AI for Minecraft

Shaohong Tian 827049, Jiahao Yu 781768, Guo Zhou 791698, Ziwei Li 7996884
shaohongt@student.unimelb.edu.au, jiahaoy@student.unimelb.edu.au,
zhoug2@student.unimelb.edu.au, ziweil6@student.unimelb.edu.au

Abstract

Minecraft is a sandbox game published by Mojang. It allows player to build with a variety of cubes in a 3D procedurally generated world. The Malmo platform is a sophisticated AI experimentation platform build on top of Minecraft. It was established by Microsoft on June 1, 2015. The purpose of Malmo is hope that using this platform can address some core research challenges in artificial intelligence. Therefore, in our final group project, We decided to make use of this platform to reproduce a famous reinforcement learning algorithm with more complex scenarios, and see if it still make sense. Our project includes two agents, which bring much more challenges but also more possibilities with agents' corporation. The two agents need to collect all the resources in the map, and then, either of them get into the final point. Based on this environment, we make two hypotheses base on our project. The first one is the performance of Double Q-learning is better than that of Q-learning, the performance here mainly concerns the distribution of Q values and the rate of convergence. Another proposal we have is that working with multiple agents, the performance of the dynamic programming is better than greedy algorithm while in resource allocation plan. The definition of performance here concerns about least training times to achieve convergence and the shortest path to goal.

1 Introduction

1.1 Project Overview

The game of Minecraft does not have exactly scenario for our project, but Minecraft supports players to build and destroy as their wish. In this project, our group set a new scenario that suits our experiment, containing resources and final points. Besides that, in order to introduce more possibilities and capabilities that we can achieve with this scenario, we set up two agents for this project. As for the future works, it can have more agents as needed. In this scenario, at first two agents cooperate with each other and collect all the resources in the map. After that, one of them achieve the final point, that is the ultimate goal. Our built scenario in Minecraft is shown above in figure 1. The floating objects are resources, and the beam shows the final point, and two men are our agents.

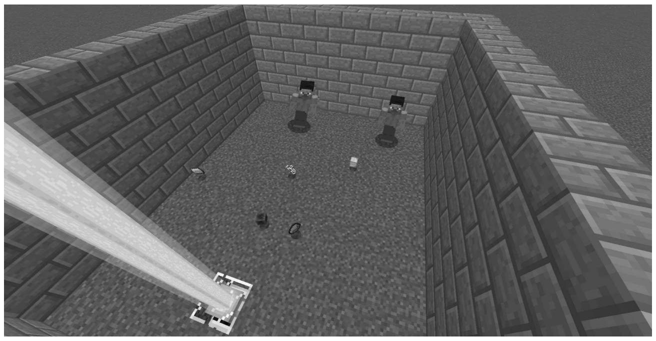


Figure 1: Scinario

1.2 Research Questions

The first proposal we raise is that the performance of double Q learning is better than that of Q learning. In order to compare the performance, the evaluation mainly focus on distribution of Q values and convergent rate. As we mentioned before, our group set two agents to work cooperatively. So the following question is how does these two agents allocate their jobs? Should they been allocated greedy at the beginning of the programming or they can do dynamic programming. We set our second proposal that in the project with multiple agents, the performance of the Dynamic programming is better than greedy algorithm used in resource allocation plan.

1.3 High-level Description

The practical significance of our research can spread to many real life scenarios. As we mentioned before, we can support more than two agents for future improvement. The allocation method can be used for some emergency situation, for example if there is a patient need ambulance, which hospital should be the better choice? This situation is like our resources should be collected by which agents. And it is really useful and important for our lives.

Some experiments have been done for the first proposal. Setting little resources as easy degree, medium resources as normal degree and plenty of food as difficult degree. And the conclusion we got is that as the difficult level increase, the performance of double Q learning is much better than Q learning. Besides that, we also set three difficult degree in experiments for our second proposal. By comparing the paths they have been allocated, we got the conclusion that dynamic programming can use less steps to achieve the goal.

2 Related Work

Lots of efforts have been made while performing the study. A clearly understanding of Minecraft utilization and a suitable environment setting for a computer is the fundamental step of this project. The most attractive and meaningful part of this project is the consideration of the practical use and the reality of this work. All of our group members discussed and brain storm about what we should study in this final project. At last, we decide to start from the basic algorithm, which is also the most popular one – Q-leraning. By studying the weakpoint of Q-learning which is overestimation, we can understand reinforcement learning algorithm much deeper. And implementing double Q-learning can help us understand the improvement it made. Ziwei Li and Shaohong Tian read lots of paper explaining the reinforcement learning algorithm, and the equations and ideas of Q-learning and why overestimate happens. Besides that, dynamic programming is also popular and highly proffered while programming. Thus we decide to simulate and study the difference between dynamic programming and greedy approach. By making use of Minecraft platform, we can intuitively observe the comparison of these approaches. Jiahao Yu and Guo Zhou first searched for many papers that related to these approach. After that they build proper scenarios in Minecraft that can test our hypothesis. We believed that combining task allocation algorithms such as dynamic approaches and greedy approach and so on, learning process can be much more efficiently and powerful.

3 The First Proposal

3.1 Methods

Ziwei Li and Shaohong Tian worked on this research question. They searched for many papers that related to these two reinforcement learning algorithms, and then set up the scenario that suits the experiment needs. Moreover, the coding are also be done by them and finally evaluate the result and draw a conclusion.

3.1.1 Introduction

In this s section, the first proposal will be discussed which is "The performance of double q-learning is better than the performance of q-learning". Before the discussion of this proposal, let's review the concept of q-learning. Reinforcement Learning [Sutton and Barto, 1998] intents to learn sequential actions that maximise the cumulative future rewards under given tasks. Q-learning is [Watkins, 1989] one of the most popular reinforcement learning algorithm. But the performance can be poor in stochastic MDPs because the overestimation of the q-learning actions values [Van Hasselt *et al.*, 2016]. If all actions values are all overestimate, that is all values are uniformly higher than realistic values, the performance might not be worse. However, the overestimation are not uniform and usually not focus on the states that we wish to learned more [Berkhemer *et al.*, 2015]. To address the overestimation problem in Q-learning, Hado van Hasselt proposed a new algorithm based on q-learning, that is double Q-learning.

3.1.2 Double Q-learning

Double Q-learning adopt the main ideas of Q-learning, that is select the action that can maximise the cumulative future rewards. But instead of one Q-table, double Q-learning like its name says, have two Q-tables. Both table consist of q-values of all states in the environment. To implement double Q-learning, first randomly choose one table to select the action, and then use the other table to calculate and update the value. The equation of update in Q-learning is shown as equation (1) and in double Q-learning is shown as equation (2) and (3)

$$Q(s,a) = Q(s,a) + \alpha(s,a)(r + \gamma Q(s', a*) - Q(s,a)) \quad (1)$$

$$Q^A(s,a) = Q^A(s,a) + \alpha(s,a)(r + \gamma Q^B(s', a*) - Q^A(s,a)) \quad (2)$$

$$Q^B(s,a) = Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s', a*) - Q^B(s,a)) \quad (3)$$

As the equation displayed, Q-learning use the only q-table for select and update the value, while for double Q-learning, let's assume there are table A and table B. If table A is used for select action, then during value update process, table B is used; for same reason, if table B is used for select action, then table A will participate the value update process. Two tables are cooperated together with each other in double Q-learning. We randomly choose table A or table B to decide the action we use, and use the another one to do the calculation part while Q-learning just use its own value to update the Q-value. By this way, the probability of both two tables are overestimate on the same action is quite low. In this situation, we ca solve the problem of overestimation.

3.1.3 Experiments Description

To verify this proposal, we set up several experiments. First, we use Minecraft to set up a map which size is 10*10. For the number of resources, we made several levels—easy, normal, hard. The flat view of these maps are shown below in figure 2. As we can see, for these difficulty level, we use the same size of the map, but the number of resources are increasing from easy to hard. We also can tell that the distribution of these resources becomes more spread and dispersed. In the easy map, all resource lies on the center of the map. And in the normal map, resources start to spread on the edge of the map. As for hard map, some resource shows up at more tricky positions (for example the back of the final goal). We believe that the more numbers and more spread the resource are, the more difficult for the agent to train and also more possible to have overestimation.

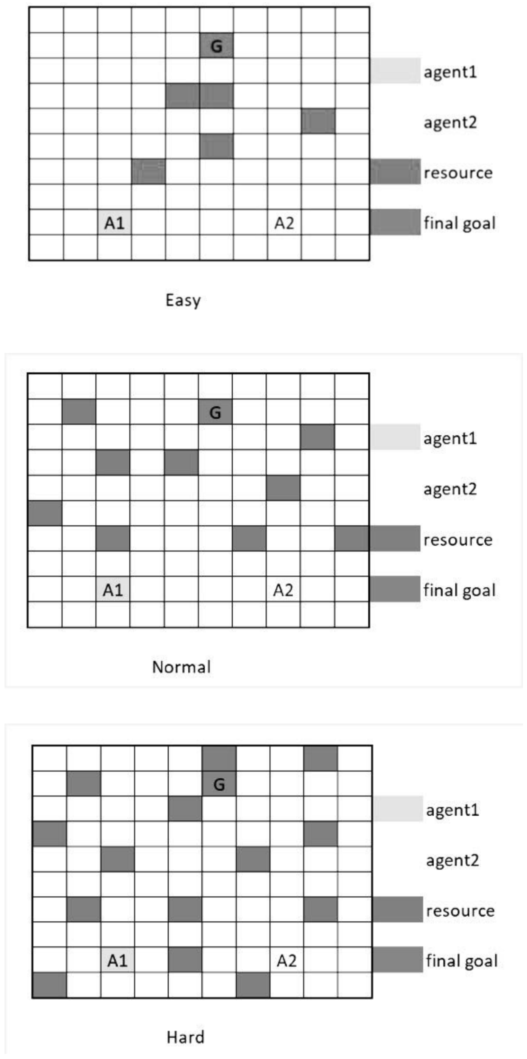


Figure 2: different maps

3.2 Experimental Evaluation

The goal of our experiment is to compare the performance of running several maps with Q-learning algorithm and Double Q-learning algorithm. And by comparing the convergent time and the values in q-table and also their distribution. The design diagram for our experiment is shown in figure 3.

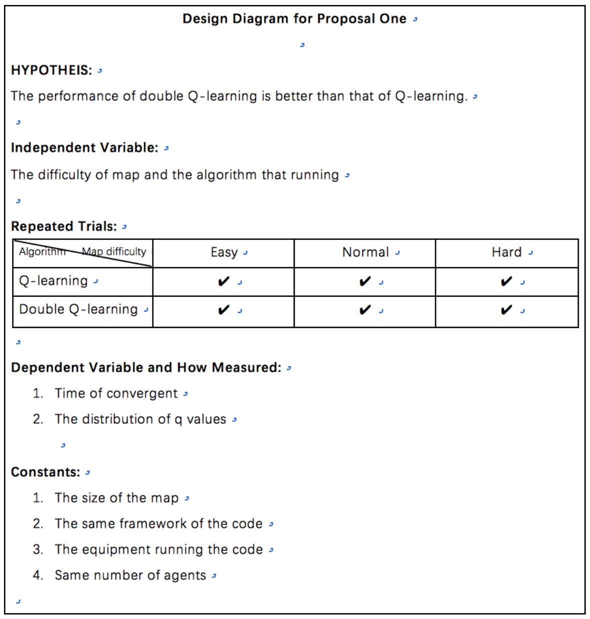
3.3 result

After running the experiments that design diagram shows, we collect many data, and we will discuss them under two main evaluations, q-values and convergent rate.

3.3.1 Q-values

According to the data from all experiment, we draw a q-value diagram of each algorithm running on different maps. The diagrams are shown in figure 4 and figure 5.

The three pictures from figure 4 are from Q-learning algorithm running on Easy, Normal and Hard maps respectively (from top to bottom). And in figure 5 shows table A and



Design Diagram for Proposal One

HYPOTHEIS:

The performance of double Q-learning is better than that of Q-learning.

Independent Variable:

The difficulty of map and the algorithm that running

Repeated Trials:

Algorithm / Map difficulty	Easy	Normal	Hard
Q-learning	✔	✔	✔
Double Q-learning	✔	✔	✔

Dependent Variable and How Measured:

1. Time of convergent
2. The distribution of q values

Constants:

1. The size of the map
2. The same framework of the code
3. The equipment running the code
4. Same number of agents

Figure 3: Proposal 1

table B (on the same row) with different map difficult level (from top to bottom is easy to hard respectively). As diagrams shown, most of the q-values are positive, but still there are some negative and zero reward. The zeros are because the certain states have not been visit. And the negative values can be caused by unintended accident like program crash, or stuck in bugs (for example, suffocate in a wall).

From the comparison of q-values between Q-learning and double Q-learning (either one in figure 5 in same row), we can clearly observe that the values are much lower in double Q-learning than that of Q-learning. That's because double Q-learning spread the values in two tables, and also reduce the times that visit the states. Moreover, if we look close at the diagrams, especially on normal and hard difficulty level maps, we can see that some unnecessary values in Q-learning diagrams are being decreased or totally eliminated. This because that using double Q-learning can prevent or reduce the times the agent from visiting somewhere not useful. The other conclusion that can discovered from these diagrams is that the two diagrams from double Q-learning values are mostly the same, that's because we think that after implementing double Q-learning, the two tables should be similar, that is whichever table we choose to use in actually scenario should be the same. So we tried a series of probabilities of selecting these two tables when training, and we figure that when the probability set to 0.5, we got two very similar Q-table. And that is the reason the tables shown below in double Q-learning are almost the same.

To be brief, the evaluation of Q-values can shows that using double Q-learning can have better Q-value distribution than using Q-learning.

3.3.2 Convergent Rate

After training for several times on each map, we calculate the average convergent times for each map running Q-learning

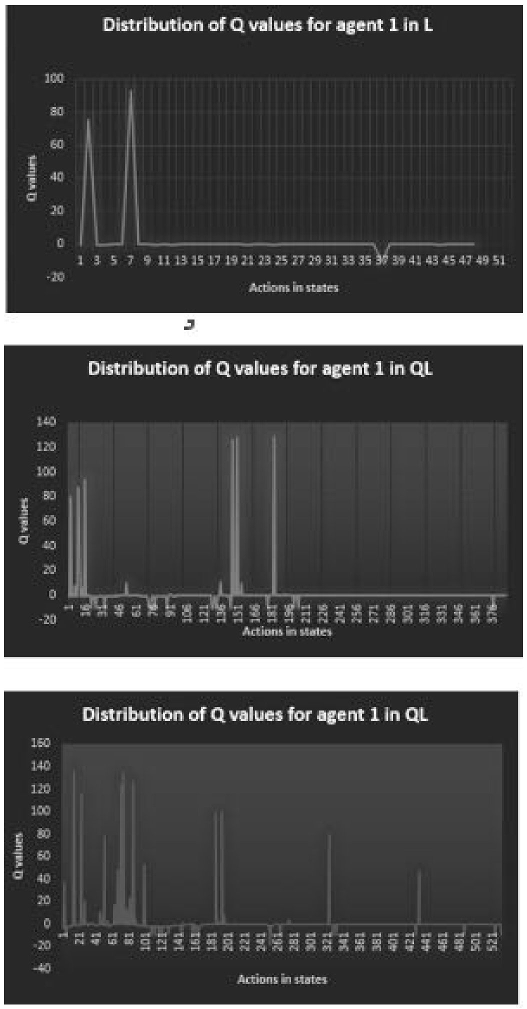


Figure 4: Q Value in QL

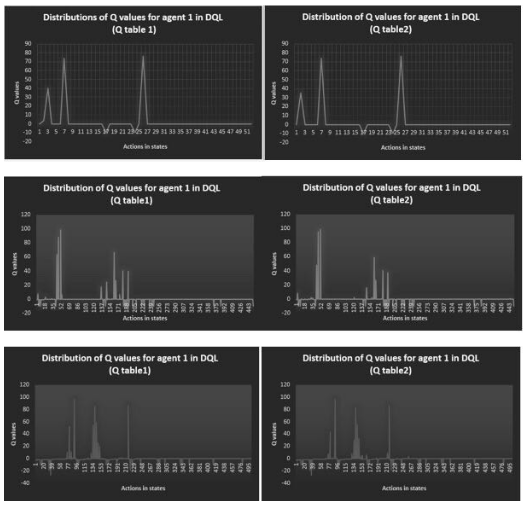


Figure 5: Q Value in DQL

and double Q-learning, we record the data in the table 1 as shown below.

	Q-learning	Double Q-learning
Easy	48.1times	69.5times
Normal	186.1times	227.3times
Hard	578.02times	654.7times

Table 1: Convergent Rate

As we can see from the form, the convergent time for double Q-learning are slightly longer than using Q-learning. The reason why that happens might cause by two reasons as for our analysis. First is that in double Q-learning we have to update two Q-tables, that might slow down the training process. Second, we think the most important reason is that those maps we are design are not complicated enough, thus although we can see the improvement from Q-value diagrams, but the overestimate problem still not obvious enough to affect the convergent rate of using Q-learning.

4 The Second Proposal

Jiahao Yu and Guo Zhou work on this research problem. They first read related papers about three different approaches towards allocation. Then they set up a series of experiments and finally discuss about the result and make a conclusion about it.

4.1 Introduction

We have already set such a scenario that two agents collaborate with each other to get all resources first and then reach the final goal. This means there will be a task allocation for these two agents when training these two agents to finish this scenario. In other words, which agent should be responsible for getting which resources, which agent should be responsible for reaching the final goal. By the way, all available actions for agent are “move east, move west, move north, move south”. When the agent passes the resource point, it can be thought that the agent gets this resource point. To obtain useful and effective methods or algorithms for task allocation, we searched a large amount of related information online and read several related papers. As a result, we got three methods of task allocation for these two agents. The first one is that before two agents beginning taking actions, which agent should get which resources and which agent should go for the final goal is decided based on the Manhattan distance from agents to resources. The second one is that before two agents taking actions, the task allocation is decided based on the greedy algorithm which will be described in detail later. The last one is that before or when two agents taking actions, the task allocation is decided based on the dynamic planning which will be explained later as well. The task allocation is constant in the first two methods, while in the last method, it is variational. Therefore, our second research question is that among these three methods, which one performs the best or worst in terms of the convergence rate of Q table and the final time cost to finish the scenario (**“the final time cost”** means after the training process for these two agents is finished successfully, the time used by these two agents to get all resources first and then reach the final goal from the starting positions) when using Q learning to train these two agents. Based on our prior knowledge on greedy algorithms[Slavík, 1995] and dynamic planning[Baxter, 2009], we made a hypothesis that in terms of convergence rate of Q table, greedy algorithm is better than the method just based on Manhattan distance which is better dynamic planning; in terms of final time cost, dynamic planning is better than greedy planning which is better than the method just based on Manhattan distance. More algorithm details about these three methods for task allocation will be introduced in the following.

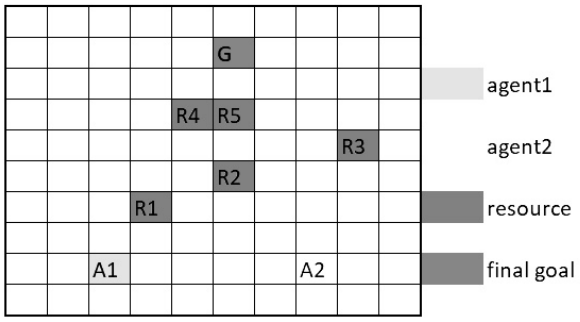


Figure 6: A map used in this scenario

4.2 Methods and Algorithms

4.2.1 The method just based on the Manhattan distance

Taking the figure 6 as an example to describe how to implement the task allocation and how to set rewards for these resource points and final goal point just based on Manhattan distance. It can be seen from figure 6 that resources R1, R4 are closer to agent 1 and resources R2, R3, R5 are closer to agent 2. So, agent 1 is responsible for getting resources R1, R4 and agent 2 is responsible for getting resources R2, R3, R5. It is easy to know that the agent will have higher probability to get the resource point nearer to it if the agent can take action randomly. So, to make the agent can get all resource points which are allocated to this agent in the process of training, the rewards of these resource points will be set lager with the increase of their Manhattan distance to this agent. The rule of setting rewards based on Manhattan distance should satisfy the following formula:

$$\text{Reward}_1 * \mathbf{C}_{\text{Mdis1}}^{\text{Vdis1}} * \left(\frac{1}{4}\right)^{\text{Mdis1}} = \text{Reward}_2 * \mathbf{C}_{\text{Mdis2}}^{\text{Vdis2}} * \left(\frac{1}{4}\right)^{\text{Mdis2}}$$

Reward1 is the reward set for one resource point, Vdis1 and Mdis1 is the vertical distance and Manhattan distance respectively from this resource point to it's owner agent ("owner agent" means that the agent is responsible to get the resource point, For example, A1 is the owner agent of R1, R4 and A2 is the owner agent R2, R3, R5). Reward2, Vdis2, Mdis2 means the same to the other resource point. ***C*** *is the symbol of permutation.*

In figure 6, the Manhattan distance from A1 to R1 is 3, from A1 to R4 is 7; the vertical distance from A1 to R1 is 2, from A1 to R4 is 5. Set the reward for R1 as 10, the reward for R4 should be set as 183 based on the becasue:

$$10 * \mathbf{C}_3^2 * \left(\frac{1}{4}\right)^3 = 183 * \mathbf{C}_7^5 * \left(\frac{1}{4}\right)^7$$

There are other four points should be known. The first one is that the task allocation and reward setting for resource point is finished before the whole process of training. The second point is that in each episode of training, the reward for the final goal is set as a positive value, for example, 100 just when all resource points are gotten. The third point is that the reward of a resource point just makes sense to it's owner agent, for example, the reward 10 for R1 just makes sense to agent1 while the reward for R1 is 0 to agent2. The last point is that rewards for these points which are not resource point or final goal point are defaults 0.

4.2.2 The method based on greedy algorithm

What will be talked about in this section is almost the same to the section 4.2.1 just except the rule for deciding which agents should be responsible for getting which agents and the rule for reward setting for resource points. Again, take the figure 6 as an example. It is obvious that the closest resource point to agent1 is R1 and the closest resource to agent2 is R2. Because the Manhattan distance from A1 to R1 is 3 which is smaller than the Manhattan distance from A2 to R2 which is 5, resource point R1 is allocated to agent1 while the allocation for other resource points remains to be done. Then assume that agent1 is on the position of R1 but the agent2 is still on it's starting position, in this case, the closest resource point to agent1 is R2, and the closest resource point to agent2 is R2 as well. Because the Manhattan distance from A1 to R1 plus the Manhattan distance from R1 to R2 are larger the Manhattan distance from A2 to R2 (A1R1+R1R2>A2R2, 3+3>5), R2 is allocated to agent2 this time. Continue to assume that agent2 is on the position of R2 and agent1 is still on the position of R1, in this case, the closest resource point to agent1 is R4 and the closest resource point to agent2 is R5. Because the Manhattan distance from R1 to R4 equals to the Manhattan distance from A2 to R2 plus the Manhattan distance from R2 to R5 (A1R1+R1R4=A2R2+R2R5, 3+4=5+2), R4 is allocated to agent1 and meanwhile R5 is allocated to agent2. Based on what was described above, assume now that agent1 is on the position of R4 and agent2 is on the position of R5. Obviously, R3 is the closest resource point to both agent1 and agent2 now. But the Manhattan distance A1R1 plus R1R4 plus R4R3 is larger than the Manhattan distance A2R2 plus R2R5 plus R5R3 (A1R1+R1R4 +R4R3>A2R2+R2R5+R5R3, 3+4+5>5+2+4), so R3 is allocated to agent2. In short, R1, R4 are allocated to agent1 and R2, R5, R3 are allocated to agent2 sequentially. Accordingly, the reward for R1, R4 and R2, R3, R5 will be set nonzero value in the same sequence. In particular, before each episode of training, just the reward of R1 and reward of R2 are set as positive values, for example 100, while the rewards for other resources points are set as 0. In the process of each episode of training, when the resource points whose rewards have already been set positive rewards are gotten, rewards for these gotten resource points will be reset as 0, and reward for the latter allocated resource point will be set as a positive value just when all former allocated resource points are gotten. For example, in the process of an episode training, when the R2 is gotten, immediately reward for R2 will be set as 0 and the reward for R5 will be set as 100. When the R2, R5 are gotten successively, the reward for R5 will be set as zero and the reward for R3 will be set 100, the reward of R2 still keeps 0 after the time when R2 is gotten. As for these following questions: rewards for non-resource and non-goal points; how and when to set reward for the final goal; the reward of a resource point just makes sense to it's owner agent. Just refer to 4.2.1

4.2.3 The method based on dynamic planning

This method just has one difference compared with the first method (described in 4.2.1). The difference is that: in the first method, the task allocation and reward setting is finished and then fixed before the whole process of training; while in this method, even though the task allocation and reward settings will be done the same as the first method does before each episode of training, the task allocation and reward setting will be reset dynamically based on the agents' current position. Take figure 6 as an example as well to explain how to do the dynamic planning. Before each episode of training, R1, R4 are allocated to agent1 and R2, R3, R5 are allocated to agent2 based on what was described in 4.2.1, and rewards for these resource points are set on the same way of the first method. In the process of each episode of training, every time when any one agent gets a resource, the task allocation and reward setting will be redone. For example, if agent 1 gets R1 and meanwhile agent2 is at this position (please see the figure 7), in this case, R2, R4, R5 are closer to agent 1 and just R3 is closer to agent2 so R2, R4, R5 are reallocated to agent1 and R3 is reallocated to agent2. Rewards of R2, R4, R5, R3 will be reset immediately based on the formula in 4.2.1: But one thing should be paid attention that the Man-

$$\mathrm{Reward}_1 * \mathbf{C}_{\mathrm{Mdis1}}^{\mathrm{Vdis1}} * \left(\frac{1}{4}\right)^{\mathrm{Mdis1}} = \mathrm{Reward}_2 * \mathbf{C}_{\mathrm{Mdis2}}^{\mathrm{Vdis2}} * \left(\frac{1}{4}\right)^{\mathrm{Mdis2}}$$

hattan distance used in the formula now is from the resource point to owner agent's **current** position. The reward of R1 who has been gotten will be set 0. Once again, as for these following questions: rewards for non-resource and non-goal points; how and when to set reward for the final goal; the reward of a resource point just makes sense to it's owner agent. Just refer to 4.2.1.

4.3 The prior evaluation of these methods and algorithms

Theoretically speaking, using Q learning to train these two agents based on any one method described above will make agents finish the scenario effectively. Each method has advantages but disadvantages as well. The first method is easy to be implemented but the training result should not be optimal. The second method is hard to be implemented but have a

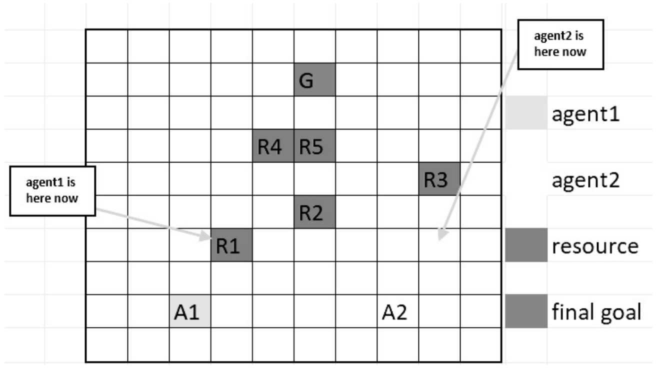


Figure 7: A map used in this scenario

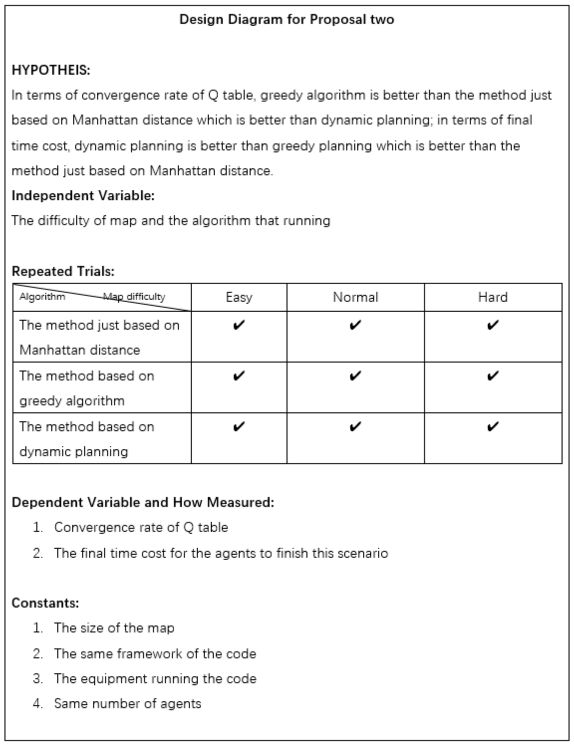
good balance on training time and training result. The last one is hard to be implemented too and training time is long but the training result should be the best. Based on these advantages and disadvantages of these three methods, it easy to infer that the convergence rate of Q table and the final time cost to finish the scenario will differ among these three methods. Our second research question is just to compare performance of these three methods in terms of the convergence rate of Q table and the final time cost to finish the scenario. So, choosing these three methods can make our second research question significant.

4.4 Experiments design

The goal of every experiment is to get two numbers:
• The number of least training episodes when two agents are trained successfully
• The number of steps used by two agents to finish the scenario after two agents are trained successfully.
The convergence rate of Q table can be represented by the first number, the final time cost by these two agents to finish the scenario can be represented by the second number. Compare and analyse these two numbers from different methods to give an answer to our research question or validate our hypothesis.

There are six steps to finish these experiments:
1. Generate a map including two agents, some resource points, and a final goal;
2. Use Q-learning to train these two agents in such a map above by the first method (4.2.1);
3. Record the number of least training episodes when two agents are trained successfully and take a video of that two agents collaborate to finish the scenario when these two agents are trained successfully;
4. Apply the second method (4.2.2) and the last method (section 4.2.3) to repeat step 2,3;
5. Generate many other maps to repeat the above steps; (All maps used are classified into three categories. "Easy": with 2 resource points distributed randomly; "Normal": with 5 resource points distributed randomly; "Hard": with 10 resource points distributed randomly)
6. Analyse all available data got from above steps.

The figure 8 below gives the outline of experiments



Design Diagram for Proposal two

HYPOTHEIS:

In terms of convergence rate of Q table, greedy algorithm is better than the method just based on Manhattan distance which is better than dynamic planning; in terms of final time cost, dynamic planning is better than greedy planning which is better than the method just based on Manhattan distance.

Independent Variable:

The difficulty of map and the algorithm that running

Repeated Trials:

Algorithm / Map difficulty	Easy	Normal	Hard
The method just based on Manhattan distance	✔	✔	✔
The method based on greedy algorithm	✔	✔	✔
The method based on dynamic planning	✔	✔	✔

Dependent Variable and How Measured:

1. Convergence rate of Q table
2. The final time cost for the agents to finish this scenario

Constants:

1. The size of the map
2. The same framework of the code
3. The equipment running the code
4. Same number of agents

Figure 8: Proposal 2

As for the implementation details required to run these experiments. Firstly, we designed and implemented the algorithm to generate maps with different difficulties. Besides, making two agents play smoothly on a same map though coding without any bug was a challenge for us but we did it. Last but not least, we also implemented these three methods successfully, which was one of our main works and took us much time.

4.5 Experiment results

We generated 30 maps (10 easy, 10 normal and 10 hard) in total. For each map, we did 10 experiments (the experiment here means the process of using Q learning to train these two agents successfully) for each method. So, we did 900 (10*3*30) experiments in the whole project. After preprocessing all experimental data, two tables are got:

	Easy map	Normal map	Hard map
The first method	48.3	186.7	578.4
Greedy algorithm	31.6	123.4	346.5
planning	58.1	247.5	754.4

Table 2: **Average** least episodes of training with different methods or maps (unit:episode)

*Take an example to explain "Average least episodes of training": there are 10*10 experiments for the second method (Greedy algorithm) with easy map, there are 10*10 least episodes of training accordingly. Sum these 10*10 numbers and divide by 10*10 is the "Average least episodes of training" for "Greedy algorithm" and "easy map"*

	Easy map	Normal map	Hard map
The first method	9.6 steps	17.1 steps	21.1 steps
Greedy algorithm	9.5 steps	14.5 steps	16.8 steps
planning	9.5 steps	12.3 steps	13.4 steps

Table 3: **Average** least episodes of training with different methods or maps (unit:step)

("Average" in table 3 has the same meaning to the "Average" in table 2)

Also, we choose a concrete map to show the path of the two agents finishing the scenario after being trained successfully by by these three methods.(Figure 9, 10, 11)

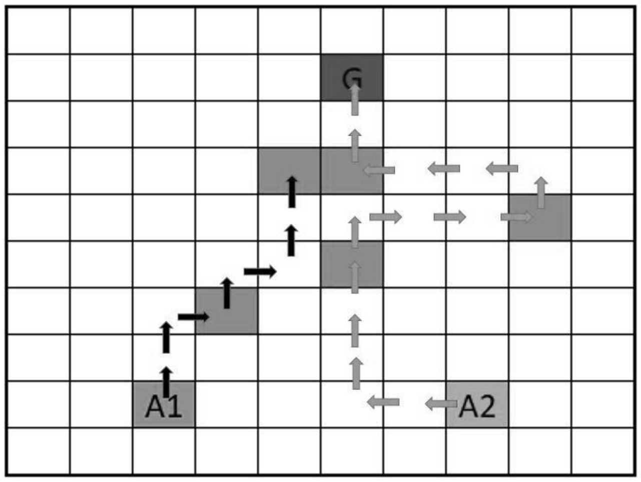


Figure 9: The method just based on Manhattan distance

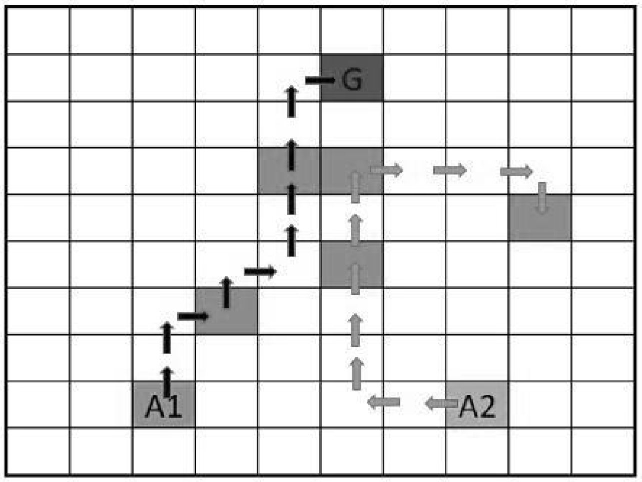


Figure 10: The method just based on greedy algorithm

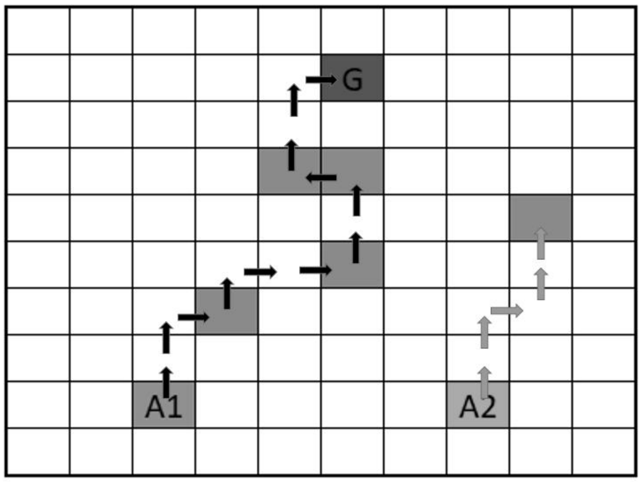


Figure 11: The method just based on dynamic planning

From table 2, it can be seen that Greedy algorithm has the fastest convergence rate of Q table while the dynamic planning has the slowest convergence rate. We think the reason is that: in greedy algorithm, no matter where a specific agent is in the process of training, the number of points with positive reward to this agent is always 1, just this one point will guide the agent where to go. But in the method just based on Manhattan distance or dynamic planning, the number of such points with positive reward to this specific agent generally is 2 or more, all such points will guide this agent where to go, as a result, useless conflicts will happen which will waste training time. In dynamic planning, task allocation and reward setting are always variational in the process of training, so such useless conflicts are more likely to happen which will waste more time.

Table 3 can tell that, in normal and hard maps, final time cost to finish the scenario of dynamic planning is less than that of greedy algorithm which is less than that of the method just based on Manhattan distance; Besides, with the increment of map difficulty, the difference will be become larger. But in easy map, the final time cost for these three methods are almost the same. The reason, we think, is that: in easy map (just with 2 resource points), task allocation is so easy that it will be the same no matter which method is used. But with the increment of map difficulty, the number of available task allocation strategies will increase exponentially, thus dynamic will have the first largest probability to find the optimal task allocation strategy and greedy algorithm second.

After finishing these experiments and analysing experiment data, a new idea comes up. it is to combine the greedy algorithm and dynamic planning to implement the task allocation and reward setting. In detail, use the dynamic planning to allocate task and then use the reward setting rule described in greedy algorithm to set rewards for resource and final goal points. In our opinion, combination greedy algorithm and dynamic planning not will only have a relatively high convergence rate of Q table but also have the least final time cost by agents to finish this scenario. This is just our opinion. Experiments are needed later to validate our views

5 Conclusion

In conclusion, this report describes the Minecraft scenario that our group built, and the two research questions we raised. Then, methods or algorithms used to study these two research questions are explained detailed and comprehensively. Besides that, the process and specifications of designing and completing corresponding experiments have also been described. At last, the experimental data is represented and some analysis results are have been discussed . As for the first proposal, combining the evaluation in section 3, although the convergent rate for double Q-learning is slightly higher than Q-learning, but double Q-learning does make Q-values much better than that using Q-learning. And the convergent time for double Q-learning only one tenth greater than using Q-learning. Therefore, we can call this experiment successfully verify our first proposal, that is double Q-learning do perform better than Q-learning under our scenario. As for the second proposal, Based on the experiment data and analysis result in section 4, it can be thought that our second hypothesis is correct as well.

References

[Baxter, 2009] L Baxter. discrete stochastic dynamic programming. *Markov decision processes*, 372(3):353–353, 2009.

[Berkhemer *et al.*, 2015] Olvert A Berkhemer, Puck SS Fransen, Debbie Beumer, Lucie A Van Den Berg, Hester F Lingsma, Albert J Yoo, Wouter J Schonewille, Jan Albert Vos, Paul J Nederkoorn, Marieke JH Wermer, et al. A randomized trial of intraarterial treatment for acute ischemic stroke. *New England Journal of Medicine*, 372(1):11–20, 2015.

[Slavík, 1995] P Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):435–441, 1995.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 1998.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[Watkins, 1989] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

6 A Individual Report - Jiahao Yu

In this project, I participated in and designed the two proposals. In the early stage of the project, I studied the research direction with my team members and constantly revise it, finally confirming our two proposals. I am mainly responsible for the implementation and research of the second proposal. My group member Guo Zhou helped me to implement the second proposal. My group member Shaohong Tian and Ziwei Li completed the first proposal. Our team members interact with each other and are very good at their respective tasks and this project. We are together addressing the challenges in this project and meet regularly to discuss the progress of our work and how the future work will proceed. I am very grateful to my team members for actively helping me solve problems and encouraging me during this period. As a member of this team, I learned how to develop my abilities in a team and how to achieve a consensus with teammates when we have different opinions on the solution of one problem. The listening is more important in a team project. Listening to other's opinions and comparing them with my opinion, and then think about how to allow others to accept my methods or combine our methods to produce a new method. Working as a member of a team let me know the art of communication and how to effectively accomplish team project with a pleasant team atmosphere.

Our research project involves artificial intelligence. Through this project, I have a deeper understanding of artificial intelligence. In the past, I thought that artificial intelligence was based on robots, allowing robots to continuously learn human knowledge and accomplish tasks that we could not accomplish. However, through this project, I realized that artificial intelligence can involve various aspects. For example, our project environment is a game I like to play. I never thought I could use it to compare the performance of AI robots trained by different algorithms and validate our hypotheses about artificial intelligence algorithms and applications. In artificial intelligence project, we can't rely too much on training iterations to improve the agent's ability in the current environment, otherwise, the agent will overfitting. From the artificial intelligence project, I learned the knowledge about how to build the environment and set the role to the agent and how to adjust agent's parameter to get better performance.

This project is a research project which is completely different from the implementation project I have done before. To be honest, make the proposal is the hardest part in the research project. From this project, I learned how to design invariants and independent variables to support a meaningful proposal. Moreover, in the early stages of the research project, blind trials are not advisable. Reading many related papers can help us broaden our thinking and come up with more interesting research direction. From doing this research projects, I known the important of ensure the research direction is correct and hypothesis is reasonable. It is futile to do more research based on an unfounded proposal. In here, I am very grateful to my supervisor, Tim and Nir, who pointed out some mistakes of our initial proposal and gave us a lot of guidance. In the research in computer science, the algorithm is an important part of computer science, just as the proposal we studied in this project are all comparisons between different algorithms. From this project, I learned the research in computer science is not a blind pursuit of the complex issues. A simple proposal that can solve practical problems is also a good research direction. And the effect of the algorithm under different conditions is worth to research and comparison.

In this project, I think I have performed well in making hypotheses and determining research directions. I have read many papers related to the project and made general assumptions and directions for our project. And I can communicate with my group member in time and help them find a solution when my group members encounter problems. I think I need to improve the ability to deeply understand and analysis of the problem. In this project, I used to make mistakes in code implementation due to not clearly understood the algorithm. I also need to improve my teamwork skills. In this project, I sometimes hope that the team members adopt the method proposed by me, so I continuously explain my views instead of seriously listening to their views. In the future teamwork, I hope I can learn to discover the advantages of other member's methods and listen before speaking.

Finally, I sincerely thank my team members for their help and company and also thank my supervisor for their patience and guidance.

7 B Individual Report - Ziwei Li

Our group project mainly contains two parts, proposal 1 and proposal 2. Just like the professor said, two of us search in one proposal and another two search in another proposal.

At the beginning of the project, the most important part is to make sure of our proposals. That is made by our four together. We look at reference and discuss together to decide our two proposals. Then the next step is to do implementation. Guo Zhou and Jiahao Yu work together to complement proposal 2, that is in the project with multiple agents, the performance of the Dynamic programming is better than greedy algorithm used in resource allocation plan. Shaohong Tian and I cooperate with each other to finish the first proposal, that is the performance of Double Q-learning is better than that of Q-learning.

From a team working, I have learnt a lot. The help from others is really important. For example, at first, I faced lots of problem in building a suitable environment in my computer. For myself, it is hard to solve it, so I come to my teammates. They are super special in this part and help me to solve it, so that I can continue going with the project.

For the aspect of artificial intelligence, I think what I have learnt most is there always has a better way wait us to detect. As for our project, we have set a goal, there are plenty of algorithms can be implemented, but we should compare two and choose a better one. However, if there is no restrict about the two algorithms, we can think about other plan which may performance better. And there are even some algorithms we did not care about before that have great performance. So we need to detect more in the way that searching in artificial intelligence.

For the aspect of computer science, I have learnt the language is a tool, but the mind is key important. When we touched a new language, we should learn a lot from others result, understand it and then use it.

As my friend Shaohong is really great in technique, so he did most of the coding task. While sometimes, he may confused about some direction, so I would discuss with him and come to a result. I think right way is also important with hardworking. I think this is my best performance in my team.

The area I need to improve is that I should learn more knowledge. Sometimes, when I have a group task, but I have some problem, I would ask help from my team members. However, if I can handle all the tasks the are assigned to me, we can save lots of time and finish the group job more efficiency.

8 C Individual Report - Guo Zhou

The whole process of completing this project includes reading related papers, identifying research questions or proposals, designing experiments, implementing experiments by coding, analysing experimental data and validating proposals finally. At the beginning, I read papers individually and then shared my opinions on how to choose proposals to my teammates. With the help of the collaboration of all group members, two proposals were identified, one of which was about the comparison of Q learning double Q learning and the other one was about the comparison of different algorithms on task allocation. Shaohong Tian and Ziwei Li were responsible for the first proposal, Jiahao yu and me were responsible for the second one. Then to validate the second proposal, Jiahao and I began again reading the papers related to the task allocation among multiple agents and then designed corresponding experiments. Designing representative and useful experiments was challenging and it took me a lot of time. The second challenge for us two was how to modify these algorithms for task allocation slightly to suit the designed experiment and implemented them by coding. In the process of dealing with this challenge, we two encountered countless detailed and technical problems, what's worse, Jiahao Yu and I had different opinions on this part. Fortunately, our hard work and patience helped us to overcome these technical problems and positive communication between us also helped us eliminate these disagreements. Since we did many times of experiment in order to avoid the accidental errors of experimental results, there were so much experimental data recorded that we two pre-processed this data first and then analysed it from different perspectives. Based on these analysis results, we two validated the second proposal. As for the final report, my main work was to finish the part about the second proposal, including the description of proposal, used algorithms or methods and the experiments designed.

I learned much from working in a team. First of all, clear assignment of tasks between group members improved our working efficiency dramatically by avoiding the overlap the some works. Besides, eliminating disagreements on specific problems is best way to keep the consistency of the whole project. For example, Jiahao Yu and I have different opinions on how to measure the performance of these different algorithms on task allocation. At beginning, we both ignored the other's views and did by our own way. But when we integrated our works, we found the results from us were not on the same channel and it was hard even impossible to integrate them. Until that time did we realise we should reach an agreement. Thanks to the agreement, our works became consistency later. Last but not least, expressing my own opinions after reading others' opinions can not only help me add some new ideas into my original opinions but also let others feel that their opinions and work are valuable and helpful. Meanwhile, a good team communicating atmosphere will be formed as well.

I thought artificial intelligence was a high technology which were far away from our lives. After experiencing this project, I have some new views on artificial intelligence. The keys of AI are searching and learning algorithms rather than amazing scenario applying AI technology. There is no doubt that unmanned car looks much more amazing than such scenario that two agents collaborate to finish a task. But Q learning are applied in both of these two scenarios. In other words, searching and learning algorithms are the base of the AI technology. If we want to apply these algorithms in difficult scenario with good performance, extra work compared with easy scenario is just to take more environment variables into consideration. Besides, I also think combining kinds of algorithms generally performs better than just using one single algorithm. Out of interest, I combined neural network and Q learning to train agents, and found in difficult scenario, the performance by using this combined method was much better than just using Q learning to train agents. So, I think developing new machine learning algorithms or combining existed algorithms reasonably is the tend to develop AI technology.

To be honest, I thought the research in computer science was just programming before. But now, I think the research in computer science is the combination of mathematical theory and coding and the mathematical theory is the key. Taking this project as an example, the part which took us most time was coming up with these algorithms for task allocation. The process of algorithm designing involved many mathematical theory analysis and calculations. As for the part of implementing these algorithms, even though encountering some bugs, the difficulty is much smaller than the part of coming up with these algorithms. Doing well in the computer science research extremely needs solid background of mathematic.

I took part in the completion of this project from the beginning to the end. When it comes to the best aspect of my performance to in my team, I think it is designing and implementing these algorithms for task allocation and organizing the final report. Of course, all our group members did a great job and made significant contributions to finish this project. In the process of finishing this project, I also found some weaknesses in myself. Especially, I always insisted on my own opinions blindly ignoring others' opinions. This behaviour not only is disrespectful but also can stop me learning something better others. I have made up my mind to drop this bad behaviour.

9 D Individual Report - Shaohong Tian

It's really so excited for me to do this project. After taking the subject AI for minecraft, I am attracted to artificial intelligence. It's really amazing that the programme or the agent can learn to achieve one task without any knowledge to the environment. The interest towards AI keep me from the start to the end of this project. Of course, there's more difficulty that I can imagine before this project. The most difficult part I guest is to choose our research problem. Minecraft is really an attractive game. I have play this game since Tim and Nir release the topic. I thought getting a quick start may help me participate in this subject, and it did. There so many possibilities in Minecraft. It can do whatever you can think of, even something you didn't. That's the beauty and also the obstacle. We don't know which direction that we should follow, which specific scenario we should choose for our final project. Lucky Tim and Nir always support us, they discuss with us every week and lead us to find our own topic. And finally, we come up with this topic, and in fact, we enjoyed it.

The other challenge was to read many related papers. As AI developed fast this years, there's so many paper we have to read. And not all of them we can understand. Thus every week we suffer from reading papers, but lucky, our group member can discuss what we have read, and help each other by discussing and brainstorming. So day by day, we make deeper understanding of related algorithms and also our own project.

Overall all challenges we have met, the most significant one should be coding. I think I am lucky that I have played this game beforehand, so I can understand the way how it works. But actually coding is a lot different from playing. Although the malmo platform provide us some examples to help us get familiar, it wasn't enough. Many methods are unknown for us, so I searched on the internet, email the malmo team for help. After lots of tiring, I can finally do what I want smoothly.

I mainly in charge of the first proposal. Ziwei and I we discuss very often, just to get a idea of how to get it done. We set up more than nine or ten scenarios before make our mind for the final one. We tried using a maze, and wondering in lava, and we also thought about making a action level game for our agent to complete. But finally we choose this one which is not only typically for our proposal, but also more intuitively for experiment. Honestly, I don't think this is my best work. I have many thoughts of how to improve our project. By introduce neural network, we can train a network that takes states as input and out the q values without training. But I guess I can always keep working even after this project.

Working on this project really benefit me a lot. First I learned how to work in a team. Team members should communicate as many as possibles. Only that we can generate new thoughts and get better understandings. Second, working on a research project made me know the procedure of doing one. I think I will do better if there's next one. Last but not least, I think that working on this project made me more attracted to AI, I think I will keep working on this field if possible.

As for a assessment for our group, I think everyone did a really great job. Ziwei Li is really active, and she always can bring us some new ideas, and she also helps a lot on paper work and setting the first proposal with me. Guo Zhou really good at programming, so sometimes I run into some problem he will nicely help me out, of course he gave very useful ideas of doing this project. As for Jiahao Yu, he has a very strong background knowledge related to AI, thus he gave us many professional options, he also helped coding. As for me, I am lucky to have these group mates, that I can participate to all the project. I programme the code for proposal one, and also analysis our result and make deep conclusion about it. In a word, I am very happy and luck to work with my friends!