Multi-Agent Reinforcement Learning for Taxi Repositioning with Attention Network

Gyuseok Lee[1*], **Hyunkuk Lim**[1*], **Sumin Lim**[1*]
[1]Institute for Software Research
gyuseokl@andrew.cmu.edu,
hyunkukl@andrew.cmu.edu,
suminlim@andrew.cmu.edu

Abstract

Recently, ride-hailing systems (i.e., Uber, Lyft) are in the spotlight, and reinforcement learning has been applied to match taxis and passengers at appropriate times. META (Liu, Chen, and Chen 2021), was proposed to alleviate the imbalance between supply and demand by rearranging taxis on a city grid scale, but it could not guarantee a realistic solution since it only focuses on one-to-one matching. In this paper, we propose an attention-based many-to-many matching solution to connect city grids. Our method performs better than the existing many-to-many approach, network-flow problem solver (NF). In addition, we conduct extensive experiments with multiple conditions, including different grid sizes and the maximum distance of grids to be considered in the matching process. Results show that our cross-attention increased the average reward by 48% than NF. Our work provides evidence that the vehicle repositioning problem can be improved with deep learning layers and recorded high increments in performance compared to existing optimization methods.

Introduction

In ride-hailing services, it is important to deal with the dilemma of ”taxi drivers hunt passengers and passengers search for unoccupied taxis” (Schaller 2017). To solve this problem, META, an existing method, was proposed (Liu, Chen, and Chen 2021). META consists of two components: *taxi demand/supply determination* and *taxi dispatching strategy formulation*. In this paper, each grid (location) is regarded as an agent and attempts to optimize the taxi demand and supply using a one-to-one matching approach with a bipartite graph architecture. This approach demonstrates the resource allocation of ride-hailing platforms, and optimizing this approach indicates the maximization of the platforms’ profit and utility. To achieve this, Liu, Chen, and Chen (2021) pairs grids in need of taxis with another grid with an excess number of taxis based on the Kuhm-Munkres (KM) algorithm, and tries to optimize the repositioning action using the deep deterministic policy gradients (DDPG) algorithm.

However, even if one-to-one matching is simple, intuitive, and effective for implementation and reduction of unnecessary actions, this situation is not realistic as it does not consider the drivers’ perspective: if a driver does not want to relocate to the designated area given by the platform, he or she has free will not to. Rather, a better approach would be to suggest multiple grids based on the need for taxis in nearby areas taking the context into consideration. To solve this problem, we plan to bridge this gap by adopting an attention mechanism to develop a many-to-many dispatching algorithm. To consider the environment’s dynamic nature, we reposition the taxis to different grids based on the attention scores between the valid grids. In other words, taxi repositioning could be performed using the relationship between source and sink grids. This allowed us to achieve better performance than the existing many-to-many method, the network-flow problem solver matching.

Our attention methods outperform network-flow problem solver in terms of average reward, matching rate, and mismatching rate. However, attention methods underperform compared to one-to-one matching. Actor loss with attention methods tends to decrease gradually after one dramatic decrement. Similarly, critic loss with attention methods converges very fast after the peak. This trend appears consistently regardless of the resolution setting. However, the actor loss for the existing methods, Kuhn-Munkres, and network-flow solver tends to increase after a dramatic decrement and tends to fluctuate for critic loss. In conclusion, our attention approach converges faster than the network-flow solver and outperforms the average reward.

As far as we know, this is the first attempt to implement many-to-many dispatching with grid agents using multi-agent reinforcement learning without any restrictions on repositioning distance. We contribute to the ride-hailing service with the MARL research stream by showing that attention algorithm can be used for many-to-many matching problems.

Firstly, we will review previous works regarding our research context. A detailed description of our approach is reported in the next section. Data, experiment settings, and results are illustrated in succession. Lastly, we wrap up our research in the last section.

[*]These authors contributed equally.

Related Work

Order Dispatching and Fleet Management

Previous works can be classified by the types of agents - an agent as a vehicle or a grid. The learning and planning process is usually employed for modeling order dispatching (Xu et al. 2018; Wang et al. 2018; Tang et al. 2019). In the planning step, usually, there are n orders and m available vehicles in the dispatching window. Kuhn-Munkres algorithm is used for a one-to-one matching regarding n orders and m vehicles as a bipartite graph. In the learning step, deep reinforcement learning methods are applied. Repositioning of vehicles in the vehicle-based approach is the same as the determination of action where to go. Overall, regarding vehicles as agents are an intuitive way of modeling. However, this approach suffers from the curse of dimensionality, burdensome computational costs, and risk of single-point of failure (Lynch 2009).

Lin et al. (2018); Li et al. (2019) utilize vehicles as agents. Lin et al. (2018) is based on centralized learning and the decentralized planning scheme based on the assumption that vehicles can decide independently and not be controlled in a centralized way. This work allows local interactions by setting a restriction on moving vehicles in the collaborative context. However, vehicles in the same grid move simultaneously and are regarded as homogeneous. Similarly, Li et al. (2019) allows local interactions by taking mean fields among neighborhoods. The interactions through a mean field lead each action to have little impact on the outcome. Furthermore, agents are also treated as homogeneous.

When treating large-scale vehicles, it is reasonable to consider agents as discretized grids. In this approach, grids are regarded as managers of vehicles so that the grids can dispatch orders and re-position the vehicles to the neighboring grids (Jin et al. 2019; Liu, Chen, and Chen 2021). In this grid-based approach, the repositioning problem is equivalent to the problem of determining how many vehicles should be re-positioned. Jin et al. (2019) employed a manager-worker system for dispatching orders and managing fleets. This approach alleviates many challenges in applying MARL for ride-hailing services research. However, the authors set the maximum distance that vehicles can move to the other grids as two. While Liu, Chen, and Chen (2021) didn't explicitly set the limitation for maximum distance, a one-to-one matching algorithm is used for fleet management. While this work mentions reported the many-to-many algorithm using the network-flow problem solver, they state that learning the optimal policy for the agents was difficult due to the dynamic environment of the many-to-many dispatching assumption. We plan to improve upon this approach.

Matching Algorithm

In multi-agent reinforcement learning literature, the KM algorithm is regarded as the primary matching algorithm in various contexts: in autonomous mobility-on-demand (Chen, Li, and Yao 2022), intelligent offloading system for scheduling (Weng, Chu, and Shi 2021), resource allocation in unmanned aerial vehicles (Zhou et al. 2022), order dispatching and fleet management in ride-hailing services (Zhou et al. 2019; Xu et al. 2018; Wang et al. 2018).

Usually, a matching algorithm is used when matching the order and the vehicles or matching the areas which lack vehicles and have abundant vehicles. The KM algorithm requires a task to be assigned to exactly one agent. This condition implies KM algorithm is appropriate for solving a one-to-one matching problem. However, from the ride-hailing services' perspective, this one-to-one matching may not be efficient and effective (Chen, Li, and Yao 2022).

There has been little research on solving this matching problem as minimum-cost flow network. The vehicle dispatching problem can be represented as the network flow problem with minimum cost in the bipartite graph. Hanna et al. (2016) compared greedy matching, centralized greedy matching, KM algorithm, and their proposed algorithm, scalable collision-avoiding role assignment with minimal-makespan (SCRAM). In this work, a bipartite graph of vehicles and passengers is constructed to find the assignment minimizing the longest distance. SCRAM recorded the minimum number of passengers waiting time among algorithms. Xu et al. (2021) constructed a bipartite graph with ride-hailing service zones. With the constructed graph, they solve multiple minimum cost flow problem to determine the number of vehicles to be dispatched. These two approaches have fundamental problems. Hanna et al. (2016)'s approach is not appropriate for our work because of the different problem settings. Xu et al. (2021) assumes that the demand and supply are inherently stable, which is a very strong assumption to be applied.

In this work, we aim to extend this one-to-one matching to many-to-many matching with deep learning methods.

Preliminaries

Most of the definitions and settings of agents and environments are followed from Liu, Chen, and Chen (2021)'s work.

Agent We set an agent as a hexagonal grid of New York City. There is a total of N grids and we will denote a focal grid, i.e., an agent as $i \in [1, \ldots, N]$.

State The state $\mathrm{s}_i^t = Num_d^i - Num_s^i$, where Num_d^i and Num_s^i refer to the number of taxi demands and supplies in an agent i at time t, respectively. This state implies our reinforcement learning aims to resolve supply-demand imbalance.

Action. Agent's action is the number of taxis that needs to be dispatched to nearby grids. There are $\left|a_t^i \times Num_s^i\right|$ taxis that need to be moved, where $-1 < a_t^i < 1$. Based on threshold ξ, we can classify each grid into three categories. If $a_t^i > \xi$, the agent i is regarded as **Source** grid, where it can reposition taxis to other agents. On the contrary, If $a_t^i < -\xi$, the agent i is considered a **Sink** grid, where it expects to receive taxis from other agents.

Strategy. The strategy is a set of repositioning pairs consisting of a source grid and a sink grid.

Reward The reward r_t^i is the revenue that an agent i obtains at the time t like below:

$$r_t^i = 1 - \frac{|Num_s^i - Num_d^i|}{\max\{Num_s^i, Num_d^i\}} \tag{1}$$

An agent aims to maximize the accumulated reward with the discount factor of γ (i.e., $\gamma = 0.9$), denoted as $\Sigma_{k=0}^{\infty}\gamma^k r_{t+k}^i$.

Approach

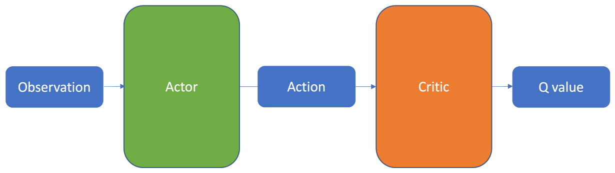


Figure 1: Overall structure of our model

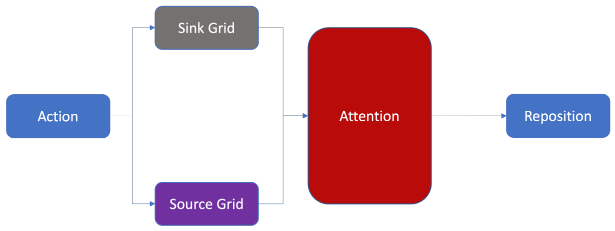


Figure 2: Taxi reposition based on Attention network

MARL

Figure 1 shows the overall structure of our model. As you can see, our model is based on the Deep Deterministic Policy Gradient (DDPG) algorithm, which has the structure of actor and critic. Specifically, the actor network outputs an action from observation, and the critic network expects Q-value from the action.

At this time, the grids are classified whether sink or source grids. Using this knowledge, we can calculate attention scores between the sink and source grids. Note that the attention score was calculated using embedding obtained through the hidden layer of the actor network. After that, we reposition the taxis by asking the source grid to the number of taxis based on the attention score multiplied by the demand of the sink grid. Through this, we can implement an attention-based many-to-many method for taxi repositioning.

The simulation design follows that of META under the MADDPG algorithm: a grid agent consists of an actor and a critic, where the actor takes action, and the action is evaluated by its own critic based on the overall change in the difference between demand and supply. However, instead of a one-to-one matching algorithm, we implement attention scores between source grids and sink grids to determine how many vehicles a sink grid receives from all source grids in a ratio to its excess demand. This approach relieves the restriction that a sink grid can only receive vehicles from a single source grid.

Attention Score

We use two different attention scores to calculate the pairwise similarity between the source and sink grid embeddings with respect to their difference in demand and supply of vehicles and reposition vehicles from source to sink grids based on the output ratio:

$$Score = \begin{cases} e_{source}^T e_{sink} & \text{Dot product} \\ Attention(e_{source}, e_{sink}) & \text{Cross attention} \end{cases} \tag{2}$$

$$Ratio = \sigma(Score) \tag{3}$$

e_{source} refers to an embedding of a source grid and e_{sink} refers to an embedding of a sink grid. The first in Equation 2 is content-based dot-product attention (Luong, Pham, and Manning 2015) between source grid embeddings and sink grid embeddings, while the second is a transformer-based cross attention (Vaswani et al. 2017) between the two. The cross attention is given by:

$$Attention = (W_q \cdot e_{source})^T (W_k \cdot e_{sink}) \tag{4}$$

However, because W_q and W_k are learnable parameters that need training, we implement a loss function based on the overall improvement in the demand-supply difference in the sink grids according to Equation 5:

$$Loss = \sum_{i=1}^{n} \left| \frac{\mathcal{D}_i - \mathcal{S}_i - (\mathcal{D}_i \cdot Score_i)}{max(\mathcal{D}_i, \mathcal{S}_i)} \right| \tag{5}$$

where n represents the number of sink grids. While the loss function for the cross-attention network can be connected to the critic loss to be updated based on each grid agent's actions at time t, classifying grids into source or sink grids does not depend on the action or its reward. Rather, because the action depends on the classification results, the attention network must be updated using an external factor that is not dependent on the actor-critic network. Thus, we perform backpropagation on the weights based on the change in the average reward.

Experiments & Results

Data

We conduct our experiments on New York City Taxi dataset from June 1st to June 15th in 2016. The data contains information on passenger pickups and drop-offs, including the time of the request, the time of pickup, and the request location in latitude and longitude.

The time space is divided into intervals of 10 minutes, giving 144 time steps in a day and $1,596$ steps in total for the dataset. The grid agents are divided in such a way that the Manhattan area is divided into equal-sized grids excluding areas vehicles cannot reach, such as bodies of water. Experiments are conducted on two different grid sizes: one with a side length of 0.01 degrees in latitude and longitude and the other with a side length of 0.005 degrees. The number of valid grids is 84 and 269, respectively.

We experiment with three settings for the maximum distance a grid can reposition vehicles to another grid: 2, 5, and 10. Under the maximum distance of 2, a grid can only send vehicles within a distance of 2 and cannot send any to grids outside the range. The grids outside the range are masked before the scores go through the softmax function to calculate the ratio.

Experiment Setup

We set the learning rates for the actor and critic networks as 0.001. The learning rate for the attention mechanism is set to 0.001. We used Adam optimizer for training the attention layer.

We conduct several experiments for comparing four methods of matching: Kuhn-Munkres (KM), network-flow problem solver, dot-product attention (DP), and cross-attention (CA). We used Google's OR-tools for solving network flow problems.

In summary, we have 24 combination of experiments in total: resolution (low and high), maximum moving distance (2, 5, 10), and four algorithms (KM, NF, DP, and CA).

Metrics

The metrics used to evaluate performance are as follows: the average reward of all agents given by Equation 1, match rate where requests from customers are properly matched with vehicles after repositioning, and mismatch rate which measures requests that have expired after 30 minutes due to long waiting time. The match rate and mismatch rate are given by

$$MatchRate_{i,t} = \frac{MatchedOrder_{i,t}}{TotalOrder_{i,t}} \tag{6}$$

$$MismatchRate_{i,t} = \frac{UnmatchedOrder_{i,t}}{TotalOrder_{i,t}} \tag{7}$$

where $TotalOrder_{i,t}$ is the sum of all orders for agent i at time t, given by $MatchedOrder_{i,t} + UnmatchedOrder_{i,t}$.

Under the assumption that idle vehicles must accept any unmatched requests, the reward reflects whether all vehicle requests are matched with idle vehicles as long as there are enough vehicles to serve the requests. Each grid agent is expected to have learned to take proper actions to increase the average reward, thereby repositioning the excess number of vehicles to source grids to sink grids with insufficient vehicles to serve its demand, increasing the match rate and decreasing the mismatch rate.

Results

Training Loss

The training actor loss and critic loss for each resolution are illustrated in Figure 3 and Figure 4, respectively. KM algorithm showed the lowest actor loss and the highest critic loss during training. DP and CA show similar patterns in actor loss and critic loss, and there was no significant difference between them. Both networks converged gradually for DP and CA, which is also consistently shown in the high-resolution setting.

One interesting pattern we notice from the actor loss is that for both KM and NF, the actor loss increases again after 500 steps in low and high-resolution settings. In contrast, actor loss in the attention-based dispatching converges between 500 and 1000 steps and does not change much after.

For all approaches, since the number of available vehicles changes due to repositioning and the random appearance of working drivers, like a driver who has just started to accept requests, there is bound to be some error in the actor loss. However, since the match and mismatch rates are converging, we assume this issue comes from misclassifying grids into a source or a sink grid when they should be classified as the opposite or neutral. The attention mechanism depends on the embedding with state information instead of the direct classification results, making it robust to the classification. Furthermore, because dispatch is on a many-to-many basis, there is less room for error than dispatching all excess vehicles to another grid after it has been classified incorrectly.

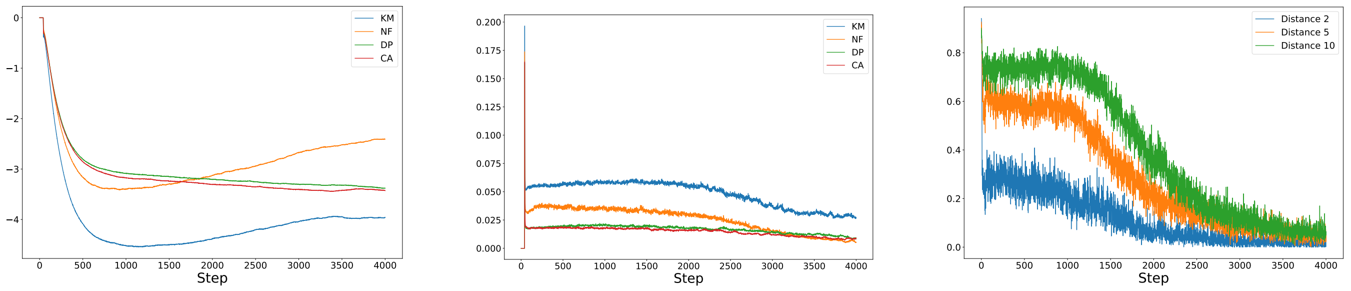
Reward

	Distance 2	Distance 5	Distance 10
Average Reward			
KM	$\mathbf{7.72}$	$\mathbf{9.50}$	$\mathbf{10.45}$
NF	4.78	6.68	7.80
(Ours) DP	6.80	8.16	9.19
(Ours) CA	7.07	8.28	9.31
Match rate			
KM	0.63	$\mathbf{0.72}$	$\mathbf{0.73}$
NF	0.54	0.54	0.54
(Ours) DP	$\mathbf{0.65}$	0.68	0.68
(Ours) CA	$\mathbf{0.65}$	0.68	0.68
Mismatch rate			
KM	0.37	$\mathbf{0.28}$	$\mathbf{0.27}$
NF	0.46	0.46	0.46
(Ours) DP	0.35	0.32	0.32
(Ours) CA	$\mathbf{0.34}$	0.31	0.32

Table 1: Results - Low Resolution

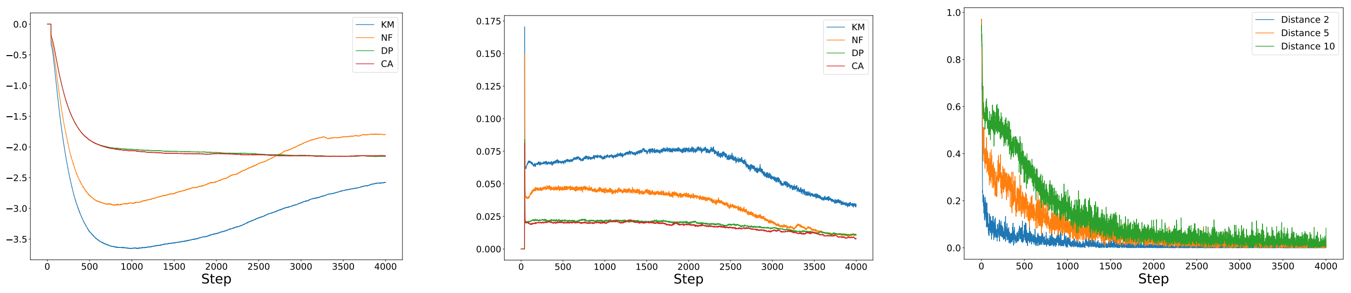
The low and high-resolution performance metrics are reported in Table 1 and Table 2, respectively.

Compared to NF, our attention networks show increased performance by 48% (CA) and 42% (DP) in average reward with distance 2. The match rate also increased using attention networks by 20% (DP and CA), and the mismatch rate decreased by 26%, implying better performance for ride dispatching on a many-to-many basis. However, the attention score approach achieves a higher match and lower mismatch rate only with a maximum distance of 2 under a low-resolution setting compared to the KM algorithm. The performance with the KM algorithm is superior in all other cases. We suspect this is because, for KM algorithms, two matched pairs of grids must exchange vehicles, while for



(a) Actor Loss (b) Critic Loss (c) Attention Loss

Figure 3: Loss - Low Resolution



(a) Actor Loss (b) Critic Loss (c) Attention Loss

Figure 4: Loss - High Resolution

	Distance 2	Distance 5	Distance 10
Average Reward			
KM	**5.36**	**7.26**	**8.02**
NF	3.85	5.25	5.67
(Ours) DP	4.87	5.29	5.82
(Ours) CA	4.81	5.53	6.38
Match rate			
KM	**0.38**	**0.40**	**0.40**
NF	0.33	0.33	0.34
(Ours) DP	0.37	0.37	0.37
(Ours) CA	0.37	0.38	0.38
Mismatch rate			
KM	**0.62**	**0.60**	**0.60**
NF	0.67	0.67	0.66
(Ours) DP	0.63	0.63	0.63
(Ours) CA	0.63	0.62	0.62

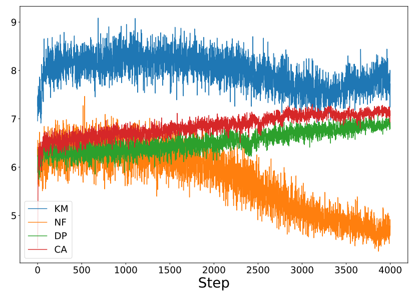
Table 2: Results - High Resolution

the attention score approaches, the order in which sink grid requests first affects other grids.

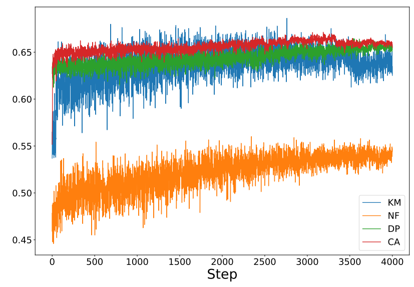
While the match rate and the mismatch rate for the attention score approaches are similar to the KM algorithm in all cases (maximum difference < 0.01), there are relatively big differences between DP and CA: 0.27 in Distance 2, 0.12 in Distance 5, 0.12 in Distance 10 with low resolution, 0.06 in Distance 2, 0.24 in Distance 5, 0.56 in Distance 10 with high resolution. Under the high-resolution setting, the difference between DP and CA in average reward is much bigger. As the maximum distance increases, the state information becomes more sparse, making the information difficult to be expressed in terms of a representative vector. However, because the cross-attention approach utilizes learned weights to further bipolarize the representations for source and sink embeddings with respect to the many-to-many dispatching results, the attention score ratios are more accurate in the number of cars to be dispatched, giving a higher average ratio.

Conclusion

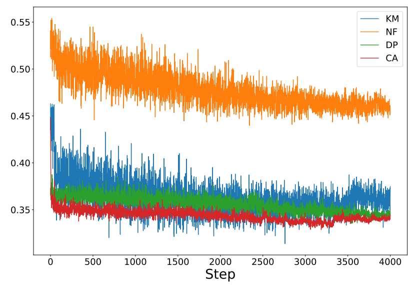
In this work, we conduct extensive experiments for many-to-many matching algorithms. Instead of using the network-flow problem solver, we employ an attention mechanism to match grids' demand and supply. Results show that our attention mechanisms outperform the NF method in terms of average reward and match rate. With cross attention method, the average reward is increased by 26.5% and the match rate is increased by 12.1%, compared to the NF method. However, our approaches underperform compared to the one-to-one KM algorithm. As analyzed in Liu, Chen, and Chen (2021), each grid's action will affect the whole agent's action decision, which makes the environment more dynamic. For example, a small sink grid with excess demand of 8 may request a source grid with an excess supply of 10 to send 7 vehicles according to the attention score. However, because there is consideration of order in the requests from the



(a) Average Reward

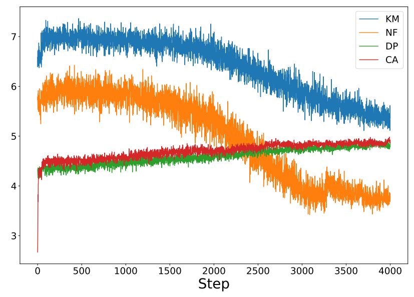


(b) Match Rate

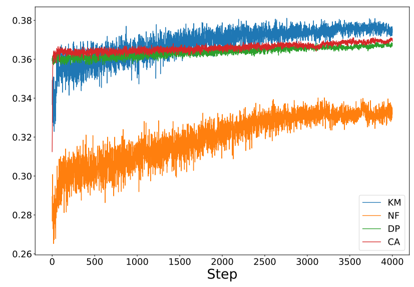


(c) Mismatch Rate

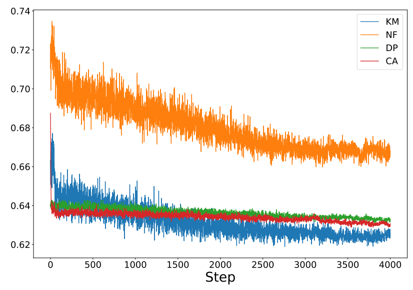
Figure 5: Performance - Low Resolution



(a) Average Reward



(b) Match Rate



(c) Mismatch Rate

Figure 6: Performance - High Resolution

sink grids, if a larger grid with 80 excess demand requests 6 vehicles from the source grid beforehand, the smaller grid can only receive the remainder, which is suboptimal. The actor and critic losses indicate that our agents have at least converged. However, there is still room for improvement as different algorithms could be applied to our many-to-many dispatching algorithm, such as ranking algorithms to the dispatching process.

Moreover, the input of the actor-critic network is constructed with one-hot encodings of state information, especially the grids, which leads to the curse of dimensionality. Furthermore, differentiating the source and sink grids is crucial, given the state information, as this provides the direction for repositioning vehicles from one grid to another. While utilizing attention scores from embeddings instead of a direct binary classification shows more stable results, it would not provide optimal results if the network does not encode the information well. For further research, compressing the input vectors as dense, representative vectors can improve the performance of the many-to-many matching approach.

In future work, we would like to make a fine-grained embedding for observation, which is used as input to the actor network. Furthermore, more research on reward shaping seems to be necessary. In addition, instead of the current approach that distinguishes sink and source grid by the action from the output of the actor network, using a separate network to predict whether sink or source would be another good study.

References

Chen, H.; Li, Z.; and Yao, Y. 2022. Multi-agent reinforcement learning for fleet management: a survey. In *2nd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2022)*, volume 12348, 611–624. SPIE.

Hanna, J. P.; Albert, M.; Chen, D.; and Stone, P. 2016. Minimum cost matching for autonomous carsharing. *IFAC-PapersOnLine*, 49(15): 254–259.

Jin, J.; Zhou, M.; Zhang, W.; Li, M.; Guo, Z.; Qin, Z.; Jiao, Y.; Tang, X.; Wang, C.; Wang, J.; et al. 2019. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1983–1992.

Li, M.; Qin, Z.; Jiao, Y.; Yang, Y.; Wang, J.; Wang, C.; Wu, G.; and Ye, J. 2019. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The world wide web conference*, 983–994.

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.

Liu, C.; Chen, C.-X.; and Chen, C. 2021. Meta: A city-wide taxi repositioning framework based on multi-agent reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Lynch, G. S. 2009. *Single point of failure: The 10 essential laws of supply chain risk management*. John Wiley and Sons.

Schaller, B. 2017. Empty seats, full streets: Fixing Manhattan's traffic problem. Technical report, Schaller Consulting.

Tang, X.; Qin, Z.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Zhu, H.; and Ye, J. 2019. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1780–1790.

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

Wang, Z.; Qin, Z.; Tang, X.; Ye, J.; and Zhu, H. 2018. Deep reinforcement learning with knowledge transfer for online rides order dispatching. In *2018 IEEE International Conference on Data Mining (ICDM)*, 617–626. IEEE.

Weng, Y.; Chu, H.; and Shi, Z. 2021. An intelligent offloading system based on multiagent reinforcement learning. *Security and Communication Networks*, 2021.

Xu, Y.; Wang, W.; Xiong, G.; Liu, X.; Wu, W.; and Liu, K. 2021. Network-Flow-Based Efficient Vehicle Dispatch for City-Scale Ride-Hailing Systems. *IEEE Transactions on Intelligent Transportation Systems*.

Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; and Ye, J. 2018. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 905–913.

Zhou, M.; Jin, J.; Zhang, W.; Qin, Z.; Jiao, Y.; Wang, C.; Wu, G.; Yu, Y.; and Ye, J. 2019. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2645–2653.

Zhou, S.; Cheng, Y.; Lei, X.; Peng, Q.; Wang, J.; and Li, S. 2022. Resource Allocation in UAV-assisted Networks: A Clustering-Aided Reinforcement Learning Approach. *IEEE Transactions on Vehicular Technology*.