

MAPDP: Cooperative Multi-Agent Reinforcement Learning to Solve Pickup and Delivery Problems

Zefang Zong^{1,2}, Meng Zheng³, Depeng Jin^{1,2}, Yong Li^{1,2,*}

¹Beijing National Research Center for Information Science and Technology

²Department of Electronic Engineering, Tsinghua University, Beijing, China

³Hitachi (China) Research Development Corporation

Abstract

Cooperative Pickup and Delivery Problem (PDP), as a variant of the typical Vehicle Routing Problems (VRP), is an important formulation in many real-world applications, such as on-demand delivery, industrial warehousing, etc. It is of great importance to efficiently provide high-quality solutions of cooperative PDP. However, it is not trivial to provide effective solutions directly due to two major challenges: 1) the structural dependency between pickup and delivery pairs require explicit modeling and representation. 2) the cooperation between different vehicles is highly related to solution exploration and is difficult to model. In this paper, we propose a novel multi-agent reinforcement learning-based framework to solve the cooperative PDP (MAPDP). First, we design a paired context embedding to well measure the dependency of different nodes considering their structural limits. Second, we utilize cooperative multi-agent decoders to leverage the decision dependence among different vehicle agents based on a special communication embedding. Third, we design a novel cooperative A2C algorithm to train the integrated model. We conduct extensive experiments on a randomly generated dataset and a real-world dataset. Experiments result shown that the proposed MAPDP outperforms all other baselines by at least 1.64% in all settings, and shows significant computation speed during solution inference.

1 Introduction

Vehicle Routing Problem (VRP) has shown its importance in formulating many real-world applications, including express systems, industrial warehousing, and on-demand delivery (Zong et al. 2021). A fleet of vehicles/couriers are managed to fulfill given demands, while the goal is to optimize the routing plan to reduce traveling expenses. In real-world scenarios, a given demand order usually has its own origination and a designated delivery destination. Furthermore, routing tasks are usually assigned to multiple vehicles simultaneously, which forms the Pickup and Delivery Problem (PDP) (Savelsbergh and Sol 1995). For instance, on-demand delivery couriers need to pick up the food first from the restaurants and then deliver it to the corresponding customers, as shown in 1. While in industrial manufacturing, large amounts of materials, productions also need to be

transported from specific factories and warehouses at multiple sites (Li et al. 2021b). Efficiently generating high-quality solutions of cooperative PDP can help reduce operating expenses, improve public efficiency and thus bring significant benefits.

Owing to the NP-hard nature, the cooperative PDP along with other VRP variants is still difficult to be optimally solved by exact methods (Toth and Vigo 2002; Madsen, Fisher, and Jornsten 1997). Even though numerous heuristic-based methods are developed to compute near-optimal solutions, the solution generation process remains time-consuming and there is still potential to find more approximate ones. The recent development of deep reinforcement learning (DRL) offers its effectiveness in solving many combinatorial optimization problems including VRPs, and thus brings another perspective to solve PDP (Bello et al. 2016; Kool, van Hoof, and Welling 2018; Nazari et al. 2018; Chen and Tian 2019; Lu, Zhang, and Yang 2019a). Benefiting from learning a parameterized model instead of relying on manually constructed rules to search for solutions, DRL has shown its appealing performance in typical routing problems. Besides, by splitting the phases of training and online inferring, DRL can generate results with much faster computation. Inspired by both high solution quality and inference speed, it is prospective to well solve cooperative PDP by constructing a DRL framework.

However, most existing DRL based methods can only be applied to typical VRPs where the demands share a common final delivery site, i.e., either to be picked up or to be delivered only. They are less effective when facing cooperative PDP with more detailed constraints and challenges from the following two aspects. First, it is not trivial to measure and structural constraint of PDP among different nodes. Compared to typical VRPs, complicated relations among all nodes should be considered. Representation upon the relation between paired pickup and deliveries should differ from that between unpaired ones. How to efficiently measure such relations is essential for framework effectiveness. Second, it is difficult to coordinate the cooperation within the vehicle fleet. As all vehicles working simultaneously to accomplish the pickup and delivery tasks, the potential decision space of one is directly influenced by others. Optimizing the coordinated routing plans as the global objective is different from optimizing the one of a single vehicle, where the dependence

*Corresponding Author

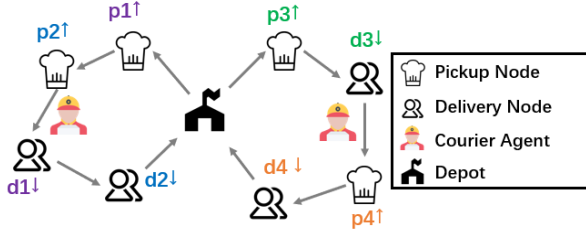


Figure 1: An application visualization of the cooperative PDP in on-demand delivery.

among them must be well modeled and measured.

To tackle the challenges above, we propose a novel end-to-end cooperative multi-agent RL (MARL) based framework to solve cooperative PDP (MAPDP). MAPDP learns to generate the next node to visit step by step for each vehicle agent and finally outputs a complete routing plan. First, we develop a paired context embedding to represent each pickup or delivery node in each order instance. The paired context embedding well models the inter-relation in-between that will further influence the potential solution space. Second, we utilize cooperated decoders to leverage the decision dependence among different vehicle agents. Given the shared instance encoding results from former module designs, each vehicle agent obtains its own partial action results based on a special communication embedding. Such multi-agent cooperation is trained and could be executed in online inference via a centralized controlling framework. Third, We also design a cooperative A2C algorithm for the integrated model training, where a joint critic value net estimates the state value.

To summarize, the main contributions of our work are listed as follows:

- To the best of our knowledge, we are the first to explore the cooperative PDP with multiple vehicle agents using MARL.
- We design a centralized MARL framework to generate cooperative decisions. We design a paired context embedding to well capture the inter-dependency of heterogeneous nodes, and train different agents based on a communication embedding via a specially designed cooperative A2C algorithm.
- We evaluate the effectiveness of MAPDP on two datasets. It outperforms other baselines by at least 1.64% in all experiment settings and shows significant computation speed during solution inference.

The remainder of this paper is organized as follows. We first introduce the related works in Section 2. We then formulate cooperative PDP in Section 3, and introduce our methodology in Section 4. The experiment results are presented in Section 5. Finally, we conclude our paper in Section 6.

2 Related Works

In this section, we review the existing literature on conventional heuristic methods for solving PDP and learning based

methods for routing problems. Besides, we also introduce the recent research in multi-agent RL.

2.1 Heuristics for PDP

As an important variant of the typical vehicle routing problem (VRP), PDP considering pickups and deliveries was first studied in (Savelsbergh and Sol 1995), and exact methods were put forward early as solutions (Ropke and Cordeau 2009; Ruland and Rodin 1997). However, trying to generate exact optimality suffers from heavy computation given then exponential complexity. As a substitution, more researchers tend to utilize heuristics to generate approximate optimal solutions, which could significantly increase the efficiency with small quality costs. A tabu-embedded simulated annealing algorithm was proposed for solving large-scale PDP with time windows (Ropke and Cordeau 2009). An adaptive large neighborhood search heuristic method was presented in (Ropke and Pisinger 2006) to solve PDP, incorporating regret insertion method and six removal strategies.

Even though the heuristic-based methods could generate near-optimal solutions within a more reasonable time instead, they heavily rely on hand-crafted rules and are greatly limited by human experience. Furthermore, the online inference time of these methods is still not satisfactory when facing high dynamics.

2.2 RL based methods for Routing Problems

Due to the potential that taking feedback reward as training signals to action attempts when interacting with outside environments, RL has shown its effectiveness in solving many decision-making problems. Bello et al. first propose an RL-based algorithm to solve combinatorial optimization problems, including the famous Traveling Salesman Problem (TSP) (Bello et al. 2016). Following this idea, Nazari et al. utilize RNN structures to solve capacitated VRP by generating decision sequences (Nazari et al. 2018). Kool et al. (Kool, van Hoof, and Welling 2018) further propose a transformer based network to fully capture the in-between relationships between different nodes. Xin et al. develop a multi-decoder based framework to generate fine-tuned solutions via a special beam-search. Despite the idea of generating sequences as the output routing results for VRP, Chen et al. (Chen and Tian 2019) formulates the solution improvement process as keeping rewriting based on the current ones. Following this idea, Lu et al. (Lu, Zhang, and Yang 2019b) further combined RL with Operation Research operators to keep updating the current solutions.

However, most existing RL based approaches can only solve typical VRPs, while the cooperative PDP problem with structural dependency and vehicle cooperation requires specific modeling. Even though Li et al. (Li et al. 2021b) proposed a framework to solve dynamic PDP, the agent model is only used to dispatch new orders to one of the vehicles, while the real routing process is processed via enumeration. Such a pure dispatching framework suffers from limits on exploration to global optimization. Li et al. (Li et al. 2021a) proposed a heterogeneous attention based network to solve single-vehicle PDP, but finding solutions for cooperative PDP remains unsolved. In contrast, our proposed

MAPDP could well model both pickup-delivery dependency and vehicle cooperations.

2.3 Multi-Agent Reinforcement Learning

In many practical decision-making problems, cooperation or competition among different agents requires specific modeling, and the multi-agent reinforcement learning (Buşoniu, Babuška, and De Schutter 2010; Lowe et al. 2017) has attracted much attention. Tampuu et al. (Tampuu et al. 2017) analyzed the cooperation and competition among two agents in reinforcement learning by carefully designing the reward schemes. Kok et al. (Kok and Vlassis 2004) tried to learn the coordinated actions of a group of cooperative agents by using sparse representation of joint state-action space. Lin et al. (Lin et al. 2018) designed a multi-agent model to manage many vehicles in the city. Lee et al. (Lee and Lee 2019) further improved cooperative models by mixing demonstrations from the centralized policy.

As for the variants of VRPs, Zhang et al. (Zhang et al. 2020) proposed a MARL-based framework to solve VRP with soft time windows for a fleet of vehicles. However, their multi-agent modeling relies on manually crafted rules in which vehicles take turns to make decisions. Furthermore, all vehicles share the same policy network, which makes the framework a single agent control indeed. In contrast, we develop independent policy networks without any manual rules in vehicle coordination and thus could enlarge the exploration space of all vehicle agents.

3 Problem Formulation

In this section, we provide the mathematical formulation of cooperative PDP.

With N delivery nodes and N pickup nodes are included in a given problem instance, the node set including the initial depot v_0 where all vehicle starts can be represented as $V = \{v_0, v_1, \dots, v_N, v_{N+1}, v_{N+2}, \dots, v_{2N}, v_{2N+1}\}$, and v_{2N+1} is the copy of the depot. For simplification, we assume that $v_i (1 \leq i \leq N)$ has parcels to be picked-up and $v_i (N < i \leq 2N)$ is the target to be delivered to. v_i and v_{N+i} form a corresponding pair. The spatial distances between different nodes can be further represented as $E = \{e_{ij}\}$, where $0 \leq i \leq 2N, 0 \leq j \leq 2N$. Each pickup order has a demand volume represented as d_i , where $d_0 = d_{2N+1} = 0$. A demand is noted as a positive value for pickups and negative for deliveries, where $d_i > 0$ if $1 \leq i \leq N$, $d_i < 0$ if $N + 1 \leq i \leq 2N$ and $d_i = -d_{i+N}$. All PDP tasks are assigned to K vehicles with C_k denoting the individual capacity of the k -th vehicle.

Let $x_{ijk} \in \{0, 1\}$ denote whether the vehicle k travels directly from node v_i to node v_j , T_i as the arrival time at node v_i . $S \subseteq V$ denotes a consecutive routing sequence from v_0 and ends at v_{2N+1} and does not include v_0 in the middle, the cooperative PDP can be formally formulated as follows:

$$\min \sum_{k=1}^K \sum_{i=0}^{2N} \sum_{j=1}^{2N+1} e_{ij} x_{ijk}, \quad (1)$$

$$s.t. \sum_{k=1}^K \sum_{j=1}^{2N+1} x_{ijk} = 1, \forall i \in [0, 2N], \quad (2)$$

$$\sum_{k=1}^K \sum_{i=0}^{2N} x_{ijk} = 1, \forall j \in [1, 2N + 1], \quad (3)$$

$$\sum_{i \in S'} d_i \leq C_k, \forall S' \subseteq S, \forall k \in [1, K], \quad (4)$$

$$\sum_{j=1}^{2N+1} x_{i,jk} = \sum_{j=0}^{2N+1} x_{i+N,jk}, \forall k \in [1, K], i \in [1, N], \quad (5)$$

$$T_i \leq T_{i+N}, \forall i \in [1, N]. \quad (6)$$

The overall objective is to minimize the total traveling distance of all vehicles. Constraint (2) and (3) guarantee that each node is visited and only visited once. (4) guarantees that a vehicle never carries parcels out of its capacity limit. (5) satisfies the structural limit that each parcel should be delivered by the same vehicle as it was picked up, and (6) guarantees that a pickup is always the precondition of its own delivery.

4 Methodology

To solve the above formulated cooperative PDP, we take advantage of MARL to explicitly learn the effective cooperation among different vehicles. We develop an end-to-end framework, MAPDP, to generate partial solution sequence continuously by combining current routing actions of different agents together, as shown in Figure 2. Different agent networks share a common public context encoder to capture problem instance representations and learn their own policy via independent decoders. We further train the entire network by computing a common critic value as an approximation to the cooperation quality.

4.1 Multi-Agent Reinforcement Learning Setting

The construction for PDP solutions can be formulated as a sequence generation process. The sequence is completed gradually and can be modeled as a Markov Decision Process (MDP). We define the essential elements within as follows.

- **State:** The state of agent k at step t includes the remaining available capacity C_k^t the current traveling trajectory S_k^t . Specifically, the current location, i.e., the last node visited by agent k is represented as $v_{I_k^t}$, where I_k^t is the node index. Note that $v_{I_k^0} = v_0$ and $C_k^0 = C_k$. In the cooperative PDP setting, we assume that all vehicles can communicate via a centralized control so that all states are fully observable.
- **Action:** The action at step t for vehicle agent k is to determine a node as its next target, represented as $v(k, t)$.

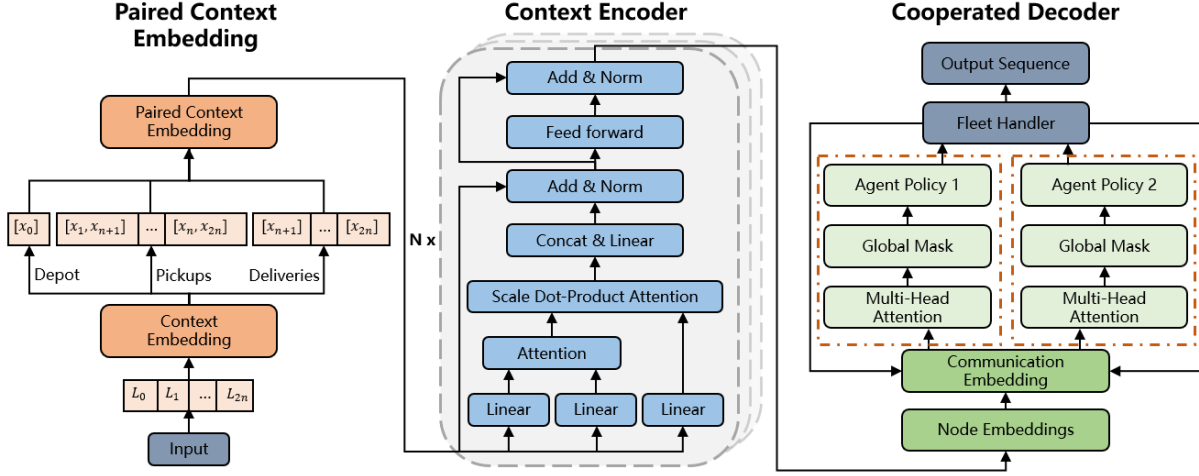


Figure 2: The overall model structure. Different agents share the same paired context embedding and context encoder structure, while learns individual policies using individual decoders.

- **Transition:** The transition between adjacent states is to replace every agent to its target node as its current action. Then we update both the trajectory and the remaining capacity of each agent: $S_k^{t+1} = (S_k^t; \{v_{I_k^t}\})$, $C_k^{t+1} = C_k^t - d_{I_k^t}$, where $;$ means concatenating the partial solution with the new selected node.
- **Reward:** To optimize the overall routing solution quality, all agents share a common objective, which is to minimize the accumulated traveling distance of all agents in the entire episode. In each decision step, the one-step reward $r_k^t = -e_{I_k^t, I_k^{t+1}}$ is the negative of the length of the newly established arc. The final episode reward R can be computed as $R = \sum_{k=1}^{K} \sum_{t=0}^{T-1} r_k^t$, where T is the decision step amount in a complete episode and $I_k^0 = 0$ means that all vehicles start from the depot v_0 .

4.2 Paired Context Embedding and Context Encoding

Due to the structural dependency in-between the nodes, it is critical to measure the inter-node relationship between one and another and provide effective representations for such a. To better capture the structural dependency in-between, we develop a paired context embedding to better characterize the attributes of different nodes.

Given the demand d_i of node v_i and its original 2-D location information \mathcal{L}_i which could be used to calculate node-wise Euclidean distances directly, we formulate the original node embeddings by concatenating the two features and map them into one dense vector as $x_i = W^x[\mathcal{L}_i, d_i] + b^x$. When an agent decides whether to adopt a pickup request and head for its pickup location v_i , it means that the agent also has to visit the corresponding paired delivery v_{i+N} afterwards. Thus, a complete representation on v_i should consider the information of its paired delivery v_{i+N} . Meanwhile, the agent is only allowed to visit v_{i+N} when v_i is already visited. A policy evaluation upon v_{i+N} does not rely on v_i any more. Motivated by this, we update the represen-

tation of pickup nodes by concatenating their paired deliveries, i.e. $x_i = [x_i; x_{i+N}]$. We further compute the linear projection of such augmented representations to generate the final paired context embeddings:

$$h_i^0 = \begin{cases} W_0^x x_i + b_0^x, & i = 0, \\ W_p^x [x_i; x_{i+N}] + b_p^x, & 1 \leq i \leq N, \\ W_d^x x_i + b_d^x, & N + 1 \leq i \leq 2N, \end{cases} \quad (7)$$

Other than intuitive context embeddings upon each node directly, we further generate encoded node embeddings considering the entire graph structure of the given instances. We adopt the Transformer Model (Vaswani et al. 2017) based encoding structure. The initial paired context embedding h_i^0 is processed through L attention layers, each of which consists of a multi-head attention layer (MHA), a skip-connection layer (He et al. 2016), a feed-forward layer, and batch normalization (BN) layers (Ioffe and Szegedy 2015). Formally, each node embedding is updated in the ℓ -th layer as follows,

$$\hat{h}_i = BN^\ell(h_i^{\ell-1} + MHA_i^\ell(h_1^{\ell-1}, h_2^{\ell-1}, \dots, h_{2N}^{\ell-1})), \quad (8)$$

$$h_i^\ell = BN^\ell(\hat{h}_i + FF^\ell(\hat{h}_i)). \quad (9)$$

The core of an attention layer above is the multi-head attention block and can be defined as follows:

$$Q_i^h, K_i^h, V_i^h = W_Q^h h_i, W_K^h h_i, W_V^h h_i, \quad (10)$$

$$A_i^h = softmax(Q_i^h K^{hT} / \sqrt{d_k}) V_j^h, \quad (11)$$

$$MHA_i = Concat(A_i^1, A_i^2, \dots, A_i^H) W_O, \quad (12)$$

where $h = 1, 2, \dots, H$ and $d_k = d_h/H$. H is to amount of attention heads, Q_i^h, K_i^h, V_i^h are the query, key and value vectors respectively. W_O is the projection matrix used to project the final MHA output. The final embedding of each node $h_i = h_i^L$ can thus be obtained from the consecutive L attention layers. Furthermore, we also generate a graph embedding $\bar{h} = \frac{1}{2N} \sum_{i=0}^{2N} h_i$ as the average of all nodes in the

problem instance, which represents the global aggregated information.

It is worthy to note that even though different agents generate their individual policies via different decoders, the paired context embedding and context encoding upon all nodes are shared. This is because, in a fully cooperative PDP scenario, different vehicle agents can share their observations under centralized agent control. Thus the representations of nodes from the environment are consistent and can be learned jointly. In addition, learning a shared context encoder also accelerates the training stage significantly and helps reduce computational costs.

4.3 Cooperative Multi-Agent Decoders

Given both node-wise and global representations, each vehicle agent learns its own policy via its individual decoder network, as depicted in Figure 2. Each agent decoder selects the next node $v_{I_k^t}$ to visit at step t based on the current observations of all agents. The agents work in a cooperative way, and each one of them only visits a part of the overall pickup-delivery pairs within each episode for a complete solution construction.

For better cooperation between different agents, we maintain a communication layer to record the updated states of different agents as follows:

$$Comm^t = [h_{I_1^t}; C_1^t; h_{I_2^t}; C_2^t; \dots; h_{I_K^t}; C_K^t] \quad (13)$$

Leveraging the up-to-date communication embedding, each agent decoder utilizes an MHA-based structure to evaluate the probability of selecting each node at step t . We first generate a special context embedding $h_{k,(c)}^t = [\bar{h}; h_{I_k^t}; C_k^t; Comm^t]$ for agent k to concatenate necessary information used in decision making, including the static global representation, the agent's current state and the states of others. The context embedding $h_{k,(c)}^t$ is further taken as the single query vector and processed via another MHA layer. Finally, the output g_k^t is used to compute the compatibility of choosing a node, and the final probabilities are computed via softmax. Such a decoding process is as follows:

$$g_k^t = MHA_{k,(c)}(h_1, h_2, \dots, h_{2N}), \quad (14)$$

$$Q_k^t, K_{k,i}^t = W_{Q,k}g_k^t, W_{K,k}h_i, \quad (15)$$

$$u_{k,i}^t = D \tanh(Q_k^t K_{k,i}^t / \sqrt{d_k}), \quad (16)$$

$$p_{\theta_k, \phi}(v(k, t)) = \text{softmax}(Mask^t(u_{k,i}^t)), \quad (17)$$

where $W_{Q,k}$ and $W_{K,k}$ are the weight matrices of the last single-head attention, $D=10$ is the clip rate for better exploration (Bello et al. 2016). $Mask^t$ resets all compatibility value of unfeasible (already visited) nodes to $-\infty$. In addition, θ_k is the parameter set for the agent decoder k , and ϕ is used to parameterize the common paired context embedding and context encoder.

During the cooperation of different agents, it is possible that several agents make the same decision to the same node. However, due to the constraint that each node can only be visited once as shown in equation (2) and (3), we design a

special fleet handler to resolve such a conflict. It randomly maintains the action of one agent from all candidates to the node and keeps the others stay at their current location $v_{I_k^t}$. When all delivery and pickup nodes are visited and all agents return to the initial depot v_0 , the episode ends.

4.4 Training via Cooperative A2C

The Advantage Actor-Critic (A2C) (Konda and Tsitsiklis 2000) is a well-known policy gradient approach that has shown its effectiveness. Leveraging the cooperation between different vehicle agents, we design a special cooperative A2C to train the proposed MAPDP.

Besides generating individual policies via individual decoders, we also formulate a centralized critic network $V_\omega^\pi(s)$ to estimate the state-values. In detail, we compute a weighted sum based on the output policy $p_{\theta_k, \phi}(v(k, t))$ at step t and the node-wise embeddings h_i . Then the weighted sum vector of all agents is processed with linear projection into a single vector v_c with $d_c = 128$. Finally, we obtain a single critic value from v_c through two dense layers. The entire critic network is parameterized by ω .

We then compute the loss of both networks and update θ_k, γ, ω as follows:

$$A^\pi(s, a) = r(s, a) + V_\omega^\pi(s') - V_\omega^\pi(s) \quad (18)$$

$$\nabla \mathcal{L}(\theta_k) = \mathbb{E} \nabla_{\theta_k} \log \pi_{\theta_k, \phi} A^\pi(s, a) \quad (19)$$

$$\nabla \mathcal{L}(\phi) = 1/K \sum_{k=1}^{K} \mathbb{E} \nabla_{\theta_k, \phi} \log \pi_{\theta_k, \phi} A(s, a) \quad (20)$$

$$\nabla \mathcal{L}(\omega) = \mathbb{E} \nabla_\omega (A^\pi(s, a))^2 \quad (21)$$

5 Experiments and Evaluation

In this section, we conduct extensive experiments on two datasets to answer the following research questions:

- **RQ1:** How does our proposed MAPDP perform on cooperative PDP compared to other heuristics and RL based methods?
- **RQ2:** How balance is the cooperation between different agents when incorporating the fleet handler?
- **RQ3:** How effective is the multi-agent formulation and the communication embedding?
- **Random Generated Dataset.** We first generate a random dataset with randomly distributed node locations with demands for efficient performance comparison. The location of node v_i , $\mathcal{L} = (x_i, y_i)$, is uniformly sampled from a 5×5 square. Both the x and y coordinates are uniformly distributed in (0,5). The demand volume of a pickup node d_i is uniformly sampled from (1,10), and the capacity limit of each vehicle is 10.
- **Real-World Dataset.** We collect real-world data from an online logistic platform providing services in Guangdong, China, including more than 100 thousand order pairs within a month. Delivery and pickup orders are provided one day in advance so that the platform could provide solutions within a rather static scenario. The capacity

Table 1: Overall performance comparison. The best result in each column is bolded. The improvement row shows the performance gain of our solution compared to the best baseline.

Model	Random Dataset								
	2N = 20, K=2			2N = 50, K=5			2N = 100, K=10		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
ACO (Gambardella, Taillard, and Agazzi 1999)	34.73	39.60%	6min	79.94	52.01%	32min	136.89	53.86%	51min
Tabu Search (Glover 1990)	29.76	19.67%	7min	64.57	22.78%	34min	112.38	26.31%	51min
OR-Tools (Google 2016)	25.91	4.18%	4min	54.64	3.90%	31min	94.25	5.93%	49min
RL-VRP (Nazari et al. 2018)	26.79	7.72%	1s	63.12	20.02%	5s	101.13	13.67%	9s
AM-VRP (Kool, van Hoof, and Welling 2018)	26.64	7.12%	1s	67.41	28.18%	4s	105.91	19.04%	8s
MDAM (Xin et al. 2021)	25.98	4.46%	8s	67.24	27.86%	25s	105.11	18.14%	51s
MAPDP	24.87	0.00%	1s	52.59	0.00%	4s	88.97	0.00%	7s
Model	Real-World Dataset								
	2N = 20, K=2			2N = 50, K=5			2N = 100, K=10		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
ACO (Gambardella, Taillard, and Agazzi 1999)	812	30.13%	6min	1205	35.39%	34min	2054	20.47%	53min
Tabu Search (Glover 1990)	834	33.65%	6min	1197	34.49%	34min	2033	19.24%	51min
OR-Tools (Google 2016)	749	20.03%	4min	1056	18.65%	31min	1811	6.22%	50min
RL-VRP (Nazari et al. 2018)	714	14.42%	1s	1130	26.97%	5s	1842	8.04%	9s
AM-VRP (Kool, van Hoof, and Welling 2018)	661	5.93%	1s	942	5.84%	4s	1759	3.17%	9s
MDAM (Xin et al. 2021)	638	2.24%	8s	941	5.73%	25s	1733	1.64%	52s
MAPDP	624	0.00%	1s	890	0.00%	4s	1705	0.00%	7s

of each vehicle is 6. We collect historical order instances that were accomplished by a fixed fleet.

In the evaluation for both datasets, we construct experiments with three different scales, $2N = 20, 50, 100$, and fix the agent amount with $K = 2, 5, 10$ accordingly. During the training process, we train 100 epochs by processing 10000 training samples within a single one. The networks are trained via Adam optimizer with $L = 3$, $d_k = 128$, $H = 8$ and learning rate $lr = 0.001$. All experiments are conducted using Pytorch 1.7 on 4 2080Ti GPUs.

5.1 Performance Comparison

Baselines We first compare our DRLPR with the three widely recognized heuristics methods:

- **Ant Colony Optimization(ACO)** (Catay 2010; Gambardella, Taillard, and Agazzi 1999) is a famous heuristic to solve the basic VRP and its variants. A number of ant colonies are established to model and optimize the objective functions.
- **Tabu Search** (Glover 1990) as a classic heuristic involves an enormous exploration space and keeps searching local solutions in the neighborhood based on the current one.
- **OR Tools** (Google 2016), Google’s vehicle routing problem solver that utilizes a set of metaheuristics, which is open online¹.

We also compare the most up-to-date RL-based approaches:

- **RL-VRP** (Nazari et al. 2018) proposed an RL-based framework using an encoder-decoder framework. The model utilizes an RNN model and generates the solution in a sequential manner.
- **AM-VRP** (Kool, van Hoof, and Welling 2018) utilize the transformer structure in both encoder and decoder. The agent network is trained via REINFORCE.

¹<https://developers.google.com/optimization/routing/vrp.html>

- **MDAM** (Xin et al. 2021) updates the encoding continuously via an embedding glimpse layer and further generates solutions based on a set of decoders with an additional training loss considering the KL divergence.

Note that all three RL baselines were proposed for the fundamental routing problems, including capacitated VRP, split delivery VRP, etc. For a fair comparison, we manually add an additional mask for pickup-delivery constraints at the decoder of the three methods to guarantee that the output solutions are feasible. As for the cooperation among different vehicles, we set a vehicle assignment order so that the vehicles will take turns to be assigned with the decoded nodes.

Overall Comparison The comparison results of the overall performance are shown in Table 1. We report the total distance, the gap of each method to the best among all, and the time spent during inference in both two datasets with three different customer scales. The best results are bolded.

In terms of the solution quality, we demonstrate that MAPDP outperforms all other baselines in all experiment settings. The closest baseline is MDAM in the Real-World dataset with $2N = 100$ but still suffers from a 1.64% performance gap from MAPDP. The state-of-the-art RL-based methods are less effective in the cooperative PDP settings due to 1) the paired dependency are not explicitly represented, and 2) the cooperation between different vehicles can not be modeled. In fact, manually fixing the assignment order of different vehicles greatly limits the potential of exploring more solutions, and thus influences the effectiveness of reinforcement learning. The cooperative multi-agent decoding process alternatively accepts more possible solution explorations. Thus, MAPDP shows its effectiveness in the practical cooperative PDP scenarios.

Besides the significant performance on solution quality, MAPDP also shows fast computation speed in solution generation. Compared to the heuristic baselines, we notice that MAPDP is able to infer solutions faster with a maximum

of 400+ times with $2N = 100$. The feature of the fast inference speed is of great importance for MAPDP. In real-world application scenarios, the logistic order requests may change continuously and even rapidly, including previous orders canceled and new orders accepted. A fast response-ability is of great significance for an online policy-making system to be adaptive to any new changing dynamics.

5.2 Case Studies on Vehicle Cooperation

To further investigate the cooperation between different vehicles in solving cooperative PDP, we compute the individual traveling distance of each vehicle with $2N = 50$ in both datasets in two case studies, as shown in Figure 3. Besides, since making decisions simultaneously might meet conflicts with other agents, the fleet handler is used to deal with such conflicts and keep the other vehicles halting where they are. We also compute and report the halting ratio = $\text{halting_times}/T$ of each agent k .

We notice that the balance among different agent’s workload is greatly affected by the dataset. In the random dataset where all nodes distribute randomly and are centered within the unit square, different agent decoder ends up with similar workloads. However, in the real-world dataset where nodes distribute unbalanced and even some locate distantly, agents show significant differences. This is because, in the same decision step, some vehicles might be assigned with one of the distant nodes, which greatly increases their total traveling distance. The agent with the longest traveling distance we investigated traveled 212 spatial units, while the one with the shortest trajectory only traveled 103 units. Meanwhile, in both datasets, we demonstrate that the agent that was halted the most ends up with the lowest workload. This is because such an agent skips the decision step while others continue to travel on the instance graph.

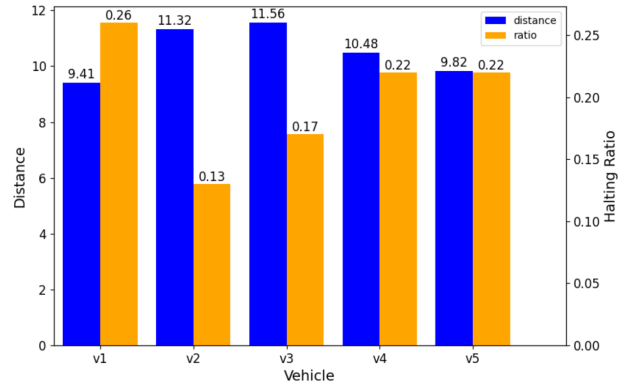
5.3 Ablation Studies on Multi-Agent Design

Table 2: Ablation study on the multi-agent structure design.

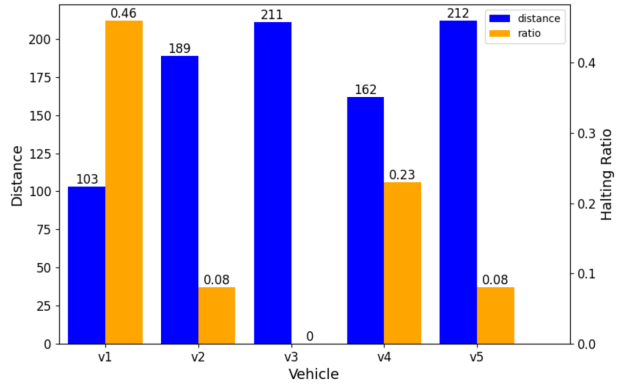
Dataset	Model	2N=20	2N=50	2N=100
Random	MAPDP	24.87	52.59	88.97
	MAPDP-SP	24.99	53.61	89.78
	MAPDP-NC	26.89	68.78	108.12
Real	MAPDP	624	890	1705
	MAPDP-SP	639	943	1721
	MAPDP-NC	731	1033	1896

We further investigate how effective each part of the MAPDP design is as shown in Tabel 2. Two additional variants of MAPDP are trained and evaluated following the same evaluation protocols: MAPDP-SP stands for the simplified model where all agent decoders share the same parameters. Thus all agents become homogeneous to generate individual solutions. MAPDP-NC stands for the multi-agent framework without consideration on the communication embedding. It generates context embeddings only based on the agent’s own state and the global graph embedding.

Results show that both MAPDP variants are outperformed by the original MAPDP. MAPDP-SP is slightly outperformed and is still superior to many other baselines. This



(a) Random Dataset.



(b) Real-World Dataset.

Figure 3: Case studies on vehicle cooperation analysis from two datasets.

shows that the multi-agent modeling along with a comprehensive communication mechanism is the core of performance improvement, while heterogeneous training can further slightly improve its effectiveness based on pure parameter sharing. However, MAPDP-NC is even outperformed by many other baselines. This is because, in a fully cooperative scenario, up-to-date communication with other agents is critical to effective coordination.

6 Conclusion

In this paper, we propose a multi-agent reinforcement learning framework to solve cooperative pickup and delivery problems (MAPDP). We design a special paired context embedding to explicitly represent the structural dependency among different nodes within the instance graph. We develop special cooperative multi-agent decoders to learn the individual policies of different agents. We also design a cooperative A2C algorithm for the integrated model training, where a joint critic value net estimates the state value. Extensive experiments demonstrate that our proposed MAPDP outperforms other baselines with at least 1.64% in all experiment settings and shows significant computation speed during solution inference.

References

- Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural Combinatorial Optimization with Reinforcement Learning. .
- Buşoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183–221.
- Catay, B. 2010. A new saving-based ant algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert systems with applications*, 37(10): 6809–6817.
- Chen, X.; and Tian, Y. 2019. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 6278–6289.
- Gambardella, L. M.; Taillard, É.; and Agazzi, G. 1999. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New ideas in optimization*. Citeseer.
- Glover, F. 1990. Tabu search - Part I. *INFORMS Journal on Computing*, 2: 4–32.
- Google. 2016. OR-Tools. <https://developers.google.com/optimization/routing>.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.
- Kok, J. R.; and Vlassis, N. 2004. Sparse cooperative Q-learning. In *Proceedings of the twenty-first international conference on Machine learning*, 61.
- Konda, V. R.; and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.
- Kool, W.; van Hoof, H.; and Welling, M. 2018. Attention, Learn to Solve Routing Problems! .
- Lee, H.-R.; and Lee, T. 2019. Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 1089–1098.
- Li, J.; Xin, L.; Cao, Z.; Lim, A.; Song, W.; and Zhang, J. 2021a. Heterogeneous Attentions for Solving Pickup and Delivery Problem via Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems*.
- Li, X.; Luo, W.; Yuan, M.; Wang, J.; Lu, J.; Wang, J.; Lü, J.; and Zeng, J. 2021b. Learning to Optimize Industry-Scale Dynamic Pickup and Delivery Problems. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2511–2522. IEEE.
- Lin, K.; Zhao, R.; Xu, Z.; and Zhou, J. 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1774–1783.
- Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.
- Lu, H.; Zhang, X.; and Yang, S. 2019a. A Learning-based Iterative Method for Solving Vehicle Routing Problems. In *International Conference on Learning Representations*.
- Lu, H.; Zhang, X.; and Yang, S. 2019b. A learning-based iterative method for solving vehicle routing problems. In *International Conference on Learning Representations*.
- Madsen, O. B. G.; Fisher, M. L.; and Jornsten, K. O. 1997. *Vehicle routing with time windows: Two optimization algorithms*.
- Nazari, M.; Oroojlooy, A.; Snyder, L. V.; and Takáč, M. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. .
- Ropke, S.; and Cordeau, J.-F. 2009. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3): 267–286.
- Ropke, S.; and Pisinger, D. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4): 455–472.
- Ruland, K.; and Rodin, E. 1997. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & mathematics with applications*, 33(12): 1–13.
- Savelsbergh, M. W.; and Sol, M. 1995. The general pickup and delivery problem. *Transportation science*, 29(1): 17–29.
- Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4): e0172395.
- Toth, P.; and Vigo, D. 2002. Branch-and-bound algorithms for the capacitated VRP. In *The vehicle routing problem*, 29–51. SIAM.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.
- Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of 35th AAAI Conference on Artificial Intelligence*.
- Zhang, K.; He, F.; Zhang, Z.; Lin, X.; and Li, M. 2020. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 121: 102861.
- Zong, Z.; Feng, T.; Xia, T.; Li, Y.; et al. 2021. Deep Reinforcement Learning for Demand Driven Services in Logistics and Transportation Systems: A Survey. *arXiv preprint arXiv:2108.04462*.