MARL-based Optimal Route Control in Multi-AGV Warehouses

Ho-Bin Choi
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
chb3350@koreatech.ac.kr

Ju-Bong Kim
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
rlawnqhd@koreatech.ac.kr

Chang-Hoon Ji
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
koir5660@koreatech.ac.kr

Ullah Ihsan
Advanced Technology Research Center
Korea University of Technology and Education
Cheonan, South Korea
ihsan@koreatech.ac.kr

Youn-Hee Han
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
yhhan@koreatech.ac.kr

Se-Won Oh
Dept. of Knowledge-converged Super Brain Convergence Research
ETRI
Daejeon, South Korea
sewonoh@etri.re.kr

Kwi-Hoon Kim
Dept. of Artificial Intelligence Convergence Education
Korea National University of Education
Cheongju, South Korea
kimkh@knue.ac.kr

Cheol-Sig Pyo
Dept. of Knowledge-converged Super Brain Convergence Research
ETRI
Daejeon, South Korea
cspyo@etri.re.kr

Abstract—Automated guided vehicles (AGVs) are an essential component for automation fulfillment centers, a kind of warehouse. Efficient control of the AGV leads to easier management of inventory in the fulfillment center. To increase the productivity of various warehouses including fulfillment centers, we propose a Multi-Agent Reinforcement Learning (MARL)-based algorithm for cooperative control of AGVs. The proposed algorithm is based on a popular cooperative MARL algorithm, and utilizes an additional technique for path control of AGVs to distinguish the sacrifices of each agent and compensate them accordingly. We evaluate the proposed algorithm in comparison with a basic MARL algorithm on two fulfillment center layouts and provide further insight via the visualization of the results.

Keywords—AGV, Warehouse, Optimal Route, Deep Learning, Reinforcement Learning, MARL

I. INTRODUCTION

The growth of e-commerce has caused many changes in the logistics market, and many companies have introduced fulfillment services to meet customer needs. In the online distribution industry, fulfillment service is a series of processes of picking, packing, and shipping products from warehouses to the customers according to their orders. These processes not only satisfy the needs of customers quickly, but also have many advantages for companies, such as delivery agency, inventory management, security, and fire insurance. Within the fulfillment center, systematic management is essential because numerous inventories must be moved in real-time.

The tasks performed in the fulfillment center such as picking, packing, and shipping can only be done by humans. However, simple transport of inventory is not dependent on humans, as automated guided vehicles (AGVs) can easily move heavy inventory. Therefore, they are essential components for automation fulfillment centers, and their coordinated control leads to efficient management of inventory and increases warehouse productivity.

Meanwhile, most real-world problems including the multi-AGV warehouses occur when many entities cooperate or compete with each other rather than a single entity [1]. Multi-agent reinforcement learning (MARL) has achieved good results in various fields as in [2-4] and has three major frameworks: (1) fully centralized learning, which is a general framework utilized in single-agent reinforcement learning, but it has a fatal problem that the action space increases exponentially as the number of agents increases, (2) conversely, fully decentralized learning, which does not have the disadvantage of increasing the action space, but the non-stationarity problem is further exacerbated by the lack of communication between agents, and (3) centralized training with decentralized execution (CTDE) [5], which combines these two frameworks well, eliminates those two drawbacks and is suitable for decentralized systems.

In this paper, we adopt the CTDE framework-based MARL to increase productivity by systematically controlling the path of numerous individual AGVs moving autonomously within the AGV warehouse. To this end, we formulate the multi-agent environment modeled by the AGV warehouse as a decentralized partially-observable markov decision process (Dec-POMDP) [6]. In addition, we present state and observation representation, action representation, and reward function along with a description of the modules constituting the system and an overall scenario. The algorithm proposed in this paper has the ability to recognize the contributions or sacrifices of individual agents. Experiments are presented with results for three metrics and we provide further insight via visualization of the results.

II. Related Work

A. Automated Guided Vehicles in Warehouses

The Amazon Robotics, formerly Kiva Systems, deals with resource allocation problems including decision making under uncertainty, robot path planning, and scheduling in the AGV warehouse. Fortunately, they provide a natural multi-agent AGV warehouse scenario for coordinated autonomy and decentralized decision making in [7]. The insights in [7] have encouraged research across various fields, among them we focus on deep reinforcement learning. The path planning problem of multi-AGV can be interpreted in various ways and a lot of algorithms have been proposed as in [8-10]. Recently, studies to control the path of multi-AGV in real time by applying reinforcement learning are being attempted, achieving good results that outperform traditional algorithms [11-12]. To the best of our knowledge, there are no papers that have studied path control or path planning of multi-AGV using multi-agent reinforcement learning.

B. Multi-Agent Deep Reinforcement Learning

In general, it is natural to use Q learning with DQN to single agent RL [13-14]. This Q learning can be simply extended to Independent Q-learning (IQL) to be applied to multi-agent RL [15]. However, a better approach is needed for Q-learning to achieve good performance in multi-agent RL with stronger non-stationarity. The value deposition networks (VDN) estimate the joint action value of the multi-agent through Q^{tot} calculated by adding Q^a of all agents [16]. This additivity constraint is relaxed by QMIX as a monotonicity constraint [17]. QMIX estimates Q^{tot} in a complex non-linear method using MLP rather than in a simple summation method. In addition, QMIX can represent a richer joint action value because it allows extra state information to be used in the mixing network. The mixing network estimating this joint action value Q^{tot} guarantees the monotonicity constraints via non-negative weights. Since the factorization of VDN and QMIX facilitates the decentralized execution of multi-agents, they are suitable as marl algorithms to deal with real-world problems.

III. System Model

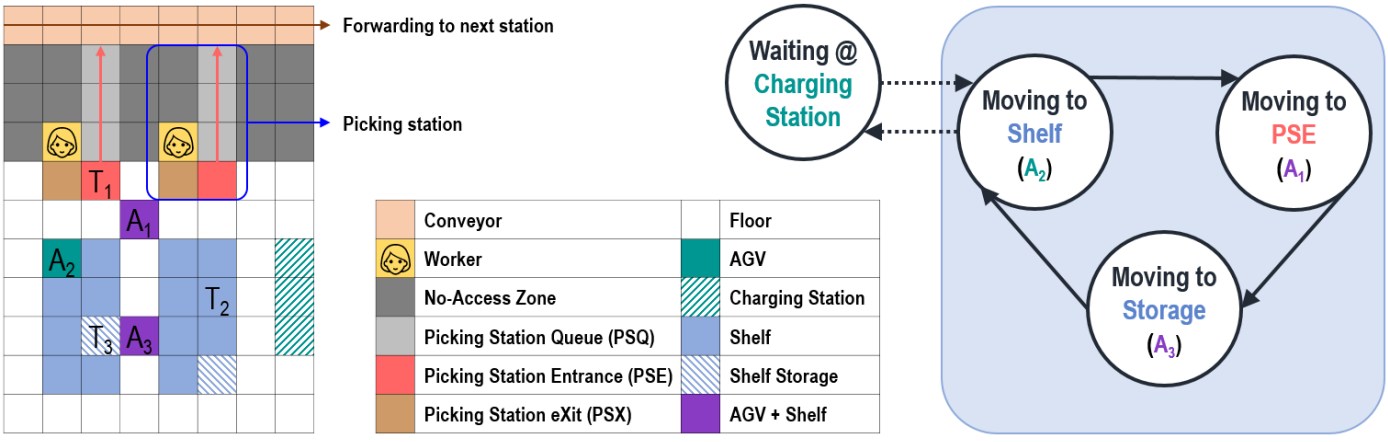
We consider a multi-agent environment to solve a task that requires collaboration between agents. This can be formulated as a Decentralized Partially-Observable Markov Decision Process (Dec-POMDP) [6], represented by a tuple $\langle A, S, O, Z, U, P, R_g, R_l, \gamma \rangle$. At each time step t, the environment outputs a global state $s_t \in S$, and observations $o_t^a = \mathrm{O}(s_t, a) \in \mathrm{Z}$, where S denotes the global state space, Z denotes the observation space for each agent $a \in \{1, \dots, n\} \equiv \mathrm{A}$, and $O: S \times A \rightarrow Z$ denotes the observation function respectively. Each agent uses its own observation to select an action $u_t^a \in \mathrm{U}$, where U denotes the action space for each agent $a \in \{1, \dots, n\}$. The environment performs the joint action $\boldsymbol{u}_t \in \boldsymbol{U} \equiv U^n$ of all agents and results in a transition according to the state transition function $P(s_{t+1}|s_t, \boldsymbol{u}_t): S \times \boldsymbol{U} \times S \rightarrow [0, 1]$. This transition is completed with a global reward $\boldsymbol{r}_t = R_g(s_t, \boldsymbol{u}_t)$ and individual rewards $r_t^a = R_l(s_t, u_t^a)$, where $R_g: \mathrm{S} \times \mathbf{U} \rightarrow \mathbb{R}$ denotes the global reward function and $R_l: \mathrm{S} \times \mathrm{U} \rightarrow \mathbb{R}$ denotes the individual reward function respectively.

In this process, each agent will try to increase its own individual reward which is aggregated into the team reward, ultimately aiming to maximize the discounted cumulative global reward $\sum_{i=0}^{\infty} \gamma^i \boldsymbol{r}_{t+i}$. Each agent generates its own observation-action history $\tau^a \in T \equiv (Z \times U)^*$ with a stochastic policy $\pi^a(u^a|\tau^a): T \times U \rightarrow [0, 1]$ as a condition, where $(Z \times U)^*$ represents the set of all possible observation-action histories. The joint policy π consisting of each agent's policy π^a has a joint action-value function, which is formulated as:

$$Q^{\pi}(s_t, \boldsymbol{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \boldsymbol{u}_{t+1:\infty}}[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{r}_{t+i} | s_t, \boldsymbol{u}_t] \qquad (1)$$

A. Problem Statement

In this section, we present the RL environment that models the AGV warehouse required by the fulfillment center. Fig. 1(a) is a snapshot of an layout example, and each cell denotes a module represented by a specific color as shown in Fig. 1(b).



(a) A snapshot of an example (b) Color-based module representation (c) AGV States transition diagram

Fig. 1. Description of the RL environment in which the AGV warehouse is modeled, where the target position of agent A_i is T_i.

The overall scenario is carried out as: (1) When the system starts operating, all AGVs wait for assignment at the charging station. (2) Each picking station's worker allocates the required shelf to a random AGV. (3) The assigned AGV moves to the requested shelf and lifts it. (4) The AGV moves to the PSE of the picking station. (5) When the work at the picking station is finished, the AGV moves to a random shelf storage. (6) The AGV lays it down and repeats this from (2). In summary, each AGV can be in one of the 4 states: Waiting, Moving to Shelf, Moving to PSE, and Moving to Storage. Fig. 1(c) shows the circulation of AGV states in this process as a state transition diagram. Note that AGV, shelf, and shelf storage are not randomly selected in the actual AGV warehouse, so the environment we present is a more difficult problem to solve because they are randomly selected.

B. State and Observation Representation

The observation of each agent consists of two-dimensional information normalized as a two-channel image for the surrounding 9×9 area centered on itself. The shape of the observation is $2 \times 9 \times 9$, and the values of each channel represent the information of the corresponding position in the layout. The first channel denotes whether the agent can move to the corresponding location, and the second channel denotes the remaining Manhattan distance from the corresponding location to the target location. Meanwhile, the shape of the state is $1 \times 20 \times 20$, which consists of a normalized cumulative number of visits to each module in the layout within the episode.

C. Action Representation

At the beginning of the episode, each agent's looking direction is randomly selected, thereafter it is determined by the action it performs. The action is performed to move one cell based on this looking direction, and the action space is defined as: {Stop, Moving Forward, Moving Right, Moving Left, Moving Back}. For this, the observation is rotated with respect to the looking direction of the agent.

The environment informs agents of available actions so that collisions can be avoided. By choosing these actions, it is guaranteed that they won't hit a wall or collide between agents. In other words, experiences related to the collision are not generated through Invalid Action masking, only high-quality experiences are accumulated in the replay buffer.

D. Reward Function

The environment returns individual rewards for each agent. After the agents perform their chosen action, each agent's individual reward is positive if the Manhattan distance to their target position gets closer or the agent arrives at their target position, otherwise it is negative. It may be considered harsh to receive a negative reward even when the agent takes a stop action, but we expect the Q value to be updated to prevent such situations. The absolute value of the individual reward is $1/The\ number\ of\ agents$, and the global reward is the sum of all individual rewards. Accordingly, the range of the global reward that can be output in one time step from the environment is -1.0 to +1.0.

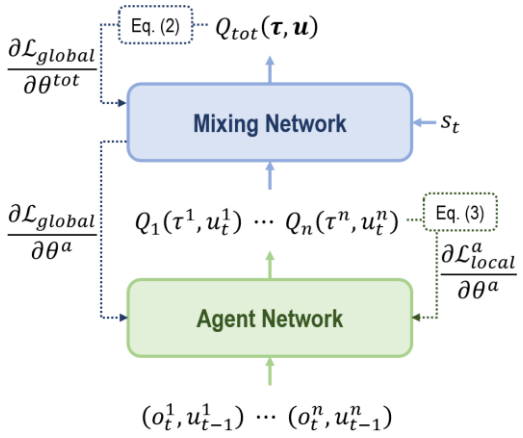


Fig. 2. The overall model architecture of the proposed algorithm.

IV. Proposed Algorithm

We propose an algorithm that applies a new technique to QMIX [17] while maintaining the centralized training with decentralized execution framework. QMIX uses the Mean Square Error between Q^{tot} and its target generated by inputting the Q value of each agent into the mixing network as loss:

$$L_{global} = \sum_{i=1}^{b} \left[\left(y_i^{tot} - Q_i^{tot}(\boldsymbol{\tau}, \boldsymbol{u}, s;\ \theta^{tot}) \right)^2 \right], \qquad (2)$$

where $y^{tot} = r + \gamma \max_{\boldsymbol{u}'} Q^{tot}(\boldsymbol{\tau}', \boldsymbol{u}', s';\ \bar{\theta}^{tot})$. For this, each agent extracts u^a and $Q^a(\tau^a, u^a)$ according to the epsilon greedy method. QMIX restricts the relationship between Q^{tot} and Q^a to the monotonicity constraint. To enforce this monotonicity constraint, the weights of the mixing network are constrained to be non-negative. After the loss of the mixing network is calculated as in (2), It is backpropagated to a feed-forward network consisting of Mixing network and Agent network. In this process, the gradient is backpropagated to the agent network, but the effect can be considered insignificant. Although monotonicity is guaranteed by the constraint of the relationship between Q^{tot} and Q^a, it is harsh to allow only the mixing network to judge the contribution of individual agents to sacrifice. Thus, we propose an additional local loss for the agent network formulated as:

$$L^a_{local} = \sum_{i=1}^{b} \left[\left(y_i^a - Q_i^a(\tau^a, u^a;\ \theta^a) \right)^2 \right], \qquad (3)$$

where $y^a = r^a + \gamma \max_{u'_a} Q^a(\tau'_a, u'_a;\ \bar{\theta}^a)$.

The global loss $\mathcal{L}_{global}$ is calculated using Q^{tot} of the mixing network, while the local loss $\mathcal{L}^a_{local}$ is calculated using Q^a of the agent network for each agent. In this study, we adopted that the agent network is shared but it is not essential. Note that each agent's observation-action history τ^a is not shared and local loss $\mathcal{L}^a_{local}$ is calculated separately. Its gradient is only backpropagated to the agent network regardless of the mixing network. Backpropagation through global loss as in (2) also affects the agent network, but backpropagation through local

loss as in (3) can clarify the feedback on the actions of the individual agents. Hence, the final loss equation is defined as:

$$\mathcal{L} = \mathcal{L}_{global} + \sum_{a=1}^{n} \mathcal{L}_{local}^{a} \quad (4)$$

Fig. 2 briefly shows the overall network structure along with the direction in which the gradient of the proposed loss in (4) is backpropagated. The agent network is composed of GRU [18] and MLP, whereas the mixing network consists only of MLP, but its weights and biases are obtained from the output of hyper-networks to ensure the monotonicity constraints. The state and observation composed of two-dimensional information are first converted into one-dimensional features through CNN and then inputted to the MLP.

Each agent interacts with the environment and stores experiences in the replay buffer, performing distributed execution using only its own observations. The mixing network is used to update the agent network by estimating the Q values of all agents as total Q values. In other words, the mixing network does not directly interact with the environment, but only requires the state and Q values of each agent sampled from the replay buffer. Note that the mixing network, the main idea of centralized training, is not used in testing after training is complete.

Even though $\mathcal{L}_{global}$ adopted in QMIX [17] is used for update, the influence of the newly added $\mathcal{L}_{local}$ is so strong that it can be considered that the influence of $\mathcal{L}_{global}$ is relatively weak. So, it is a reasonable doubt that there will be no significant difference from IQL which uses only $\mathcal{L}_{local}$ as the final loss. In Section 5 we conducted an experiment comparing the algorithm proposed in this paper with IQL and also verify its validity by analyzing the results.

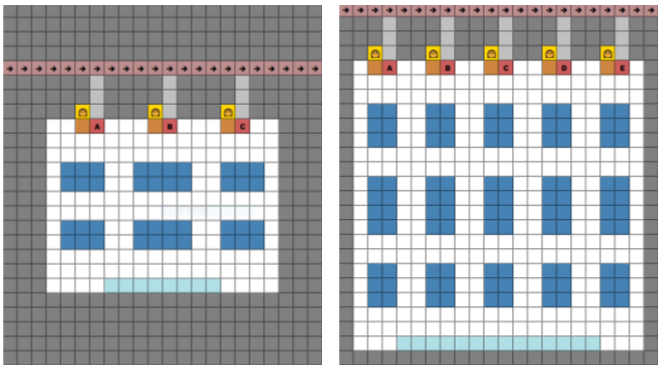
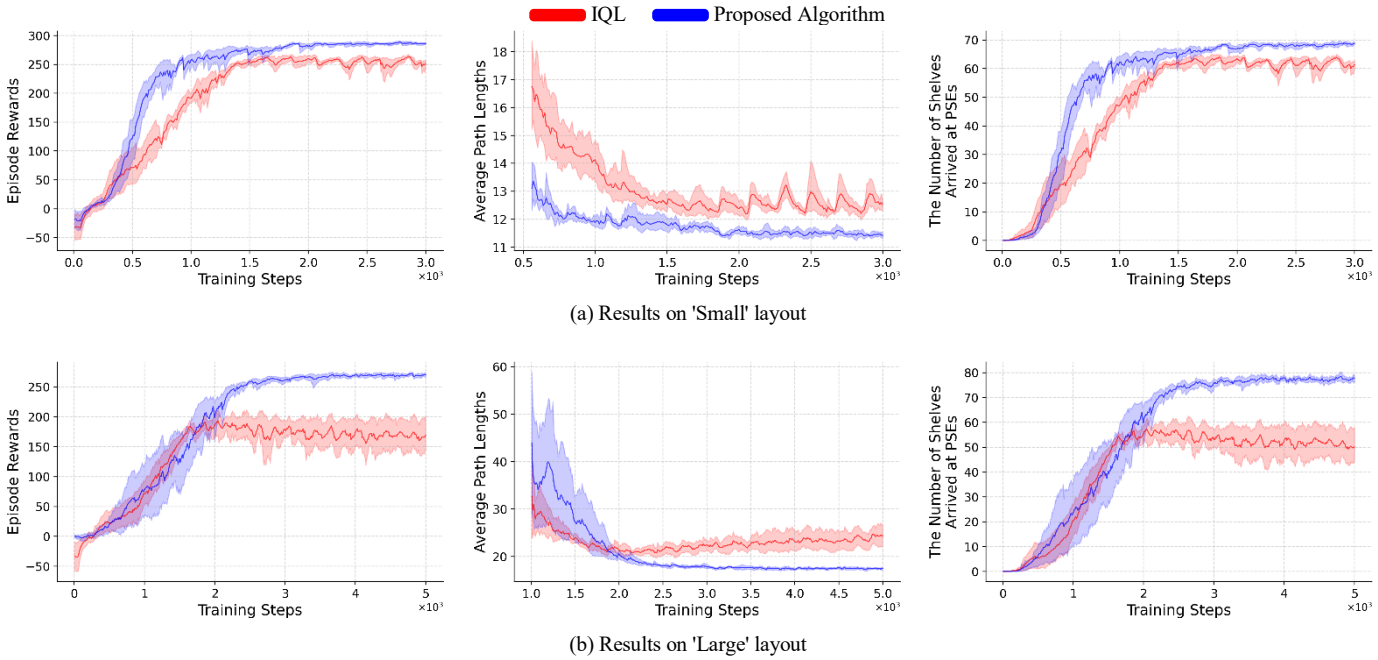


Fig. 3. Layout configuration: 'Small' layout (left) and 'Larget' layout (right).

V. EXPERIMENTS

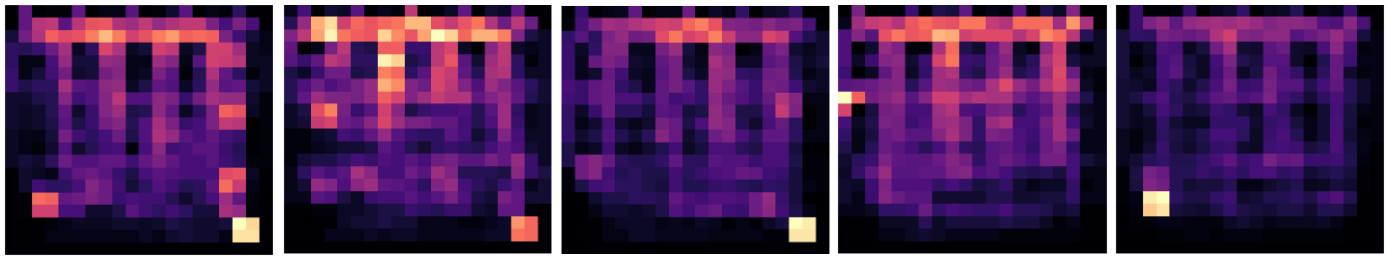
In this section, the performance of the algorithm proposed in this paper is presented in comparison with IQL, which is considered the most basic marl algorithm. The experiments are performed in 'Small' layout and 'Large' layout as shown in Fig. 3. The 'Small' layout consists of 8 AGVs and 3 picking stations in 12x16 grid size, while the 'Large' layout consists of 14 AGVs and 5 picking stations in 20x20 grid size. For detailed evaluation we adopted three metrics: episode rewards, average path lengths, and number of shelves arrived at PSEs. During training, for every 10 training steps each metric is measured as the average value of 5 episodes in which all agents took a greedy action. We plot the average of 5 runs with 95% confidence intervals for every metric on both layouts. Note that 1 training step means a single model update by the final loss that aggregates the global loss and the local losses as in (4), and it is calculated using the experience sampled from the replay buffer.



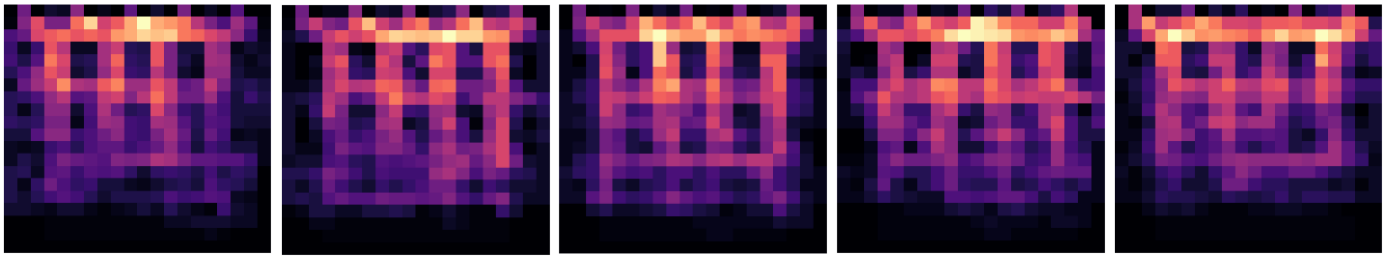
(a) Results on 'Small' layout

(b) Results on 'Large' layout

Fig. 4. Performance comparison for three metrics. Note that the front part of the graphs about average path lengths is omitted for the readability of the graph.



(a) IQL



(b) Proposed Algorithm

Fig. 5. Heat maps on 'Large' layout.

A. *Performance Comparison*

Fig. 4 shows the performance of the algorithm proposed in this paper compared with IQL as the change of three metrics according to the training steps, and Fig. 4(a) and 4(b) are the results on 'Small' layout and 'Large' layout, respectively. As shown in the graphs about the two metrics, episode rewards and number of shelves arrived at PSEs, we can know that our algorithms learn faster and more consistently. More specifically, our algorithms achieve unshakable convergence at better values, while IQL tends to be unstable, shaking even at worse values. This contrast is more evident in 'Large' layout than in 'Small' layout. Since IQL has no information exchange between agents, the performance of IQL decreases sharply as the number of agents increases. On the other hand, even if the number of agents increases, our algorithm proves their robust performance through efficient information exchange and cooperation between agents. These results can be easily confirmed from the graphs about the metric, average path lengths. For our algorithm, it becomes shorter and more stable as training progresses, but for IQL it does not converge and exhibits a very unstable appearance, and even in 'Large' layout it becomes longer. As a result, we can infer that the more complex the structure of the layout (i.e., the more AGVs), the more important is the collaboration between agents through communication.

B. *Heat Maps*

In order to analyze the actual behavior patterns of agents, we visualize the cumulative number of visits to the movement of AGVs as shown in Fig. 5. Episodes during 500 time steps were drawn five times each for our algorithm and IQL on the 'Large' layout. The relatively larger the cumulative number of visits, the brighter the corresponding position. From this analysis, we found a very interesting fact about IQL that certain AGVs are in an idle state at arbitrarily inappropriate positions. Therefore, the heat map of IQL has a very bright specific position, not the main path of the entire system, and can be interpreted that the absence of communication between agents develops into a fatal problem. In contrast, our algorithm has been confirmed that, as we expected, all AGVs mainly move between picking stations and shelves, the main paths of the entire system. When comparing these results, we emphasize that all AGVs learned with our algorithms move primarily within a limited range, which is the main path of the entire system, but do not cause congestion. Accordingly, state circulation of all AGVs occurs smoothly and high performance follows.

VI. CONCLUSION

In this paper, we proposed a CTDE-based MARL algorithm that can efficiently control the routes of AGVs which are essential components to increase the productivity of fulfillment centers. To evaluate the proposed algorithm, we presented state and observation representation, action representation, and reward function along with a description of the modules constituting the system and an overall scenario. The evaluation results are that the proposed algorithm outperforms IQL for three metrics on two different 'Small' and 'Large' layouts. We also provide insight into the importance of communication between agents via the visualization of the results.

ACKNOWLEDGMENT

This work was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIP) (No. CRC-15-05-ETRI), and also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2020R1I1A3065610).

REFERENCES

[1] L. Buşoniu, R. Babuška, and B. Schutter, Multi-agent reinforcement learning: An overview, Innovations in multi-agent systems and applications-1, pp. 183–221, 2010.

[2] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," IEEE Transactions on Wireless Communications, vol. 19, no. 2, pp. 729–743, 2019.

[3] X. Li, J. Zhang, J. Bian, Y. Tong, and T. Liu, "A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network," In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019.

[4] X. Li, X. Hu, W. Li, and H. Hu, "A multi-agent reinforcement learning routing protocol for underwater optical sensor networks," In Proceedings of IEEE International Conference on Communications, 2019.

[5] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," Journal of Artificial Intelligence Research, vol. 32, pp.289–353, 2008.

[6] F. A. Oliehoek and C. Amato, A concise introduction to decentralized POMDPs, SpringerBriefs in Intelligent Systems, Springer, 2016.

[7] J.J. Enright and P.R. Wurman, "Optimization and coordinated autonomy in mobile fulfillment systems," In Proceedings of the AAAI workshop on automated action planning for autonomous mobile robots, pp. 33–38, 2011.

[8] J. Bae and W. Chung, "A heuristic for a heterogeneous automated guided vehicle routing problem," International Journal of Precision Engineering and Manufacturing, vol. 18, no. 6, pp. 795–801, 2017.

[9] Z. Han, D. Wang, F. Liu, and Z. Zhao, "Multi-AGV path planning with double-path constraints by using an improved genetic algorithm," PloS one, vol. 12, no. 7, 2017.

[10] Y. Lian and W. Xie, "Improved A* multi-AGV path planning algorithm based on grid-shaped network," In 2019 Chinese Control Conference, 2019.

[11] R. Kamoshida and Y. Kazama, "Acquisition of automated guided vehicle route planning policy using deep reinforcement learning," IEEE International Conference on Advanced Logistics and Transport (ICALT), 2017.

[12] Y. Yang, J. Li, and L. Peng, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," CAAI Transactions on Intelligence Technology, vol. 5, no. 3, pp. 177–183, 2020.

[13] C.J.C.H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, pp. 279–292, 1992.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[15] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," In Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337, 1993.

[16] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," In Proceedings of 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 2018.

[17] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 2018.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," In NIPS 2014 Workshop on Deep Learning, 2014.