

DQN–MADDPG Coordinating the Multi-agent Cooperation

Zhisheng Chen *

School of Software, South China Normal University, Foshan, China

* Corresponding author email: 20202031031@m.scnu.edu.cn

Abstract. Multi-agent coordinating aims for many agents to finish the same or more mutual goals. To achieve this function, there are two main ideas; the first is making multi-agents communicate with each other and acquire the information for other agents. Another is sharing the same environment, dividing the work, and cooperating to achieve the goal. “Gym-cooking” is an excellent model to test the algorithms’ performance in network coordination and game theory; this is a sharing environment. Based on sharing information, the agent have two networks(policy and target) and tries to do something more efficiently. This article will increase the complexity of the environment and use different algorithms to process the experiment. Specially, this paper will use the MADDPG model as the primary model to show its performance in a complex environment and contrast other models like DQN. The MADDPG model in this experiment is unlike the traditional MADDPG; the work trains the MADDPG network to deal with emergencies and accidents.

Keywords: DQN; MADDPG; Multi-agent; Cooperation.

1. Introduction

Multi-agent coordination has extensive prospection, such as self-driving, industrial control, game theory, robot coordination, etc. With the development of the neuron network, people can solve more complex problems and consider more complicated situations. Multi-agent coordination is a crucial aspect of multi-agent systems, as it enables agents to work together effectively to achieve their common goals. Research in this area has focused on developing algorithms, protocols, and models to facilitate effective coordination among agents [1]. This work has included studies of communication and cooperation strategies and the application of game theory to multi-agent systems. In addition, researchers have also explored the use of machine learning and artificial intelligence techniques to improve the coordination of agents.

Multi-agent coordination is confronted with some main challenges: 1. the complexity of the coordination; 2. The change in the environment; 3. Lack of common sense; 4. Too much information cause coordinate difficultly; 5. Cheat behaviors, and so on. [2]. The coordination environment needs robustness and stability when encountering mistakes or accidents. It needs to be considered when building the model to deal with the actual situation [3]. Multi-Agent Deep Deterministic Policy Gradient (MADDPG) has good performance in the case of multi-agent coordinating. MADDPG is based on DQN; it adds the actor-critic mechanism to improve the performance, and it still has an excellent performance in solving the more dimension of action_space and state_space. This paper will use the MADDPG model as the primary model to show its performance in a complex environment and contrast other models like DQN. The MADDPG model in this experiment is unlike the traditional MADDPG; the author trains the MADDPG network to deal with emergencies and accidents [4]. The experiment will consider a more complex environment, increase the ratio to an emergency occur, optimize the model, and improve the mechanism to solve these problems. As for the environment in this experiment, it is imitating the gym package from OpenAi, which is based on the model of a restaurant having many orders and recipes that agents need to divide the work to subtask; furthermore, the restaurant will suffer some accidents, which require agents respond immediately and adjust their plan and restart divide the task into subtasks.

Overall, this research has helped advance our understanding of how agents can coordinate their actions and has led to the development of more effective multi-agent systems. The researchers use two central policies to solve the problem of multi-agent coordination, one is game theory, and the other one is machine learning algorithms; the game theory tries to analyze each agent’s goals and

behavior to infer their possible actions and guide them to coordinate. The machine learning algorithms use the neuron network to study the agents' behaviors and imitate the agents to improve coordination efficiency [5]. Unlike other experiments, this experiment's environment has more variable and random accidents to improve the performance of the MADDPG model. And use the experience replay to e, breviate the time for training the network. Inspired by Too many cooks and their bayesian delegation, this environment adds more dimension of state and action to coordinate the agents.

2. Method

The paper uses MADDPG to measure how multi-intelligent agents divide work. We use the restaurant cooking (environment), which has two agents and how to separate work to cook and finish more than 20 orders by cooperation. We refer to the environment as "too many cooks" and create our environment to increase two dimensions [6]. One of them is when agents divide the task, the priority of the subtask, which means in a subtask, agents do first, and the other is next. The agent who finishes his portion can depart from this subtask, or the agent does not wait for another agent. Another function is how agents perform more than one task. As for these dimensions, we can measure more situations by dividing the work of subtasks [7]. Furthermore, the mechanism is added to change the environment variable; there are two main changes; one is that the order is asked for in advance, and the agents need to replan their subtasks, and the other change is material will rot. The agents need to reprepare for the new material.

2.1 Environment Information

We create the environment referring to the package gym of OpenAI; we write the several main functions that env. step() env. reset(). As follow, the introduction of parameters of ENV.

There is Action_space, State_space, Observation.

Action_space = [0: cut tomato 1: cut lettuce 2: prepare bacon, 3: prepare beef 4: cooperate 5: mix]
total 4 actions

```
State_spac=[agent1(none, cut1,cut2,cut3,cut4)
            agent2(none,cut1,cut2,)
            tomato(notcut,1,2...)
            lettuce(not,cut,1,2...)
            bacon(not,cut,1,2...)
            beef(not, cut,1,2...)
            recipe1(none, 1, 2)
            recipe2(none,1,2)
            recipe3(none,1,2)
            recipe4(none,1,2)
            recipe5(none,1,2)
            recipe6(none,1,2)
            order_sequential]
```

And the observation structure is the same as the State_space. The explanation of the actions in the Action_space [8]. The agent has 5 acts; when the agent executes the cut tomato, the agent's state will become cut1, and the tomato's state will add "1", which means the number of tomatoes is ready. The act-cut lettuce is the same as the act-cut tomato. Still, the reward differs when the agent chooses to cooperate when another state is cut1 or cut2. When the agent act and collaborates with another agent, the state is cut1, and the reward is 0.1, while will cut2, the reward is 0.01. It can be distinct the efficiency of the cooperation, avoiding being trapped in constantly choosing the association. Then, when the agent wants to act mix, the tomato state and lettuce state need to be up to 1, and they will be minus one, and the salad state adds 1. Finally, the condition of finishing an episode is salad's state of more than two or equal to two, and this step deserves one reward [9]. Above all, the restaurant has

six recipes and order_sequential; the environment adds a time dimension and cooperates priority to improve the performance of the DDPG model to deal with the complex situation.

2.2 MADDPG Model

MADDPG expands the actor-critic function in DDPG [2]. In DDPG, its action space is constant, but it is not suitable to solve the problem whose action space is dispersal. MADDPG is appropriate to solve continuous action environment. Amid the algorithms, Gumbel-Softmax Trix transfers the distributed policy network into concrete action and irrelevant noise to the policy network so that the gradient can spread.

MMDDPG includes two main parts, the Multi-Agent Decision Process and DDPG, each agent in MADDPG has its policy network; they can share the same environment and make their own decisions for every step, and the other part is to share the value network to help them get the highest reward base on their policy network [10]. Besides, unlike different distributed deep learning algorithms, MMDDPG can study without knowledge about other agents, which helps this model deal with a situation with many agents (Figure 1).

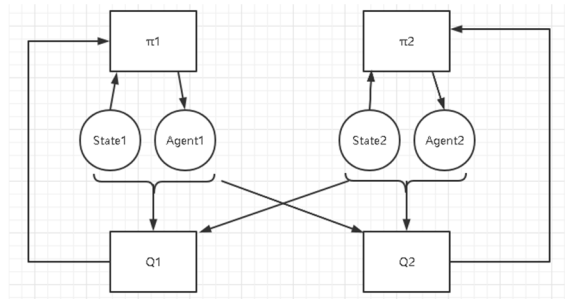


Fig 1. MADDPG model

π represents the agent's policy and design its parameter as θ , $J(\theta)$ represents the accumulated expected reward for i agent. The gradient of i agent can

$$\nabla_{\theta_i} J(\theta_i) = E_{s \sim p^\mu, a_i \sim \pi_i} [\nabla_{\theta_i} \log \pi_i(a_i | o_i) Q_i^\pi(x, a_1, \dots, a_N)] \quad (1)$$

And O_i means the observation of i agent $x = [O_0, \dots, O_I]$, Q_1^n , is the critic network.

As for the determined policy, its gradient can be represented as:

$$\nabla_{\theta_i} J(\mu_i) = E_{x, a \sim D} [\nabla_{\theta_i} \mu_i(a_i | o_i) \nabla_{a_i} Q_i^\mu(x, a_1, \dots, a_N) | a_i = \mu_i(o_i)] \quad (2)$$

It is utilized to update the actor network; for the critic network, it needs to calculate the average variance as the loss to update the function.

$$\mathcal{L}(\theta_i) = E_{x, a, r, x'} [((Q_i^\mu(x, a_1, \dots, a_N) - y)^2], \quad y = r_i + \gamma Q_i^\mu(x', a'_1, \dots, a'_N | a'_j = \mu'_j(o_j)) \quad (3)$$

Q^μ as the target_critic_network.

2.3 Interaction

Cooperation efficiency. In the experiment environment that creates two agent networks, it means the agent observation includes another agent's state; it will consider whether cooperation with other agents, so the agents have five states, two of them the cut1 and cut4, having high efficiency if the other agent chooses to act as cooperation [5]. Typically, the state of one material adds one with the corresponding action, but if they decide efficient action combination, it will add three materials to represent that they save time.

The structure of the network. The agent network still has the experience of replay institution to accelerate and train the network to converge. All of the agents will share one experience pool. Each agent's policy and target network consist of two layers, the input layer, and the output layer, and the input layer consists of (13, 64), and the output layer consists of (64, 10).

Random accident. In order to simulate the actual situation, a function will cause the ready material to disappear and change the order sequentially. It requests that the agents can respond immediately to the changing state. The accident will probably occur after every step; the environment has a list of order_sequential to ask for the correct sequence. It will imitate the accident that the order will be canceled.

Finish condition. The order list includes six recipes; if the agents can finish 30 tasks, the game will end, and the process will be the episode.

3. Result

This experiment implements the DQN and the MADDPG model, both suitable for resolving the dispersal action space (Figure 2-3). The main difference between this model is the DQN only have one network, while each agent has their own network in MADDPG.

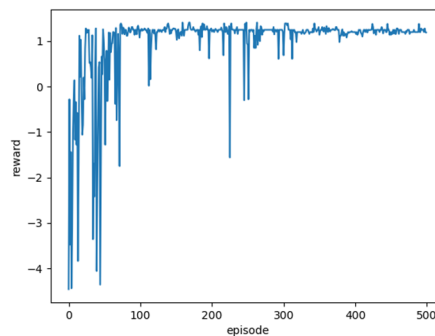


Fig 2. The result of DQN

The episode and the value picture show that the DQN model converges after about the episode of 300, while the MADDPG converges after about 180 episodes; in the same environment, the MADDPG face the complex situation has better performance than the DQN; in the field of multi-agent coordination, MfacesG faces the random accident still more vital longer able to solve it while the DQN.

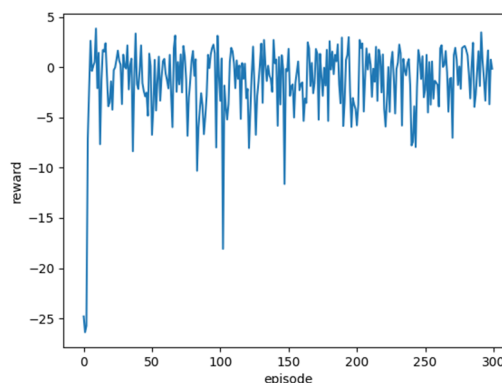


Fig 3. The result of MADDPG

In more complex environments, the MADDPG model can converge quickly, while the DQN model is hard to a stable value.

The parameter of the DQN. These two environments' structure is similar, but their action space and the state space 's dimension are different, which test the complexity of the environment in the two models. Environment1's action space has four dimensions, state space has five states, environment2's s action space has five actions, and state space has thirteen dimensions.

Their network initialization is the same. The learning rate, the batch size of experience replay, and neuron number are the same; they are only different in the number of networks. DQN's agents share the target and evaluation network, and the MADDPG's agents each have two networks.

MADDPG compared with the Bayesian Delegation. both are based on each agent deciding on action. Bayesian Delegation is based on the odds, and the MMDDPG is based on the neuron network and policy gradient. The result of the experiment, the Bayesian Delegation can train fast and spend less time than the MADDPG because of its need to base on the fundamental recognition of the environment so that the MADDPG may perform well in strange environments. Bayesian will perform well in a familiar environment.

4. Conclusion

This experiment aims to compare the two-reinforcement learning algorithm and one probability algorithm; in the cooking game, the MADDPG is better than DQN, and the Bayesian Delegation has the advantage of spending less time to train while it needs prior experience. The MADDPG can study without any former knowledge. In a complex, exceptional multi-agent environment, the MADDPG is well over DQN. MADDPG has been successfully applied in a number of applications, such as controlling multi-drone systems, collaborative robots, and more.

In the future, MADDPG may continue to evolve and be applied in even more applications. For example, it could be used to control autonomous vehicles or more complex collaborative robot systems. Additionally, there may be new developments, such as new multi-agent reinforcement learning algorithms or improvements to MADDPG, that make it more suitable for specific applications.

References

- [1] Rose E. Wang, Sarah A. Wu, James A. Evan, Joshua B. Tenenbaum, David C. Parkes, Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi_agent collaboration, 2020,7(2):23-31.
- [2] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, Igor Mordatch, Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, 2020,9:323-342.
- [3] R.J.Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 1992, 8(3-4):229–256.
- [4] J. Hu and M. P. Wellman. Online learning about other agents in a dynamic multiagent system. In Proceedings of the Second International Conference on Autonomous Agents,1998, 239–246.
- [5] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge. Morgan Kaufmann Publishers Inc., 1996, 195–210.
- [6] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. PloS one, 2017, 12(4):e0172395.
- [7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. 2015, arXiv preprint arXiv:1509.02971.
- [8] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In Proceedings of the 31st International Conference on Machine Learning, 2014, 387–395.
- [9] Eithan Ephrati and Jeffrey S Rosenschein. Divide and conquer in multi-agent planning. In AAAI, 1994. 1. 80.
- [10] Barbara J Grosz and Sarit Kraus. 1996. Collaborative plans for complex group action. Artificial Intelligence 1996, 86, 2: 269–357.