Maze Self-Exploring Bot With Reinforcement Learning

Kah Ming Leong
Faculty of
Computing And Informatics,
Multimedia University
wilsonlkm98@gmail.com

Timothy Tzen Vun Yap
Faculty of
Computing And Informatics,
Multimedia University
timothy@mmu.edu.my

***Abstract*—The classical maze pathfinder is a supervised method and it showed no sign of intelligence when the program robot acts. Hence, reinforcement learning was introduced to replace the classical method. Reinforcement learning is one of the machine learning techniques used to create an agent to learn what to do, what kind of action needs to perform, focusing on obtaining the reward, and find the objective. The purpose of this research project is to investigate the performance of the various reinforcement learning algorithms in maze runner, identify the optimal learning parameter for each reinforcement learning algorithm, and make the comparison then decide which algorithm with optimal value is performed the best in both mazes. We applied 4 types of reinforcement learning algorithms in this research project. Besides, we are focusing more on the epsilon learning parameter where epsilon refers to the probability of randomly choosing to explores by choosing random action and exploits by choosing best-known action inside the maze. In the conclusion, the deep Q network performed the best in both mazes where this algorithm learning ability for the agent is better when comparing the rest of the algorithms.**

***Index Terms*—Reinforcement learning**

I. Introduction

The classical maze pathfinder showed no sign of intelligence when they decide some action, these approaches are more on supervising the program robot to find the objective. In another word, everything is already pre-computed by the algorithm. Hence, we want to create a program robot that able to make its own decision without help from the human, this is why reinforcement learning was selected in this study to replace the classical method. Before getting into the detail, we will start introducing Artificial intelligence first before introducing reinforcement learning.

Artificial intelligence (AI) refers to the simulation of human intelligence in the programmed machine. This technique allows the machines capable to execute the task, learn the knowledge, and impersonate human behavior. We can consider this AI as an artificial life in the real world. However, many researchers failed to achieve this goal as it is difficult to create and required huge resources for the project. Since machine learning is a subset of artificial intelligence, we may make use of the reinforcement learning technique to solve the problem. Since both of the technique, concepts are quite overlapping and they shared a similar goal. Therefore, reinforcement learning is one of the machine learning methods approaches towards sentient AI.

Reinforcement learning is part of the machine learning technique, alongside supervised–learning and unsupervised–learning. Reinforcement learning is telling the learner to learn what to do in the environment, what kind of action needs to perform based on the current situation and look for the goal. The goal might not be given immediately as it requires the time to figure out the possible path that will lead to the goal depending on how challenging of the task assigned and environment complexity. The "Trial–error and delayed reward" is the characteristics of reinforcement learning for the learner. In computer science, we often called the learner an agent, the main protagonist in reinforcement learning. The agent will learn by trying if fail then make some improvement by choosing another actions, and figure the goal by itself.

This project will focus on the future direction of using reinforcement learning to finding the solution for the AI, to match human performance, and create an AI to replace humans to explore the undiscovered region in the real world.

A. Project Objectives

The first objective is to investigate the performance of the various reinforcement learning algorithms in maze runner. We selected a different kind of reinforcement learning algorithms but we do not know which algorithms are the most suitable in terms of performance and complexity. Some algorithms could perform very well, but they take a very long time to train and vice-versa. So, we need to figure out which reinforcement learning variant is good in exploring the maze.

The second objective is to identify the optimal parameters for each reinforcement learning algorithm variant. The learning parameters consists: epsilon, alpha, and gamma. But, we emphasize more on epsilon learning parameter since it plays an important role to balance the probability of the agent choosing to explore or exploits inside the environment. Yet, we do not know which is the most optimal value for each reinforcement algorithm variant in maze runner. So, we have to test a different kind of value on each reinforcement learning variant and collect the results.

The last objective is to benchmark the various reinforcement learning algorithms against the baseline. Once we have done the testing on the experiments, we collect the data which is the result of reinforcement learning algorithms variants with different kinds of learning parameter values. We analyze the data, make a comparison, then determine and conclude which reinforcement learning algorithm variant with optimal parameter is performed the best.

II. Literature Review

A. Classical Methods To Explore The Maze

1) Pathfinder:

Pathfinder algorithm is used to solve the shortest path problem in graph theory. It searches a graph by starting at one vertex and exploring adjacent nodes until the destination node is reached, generally with the intent of finding the cheapest route. However, this algorithm is unstable, time-consuming and it does not always provide reliable result while exploring the maze. Therefore, some research papers stated this algorithm able to improve furthermore by applying extra calculation to find the goal by calculating the coordinate of the location-based on the size of the maze. They extracting the information from spatial navigation tracking files that contain XY coordinates over time [1]. and applied six variables to increase the accuracy of path-finding. The variables included: idea path error, heading error, angular corridor width, chaining annulus width, thigmotaxis zone size, and add a goal. With the help of extra variables, the algorithm has higher accuracy to find the goal at the correct location.

2) Route-Searching Method:

This method is similar to the pathfinder algorithm, both of them are sharing the same concept by solving the shortest path problem in graph theory. However, this method applied a depth-first searching method to study the grid of the maze. It can spot the dead-end grid then return along the previous paths to crossway, repeat until all the reachable grids are visited. Then, by using the information collected to determine which path is the shortest and optimal [2].

3) Image-Processing Approach To Solve The Maze:

This method is using an image processing approach to pre-processing the data based on the entire maze's image and capturing the possible paths that lead to the goal [3]. In other words, the robot already knows the solutions/paths in advance then starts to move to that certain location. This method is nice because it avoided the robot traversing long pathways and saves time and memory. However, in the current research paper that we studied they train the robot on 2D environment, not 3D environment. Therefore, we are doubting whether this method is able to work in the 3D environment. This method is more on supervising the robot to find the goal and it is not intelligent since the robot already knew the solutions/paths in advance when the algorithm ran and scanned the structure of the maze's image.

B. Neural Network

The word "network" means the connection or information processing mechanism. They are very alike to neurons of the human brain. Imagine there are a lot of neurons and each neuron can hold a different value, the value inside the neuron is called the activation and the networks are used to operate the activation between the layers.

Therefore, the neural network is about the way of the network operates the activations in the input layer determine the activations of the hidden layer, and then from the activations in the hidden layer determine the output of the output layer. The hidden layer mainly uses the data/activations from the input layer or hidden layer to perform the mathematical computation then predict the result.

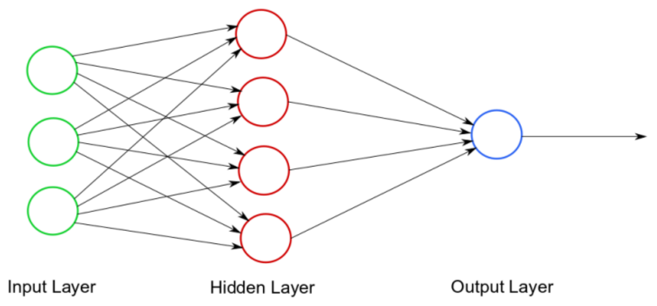

Fig. 1. Example of a neural networks.

C. Reinforcement Learning

Reinforcement learning is about learning what to do by trying them, what kind of action needs to perform based on the state at the same time focusing on obtaining the reward value and find out the goal. Besides, trial–error is the common characteristic of reinforcement learning.

In trial–error, we expect the agent (a computer program or the bot) to act wrongly in the early stage, from the mistake then refine their understanding and try another attempt until it found the goal. In the stimulation, the agent might not give the correct result immediately as it needs the time to discover the possible action that will lead to the goal, it will loop through many times to give an optimized result.

D. The Exploration And Exploitation

The exploration means the agent selects the random action explores all the possible regions to collect the data and make recommended action selections (exploitation) in the future. The exploitation means the agent selects the best-known action, in another word the agent chooses the action by exploiting the previous states/action-value that the agent learned so far to discover the optimal actions to obtain the reward.

However, we cannot balance the exploration and exploitation in the real situation by exploiting the past actions that are performed then adapt the knowledge to explore the other region and these issues remain unsolved. The data scientist named this problem as a dilemma of exploration and exploitation. The dilemma of exploration and exploitation knew as the problem of balancing the ratio of exploration and exploitation.

E. Reinforcement Learning Elements

The reinforcement learning agent is a computer program or learner. Reinforcement learning heavily depends on the action, state, policy, reward signal, value function, action–value, and model of the environment.

The action (A) is mean the set of all feasible options that the agent can make to navigate itself in the environment. An example of the actions is the agent can go up, down, left, right, or jump depended on the learning requirement.

The states (S) mean the observation in the environment which describing the agent's current situation in that certain coordinate.

The policy (π) can consider as an agent strategy or action selection. It means the way of an agent behaving in the environment at the given time.

The reward (R) signal is known as sense primary. It is a goal that we aimed at in reinforcement learning. Every time an agent makes an action, the environment will send a numeric value which we called it a reward to an agent. This is the main objective for an agent to obtaining the reward value as much as it can in the long run. The reward can be positive or negative rewards. The positive reward means the computer is telling an agent "you done a good job on this task" then adding the agent's R-value, as for the negative reward it can consider as punishment (made a mistake) which will result in deducing some R-value from the agent.

The value (V) function is known as a prediction of rewards in the short run. "RVπ(S)", this is the equation for value function represented and it means a return of the current state under the policy. The purpose of this equation is to achieve more reward by predicting the value, in which the value is mainly used to make and evaluating the decision.

The model of an environment is a simulation for an agent to interact with and solving the problem. The environment is created by a list of states and the reward system.

F. Q-Learning

The Q–learning or state action-value function is an algorithm for the agent to learn the policy by telling it to perform which action under that state. This the step for updating the value in the Q–learning algorithm.

Algorithm 1 Q–learning algorithm

```
1: Firstly, we initialize the Q – table at 0.
2: Let the agent choose an action.
3: The agent start to perform an action.
4: Measure the reward after it performed.
5: Update the Q – table value.
```

This the step for choosing an action in the Q–learning.

Algorithm 2 Algorithm Choose an action to explore or exploit

```
1: P = random()
2: if P less than epsilon:
3: then perform random action
4: else:
5: then perform best-known action
```

The Q–learning require some time to train. After it finished the train, the agent will evaluate the latest value on the Q–table and choose the better way (action) to achieve the goal.

G. Reinforcement Learning Platform

Reinforcement Learning Platform means the existed library that is used to train a reinforcement learning (RL) agent. There are many reinforcement learning platforms like pygame, Atari, and openAI gym but we are focused more on Malmo.

1) Malmo Platform:

The Malmo platform is an advanced artificial intelligence study built on top of a sandbox video game, Minecraft [4]. This platform allows AI technology to address real-world application. It is an open-source platform and it is mainly designed to assist the research in AI, reinforcement learning, and cognitive science. The testing environment is using Minecraft to train the agent.

III. Methodology

A. Platform Used And The Reason Of Choosing This

For the method to create the prototype, we preferred to use the Malmo environment to create the reinforcement learning agent. There is plenty of reasons why we are choosing Malmo as our default reinforcement learning platform, we listed it down below and provided the explanation based on our opinion.

1) Free and open-source platform
   - It is totally free, with no heavy setup, and easily downloads it from online. Which encouraged the programmer/researcher to do various artificial intelligence or reinforcement learning experiments inside the Minecraft.

2) Physics science
   - Minecraft has physics but it is quite simple physics and it is nowhere near the complexity of physics in the real world. No risks are involved when they run the experiment in the virtual environment (Minecraft), unlike the real world. If they run their experiment by using the actual robot in the real world, they must consider a lot of things like safety issues and repair costs if the robot is damaged, not only limited to these factors, it is just an example. Although the agent can die accidentally inside the Minecraft, it can re-spawn many times and it would not give us any problem in the real world since it is run in the virtual environment. It is much safer to run the experiment in the virtual environment to see the result before deploying it as a robot by applying the algorithm that we created in the real world.

3) Zero cost to set up for the experiment
   - As we mentioned at the beginning, the project will focus on the future direction of using reinforcement learning to match human performance and create an AI to replace humans to explore the undiscovered region in the real world. But, to create a robot for this required huge resources, resources included money, knowledge, material, training ground, and times which it is difficult for us to do as a normal university student. Fortunately, this platform can be done anything that we expected inside the game, it is free, and the interface is user-friendly, just required spending some time running the experiment and do the coding. However, the major drawback is my laptop barely support this platform, when we run some experiment the video frame rate is slow since we are using the low spec laptop, but still, we were able to run for it.

4) Can create the environment/maze difficulty based on our idea
   - We were able to create our own environment which is the maze based on our idea inside Minecraft. The objectives, blocks, and rules are easy to define and can code inside the XML file, it is very convenient since it no needs to write a lot of code. Documentation is provided when we want to add a function inside the XML, so it is easy for us to refer. After finished setting up the environment, we can use functions built inside the Malmo library (Python) to run the experiment.

B. Overview Of The Environment

In this research project, we created two types of mazes and both of these mazes are monster free. The width and length of the mazes are 30 x 30 per block size:

1) Evaluated maze
   - This map has a fixed maze structure but with a complicated design. Which means the exit is not easy to be found by the agent.

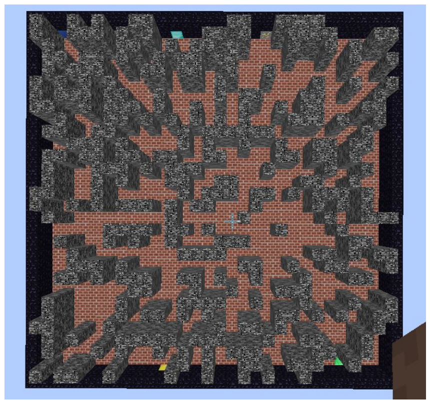

Fig. 2. The overview of the evaluated maze.

2) Random maze
   - This map has a random maze structure depend on the random seed generated. Sometimes the maze structure is less complex which easy for an agent to find the goal and sometimes it can be very difficult. Whenever the round is finished (either the agent found the goal or timeout), the program will generate a different maze structure for next the round.

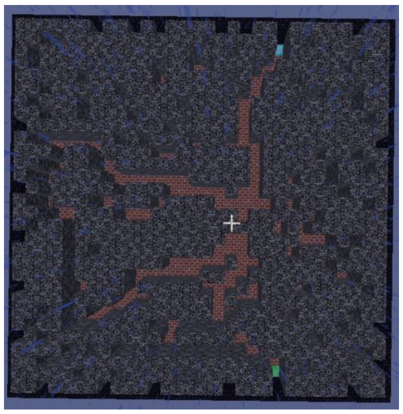

Fig. 3. The overview of the random maze.

The agent will be spawned at the emerald block on a fixed coordinate when the experiment is started, the agent needs to look for the special blocks to collect the mini reward and at the same time looking for the objective block to end the game.

1) Special blocks with the reward value (R = 500)
   - Iron block
   - Quartz block
   - Red stone block
   - Diamond ore
   - Gold ore
   - Iron ore
   - Lapis block
   - Gold block

2) Objective block with the reward value (R = 10000)
   - Diamond block

Regarding the agent's action, there are four actions that the agent can make: move forward, move backward, turn to the left, and turn to the right. Every time the agent sent a command to perform some action, it will reduce the agent reward value by 1, so in meantime, the agent must look for the special blocks in order to collect the reward value from the block.

C. Reinforcement Learning Algorithms

In this research, we applied four types of reinforcement learning algorithms to train the agent inside the maze. Algorithms included:

1) Greedy epsilon Q-learning
2) Reward-based greedy epsilon Q-learning
3) Q-learning with the Epsilon, Alpha, and Gamma learning parameters
4) Deep Q network

1) Greedy epsilon Q-learning (GEQL):
The most basic algorithm for an agent to perform exploration and exploitation inside the maze. It used the epsilon learning parameter to balance the probability of choosing to explore and exploits. The epsilon value is the range between 0 to 1. If epsilon is set to 0, we never explore but always exploit the knowledge we already have. On the contrary, having the epsilon set to 1 forces the algorithm to always take random actions and never use past knowledge. Usually, epsilon is selected as a small number close to 0.

2) Reward-based greedy epsilon Q-learning (RB-GEQL):
This algorithm is the same as the greedy epsilon Q-learning that we mentioned in the previous paragraph, the only difference is we are using the reward that the agent collected to update the Q-table. So, the agent is able to keep track of the correct path based on the reward collected on every round.

3) Q-learning with the Epsilon, Alpha, and Gamma learning parameters (QL-3LP):
We have learned epsilon which is mainly used to balance the ratio of exploration and exploitation. Alpha is known as the learning rate, which means the agent's learning ability and decision-making when performing some action inside the environment. The alpha value is the range between 0 to 1, if we set alpha to zero, the agent learns nothing from new actions. Conversely, if we set alpha to 1, the agent completely ignores prior knowledge and only values the most recent information. Higher alpha values make Q-values change faster. Gamma is known as the discount factor, which is multiplied by the estimation of the optimal future value. The next reward's importance is defined by the gamma parameter. The gamma value is the range between 0 to 1. If we set gamma to zero, the agent completely ignores the future rewards. Such agents only consider current rewards. On the other hand, if we set gamma to 1, the algorithm would look for high rewards in the long term. A high gamma value might prevent conversion: summing up non-discounted rewards leads to having high Q-values.

However, for this reinforcement learning algorithm we only do parameter tuning on epsilon while the other learning parameters like alpha and gamma are assigned the default value of 0.1. The reason behind this is we do not have much time to perform more experiments because it is very time-consuming and needs parameters turning a lot of value.

4) Deep Q network (DQN):
A deep Q network is a combination of Q-learning and neural networks. This algorithm used a neural network to improve the Q-learning performance through a technique called experience replay. Experience replay means instead of running Q-learning on state/action pairs as they occur during simulation or actual experience, the system stores the data discovered for [state, action, reward, next state] in a large table. Then we use these to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output.

Deep Q network also using learning parameters like epsilon, gamma, and epsilon decay to do some optimization. However, we perform parameter tuning on epsilon only while we set a default value with 0.1 to the gamma variable and a default value with 0.45 to the epsilon decay variable. We are using PyTorch/torch and PyTorchvision/torchvision python libraries to use neural network function inside our program and we set the network layers equal to 64.

D. Q-Table

This is called the action-value table or Q-table. The function approximates the value of selecting a certain action in a certain state. Regarding updating the Q-table, we wrote a code by collecting the information generated from the Q-table and assign it to a new q variable. Lastly, update the Q-table based on the new q. This Q-table is used to keep track of the agent's previous explored area and also see the agent's current state.

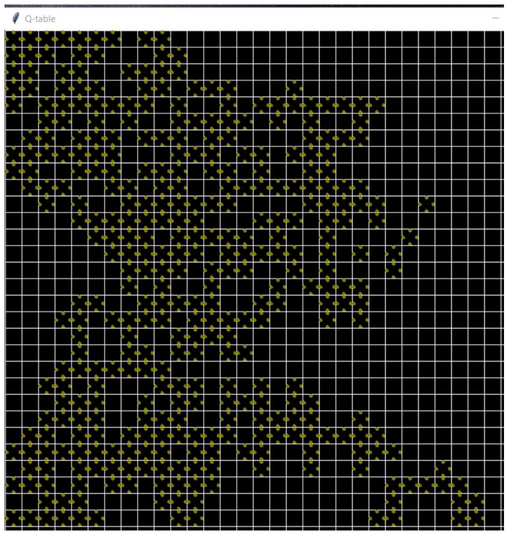

Fig. 4. Sample of the Q-table.

E. Testing

Previously we mentioned we only do the parameter tuning on the epsilon learning parameter for the 4 algorithms. Epsilon value is the range between 0 to 1. Therefore, we run the experiment 30 times with the epsilon value of 0.0, and with each round run 5 minutes, we will be collecting the important data like rewards and time taken on each round for every experiment. Once the experiment is finished, we manually increase the epsilon value to 0.1 and re-train it again. This process will be repeated until we finished training the experiment with the epsilon value of 1.0 for every algorithm except the deep Q network. We train this on the random maze and evaluated maze. Once we collected all of the results for the random maze and evaluated maze on the different types of algorithms except the deep Q network, we analyze and look for the average reward on different experiments and pick the epsilon value with the highest average reward. Then, we do the training again but this time is training up to 100 times, and with each round runs 1 hour for every algorithm.

Meanwhile, for the deep Q network algorithm, we directly run the experiment 100 times with the epsilon value of 0.0, and with each round runs 1 hour for both mazes. Once the experiment is finished, then we manually increase the epsilon value to 0.1 and re-train it again. This process will be repeated until we finished train the experiment with the epsilon value of 0.9. All of the results will be collected and stored in the CSV file for future analysis.

IV. Result And Discussion

We collected the results in which the epsilon value is the range in between 0 to 1. This results will be used to analyze and determine the agent with which epsilon value performed the best inside the random maze and evaluated maze. This result is only used 5 minutes to train the agent for 30 rounds. The reward on each round (up to 30 rounds) are accumulated and divided by 30 to find the average reward. Table below showed the highest average reward together with the epsilon value on each reinforcement learning algorithm variant.

	Random Maze		
	Algorithms	***Epsilon***	***Average Reward***
1	GEQL	1.0	3219.53
2	RB-GEQL	0.7	2950.27
3	QL-3LP	0.3	3799.97

	Evaluated Maze		
	Algorithms	***Epsilon***	***Average Reward***
1	GEQL	0.8	54.07
2	RB-GEQL	0.7	48.93
3	QL-3LP	0.5	290.7

Next, we do the training again but this time is training up to 100 times, and with each round runs 1 hour for every algorithm. The reward on each round (up to 100 rounds) are accumulated and divided by 100 to find the average reward.

	Final Evaluation For Evaluated Maze			
	Algorithms	***Epsilon***	***Average Reward***	***Success Rate***
1	GEQL	1.0	3920.33	79%
2	RB-GEQL	0.7	3772.26	65%
3	QL-3LP	0.3	3582.11	78%
4	DQN	0.5	4011.0	78%

	Final Evaluation For Evaluated Maze			
	Algorithms	***Epsilon***	***Average Reward***	***Success Rate***
1	GEQL	0.8	2772.74	29%
2	RB-GEQL	0.7	2826.13	19%
3	QL-3LP	0.5	2920.22	21%
4	DQN	0.5	258.5	35%

A. Discussion

We applied four types of reinforcement learning algorithms for the agent to explore the environment that we created, algorithms included: greedy epsilon Q-learning, reward-based greedy epsilon Q-learning, Q-learning with three learning parameters, and lastly deep Q network. We do parameter tuning on epsilon variable on these algorithms to find the most optimized value, in another word, we are expecting these optimized values would help us reduce the dilemma of exploration and exploitation issue.

We created two types of mazes named random maze and evaluated maze which we used to do the comparison. Nevertheless, our main focus is more on the evaluated maze since that maze has a complicated structure and it not easy to look for the exit/objective.

For the random maze, the deep Q network algorithm with the epsilon value of 0.5 performed the best and it achieved the highest average reward (R = 4011) while the average reward for other algorithms is the range between 3000 to 3900. For the success rate of looking at the goal or objective point, in 78 out of 100 rounds which means 78% the agent managed to found the diamond block to finish the game early for the deep Q network. Similar to greedy epsilon and Q-learning with three learning parameter algorithms, they shared the same success rate to finish the game early while the reward-based greedy epsilon Q-learning performed the worst, it has the lowest average reward and lower success rate to find the goal. The DQN agent managed to collect the reward beyond 2500 and it also able to finish the game within 1 hour while other algorithms were not able to achieve better than DQN. However, this is not a fair comparison since the maze structure is randomly generated, sometimes the maze can be very straightforward, less challenging, or very complicated depending on the random seed numbers.

For the evaluated maze, if we evaluated the performance based on average reward, the Q-learning with alpha, epsilon, and gamma learning parameter with the epsilon value of 0.5, alpha value of 0.1, and gamma value of 0.1 performed the best with the highest average reward (R = 2920.22) compared to the rest of the algorithms. However, the Q-learning with three learning parameter has the lowest success rate of looking the goal or objective point which it is equal to 21 while the deep Q network with the epsilon value of 0.3 has the highest success rate of looking the goal which it is equal to 35 although it has the lowest average reward (R = 258.5). The DQN agent sometimes received negative reward or positive reward at that rounds while other agents received positive reward at most of the times. The reason why the agent is having a negative reward is that for the deep Q network, the agent performed more actions compared to the general Q-learning algorithms. The agent keeps finding the most optimal actions to look for the special block and objective block through the neural network, which it involved complex mathematical formula while general Q-learning only involved simple mathematical formula. That the reason why general Q-learning performed fewer actions while deep Q network performed more actions because deep Q network is better when coming to decision making and it has some intelligence to do something. Although, the agent able to collect the rewards from special blocks or objective block, but sometimes the accumulated reward will still end up with a negative reward because the agent performed too many actions. Therefore, we are considering the deep Q network with the epsilon value of 0.3 is performed the best compared to other algorithms.

Regarding the training time, the evaluated maze takes more time to train compared to the random maze. The reason is that the evaluated maze structure is far more labyrinthine than the random maze. We can use the success rate to determine the difference. The success rate of looking at the goal and objective point is lower than the random maze, which the highest is only 35% (deep Q network) for the evaluated maze while most of the algorithms for random maze are above 70%. Nonetheless, the deep Q network algorithm performed the best in both mazes.

V. Conclusion

Nowadays, technology is getting more advanced, we intended to start working with a mini-experiment before we create artificial intelligence in the near future. This research project focused on creating the prototype by applying the concept of reinforcement learning instead of AI. Reinforcement learning is an alternative and cheapest way to set up an experiment, we used the Malmo library as our reinforcement learning platform to create the reinforcement learning agent.

We were come up with several reinforcement learning algorithms for the agent to explore the environment that we created. We do parameter tuning on epsilon variable on these algorithms to find out the most optimized value, in another word, we are expecting these optimal values would help us reduce the dilemma of exploration and exploitation issue. Lastly, based on the result that we collected and analyzed, the deep Q network reinforcement learning algorithm is performed the best with the epsilon value of 0.5 for the random maze and the epsilon value of 0.3 for the evaluated maze. Meanwhile, the result of the reinforcement learning algorithms in terms of average reward is close to the result of the deep Q network.

A. Future Works

At this stage of the project, we are satisfied with the outcome as we applied the fundamental reinforcement learning algorithms inside the prototype that we created and also done some parameter tuning on epsilon to figure out the best value for each algorithm. However, there are areas in our prototype and program that we could improve upon further.

Perhaps, we are considering doing parameter tuning on alpha and gamma learning parameters in the future. Since both learning parameters also played an important element inside the reinforcement learning. Moreover, we will try to add the obstacles and dangerous monsters/beasts to increase the maze difficulty, write some extra function allow the agent to evade the obstacles or attack the monsters. We would like to see what kind of decision that the agent will make at the same time looking for the mission objective to end the game. Besides, we planned to write a program to allow the agent to explore the open-world in Minecraft, which we mentioned our future direction from the beginning as using a reinforcement learning agent to explore the undiscovered region, instead of real-world we test it in the virtual world.

References

[1] B. Matthew and P. Timothy and E. Richard, Pathfinder: open source software for analyzing spatial navigation search strategies, 2020.

[2] C. Wu and D. Liaw and H. Lee, A Method for Finding the Routes of Mazes, 2018.

[3] B. Rahnama and A. Elçi and S. Metani, An Image Processing Approach to Solve Labyrinth Discovery Robotics Problem, 2012.

[4] J. Matthew and H. Katja and T. Hutton and D. Bignell, The Malmo Platform for Artificial Intelligence Experimentation, 2016.