the reward structure, and encouraging faster deliveries.

3.2 State Encoder

The state encoder is composed of three parts: the base embedding, the node encoder, and the agent context encoder. The base embedding consists of a package embedding matrix $h^{\boldsymbol{P}}$ and a base node embedding matrix $h^{\boldsymbol{N}}_{(0)}$. The package embedding matrix is composed of a random vector sampled from a standard normal distribution for each package destination, while the base node embedding matrix is derived by applying a projection layer to the two-dimensional locations of the nodes. This base embedding matrix is shown in the far left of Figure 1.

The node encoder is provided with the $h^{\boldsymbol{P}}$, $h^{\boldsymbol{N}}_{(0)}$, $M^{\boldsymbol{NP}}$, $M^{\boldsymbol{NN}}$ and it outputs a node embedding matrix $h^{\boldsymbol{N}}$. Specifically, the node encoder is composed of L node encoder blocks, where each block consists of a multi-head cross-attention block and a multi-head self-attention block. The node encoder can be seen on the mid-left side of Figure 1.

The agent context encoder takes input $n_{\text{src}}, n_{\text{tgt}}, d_{\text{src}}$ from the state along with $h^{\boldsymbol{N}}$, and outputs an agent context embedding vector $h^{\boldsymbol{V}}$ for each agent. $h^{\boldsymbol{V}}$ is composed of three distinct embedding vectors that capture different aspects of a vehicle. An agent position embedding vector h_{pos} represents the agent's current location, while an agent direction embedding vector h_{dir} captures the agent's current movement direction. An agent container embedding vector h_{con} encodes the condition of the agent's container.

The node embedding vectors in $h^{\boldsymbol{N}}$ are added, subtracted, and concatenated to generate $h_{\text{pos}}, h_{\text{dir}}, h_{\text{con}}$, as briefly illustrated on the right side of Figure 1. h_{pos} is defined as a weighted sum of $h^{\boldsymbol{N}}_{\text{src}}$ and $h^{\boldsymbol{N}}_{\text{tgt}}$, where $h^{\boldsymbol{N}}_{\text{src}}$ and

$h^{\boldsymbol{N}}_{\text{tgt}}$ are corresponding node embeddings for n_{src} and n_{tgt}. Using d_{src} as a weight, h_{pos} can accurately represent the agent's position on the edge and its relative location along that edge. Similarly, h_{dir} is defined as the difference between $h^{\boldsymbol{N}}_{\text{tgt}}$ and $h^{\boldsymbol{N}}_{\text{src}}$, effectively capturing the orientation of the edge and the agent's relative direction along it. h_{pos} and h_{dir} are then computed as follows:

$$h_{\text{pos}} = (1 - d_{\text{src}})h^{\boldsymbol{N}}_{\text{src}} + d_{\text{src}}h^{\boldsymbol{N}}_{\text{tgt}}, \tag{5}$$

$$h_{\text{dir}} = h^{\boldsymbol{N}}_{\text{tgt}} - h^{\boldsymbol{N}}_{\text{src}}. \tag{6}$$

h_{con} for a given agent is created by selecting the corresponding vectors in $h^{\boldsymbol{P}}$ based on container status. Note that if a compartment is empty, an empty package embedding vector is selected to ensure consistency. The agent container embedding vector is then computed as follows:

$$h_{\text{con}} = [h^{\boldsymbol{P}}_{f_{\text{con}}(c_1)}, \ldots, h^{\boldsymbol{P}}_{f_{\text{con}}(c_{\Delta_{\boldsymbol{C}}})}]. \tag{7}$$

Finally, $h^{\boldsymbol{V}}$ is a concatenation of all three agent embedding vectors $[h_{\text{pos}}, h_{\text{dir}}, h_{\text{con}}]$. The agent context encoder is depicted on the mid-right side of Figure 1. Additionally, an encoded state s_{enc} shared among the MARL agents is obtained by combining the agent context embedding vectors of all agents and can be expressed as $[h^{\boldsymbol{V}}_1, \ldots, h^{\boldsymbol{V}}_{\Delta_{\boldsymbol{V}}}]$.

3.3 Multi-Agent Tsallis Actor-Critic for Efficient Delivery

Tsallis actor-critic (TAC) [24, 25], a variant of soft actor-critic (SAC) [26, 27], achieves superior performance over other RL algorithms when an appropriate entropic index q is applied. We extend TAC to a multi-agent setting by incorporating a centralized policy model $\boldsymbol{\pi}_\phi : \mathcal{S} \rightarrow [0,1]^{\Delta_V \times |\mathcal{A}|}$ and a Q-network model $\boldsymbol{Q}_\theta : \mathcal{S} \rightarrow \mathbb{R}^{\Delta_V \times |\mathcal{A}|}$, parameterized by ϕ and θ, respectively. This extension of TAC is referred to as MATAC in this paper.

$\boldsymbol{\pi}_\phi$ and $\boldsymbol{Q}_\theta$ are integrated with the state encoder to solve the delivery problem. An individual policy $\pi_{\phi,v}$ and a Q-value $Q_{\theta,v}$ of an agent $v \in \boldsymbol{V}$ is defined as a v-th row of the output value of $\boldsymbol{\pi}_\phi$ and $\boldsymbol{Q}_\theta$. For each agent, both $\pi_{\phi,v}$ and $Q_{\theta,v}$ are trained to optimize the individual objective described in Section 2.2. Furthermore, each agent is assigned a temperature parameter α_v, allowing α_v to regulate the Tsallis entropy of $\pi_{\phi,v}$ to a target value. Performance improvement can be made by scheduling the target Tsallis entropy. We claim this modified MATAC, integrated with the state encoder, as MATAC-RG.

MATAC-RG is considered a centralized algorithm since it leverages the global state during both training and execution. Centralized training allows each agent to observe all other agents, enabling stronger consideration of others during learning. Additionally, the reward $r_{v,p}$ encourages cooperation among agents, even when each policy is optimized to maximize its own Q-value. These two aspects lead MATAC-RG to learn policies that enable efficient collaboration and rapid delivery completion.

4. EXPERIMENT

4.1 Baselines

We evaluated our proposed approach against four baselines: a MARL method, an exact optimization technique, and two heuristic strategies. First, we implemented a modified multi-agent deep Q-network (MADQN) integrated with the state encoder, referring to it as MADQN-RG. MADQN-RG adopts the same centralized network structure and individualized objectives as MATAC-RG. Second, we adapted the optimization problem introduced in Section 2.1 into a binary integer programming (BIP) [28,29] formulation and solved it using the GUROBI optimization solver [30].

For heuristic baselines, we developed two greedy heuristics for assigning packages, each differing in the loading and unloading sequence. They are called *greedy load first* (GLF) and *greedy unload first* (GUF) strategies. In the GLF strategy, a set of packages corresponding to the vehicle's container capacity is assigned to each vehicle, prioritizing shorter delivery distances. The vehicle first visits the warehouses to load the assigned packages and then proceeds to the destinations. The remaining packages are assigned when the vehicle container becomes empty again. In contrast, the GUF strategy assigns only one package with the shortest delivery distance to each vehicle at a time. Once the unloading of that package is complete, another remaining package is assigned to the vehicle.

4.2 Experiment Setup

To compare the ability of MATAC-RG and baselines to solve problems in various scenarios, we created maps of varying complexity. We generated three types of graphs: Map0, Map1, Map2. Each graph consists of 30 nodes with different locations and connections, designating 4 nodes as warehouses and 8 as destinations. Additionally, we created a small replica based on the map of Sangam in Seoul, South Korea. This map, Sangam, is designed to evaluate performance on a road graph with realistic complexity. It contains 63 nodes with 5 nodes designated as warehouses and 10 as destinations. In total, four types of maps shown in Figure 2 were used for evaluation. In the experiments, we set $\Delta_{\boldsymbol{V}} = 3$, $\Delta_{\boldsymbol{C}} = 2$, and $\Delta_{\boldsymbol{P}}$ for a number of destination nodes of each map.

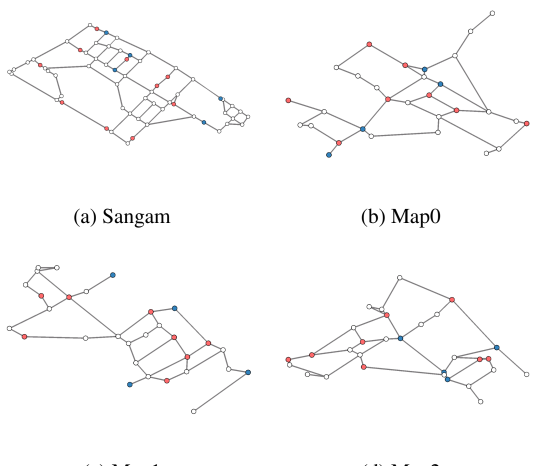

Fig. 2.: Maps used to evaluate algorithms. Warehouses are denoted as blue, and destinations are denoted as red.

Table 1.: Makespan (steps) of baselines and MATAC-RG. Lower value indicates better performance. Cases where the time limit was exceeded are marked as '-'. The best results are highlighted in **bold**, and the second-best results are highlighted in blue.

Map	BIP	GUF	GLF	MADQN-RG	MATAC-RG
Sangam	-	55.4±4.78	54.0±6.04	53.7±6.34	**49.0±3.95**
Map0	**20.0±1.73**	33.0±6.32	35.2±4.09	29.1±4.36	27.4±3.24
Map1	**23.4±0.894**	39.2±5.50	41.0±6.40	32.8±3.09	32.6±3.18
Map2	**22.8±3.70**	38.4±3.91	37.0±7.28	31.2±4.75	30.8±2.61

Table 2.: Inference time (seconds) of baselines and MATAC-RG. Cases where the time limit was exceeded are marked as '-'. The best results are highlighted in **bold**, the second-best results are highlighted in blue, and the worst results are highlighted in red.

Map	BIP	GUF	GLF	MADQN-RG	MATAC-RG
Sangam	-	**5.29e-5±3.01e-6**	5.80e-5±6.50e-6	2.39e-3±3.79e-5	2.24e-3±2.99e-5
Map0	4.77e1±2.12e1	**3.19e-5±4.11e-6**	4.63e-5±1.83e-5	2.27e-3±2.84e-5	2.29e-3±3.44e-5
Map1	8.66e1±6.90e1	4.76e-5±2.60e-5	**3.10e-5±4.03e-6**	2.32e-3±1.97e-5	2.27e-3±2.35e-5
Map2	3.85e2±2.91e2	**5.20e-5±4.66e-5**	6.65e-5±5.80e-5	2.35e-3±1.41e-5	2.28e-3±2.81e-5

4.3 Evaluation metric

The comparison is conducted using two criteria: makespan and per-step inference time. Makespan refers to the delivery completion time objective of the multi-vehicle road graph delivery problem, while per-step inference time represents the time needed for the algorithm's inference in a single environment step.

We evaluated the proposed method using a fixed map and five distinct initial conditions, which varied in terms of the initial positions of vehicles, f_{wh}, and f_{dst} associated with each scenario. Performance metrics were computed for each condition and averaged to obtain a representative evaluation. For MARL algorithms, we trained each model for 5,000 episodes, and further trained five independent models using different random seeds to account for variance in learning outcomes. The final results are reported as the mean and standard deviation of the evaluation metrics, aggregated across both the initial conditions and the independently trained models.

5. RESULT

Table 1 presents the evaluation results for makespan, a key indicator of solution quality. In terms of makespan, MARL algorithms outperform heuristic methods by an average reduction of over 11%. Additionally, in Map0, Map1, and Map2, MATAC-RG outperforms MADQN-RG in terms of makespan by a small margin, while the performance gap becomes significant in the larger Sangam map. This is likely because the Tsallis entropy regularization and target entropy scheduling used in MATAC-RG provide more effective exploration compared to MADQN-RG.

As shown in Table 2, in terms of inference time, MARL algorithms significantly outperform the exact optimization method. In BIP-solvable three maps, per-step inference time of BIP is at least 20,000 times bigger than that of the MARL algorithms. As the solution space expands, the inference time for the exact optimization method, such as BIP, becomes considerably longer compared to other methods. Even when the problem complexity is fixed, the inference time for BIP exhibits substantial variability, making it nearly impossible to predict in real time. In contrast, MARL methods require less inference time than BIP because they learn a generalized policy through exploration over various initial states during the training process, thereby reducing the need for extensive search at each inference step. In conclusion, since it requires relatively low inference time while maintaining solution quality, MATAC-RG demonstrates potential for use in large-scale real-time delivery applications.

Furthermore, for the Sangam map, BIP was unable to solve the problem due to a time limit, whereas our method, although suboptimal, outperformed the heuristic methods. This demonstrates that the encoding method we proposed retains its expressiveness and operates in a scalable manner, even for larger problems.

6. CONCLUSION

We present the multi-vehicle road graph delivery problem, addressed through the need for autonomous vehicle fleets in delivery. We also introduce a MARL environment, a state encoder, and MATAC to solve the problem effectively. The attention-based state encoder transforms road graph data into a rich encoded representation. At the same time, MATAC integrated with the state encoder, combined with target entropy scheduling, effectively manages the exploration-exploitation trade-off in the large discrete state space of this MARL environment. Experimental results demonstrate that the MARL algorithms outperform exact optimization approaches in inference time and exceed the performance of heuristic methods in terms of makespan, highlighting the effectiveness of the MARL framework. Future research aims to improve the MARL approach to generate high-quality plans under dynamic traffic conditions and road constraints.

REFERENCES

[1] C. Chen, E. Demir, Y. Huang, and R. Qiu, "The adoption of self-driving delivery robots in last mile logistics," *Transportation research part E: logistics and transportation review*, vol. 146, p. 102214, 2021.

[2] M. Salazar, N. Lanzetti, F. Rossi, M. Schiffer, and

M. Pavone, "Intermodal autonomous mobility-on-demand," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3946–3960, 2019.

[3] C. Ruch, C. Lu, L. Sieber, and E. Frazzoli, "Quantifying the efficiency of ride sharing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5811–5816, 2020.

[4] YingLu, Y. Liang, Z. Ding, Q. Wu, T. Ding, and W.-J. Lee, "Deep reinforcement learning-based charging pricing for autonomous mobility-on-demand system," *IEEE Transactions on Smart Grid*, vol. 13, no. 2, pp. 1412–1426, 2021.

[5] S. Wollenstein-Betech, M. Salazar, A. Houshmand, M. Pavone, I. C. Paschalidis, and C. G. Cassandras, "Routing and rebalancing intermodal autonomous mobility-on-demand systems in mixed traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 263–12 275, 2021.

[6] D. G. B and R. J. H, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[7] J. Renaud, F. F. Boctor, and J. Ouenniche, "A heuristic for the pickup and delivery traveling salesman problem," *Computers and Operations Research*, vol. 27, no. 9, pp. 905–916, 2000.

[8] H. Tang and E. Miller-Hooks, "A TABU search heuristic for the team orienteering problem," *Computers and Operations Research*, vol. 32, pp. 1379–1407, 2005.

[9] H. B. Ticha, N. Absi, D. Feillet, and A. Quilliot, "Vehicle routing problems with road-network information: State of the art," *Networks*, vol. 72, no. 3, pp. 393–406, 2018.

[10] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. of the Advances in Neural Information Processing Systems*, 2018.

[11] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. of the International Conference on Learning Representations*, 2019.

[12] Y. Kwon, J. Choo, I. Yoon, M. Park, D. Park, and Y. Gwon, "Matrix encoding networks for neural combinatorial optimization," in *Proc. of the Advances in Neural Information Processing Systems*, 2021.

[13] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7208–7218, 2021.

[14] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang, "Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2306–2315, 2021.

[15] X. Li, W. Luo, M. Yuan, J. Wang, J. Lu, J. Wang, J. Lü, and J. Zeng, "Learning to optimize industry-scale dynamic pickup and delivery problems," in *Proc. of the International Conference on Data Engineering*, 2021.

[16] G. Bono, J. S. Dibangoye, O. Simonin, L. Matignon, and F. Pereyron, "Solving multi-agent routing problems using deep attention mechanisms," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7804–7813, 2020.

[17] R. Gama, D. Fuertes, C. R. del Blanco, and H. L. Fernandes, "Multi-agent environments for vehicle routing problems," *CoRR*, vol. abs/2411.14411, 2024.

[18] H. B. Ticha, N. Absi, D. Feillet, and A. Quilliot, "Vehicle routing problems with road-network information: State of the art," *Networks*, vol. 72, no. 3, pp. 393–406, 2018.

[19] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 2, pp. 156–172, 2008.

[20] Z. Kaiqing, Y. Zhuoran, and B. Tamer, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

[21] D. Huh and P. Mohapatra, "Multi-agent reinforcement learning: A comprehensive survey," *CoRR*, vol. abs/2312.10256, 2023.

[22] J. Orr and A. Dutta, "Multi-agent deep reinforcement learning for multi-robot applications: A survey," *Sensors*, vol. 23, no. 7, p. 3625, 2023.

[23] H. Fu, K. Tang, P. Li, G. Deng, and C. Chen, "Multi-agent reinforcement learning with hybrid action space for free gait motion planning of hexapod robots," in *Proc. of the Conference on Robot Learning*, 2024.

[24] K. Lee, S. Kim, S. Lim, S. Choi, M. Hong, J. I. Kim, Y.-L. Park, and S. Oh, "Generalized tsallis entropy reinforcement learning and its application to soft mobile robots." in *Proc. of the Robotics: Science and Systems*, 2020.

[25] K. Lee, S. Kim, S. Lim, S. Choi, and S. Oh, "Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning," *CoRR*, vol. abs/1902.00137, 2019.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of the International Conference on Machine Learning*, 2018.

[27] P. Christodoulou, "Soft actor-critic for discrete action settings," *CoRR*, vol. abs/1910.07207, 2019.

[28] A. M. Geoffrion and R. E. Marsten, "Integer programming algorithms: A framework and state-of-the-art survey," *Management Science*, vol. 18, no. 9, pp. 465–491, 1972.

[29] K. Genova and V. Guliashki, "Linear integer programming methods and approaches–a survey," *Journal of Cybernetics and Information Technologies*, vol. 11, no. 1, 2011.

[30] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com