Transactions on Computational and Scientific Methods | Vo. 4, No. 8, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Optimized Deep Reinforcement Learning for Cooperative Path Planning and Obstacle Avoidance in AGV Dynamic Environments

Zayden Connors
Department of Electrical Engineering, University of Tennessee, Knoxville, USA
zayden.connors5499@utk.edu

Abstract: As smart logistics and factories advance, Automated Guided Vehicles (AGVs) face increasingly complex and dynamic environments, requiring more intelligent path planning and obstacle avoidance. Traditional path planning algorithms, while widely used, often suffer from high computational costs and limited generalization capabilities. This paper introduces an optimized approach utilizing Deep Deterministic Policy Gradient (DDPG) and Multi-Agent DDPG (MADDPG) algorithms to address AGV cooperative path planning in dynamic scenarios. By modeling AGVs as agents within a deep reinforcement learning framework, a centralized training and decentralized execution approach is employed. The algorithms are enhanced through optimized experience buffer sampling methods, allowing agents to autonomously navigate and avoid static and dynamic obstacles.

Keywords:AGV; Deep Reinforcement Learning Algorithm; TD-error; Intelligent Collaborative PathPlanning.

1. Introduction

In recent years, with the development of smart logistics and smart factories[1] , the AGV operation environment is more complex and dynamic, and the requirements for AGV intelligence are higher, so it is especially important for the implementation of AGV obstacle avoidance and cooperative path planning in dynamic environments. The current traditional path planning methods include algorithms based on fuzzy logic[2] , simulated annealing[3] , artificial potential fields[4] , and intelligent path planning algorithms such as ant colony algorithm [5] , A* algorithm [6] , genetic algorithm [7] and particle swarm algorithm [8] and so on.

Through analyzing the current situation of domestic and foreign research, the traditional path planning algorithm is computationally intensive, with a single task and weak generalization ability of intelligent algorithms based on grid method construction map construction. Deep reinforcement learning has achieved breakthrough success in mobile [9–11] , offering unlimited possibilities to achieve intelligent control and autonomous learning of robots[12]. Instead of relying on data models and raster map construction, deep reinforcement learning algorithms only need to set planning goals and let the agents keep exploring to achieve problems such as obstacle avoidance, path planning, and navigation.

Deterministic Policy Gradient algorithm [13] in deep reinforcement learning is an artificial intelligence method for continuous policy learning, which is naturally applicable to a class of dynamic obstacle avoidance type continuous decision problem of achieving stable operation of the AGV in

dynamic scenarios[14]. In the deep reinforcement learning framework, by considering AGVs as agents interacting with the environment, a centralized training and decentralized execution approach are used to set a reasonable reward function and optimize the experience pool so that the agents update their strategies to obtain the maximum reward. Thus, the learning of the optimal strategy for the AGV to reach the target points in a shorter time is achieved.

In this topic, the MADDPG [15] and DDPG algorithms based on the deep learning framework are used to train AGV operation models for the problem of achieving cooperative and stable AGV operations in dynamic scenes. The optimized MADDPG algorithm has a better strategy compared with the MADDPG and DDPG algorithms before the improvement, which can realize the task of cooperatively reaching the destination and avoiding static and dynamic obstacles in the dynamic environment of AGV.

2. AGV Markov Model Description

2.1. AGV State Space Description

Taking the AGV as the agent body, the environmental information perceived during the travel of the AGV is described as the state space, and the state space considers the location of the destination as well as the location of the obstacles.

$$\dot{s}(t) = \frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \varphi(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} v(t) \cdot \cos\theta\,(t) \\ y(t) \cdot \sin\theta\,(t) \\ \alpha(t) \\ \omega(t) \\ v(t) \cdot \tan\varphi\,(t)/L_A \end{bmatrix}, t \in [0, t_e]$$

The S_{aai} represents the communication information between AGVs, which is obtained by the angle matrix and the position coordinates of AGVs.

The state space of AGV is described as S= (S_{ga} , S_{oa} , S_{aai} , Dis_{ga} , Dis_{oa})

2.2. AGV Action Space Description

The action in AGV path planning is mainly the control of speed and angular velocity, through the speed control can realize the acceleration when far from the target point and deceleration when close to the obstacle; through the angular velocity control can realize the orientation conversion of AGV. According to the characteristics of AGV, the action of AGV is discrete, the speed is discrete as 0.5m/s, 0.2m/s, 0m/s, and the angular speed is discrete as -1rad/s, 1rad/s, -0.5rad/s, 0.5rad/s, 0rad/s. In practice, the specific action is shown in Table 1.

Table 1: Action space

Default	Parameters	Angular velocity
0	0.5m/s	-1rad/s
1	0.2m/s	-1rad/s
2	0m/s	-1rad/s
3	0.5 /s	-0.5rad/s
4	0.5m/s	0rad/s
5	0.2m/s	0rad/s
6	0.5m/s	0.5rad/s
7	0.5m/s	1rad/s
8	0.2m/s	1rad/s

9	0m/s	1rad/s

2.3. MADDPG Algorithm

The MADDPG algorithm is a multi-agent algorithm that extends the DDPG algorithm and the Actor-Critic method. The framework of the MADDPG-based AGV training model is shown in Fig.1.The AGVs running in the operational environment consist of an Actor-Critic dual network, and the AGVs interact with the environment and execute actions at all times. In the process of training path planning strategies for AGVs, a centralized training and decentralized execution approach is used, i.e., when the number of AGVs exceeds one, the AGVs not only rely on their own strategies to execute actions, but also extract the experience of other agents from the experience buffer to train their own neural networks.

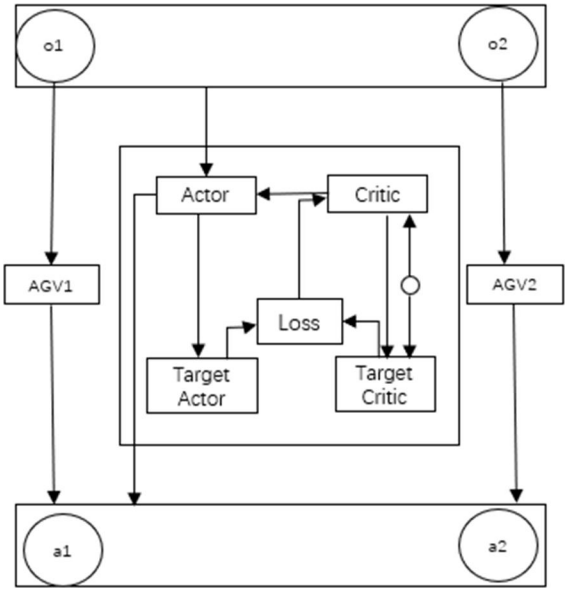


Figure 1. AGV training model framework for MADDPG

3. Experiment Simulation

3.1. Environment and Parameter Setting

This experiment was simulated for single AGV, double AGV in the experimental environment, and the starting point of AGV as well as the unknown of the target point and obstacles were initialized and updated for each episode of the experimental environment. At the same time, in order to reduce the problem of increasing distance from the target point caused by the inability of the AGV to explore effectively at the early stage of training, this experiment sets the maximum step size of 25 per episode. specific experimental parameters are shown in Table 2.

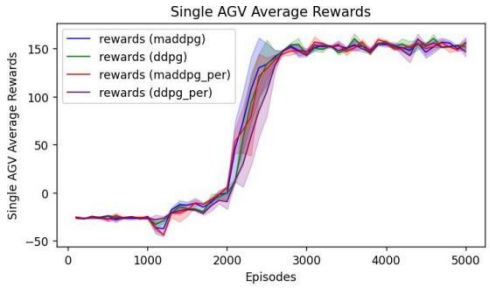
Table 2: Hyperparameters of path planning method

Number	Speed	Description
lr	0.01	Rate of learning
Mini-batch	1024	Mini-batch gradient descent
D	1e6	Experience buffer capacity
γ	0.95	Decay factor
α	0.6	Mini-batch
β	0.442	Priority weighting amplification factor

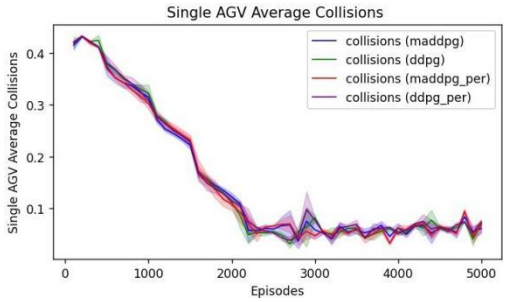
γ	0.95	Offset probability

3.2. Experimental Results and Discussion

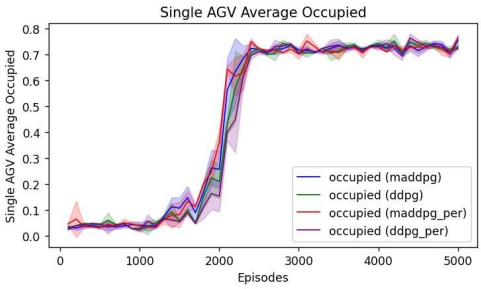
The average rewards, collision rates and average coverage rates are shown in Fig. 2 and Fig. 3 after 5000 episodes and 30 000 episodes of training with the improved MADDPG, MADDPG-per, DDPG, and DDPG-per, respectively, in the environments with the number of AGVs N=1, and N=2.



(a) Single AGV Average Rewards



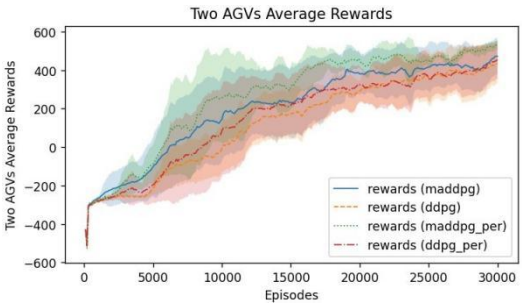
(b) Single AGV Average Collisions



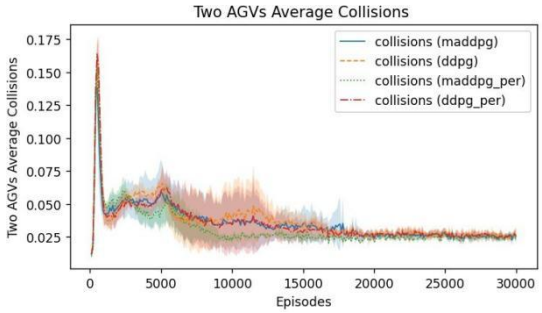
(c) Single AGV Average Occupied

Figure 2. Single AGV strategy training

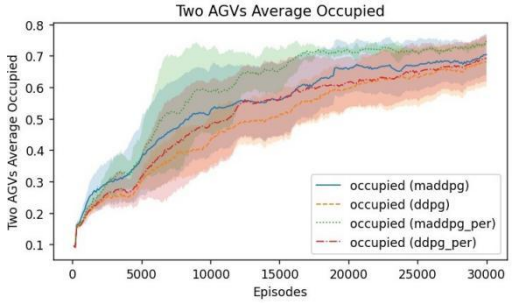
In Fig. 2, it can be seen that for single a AGV system in the MADDPG-per algorithm and DDPG-per algorithm using the preferred experience mechanism reward convergence with small improvement, and the target coverage and collision rate are almost the same.



(a) Two AGVs Average Rewards



(b) Two AGVs Average Collisions



(c) Two AGVs Average Occupied

Figure 3. Two AGVs strategy training

As can be seen from Fig. 3, for the dual AGV system, the reawrds with the MADDPG-per algorithm have a greater improvement in the middle of training, smoother reawrd values after convergence, and higher target coverage and lower collision rate with the same episode.

For the single AGV system, after 3000 episodes, the AGV basically learns the path planning strategy. For the dual AGV system, after 25 000 episodes, the AGV basically learns the path planning strategy. the specific strategy indexes of the AGV are shown in Table 3 and Table 4.

Table 3: Reward Comparison

Number of AGVs	Algorithm	Reward Convergence Episode	Reward Convergence Value
Single AGV	MADDPG-per	2650	159.63
	MADDPG	2700	154.83
	DDPG-per	2770	153.52
	DDPG	2800	151.52
Two AGVs	MADDPG-per	16000	515
	MADDPG	18500	395
	DDPG	26000	355
	DDPG-per	27000	350

Table 3 shows the reward training of a single AGV and Two AGVs in an unknown dynamic environment. The MADDPG-per algorithm for a single AGV has the fastest reward convergence, with a basic convergence at episode 2650 and a convergence reward of 159.63. Compared to the MADDPG algorithm the payoff convergence is 1.9% faster with a 3% increase in reward and compared to the DDPG algorithm the reward convergence is 5.7% faster with a 5.08% increase in reward. The DDPG- per algorithm with a single AGV has a faster reward convergence rate, basically

converging at 2770 episodes with a convergence reward of 153.52, which is 1.08% faster and 1.3% more rewarding compared to the DDPG algorithm reward convergence rate. Over all the single AGV with DDPG-per and MADDPG-per has less return improvement. The MADDPG-per with two AGVs has the fastest reward convergence, with a basic convergence at 16 000 episodes and a convergence return of 515. compared to the MADDPG algorithm return convergence speed is improved by a 15.6% reward increase of 23.3%, compared to the DDPG algorithm reward convergence speed is improved by 62.6% and a reward increase of 31.1%. Over all the reward performance of MADDPG-per for two AGVs has a more significant improvement than a single AGV, and the reward performance of DDPG-per has a slight decrease than DDPG.

Table 4: Comparison of task indicators

Number of AGVs	Algorithm	Target coverage rate	Impingement rate
Single AGV	MADDPG-per	78.1%	0.048
	MADDPG	75.4%	0.052
	DDPG-per	73.4%	0.062
	DDPG	73.2%	0.069
Two AGVs	MADDPG-per	76.6%	0.026
	MADDPG	65.4%	0.027
	DDPG	62.8%	0.029
	DDPG-per	59.1%	0.029

In Table 4, the target coverage and collision rates for a single AGV and two AGVs at the convergence of training rewards are shown. Among them, for a single AGV, MADDPG-per achieves the highest target coverage of 78.1%, which is 3.46%, 6.01%, and 6.3% higher compared to the other algorithms, respectively. The collision rate reaches a minimum of 0.048, which is 8.3%, 29.1%, and 43.8% lower than the other algorithms, respectively. the target coverage of DDPG-per reaches a maximum of 73.4%, which is 0.27% higher than the DDPG algorithm, respectively. The collision rate reaches 0.062, which is reduced by 11.3% compared to the DDPG algorithm. Among them, for dual AGVs, MADDPG-per achieves the highest target coverage of 76.6%, which improves 14.6%, 18.1%, and 22.8%, respectively, compared to other algorithms. The lowest collision rate reaches 0.026, which is 17.4%, 26.1%, and 26.1% lower compared to other algorithms, respectively. Among them, DDPG-per takes the sample importance measure only for the critic input of one AGV, while the transition of different AGVs is different, and the sample importance of AGV_1 and AGV_2 are different, and the number of training for the important sample of another AGV becomes less. Therefore, DDPG-per has a better strategy in a single AGV, while the performance of the algorithm decreases in double AGV training. Overall for the MADDPG-per algorithm, both single and dual AGVs have higher target coverage and lower collision rates.

4. Conclusion

In this paper, Markov model is constructed for the characteristics of AGV operation. And two continuous policy algorithms of deep learning algorithms, DDPG and MADDPG, are used to treat AGVs as agents, while the experience buffer sampling methods of the two algorithms are optimized. The experiment uses the gym platform to simulate the unknown dynamic AGV operation environment, and each episode is randomly assigned with unknown obstacles and target points. The experiments compare the results of the two algorithms before and after improvement, with reward convergence, collision rate and coverage rate as performance indicators. Finally, the visualization results of the model training show that MADDPG-per is able to learn better strategies and has the ability of autonomous path planning to better achieve the task of obstacle avoidance and cooperative operation of AGVs in unknown dynamic environments.

References

[1] L. D. Evjemo, T. Gjerstad, E. I. Grøtli, and G. Sziebig, "Trends in smart manufacturing: Role of humans and industrial robots in smart factories," Current Robotics Reports, vol. 1, no. 2, pp. 35–41, 2020.

[2] G. T. Zoumponos and N. A. Aspragathos, "Fuzzy logic path planning for the robotic placement of fabrics on a work table," Robotics and Computer Integrated Manufacturing, vol. 24, no. 2, pp. 174–186, 2008.

[3] H. Miao and Y.-C. Tian, "Dynamic robot path planning using an enhanced simulated annealing approach," Applied Mathematics and Computation, vol. 222, pp. 420–437, 2013.

[4] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," Autonomous Robots, vol. 13, no. 3, pp. 207–222, 2002.

[5] G. Z. Tan, H. E. Huan, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," ACTA AUTOMATICA SINICA, vol. 33, no. 3, pp. 279–285, 2007.

[6] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric a-star algorithm: an improved a-star algorithm for agv path planning in a port environment," IEEE Access, vol. 9, pp. 59 196–59 210, 2021.

[7] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," Computers & Electrical Engineering, vol. 38, no. 6, pp. 1564–1572, 2012.

[8] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826), vol. 4. IEEE, 2005, pp. 2473–2478.

[9] M. Feng and H. Xu, "Deep reinforecement learning based optimal defense for cyberphysical system in presence of unknown cyber-attack," in 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017, pp. 1–8.

[10] B. H. Abed-alguni and M. A. Ottom, "Double delayed q-learning," International Journal of Artificial Intelligence, vol. 16, no. 2, pp. 41–59, 2018.

[11] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3221–3226, 2020.

[12] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013.

[13] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in international conference on machine learning. PMLR, 2014, pp. 387–395.

[14] Y. Li, X. Wang, W. Wang, Z. Zhang, J. Wang, X. Luo, and S. Xie, "Learning adversarial policy in multiple scenes environment via multi-agent reinforcement learning," Connection Science, vol. 33, no. 3, pp. 407–426, 2021.

[15] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multiagent actor-critic for mixed cooperative-competitive environments," Advances in neural 13 information processing systems, vol. 30, 2017.

[16] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. arXiv preprint arXiv:1511.05952.

[17] Ramicic and A. Bonarini, "Correlation minimizing replay memory in temporal difference reinforcement learning," Neurocomputing, vol. 393, pp. 91–100, 2020.