Learning to Cooperate using Deep Reinforcement Learning in a Multi-Agent System

Nabil Khan and Maria Gini

University of Minnesota-Twin Cities, Minneapolis MN 55455, USA
khan0096@umn.edu,gini@umn.edu

Abstract. We address the problem of emergence of cooperation between agents that operate in a simulated environment, where they need to accomplish a complex task, moving a heavy load, that is decomposed into sub-tasks and that can be completed only with cooperation. A deep reinforcement learning approach using a multi-layered neural network is used by the agents. The goal of this work is to empirically show that cooperation can emerge without explicit instructions, whereby agents learn to cooperate to perform complex tasks, and to analyze the correlation between task complexity and training time. The series of experiments we conducted helps establish that cooperation can emerge but becomes unlikely in partially observable environments as the environment size scales up. Another series of experiments shows that communication makes the cooperative behavior more likely, even as environments scale up, when the environment is only partially observable. However, communication is not a necessary condition for cooperation to emerge, since agents with knowledge of the environment can also learn to cooperate, as demonstrated in the fully observable environment.

Keywords: cooperation · reinforcement learning · multiagent system.

1 Introduction

Multi-agent systems are increasingly popular and are often tasked with accomplishing complex tasks that require cooperation. People experience the reality of their physical limitations as individuals when trying to move a bulky or irregular sized object which requires them to coordinate with others to accomplish the task. The need to work cooperatively arises from trying to interact in the environment with artifacts that are larger and more complex than what a single person can reasonably manipulate. Given the complexity and multitude of tasks that have to be performed all around us, cooperative team work is inevitable for successful functioning in organizations, sports, warehouses and almost every facets of society. In an ever increasing pace of automation we see that autonomous agents need to work cooperatively, where they operate in an environment with objects that are bigger than themselves and hence cannot be manipulated by a single agent.

Let us consider a team of warehouse robots that are tasked with cooperatively moving objects to a specific location. The robots have to learn the optimal path to the object itself and then synchronously move the object to the goal location. Robots can be easily programmed to do this but if a robot were to be made truly adaptive it needs to have the ability to learn the optimal set of actions in any environment. The robot will explore the space and, given the correct set of rewards, will learn to navigate to the location of the object and then possibly learn to cooperatively move the object to the target location. The purpose of this work is to investigate how robots learn to cooperate in a simulated environment that has been broadly inspired by a factory warehouse. The series of experiments have been completed in three phases with different conditions of the environment: (1) fully observable, (2) partially observable, and (3) partially observable with communication.

The focus and main contribution of this paper is to comprehensively study how properties of the environment affect learning in a new problem scenario and how communication affects the overall learning in a partially observable environment when compared to a fully observable environment. To the best of our knowledge, the use of neural networks with reinforcement learning for the purpose of studying the emergence of cooperation in the three categories listed above, based on observability and communication, is a first. This work also shows that the availability of complete information in a fully observable environment does not guarantee cooperation but can be rather detrimental to the learning effort in a large environment whereas communication provides a more efficient way to make the relevant information available to agents. The work described in this paper is largely based on [6].

2 Related Work

Multi-agent cooperation has seen significant progress in the past two decades. Recent application of deep learning in fields of computer vision and natural language processing has contributed to its application in multi-agent task environments. One of the earlier papers by Mahadevan et al. [10] explores behavior-based robots using reinforcement learning from a single agent perspective. A real robot called OBELIX combined with a simulator for gathering data is used for tasks that involve pushing boxes. Q-learning is used as the reinforcement learning algorithm. An emphasis on the learning of actions via reinforcement learning is made in the paper because it removes the burden of programming the behavior into the agents.

In further development, Tan [14] studied cooperative agents with independent agents as a benchmark. Agents can be cooperative by communicating different types of information such as perception, rewards, action, episodes. The paper presents three case studies each with three different types of information, instantaneous information, sequential information, learned decision policies. In the first case study the information shared is the perception of another agent. The second case study shares learned policies and episodes. The third case study involves

joint tasks where more than one agent is needed to accomplish the task. Tan's paper makes the claim that cooperative agents outperform independent agents although learning might take longer. The experiments are setup to measure success in terms of the discounted reward per action. The extension of reinforcement learning from single agents in prior work to multiple agents is defined by Tan as the comparison between n independent agents and n cooperative agents. The main thesis of the paper is that agents can benefit from the information shared between them.

Kasai et al. [5] further extended the work in [14] by enabling the learning of communication codes. The pursuit problem where agents are tasked with cooperatively capturing a prey is used for the purpose of learning communication codes. Learning of communication code is called Signal Learning in the paper and it is accomplished by encoding key features of the state space such as location of prey as a sequence of bits of fixed length. The paper examines the effect on performance when the signal length is varied between experiments. Kasai et al. claims that learning relevant information to share between agents is crucial because irrelevant information can have a detrimental effect on the overall learning process. The work in this paper essentially automates the information that is shared between agents whereas communication code was predefined in [14].

In [1] agents learn communication protocols to solve switch riddle and games in a fully cooperative, partially observable environment. Two approaches are used Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL). Since agent behavior has to be coordinated, agents have to discover a communication protocol. During learning the communication bandwidth is not restricted, that is learning is centralized and execution is decentralized. RIAL uses Q-learning with a recurrent network and two different approaches are used within this method. The first approach is independent Q-learning where agents learn their own network parameters. The second approach of independent Q-learning parameters are shared. In DIAL gradients are passed between agents as real valued messages. During execution, since only limited bandwidth channels are available, the real valued messages are discretized. Experimental results show that DIAL outperforms both RIAL methods. Parameter sharing is also found to be crucial for learning the optimal protocol. The baseline also shows that no learning takes place without any communication. The paper establishes that differential communication is essential to utilize the full potential of centralized learning. The paper claims to be a first attempt at learning communication and language with deep learning approaches.

Gupta et al. [4] consider cooperation in partially observable environments without communication. The problem of cooperation can be formulated as a decentralized partially observable Markov decision process as seen before. Gupta et al. claim that policy gradient methods outperform both temporal difference and actor-critic methods. However, since exact solutions are intractable, approximates have to be applied. It is worth considering that since approximate methods are limited to discrete action spaces applying them to real-world problems require careful design considerations. Further, the combination of deep learning

and approximate methods has given rise to deep reinforcement learning and has seen considerable success in complex tasks such as robotic locomotion.

Lan et al. [8] show that discrete language appears even when using a continuous space. Their work uses signaling games where an agent perceives some information and produces a message for a receiver that takes an action. Discreteness, which is clustering of words in acoustic space and displacement, which occurs with communication beyond the immediate context are the focus here. Besides the emergence of discrete language, using a continuous communication protocol means that standard back-propagation can be used for end-to-end training.

Communication has been shown to be important in ad-hoc teamwork [13] to get the agents to learn how to work together [11]. Stone [13] discusses coordination without preconditions where an ad-hoc team of agents with different capabilities coordinate on a set of tasks with perfectly aligned goals. Robot soccer with the variation of a pickup game is an example of such an ad-hoc cooperative strategy that can be studied within the framework of game theory. The paper elaborates on the collaborative multi-arm bandit with a teacher and a learner and challenges the community to develop other approaches. Multiple papers have been written since that initial paper on ad-hoc team.

A relevant related research topic is for agents to learn how to communicate. Multiple methods have been proposed, such as using deep reinforcement learning [1] or learning from perceptual data and by following instructions [3]. In this work, when we use communication the agents use messages to share information with other agents. Learning how to communicate is a topic we will address in future work.

Gerkey [2] presents a formal study of multi-agent task allocation which essentially determines which robot should execute which task. Task independence is assumed and order constraints are not permitted. Two type of cooperative behavior are identified, intentional and emergent. Intentional cooperation occurs through communication and negotiation. In an emergent approach agents cooperate through interaction and with no explicit communication or negotiation. The paper only considers intentional cooperation and does not propagate to the level of task execution. Korsah [7] deals with intentional cooperation where tasks are explicitly assigned to agents as a follow up to the taxonomy provided by [2] and provides a more complete taxonomy involving interrelated utilities. The new taxonomy, called iTax, addresses different levels of dependencies between tasks for different problems and breaks it down into four categories – no dependency, in-schedule dependency, cross schedule dependency, and complex dependency. In cross-schedule dependency the utility of a robot depends not only on its own schedule but also on the schedule of other robots and is relevant to the simulation experiments performed for this paper, although task execution is our main focus. The more recent paper by Nunes et al. [12] focuses on task allocation with an emphasis on temporal and ordering constraints, and extends the taxonomy to include different types of temporal and ordering constraints.

3 Experimental setup

3.1 Problem environment

Given the different approaches outlined in the related work section, the problem of cooperation can be broadly split into three categories of the environment which affects how much relevant information is available to the agents.

FOE which is the fully observable environment has all state information available to the agents, however, some of this information may not be relevant for learning the cooperative task.

PONOCOM is the partially observable environment with no communication between agents. Agents may not have all the relevant information available at their disposal for learning the cooperative task at hand.

POCOM is the partially observable environment with communication where state information may not be available but agents have the ability to send and receive discrete messages consisting of their location coordinates which is relevant to the task at hand. We want to investigate if agents under each of these conditions tasked with a novel problem will learn to cooperate.

The specific problem created to investigate these conditions is the "warehousing problem" which is a simulated environment motivated by a real-world warehouse where robots might be tasked to find an object and then cooperatively move it using the help of another agent. Fully observable environments by definition make all the state information available to the agents, but the system is not expected to be scalable. In a large environment the additional information that is not relevant to the learning task would potentially have a negative impact on learning because the state space would grow exponentially. Hence, given a large observation space we also want to investigate the scalability of the learning schemes in a fully observable model. We use neural networks with reinforcement learning for scalability. We also want to investigate the feasibility of training a neural network, since learning with a large state space requires additional parameters.

The second problem we wanted to explore is whether cooperation will emerge in a partially observable environment using the same warehouse problem. In this case the visibility of the agent is limited only to its current location, which ensures a very small observation space regardless of the size of the environment. The agent is also given information about the location of the load and that of the goal. No information about the location of the other agents is available in the partially observable version of the warehouse problem. We want to investigate if a partially observable environment with no communication is conducive to learning. Moreover, the same question of scalability concerning the environment and the neural network still hold in this scenario.

The third problem we wanted to address also involves a partially observable environment but where communication between the agents is enabled. Communication involves placing discrete messages into the environment that is then perceived by other agents. In a partially observable environment, agents may

not have all the information needed to learn the optimal action for each given state. Communication gives the agents the ability to share information that the other agents may not possess. Besides the question of whether communication will emerge, we also investigated if communication will allow the agents to learn more efficiently, specifically using a neural network.

3.2 Artificial Neural Network Architecture

The artificial neural network architecture used for the training assumes an input from the agent percepts at each time step, including the rewards received in the given state. There are two hidden layers in the network that are densely connected to each other with 30 nodes. The output of the neural network are the Q-values for each action and the action corresponding to the maximum Q-value is chosen as the learned optimal action. Two separate but identical neural networks were used for each of the learning phases, search where agents have to find the object and haul where agents have to cooperatively move the object. However, the search phase has one fewer output since the lift action is not required in the search phase. Figure 1 shows the reinforcement learning approach where the agents take actions in the environment impacting the state which is fed as input into the neural network which produces the Q-values as output.

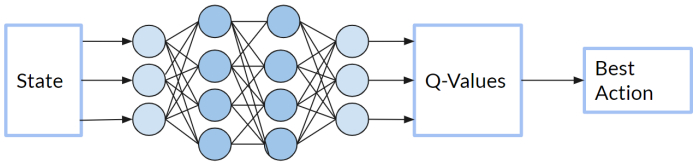


Fig. 1. Artificial Neural Network based reinforcement learning

3.3 Formal Specification of the Problem

More formally, we define our environment as a $n \times m$ grid with discrete space identifiable by an x and y coordinate. Agents $r_i, r_{i+1}, ..r_{i+j} \in R$ are randomly placed in this discrete space. Loads l_i, l_{i+1}, l_{i+k} are also available in the discrete space in a fixed location, along with a fixed goal location g. Loads are allocated to one or more agents. The task to be completed by agent r_i is to first learn the sequence of actions, $a_i \in A$ to move to the location of the load, l_i that the agent is assigned to. Loads can have a minimum threshold number of agents to be moved, typically 2, so assigned agents have to cooperatively move the load to target g. The action space A includes the four directional movements and lift. At each time step t, agents take an action. Moving the load requires a synchronous

series of actions by different robots, including the lift action. If agents move in different directions after lifting an object, they lose hold of the object and the object has to be lifted again for moving. The robots will likely learn an optimal, i.e., shortest, path from the load location to the goal, but optimality of the path is not a requirement, unlike what happens often in robot motion planning [9].

4 Experimental results

We conducted a number of experiments in the simulation environment. In the experiments, a varying number of agents are trained in a grid world to perform tasks in two phases: (1) search phase where agents find the load assigned to them and (2) haul phase where agents move the load to their destination with the help of another agent. The agents use an emergent approach to learn the behavior, since there is no direct communication or negotiation to induce cooperation. Reinforcement learning, more specifically Q-learning, was chosen to see if cooperative behavior would emerge. A multi-layered neural network was used to see if cooperative behavior would emerge.

The agents can execute an action $a \in up, down, left, right, lift$. The loads that are already on the grid have to be picked up using the lift action and synchronously moved to the target location. The loads are not allowed to move without the aid of agents and may require multiple agents for the lift to produce the intended effect on the object. Learning to cooperate between agents involves performing the lift operation and also learning to synchronize the move actions after picking up the load to cause the object to move in a specific direction.

At each step, the agents are either penalized or rewarded following the completion of an action. A collision with the wall does not change the state of the agent but penalizes more heavily (penalty is -5 for collision) compared to a non-rewarding step (penalty is -1). Agents can simultaneously occupy the same location in space so there is no penalty for agent collision. Successfully finding a load rewards the agents significantly (+50 reward) and moving the load to the target location results in the maximum possible reward (+100 reward). All motion are deterministic and the environment is fully observable in the first set of experiments. The series of experiments are then repeated in a partially observable environment with and without communication.

5 Experiments and Results for Fully Observable Environments

5.1 Results for 4 × 3 grid with a single agent and a single load

The first set of experiments with a single agent and a single load is a relaxed version of the problem, where the load is allowed to be lifted by a single agent. All subsequent experiments involve more than one agent and require more than one agent to move the load. Figure 2 shows the initial and final state.

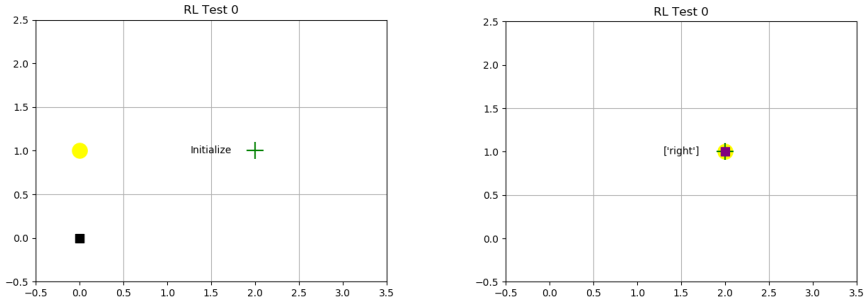


Fig. 2. Initial state and final state for a 4 × 3 grid, a single agent, and a single load

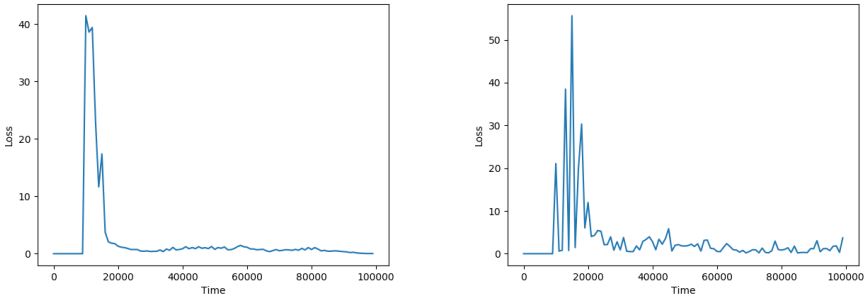


Fig. 3. Training on a 4 × 3 grid with a single agent and a single load, showing the loss over time steps. The left figure is for the search part and the right for the haul part

Figure 3 shows convergence within 200,000 epochs for the agent to learn the task. The agent successfully completes the task following the training and achieves the maximum possible reward, as seen in Figure 2, where the plus sign indicates the target location. We ran the experiments 10 times, each with different initial locations for the agents. This prevents the neural network to memorize the path of the agent from the initial location to the load.

The results were used to establish baseline numbers for the training time and to prove that a single agent can successfully learn to find the load, pick it up, and move it to the target location. The constraint for the load to require multiple agents was relaxed only for this experiment. Even for this relatively straightforward task involving a single agent, two separate neural networks were necessary, the first for the search task, where the agents find the load, and the second for the haul task, where the load is moved to the target location.

5.2 Results for 4 × 3 grid with two agents and a single load

Multi-agent cooperation was established in this phase where two agents were trained simultaneously, following an exploration phase of the grid. Similar to the

single agent training, the task of search and haul were separately handled using two distinct neural networks. The constraint for requiring at least two agents to handle the load was applied in this phase. The convergence was expected to be slower, because the probability of hitting the target is greatly reduced with two agents, so a larger training set was used. Training between a single agent and multi-agents differs essentially in the way data points are generated during the training session. Every step which occurs simultaneously generates two intermediary states that are fed into the agent percepts, and appear as a training data point within the neural network. An alternate solution might have been to simply train a single agent longer. However, the synchronization required for cooperation may not occur if agents are trained separately. The agents successfully completed the task following the training and achieved the maximum possible cumulative rewards.

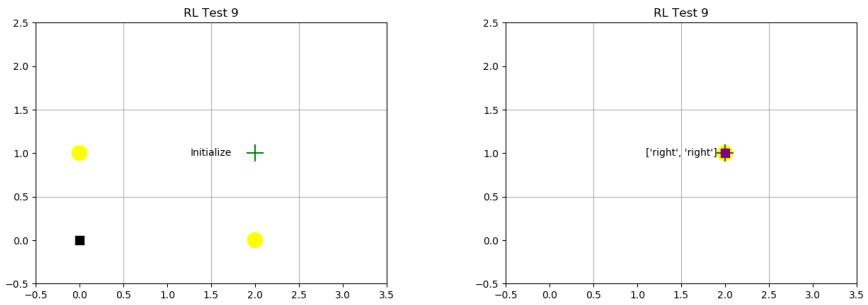
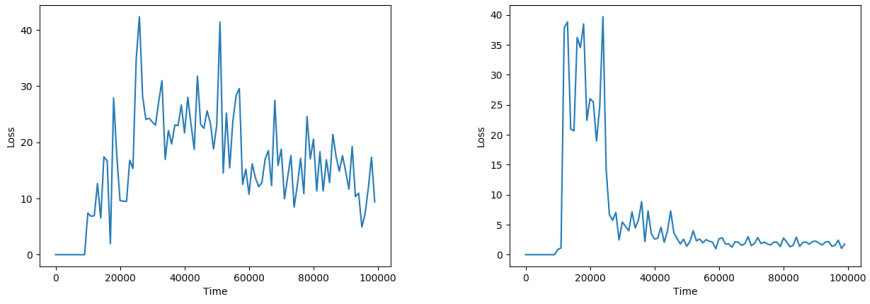


Fig. 4. Initial and final state for a 4 × 3 grid with two agents and a single load



!ht

Fig. 5. Training for the experiment with a 4 × 3 grid, two agents and a single load showing loss over time steps. The left figure is for the search part and the right figure is for the haul part

5.3 Results for 8 × 6 grid with 8 agents with multiple loads in pairs

Training for the 8 × 6 grid with 8 agents required the pre-allocation of loads to agents. This allocation remained fixed through the learning process. The agents find the loads they are assigned and move them to the target location indicated in Figure 6 by a green plus symbol.

With the expanded search space and more agents training had to be performed on a large amount of data to achieve convergence. However, in one case one of the agents failed to reach its target location in all the test runs. All the other agents successfully completed the task after the training and achieved the maximum possible cumulative rewards.

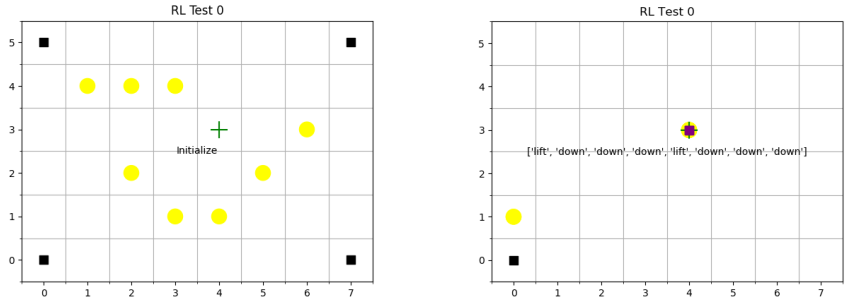


Fig. 6. Initial and final state for a 8 × 6 grid world with 8 agents and 4 loads

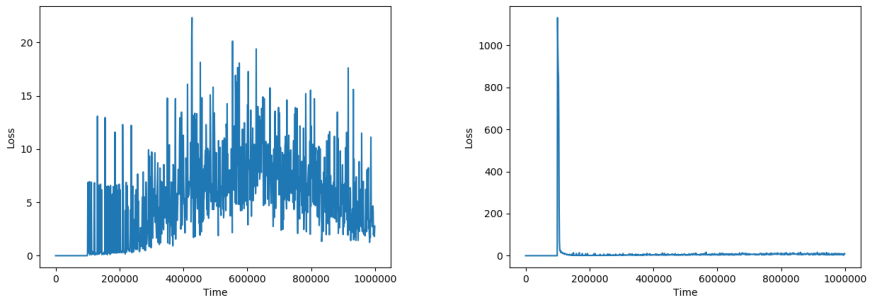


Fig. 7. Training for the experiment with a 8 × 6 grid, with 8 agents and 4 loads. The left figure is for the search part and the right for the haul part

6 Experiments for Partially Observable Environments

6.1 Results for 4 × 3 partially observable grid with 2 agents and a single load

The same experiment was repeated in a 4 × 3 grid in a partially observable environment, where agent perception was limited to its own location, the goal location and the load location. No information about other agents was made available in the percept sequence.

The neural network architecture used was similar to the one for the previous experiments, with one network for the search and a second one for the haul task. However, fewer layers were needed to achieve learning. Since, the number of input parameters were reduced, the neural network had fewer overall parameters and hence could be trained more efficiently compared to the fully observable model. After 100, 000 iterations the agents learned to do the task of finding the object and cooperatively moving it to the goal location. In Figure 8, the initial state and final state are shown in the test run following training. Figure 9 shows that training was achieved within just 100,000 iterations.

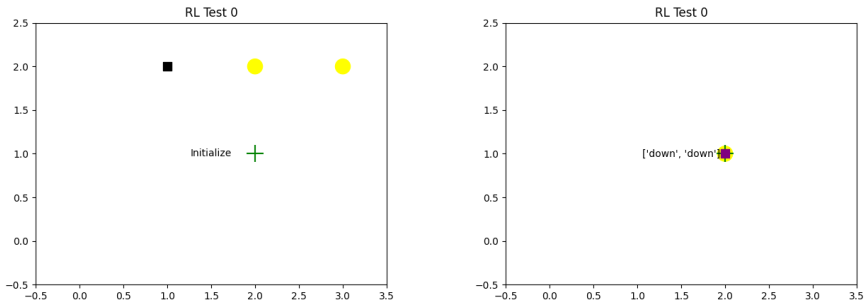


Fig. 8. Initial and final state for a 4 × 3 grid with two agents and a single load in a partially observable environment with no communication

6.2 Results for 5 × 5 partially observable grid with 2 agents and a single load

In a 5 × 5 grid with a partially observable environment, the agents also learned to find the object and cooperatively move it to the goal location. However, the training time doubled compared to the 4 × 3 case for the learning to take place. This is expected because the environment size is about twice as large in terms of the number of discrete locations compared to the 4 × 3 case. Figure 10 shows the initial and final states from the test run following the training of 200, 000 epochs shown in Figure 11.

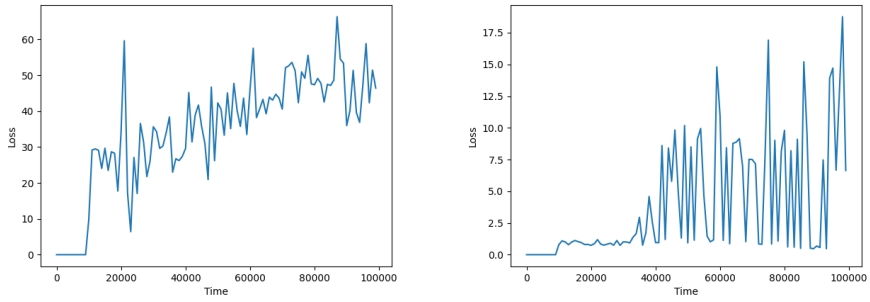


Fig. 9. Training for a 4 × 3 grid, two agents, and a single load showing loss over time steps. The left figure is for the search part and the right for the haul part

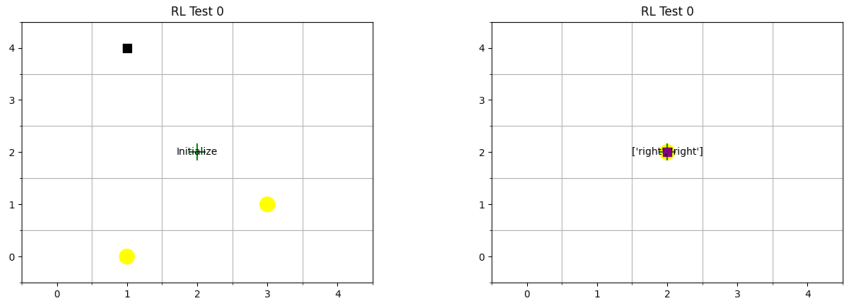


Fig. 10. Initial and final state for a 5 × 5 grid with two agents and a single load in a partially observable environment with no communication

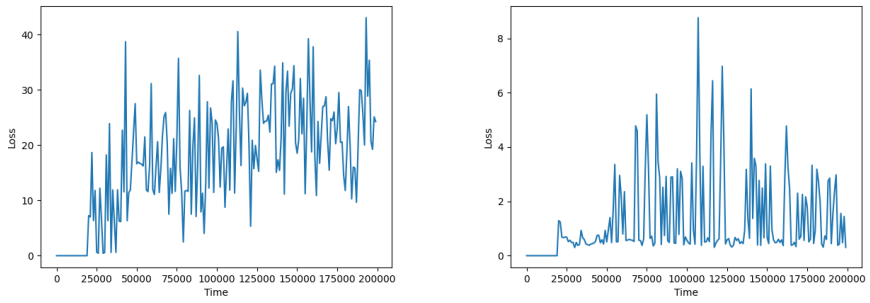


Fig. 11. Training on a 5 × 5 grid, two agents, and a single load showing loss over time steps. The left figure is for the search part and the right for the haul part

6.3 Results for 8 × 6 partially observable grid with 2 agents and a single load

Cooperation was not achieved in the partially observable case of the 8 × 6 grid as seen in Figure 12. The training time was increased to 1,000,000 epochs which

is shown in Figure 13. Despite a 5-fold increase over the 4×3 case the agents failed to complete the tasks in the test runs. The rewards achieved were also much smaller than in the optimal case but still better than a random test run.

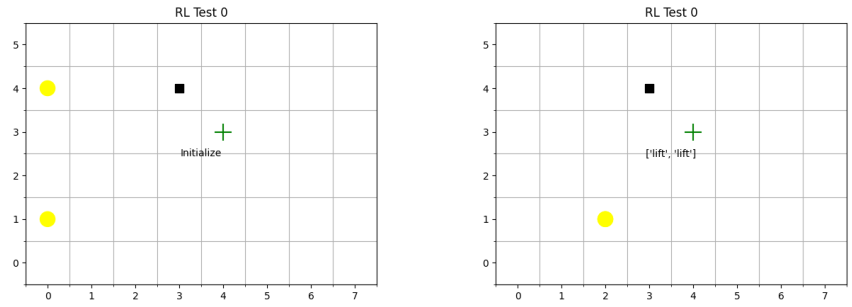


Fig. 12. Initial and final state for a 8×6 grid with two agents with a single load in a partially observable environment with no communication

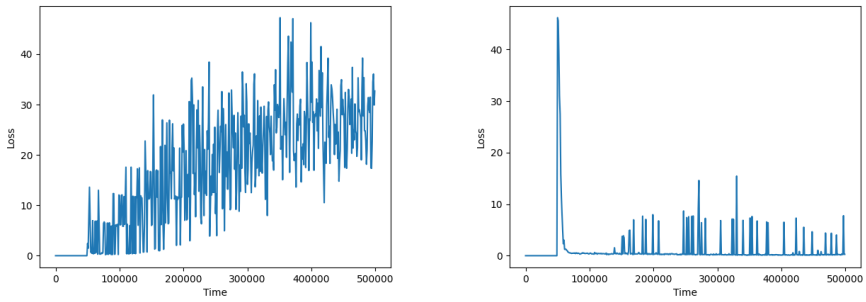


Fig. 13. Training for 8×6 grid, two agents, and a single load showing loss over time steps. The left figure is for the search part and the right for the haul part

7 Experiments and Results for Partially Observable Environments with communication

7.1 Results for 4×3 partially observable grid with 2 agents and a single load and with communication

The series of experiments from the partially observable model were repeated using communication. In the first experiment with the 4×3 grid the agent

perception was just limited to the goal location and obstacle location. However, agents were allowed to send and receive messages from other agents. The neural network architecture used was the same as the partially observable case with two separate networks for the search and for the haul phase. The number of input parameters was increased since the communication messages were also included in the training. Despite the increase in the overall training parameters, after $200,000$ epoch, as seen in Figure 15, the agents learned to accomplish the task of finding the object and cooperatively moving it to the goal location. The initial and final location are seen in Figure 14.

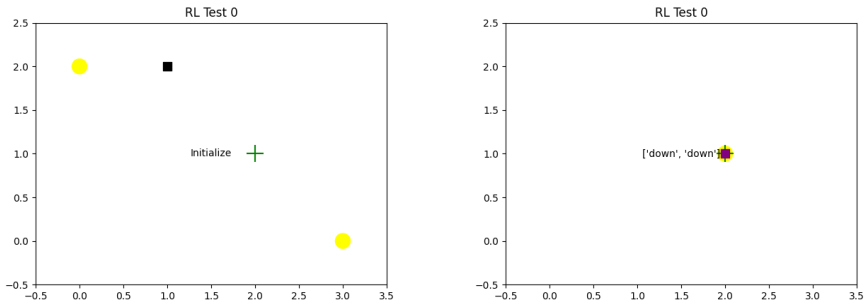


Fig. 14. Initial and final state for a 4×3 grid with two agents and a single load in a partially observable environment with communication

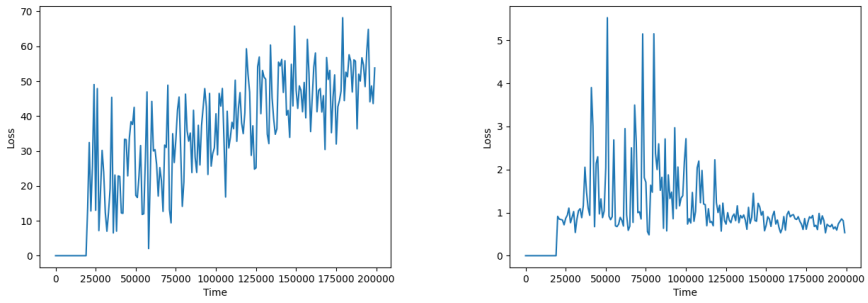


Fig. 15. Training for the experiment with a 4×3 grid, two agents, and a single load showing loss over time steps. The left figure is for the search part and the right figure is for the haul part

7.2 Results for 5 × 5 partially observable grid with 2 agents and a single load and with communication

In a 5×5 grid with a partially observable environment and with communication, the agents also learned to find the object and cooperatively move it to the goal location, as seen in Figure 16. However, the training time shown in Figure 17 was significantly greater than for the partially observable case with no communication. This was likely due to the increase in the number of input parameters to the neural network.

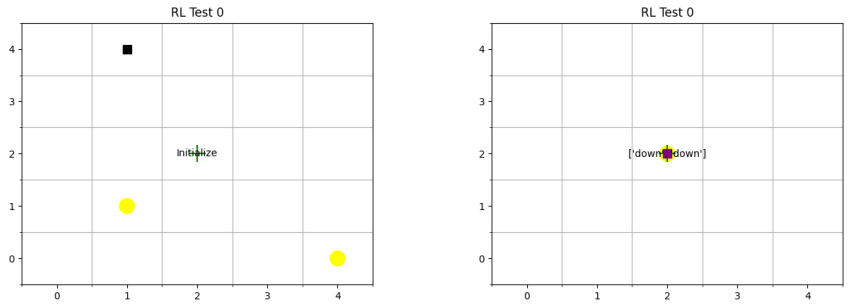


Fig. 16. Initial state and final state for a 5 × 5 grid with two agents that have to move a single load for the partial observable environment with communication

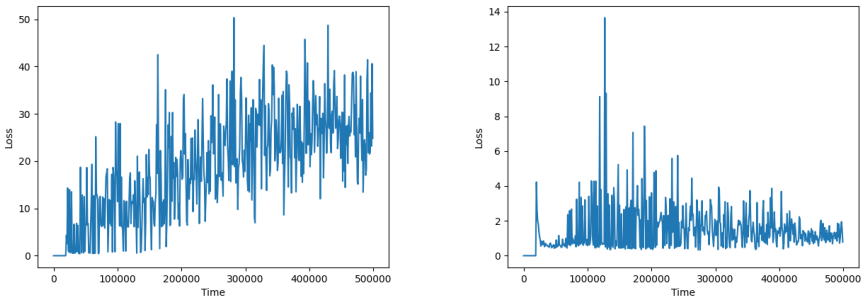


Fig. 17. Training for the experiment with a 5 × 5 grid, two agents, and a single load showing loss over time steps. The left figure is for the search part and the right figure is for the haul part

7.3 Results for 8×6 partially observable with 2 agents and a single load and with communication

Learning was successfully achieved in the partially observable case with communication with the larger grid size of 8×6. However, the training time increased to about $1,000,000$ epochs. The rewards achieved were smaller compared to the previous cases but still close to the expected maximum reward achievable. This is because since even the optimal policy would take more steps in a larger grid space. Figure 18 shows the initial and final state of the test runs for the 8×6 case. The increased training time as seen in Figure 19 is comparable to the training time seen for the partially observable case with no communication.

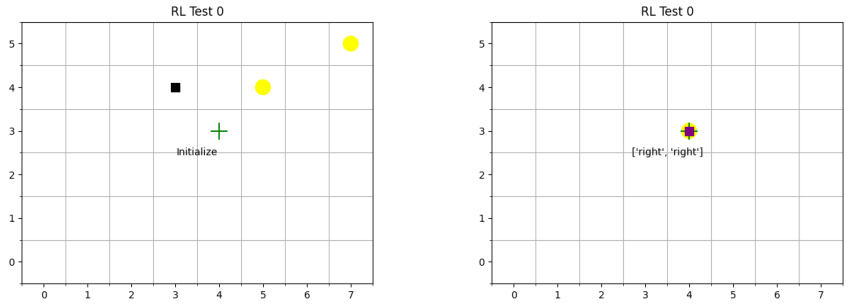


Fig. 18. Initial state and final state for a 8×6 grid with two agents that have to move a single load for the partially observable environment with communication

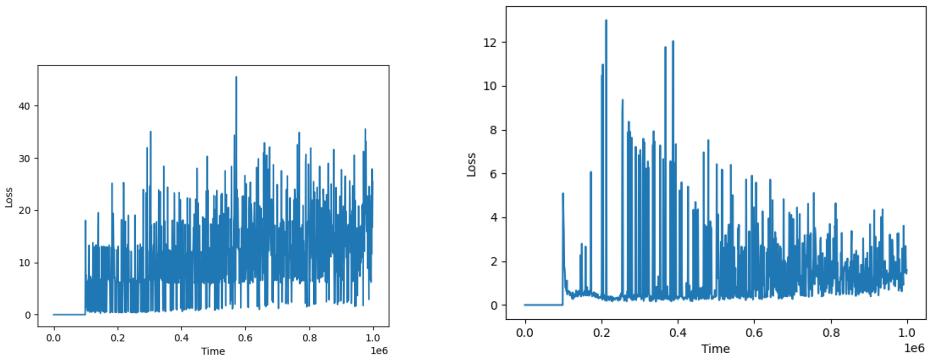


Fig. 19. Training for the experiment with a 8×6 grid, two agents, and a single load. The left figure is for the search part and the right figure is for the haul part

8 Analysis

The series of experiments in the three different cases of fully observable, partially observable with no communication, and partially observable with communication, show that cooperation is achieved in the case of the 8×6 partially observable environment. In all the other cases the agents learned the optimal set of actions in both the search phase and the haul phase. Of particular importance is that the agents learned to synchronize their actions to haul the object at each step exhibiting cooperation. In the specific case of the 8×6 fully observable environment, one set of agents consistently failed to move the load to the goal location whereas the other set of agents succeeded in all the test runs. Cooperation did not emerge in the two cases with no communication between agents but with communication we see that it is possible for cooperation to emerge even in a large environment. This seems to indicate that communication is necessary for cooperation in large partially observable environments. The training time across all the experiments were consistently kept below $1,000,000$ time steps. It appears that the slightly better performance with the larger grid in the partially observable environment with communication can be attributed to the agent's ability to share information. However, more experiments would be needed to establish this conclusively, in particular with even larger environments. Although the series of experiments we performed does not provide conclusive evidence, it motivates further work in this area, particularly on the observation space and information that is shared between agents.

9 Conclusions and Future Work

We have shown that the agents successfully achieved the desired cooperative behavior by working as a team to accomplish a complex task which included two subtasks. Although two separate neural networks were needed to learn each phase of the task, we have shown that complex tasks that can be broken down into simpler subtasks and there is potential for emergent cooperation, that is, without explicitly teaching the agents to cooperate, agents can learn to cooperate.

The Q-learning algorithm that was used proved to be effective and maximized the utility values in all the experiments performed in the fully observable environment. In all cases, the learning agents outperformed a random agent, including the case where a set of agents failed to learn the overall task. Similarly, in the partially observable cases all agents succeeded in learning the task, except in the case of the larger grid. However, this was remedied by allowing the agents to communicate which made the necessary information available for learning to occur. Given the limitation of learning in a fully observable environment and a partially observable environment we can conclude that agents with the ability to communicate will generally outperform those with no means of communication.

The work presented in this paper provides several avenues to investigate the role of communication in a partially observable environment. Exploring even larger grid sizes could conclusively determine the limitation of discrete environments and provide insight on the feasibility of training in such an environment

using neural networks. Additional experiments with more variations on the information available to the agents can further clarify the role of relevant information when it comes to learning. As an example, during the simulation runs, agents never learned when critical pieces of information were suppressed, such as the location of the goal. Investigating further into these variations would provide more insight on the impact on cooperation when critical pieces of relevant information is not available as part of the agent percept.

References

1. Foerster, J., Assael, I.A., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 2137–2145 (2016)
2. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. International Journal of Robotics Research **23**(9), 939–954 (2004)
3. Gopalan, N., Rosen, E., Konidaris, G., Tellex, S.: Simultaneously learning transferable symbols and language groundings from perceptual data for instruction following. In: Robotics: Science and Systems (2020), http://www.roboticsproceedings.org/rss16/p102.pdf
4. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: Proceedings International Conference on Autonomous Agents and Multiagent Systems. pp. 66–83 (2017)
5. Kasai, T., Tenmoto, H., Kamiya, A.: Learning of communication codes in multi-agent reinforcement learning problem. In: Proceedings IEEE Conference on Soft Computing in Industrial Applications. pp. 1–6. IEEE (2008)
6. Khan, N.: Learning to cooperate using deep reinforcement learning in a multi-agent system. Master's thesis, University of Minnesota (2020)
7. Korsah, G.A., Stentz, A., Dias, M.B.: A comprehensive taxonomy for multi-robot task allocation. International Journal of Robotics Research **32**(12), 1495–1512 (2013)
8. Lan, N.G., Chemla, E., Steinert-Threlkeld, S.: On the spontaneous emergence of discrete and compositional signals. arXiv:2005.00110v1 [cs.CL] (2020)
9. LaValle, S.M.: Planning Algorithms. http://planning.cs.uiuc.edu/ (2006)
10. Mahadevan, S., Connell, J.: Automatic programming of behavior-based robots using reinforcement learning. Artificial Intelligence **55**(2-3), 311–365 (1992)
11. Mirsky, R., Macke, W., Wang, A., Yedidsion, H., Stone, P.: A penny for your thoughts: The value of communication in ad hoc teamwork. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence. pp. 254–260 (2020)
12. Nunes, E., Manner, M., Mitiche, H., Gini, M.: A taxonomy for task allocation problems with temporal and ordering constraints. Robotics and Autonomous Systems **90**, 55–70 (Apr 2017)
13. Stone, P., Kaminka, G.A., Kraus, S., Rosenschein, J.S., et al.: Ad hoc autonomous agent teams: Collaboration without pre-coordination. In: Proceedings AAAI Conference on Artificial Intelligence (2010)
14. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning. pp. 330–337 (1993)