Deep Reinforcement Learning-based Task Assignment and Path Planning for Multi-agent Construction Robots

Xinghui Xu [1,*], Borja García de Soto [2,*]

***Abstract*—Recent developments in deep learning have enabled reinforcement learning (RL) methods to drive optimal policies for a sophisticated high-dimensional environment, which is suitable to overcome the challenges of implementing on-site construction robots, such as the dynamic nature of the construction environment and inherent complexity to solve the multiple decision-makers interacting simultaneously. In this research, we are trying to propose a systematic framework to adopt deep reinforcement learning (DRL) algorithms into on-site construction robotic applications (e.g., bricklaying platforms). This research has two main objectives: 1) Implement a multi-agent path-planning (MAPP) method for on-site robots that allow multiple mobile robots to navigate through the environment toward the assigned goal position and conduct the desired task logic while avoiding collisions, and 2) integrate the multi-agent task allocation (MATA) framework to solve complex tasks (e.g., laying bricks or delivering materials) through the cooperation of individual agents by assigning different tasks and roles to individual robots, which allows multiple robots to work simultaneously, just as how human workers act on a job site to make the best advantages of the productivity gains.**

I. Introduction

Construction tasks involve interaction among multiple agents, where emergent behavior and complexity arise from agents co-evolving together. Construction workers are assigned different roles with a given set of tasks to accomplish. During this process, different teams work in the same environment. They either collaborate in reasonable sequences or compete for resources (e.g., space) to get the job done. When we implement multi-robot systems to finish a task, considerations must be made to path planning [1], resource allocation [2], sequence arrangement for operations [3], and scheduling among robots [4]. The working schemes of collaborative robots are essential to be investigated, which allow them to work in a group just as humans to boost productivity. However, most robot systems in the construction domain are developed with a single-agent setting [5]. Previous research has not investigated controlling robots simultaneously to finish a construction task with the same goal. This has been achieved in other areas for a long time, such as game simulation [6], collaborative manufacturing [7], network communication [3], warehouse delivery [8], etc. There is a lack of knowledge of the broad adoption of robotic components on construction sites.

As an essential branch of machine learning, Reinforcement Learning (RL) can realize sequential decision-making under uncertainties through end-to-end education and has made a series of significant breakthroughs in robot applications. With the development of deep learning, studies on Deep Reinforcement Learning (DRL) algorithms with robots have attracted researchers' interest because of their ability to handle high-dimensional data [9] with stochastic dynamic reward functions [10]. It has led to a wide range of impressive progress in various domains, such as industrial manufacturing [7], robot control [11], autonomous driving [12], multi-agent task allocation (MATA) [13], and multi-agent path-planning (MAPP) [14] problems.

However, construction research has not investigated approaches combining MATA and MAPP problems using DRL. Previous research either looks into the stationary MATA problem to find the optimal policy for the job scheduling problem [15], or the navigation pathfinding methodology [16] using DRL algorithms, which is insufficient to adopt the DRL combined robots in the construction domain. This study proposes a systematic framework incorporating MATA and MAPP for a multi-agent construction robotic system. We are adapting the multi-agent proximal policy optimization (MAPPO) method for the MATA and MAPP to solve a construction bricklaying problem by giving different roles to the robots, such as bricklayer, material deliveryman, and inspector (e.g., to monitor progress and quality). Robots will follow a logic based on their roles, find the optimal schedule to start the navigation and do the pathfinding from initial points to execution areas. After reaching the goals, a predefined action will be conducted with a determined schedule. The final goal is to lay the bricks to specific points with the shortest timespan and shortest distances navigated during construction.

II. Related Work

A. *Multi-agent Task Allocation (MATA)*

Gradually, much research has started to investigate MATA. However, in the construction domain, it still needs to be explored. The MATA problem has been systematically defined in other areas, such as industrial manufacturing. Liu et al. [17] solved a planning problem with a multi-agent reinforcement learning (MARL) algorithm based on an options framework to handle a cooperative multi-robot system in an aircraft painting application. Agrawal et al. [18] suggested the RL method for multi-robot task allocation in warehouse environments. Recently a new framework by Lee et al. [19] proposed a digital twin-driven DRL for adaptive task allocation in robotic construction and tested it to assemble prefabricated concrete bricks using stationary robotic arms in a simulated environment. However, to make the robotic system more suitable for construction tasks and more capable of other roles on the construction site, the mobility of the construction robots must also be implemented.

B. *Multi-agent Path-Planning (MAPP)*

The MAPP robotic problem has caused a lot of interest in different areas to handle dynamic environments. Shang et al. [20] developed a collaborative path planning of a carrier-based aircraft to improve the scheduling efficiency on the aircraft carrier deck using MARL. Hu et al. [21] proposed an Automated Guided Vehicle (AGV) conflict prevention path planning method to enhance the container terminal's transportation cost and operational efficiency. Long et al. [22] proposed an efficient multi-agent navigation in dynamic environments, which is of great industrial value when

[*]S.M.A.R.T. Construction Research Group, Division of Engineering, New York University Abu Dhabi (NYUAD), Experimental Research Building, Saadiyat Island, P.O. Box 129188, Abu Dhabi, United Arab Emirates
[1]PhD Student, xx927@nyu.edu
[2]Assitant Professor, garcia.de.soto@nyu.edu

deploying a large-scale fleet of robots to real-world applications. We could adopt these algorithms for the same use on the construction site to allow robots to navigate to the assigned position while avoiding collisions.

C. MATA combined with MAPP

As indicated in the previous section, using a DRL-based combination of MATA with MAPP is rare in the construction literature. However, it has been investigated in other domains. For example, various algorithms are proposed to solve the task allocation and path planning problem in warehouse delivery. Although these are used in controlled environments compared to construction sites, these algorithms could be transferable with modifications. For instance, Chen et al.[8] proposed a marginal-cost assignment heuristic improvement strategy based on Large Neighborhood Search, which allows the task assignment and path planning to be performed simultaneously in a warehouse package delivery system. Liu et al. [23] addressed the combination problem with a sequential two-stage problem: (1) task assignment followed by (2) planning. The cost of the path planning task may be far higher than the task assignment solver anticipated. But they didn't consider the uncertainty influence from the path planning to the task allocation part. Elfakharany and Ismail [24] introduced a method of decentralized sensor-level policy that performs multi-robot task allocation and navigation from end to end. However, it mostly considered the path planning part, which is not suitable for the dynamics task logic of a construction robot.

III. Methodology

A. Problem Identification

We combine MATA and MAPP in a simulated environment to solve a construction bricklaying problem. The main elements of the methodology, related steps, and objectives are summarized in Table 1.

Different roles are given to the robots, such as bricklayer, material deliveryman, and inspector. Thus, three robots are deployed in a collaborative setting with different locations to navigate and then do the execution. They will follow the basic task logic, such as finding the optimal schedule to start the navigation, considering the uncertainty of the path planning, and doing the pathfinding from initial points to the goal position. Robots are trained to understand their tasks and then navigate to the corresponding goal. After reaching the areas, a predefined action will be conducted with a determined schedule and motion trajectory.

In this study, we are using DRL algorithms. The problem

Table 1. Research methodology, steps, and objectives

Steps		Objectives
1.	Simulator set up	Address dynamic changes of the site.
2.	Communication network	Allow different robot settings trained in the same framework.
3.	DRL & MARL algorithm	Investigate RL algorithms for construction tasks.
4.	Integrated MATA and MAPP simulation	Allow mobility of multi-agent robots. Allow task allocation and scheduling of multi-agent robots

is formulated as a partially observable Markov decision process (POMDP). Each robot does a sequence of observations and actions, forming a trajectory from its start to the goal position chosen by the policy. The POMDP is episodic, and each episode ends either with one of the robots having a collision, all the robots successfully reaching the goals, or the maximum episode duration being exhausted. The goal is to finish the path-finding process and arrive at the waypoint on an optimal schedule to start the construction activities with the shortest time span and distances navigated.

B. Simulator and communication network setup

Training on such algorithms is often required. Testing the algorithm directly on a real robot is unrealistic. Thus, a simulator is needed to generate robot navigation and manipulator motions with quantitative evaluations and records. Middleware is required to communicate between the simulator and the environment that facilitates implementing a complex distributed application involving interacting components and logic. The most popular middleware used in robotics is the Robot Operating System (ROS). We use the ROS-embedded simulator software (Gazebo) to simulate the environment. Besides, we utilized a framework where a developer combines numerous ROS node sub-processes into an application package. In this study, ROS nodes with RL training algorithms were developed and linked with other ROS nodes to control the robots in the system architecture. These nodes and their interactions will help the robot feed forward the information from its sensors and wait for the feedback from the training scripts to understand the optimal policies in each step.

C. MAPPO algorithm

We use the multi-agent proximal policy optimization (MAPPO) method for the MATA, and MAPP adapts it to complicated task allocation and path planning problems (Figure 1). While various forms of the policy gradient (PG) method exist, proximal policy optimization (PPO) has demonstrated comparable or better performance than recent PG approaches, and they are simpler to implement [25]. PPO enables multiple updates per minibatch sample to promote sample efficiency and guarantees policy optimization's stability by limiting the policy's update amplitude [26]. It ensures that the updated policy is not too different from the old one to ensure low training variance suitable for construction robot settings.

```
1.  Initialize policy network $\pi_\theta$ old policy network $\pi_{\theta old}$ and value network $V_\varphi$
2.  for iteration = 1,2,...do
3.      # Data collection
4.      $T_{total} \leftarrow 0$
5.      for episode = 1,2,... do
6.          # Robots are running in parallel
7.          for robot = 1,2,... N do
8.              Run policy $\pi_\theta$ for $T_i$ time steps, collecting $(O_i^t, a_i^t, r_i^t)$, where $t \in [0, T_i]$
9.              Calculate $A_i^t = \sum_{t=0}^{T_i} (\gamma\lambda)^t \left( r_i^t + \gamma V_\varphi \left( O_i^{t+1} \right) - V_\varphi \left( O_i^t \right) \right)$
10.             Returns $R_i^t = A_i^t + V_\varphi(O_i^t)$
11.             $T_{total} \leftarrow T_{total} + T_i$
12.             Break if $T_i \geq T_{max}$ or a collision happens or each robot reaches a goal
13.         end for
14.         Break if $T_{total} \geq T^{th}$N, and send the data rollout to train the centralized networks
15.     end for
```

Figure 1. MAPPO Training Algorithm.

We adopted the centralized learning, decentralized execution paradigm [27] in which each robot has a copy of the policy π_θ and the value V_φ networks. The policy and the value networks have different weights θ and φ. The algorithm (Figure 1) summarizes the data collection and training process. At each time step, each robot receives its observation O_i^t and uses its copy of the policy π_θ to generate the action a_i^t. In the following step, the observations measure the distance between robots and goals, calculate the total and idle time in each action, and make 2D laser scanner measurements of the obstacles to rule out the collisions. The action space is just a 1D vector representing the linear velocity of the robot heading towards the designed goals in each timestamp $[v_i^t]$. Then the reward function r_i^t is calculated. Each robot collects its data $(O_i^t,\ a_i^t,\ r_i^t)$ from the environment. Once the amount of data exceeds a certain threshold T^{th}, the rollouts of data are sent to a centralized copy of the policy and the value networks. Then, the gradients of the objective function L with respect to the centralized policy network weights θ and value network weights φ are computed. Then, the Adam optimizer updates the weights θ and φ using the learning rate l_r for E epochs. After each update, each robot receives a copy of the update weights θ and φ to start collecting a new batch of data. Thus, the policy and value networks are trained on the experiences collected by all the robots simultaneously.

D. *Reward function*

Instead of precisely designing algorithms to manually calculate and build up an optimal solver, we focused on the reward function. Rather than defining each step for the robot to find the optimal solution, we defined the final goals and actions the robot could take in each step by using the DRL algorithms to train the robot to act towards the desired goals. In this specific problem of bricklaying robots, the reward function incentivizes each robot to understand the correct working logic and schedule, choose and move towards a unique goal position, and decrease the total idle time and the total distances with the shortest timespan and working distances. It penalizes getting near obstacles and colliding with them, multiple robots reaching the same goal position and consuming extended amounts of time to reach the goal. These are summarized as follows:

$$(r_s^{logic})_i^t = \begin{cases} a, \textit{When the robot team understand the assigned schedule network} \\ -b, \textit{In case the robot receives the assigned task but the order is incorrect} \\ -c, \textit{in case the robot could not find the corresponding task} \end{cases}$$

$(r_s^{logic})_i^t$ is used to measure if the assignment of the task follows the task logic of the construction work.

$$(r_s^{idlet})_i^t = -dmax((\frac{t_{idle}}{t_{total}} - e),0)$$

$(r_s^{idlet})_i^t$ is used to calculate the percentage of the idle time, and give a certain penalty for delaying the total time.

$$r_i^t = \begin{cases} (r^h)_i^t + (r^o)_i^t + (r^{ob})_i^t \\ f,\ \textit{In case of reaching a unique goal} \\ -g,\ \textit{In case of reaching a goal occupied by another robot} \\ -h,\ \textit{In case of a collision} \end{cases}$$

r_i^t is the general reward function of path planning, where $(r^h)_i^t$ is a reward that increases in value when the robot is heading toward the goal, $(r^o)_i^t$ is the reward that penalizes the robot in case it is moving toward a goal position that another robot is moving towards, $(r^{ob})_i^t$ is the reward that penalizes the robot for getting near an obstacle.

$$(r_s^{dis})_i^t = -i\max((\frac{dis_i^t}{dis_{shortest}} - j),0)$$

$(r_s^{dis})_i^t$ is a reward function to measure how good the distance is compared to the shortest path between the initial points to the assigned goals.

$$R_i^t = (r_s^{logic})_i^t + (r_s^{idlet})_i^t + r_i^t + (r_s^{dis})_i^t)$$

R_i^t is used to measure the overall performance of the actions for robot i at time step t with 4 subparts' reward on logic, time, and distances. In order to distinguish the effects of these subparts to clarify the goals (i.e., reward or penalty behavior), we define the values (i.e., weights) for $a, b, c, d, e, f, h, i, j$ to represent the reward/penalty received. For example, if we are concerned only with the least time to get the optimized schedule, we could give higher weights to the task logic and idle time and lower weights to long-distance traveling. This could help to train the algorithms with different settings under multiple circumstances.

IV. Result and Conclusion

A. *Results*

To test the proposed framework, we built a simulator in the ROS Gazebo (Figure 2) that allows robots with different roles to collaborate within the same environment. Three areas (execution area, material area, and monitor area) were defined, corresponding to different roles of activities performed by different robots (TutleBot 1-*r1*, TutleBot 2-*r2*, TutleBot 3-*r3*) in the simulator. Currently, the experiment is implemented as a proof of concept to verify the feasibility of combining path planning with the task allocation problem. It is hard for the robot to navigate randomly inside this model and find the optimal path without causing any collisions. Most of the trails will exit without even starting the task allocation part. Thus, we have simplified the problem to decrease the uncertainties of the path planning part to ensure the robot can arrive at the area in a specific time. The rigid body of the obstacles is removed in the current experiment and $(r^{ob})_i^t$ is ruled out to ensure the algorithm can run entirely in each trial. Besides, to simplify the navigation part, the robot will first change its orientation to the assigned goals from its starting point in each trial. Thus, this simple setting does not consider the penalty function of path planning r_i^t and the shortest path reward function $(r_s^{dis})_i^t$. The only matter that causes the uncertainties from the path-planning process is caused by the random velocities assigned to the robot in each step. After several iterations, the robots will arrive at the goal areas without any problem. In future experiments, $(r^{ob})_i^t$ will be considered by using laser scanner data to address the static and dynamic obstacles.

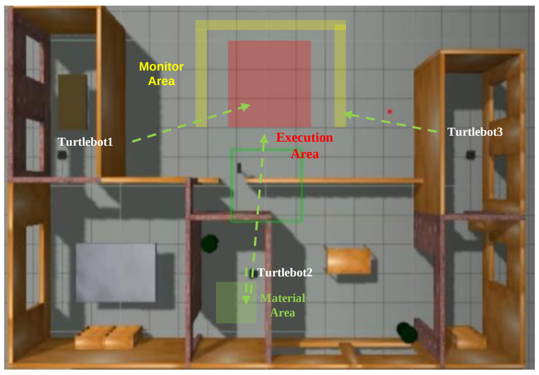

Figure 2. Simulator for combing MATA and MAPP in ROS Gazebo

Figure 3 shows two measurements in this experiment, $(r_s^{logic})_i^t$ and $(r_s^{idlet})_i^t$ to evaluate the training result of the task schedule assignment and path planning. For the simplicity of showing the subparts' performance, we are using a fixed seed to report the result. The numerical numbers come from the reward functions with different weights. The higher the cumulative reward (y-axis), the better the performance of the robots. Overall, the cumulative reward is stabilized after several episodes of training.

Robot 3 (*r3*) performs well on the total idle time and task logic; the optimal policy is quickly gained, resulting in the reward converging fast, as shown in Figure 3. This is due to the independence of *r3*'s task of monitoring the other two robots. In the first 2,000 episodes, Robot 1 (*r1*) and Robot 2 (*r2*) have some penalties on $(r_s^{logic})_i^t$ attributed to the interaction of the two robots because of a misunderstanding of the orders causing the tasks to be performed following the wrong schedule. However, in the end, after around 5,000 episodes, they find a way to generate the optimal policy to finish the task to gain the maximum reward. This means that, as expected, the goals are understandable and reachable after training. The falling of the yellow line $(r2 - logic)$ is due to the collision between the material delivery (assigned to $r2$) with the assigned execution point generated for $r1$. This causes a penalty on the result of the task assignments.

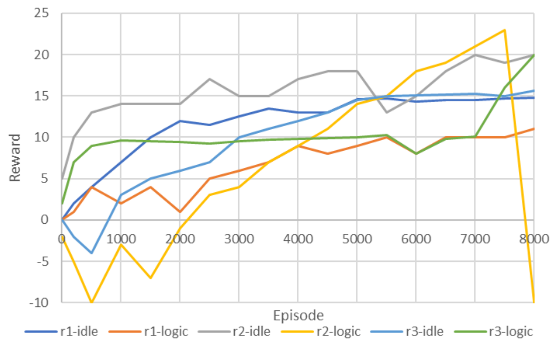

Figure 3. Training result of MAPPO on 3 robots

B. *Conclusion and outlook*

In this study, we propose a multi-agent proximal policy optimization (MAPPO) method that combines MATA and MAPP problems for a collaborative construction robotic application. The training results show the feasibility of the proposed framework in a simple setting where the path planning functionalities are simplified. To solve problems in the construction domain, such as the bricklaying problem, still, this work has many limitations. For example, path planning with obstacles and moving objects should be considered. In this way, more than the current driving of robots from point to point is needed to address this challenge. The robot's action should be modified, and it will take longer to find the optimal path. Second, the construction tasks should be modeled in the simulator with the pick-and-place function embedded. Third, more uncertainties need to be considered. Currently, the goal position is randomly generated inside the execution area, which is insufficient to handle this issue to mimic the actual construction. Besides, the roles of robots are now fixed; the switching roles of robots in different episodes should also be considered in a future implementation. Future work includes enriching the elements inside the simulator and the algorithm, and tests will be conducted in a simulated environment alongside real-world verification.

References

[1] V. I. Gorodetskii, "Self-organization and multiagent systems: I. Models of multiagent self-organization," *Journal of Computer and Systems Sciences International*, vol. 51, no. 2, pp. 256–281, 2012, doi: 10.1134/S106423071201008X.

[2] J. Edmondson and D. Schmidt, "Multi-agent distributed adaptive resource allocation (MADARA)," *International Journal of Communication Networks and Distributed Systems*, vol. 5, no. 3, pp. 229–245, 2010, doi: 10.1504/IJCNDS.2010.034946.

[3] H. A. Al-Rawi, M. A. Ng, and K.-L. A. Yau, "Application of Reinforcement Learning to Routing in Distributed Wireless Networks: A Review," *Artif. Intell. Rev.*, vol. 43, no. 3, pp. 381–416, Mar. 2015, doi: 10.1007/s10462-012-9383-6.

[4] J. Parker, "Task allocation for multi-agent systems in dynamic environments," in *AAMAS*, 2013.

[5] T. Bock and T. Linner, "Single-Task Construction Robots by Category," in *Construction Robots*, Cambridge University Press, 2016, pp. 14–290. doi: 10.1017/CBO9781139872041.002.

[6] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: 10.1038/nature16961.

[7] S. Mahadevan and G. Theocharous, "Optimizing Production Manufacturing Using Reinforcement Learning," in *The Florida AI Research Society*, 1998.

[8] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery," *IEEE Robot Autom Lett*, vol. 6, no. 3, pp. 5816–5823, 2021, doi: 10.1109/LRA.2021.3074883.

[9] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications," Dec. 2018, doi: 10.1109/tcyb.2020.2977374.

[10] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annu Rev Control*, vol. 46, pp. 8–28, 2018, doi: https://doi.org/10.1016/j.arcontrol.2018.09.005.

[11] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int J Rob Res*, vol. 32, no. 11, pp. 1238–1274, 2013, doi: 10.1177/0278364913495721.

[12] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning," May 2017, doi: 10.48550/arxiv.1705.01196.

[13] Q. Zhu and J. Oh, "Deep Reinforcement Learning for Fairness in Distributed Robotic Multi-type Resource Allocation," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 460–466. doi: 10.1109/ICMLA.2018.00075.

[14] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized Non-communicating Multiagent Collision Avoidance with Deep Reinforcement Learning," Sep. 2016, Accessed: Apr. 26, 2023. [Online]. Available: https://arxiv.org/abs/1609.07845

[15] C. Shyalika, T. Silva, and A. Karunananda, "Reinforcement Learning in Dynamic Task Scheduling: A Review," *SN Comput Sci*, vol. 1, no. 6, p. 306, 2020, doi: 10.1007/s42979-020-00326-5.

[16] J. Lin, X. Yang, P. Zheng, and H. Cheng, "End-to-end Decentralized Multi-robot Navigation in Unknown Complex Environments via Deep Reinforcement Learning," Jul. 2019, Accessed: Apr. 26, 2023. [Online]. Available: https://arxiv.org/abs/1907.01713

[17] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments," Jul. 2020, doi: 10.48550/arxiv.2007.15724.

[18] A. Agrawal, A. S. Bedi, and D. Manocha, "RTAW: An Attention Inspired Reinforcement Learning Method for Multi-Robot Task Allocation in Warehouse Environments," Sep. 2022, doi: 10.48550/arxiv.2209.05738.

[19] D. Lee, S. Lee, N. Masoud, M. S. Krishnan, and V. C. Li, "Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction," *Advanced Engineering Informatics*, vol. 53, p. 101710, 2022, doi: https://doi.org/10.1016/j.aei.2022.101710.

[20] Z. Shang, Z. Mao, H. Zhang, and M. Xu, "Collaborative Path Planning of Multiple Carrier-based Aircraft Based on Multi-agent Reinforcement Learning," in *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, 2022, pp. 512–517. doi: 10.1109/MDM55031.2022.00108.

[21] H. Hu, X. Yang, S. Xiao, and F. Wang, "Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning," *Int J Prod Res*, vol. 0, no. 0, pp. 1–16, 2021, doi: 10.1080/00207543.2021.1998695.

[22] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," Sep. 2017, doi: 10.48550/arxiv.1709.10082.

[23] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and Path Planning for Multi-Agent Pickup and Delivery," in *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*, in AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1152–1160.

[24] A. Elfakharany and Z. H. Ismail, "End-to-End Deep Reinforcement Learning for Decentralized Task Allocation and Navigation for a Multi-Robot System," *Applied Sciences*, vol. 11, no. 7, 2021, doi: 10.3390/app11072895.

[25] N. D. Nguyen, S. Nahavandi, and T. Nguyen, "A Human Mixed Strategy Approach to Deep Reinforcement Learning," Apr. 2018, doi: 10.1109/SMC.2018.00682.

[26] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," Dec. 2015, doi: 10.48550/arxiv.1512.04150.

[27] Y. and P. W. and K. T. K. S. and K. S. and C. H. Sartoretti Guillaume and Wu, "Distributed Reinforcement Learning for Multi-robot Decentralized Collective Construction," in *Distributed Autonomous Robotic Systems*, M. and O. M. Correll Nikolaus and Schwager, Ed., Cham: Springer International Publishing, 2019, pp. 35–49.