Final Report

Chen, Changqi
Department of Computer Science
University of California, Irvine
Irvine, CA 92617
changqc@uci.edu

Douglass, Michael
Department of Computer Science
University of California, Irvine
Irvine, CA 92617
mrdougl1@uci.edu

Liu, Mark
Department of Computer Science
University of California, Irvine
Irvine, CA 92617
youremail@uci.edu

Abstract

Our project for the class is based around the Malmo platform for Minecraft. We want an AI agent to perform steps within Minecraft that a normal human player could perform themselves. That includes mining different types of materials, crafting different items, and crafting different types of pickaxes (wood, gold, diamond) all from first finding the specific materials to craft with (wood logs, gold blocks and diamond blocks). The AI agent has to perform these tasks in specific orders (E.G mining wood first, creating wood planks and sticks second then finally creating a wooden pickaxe). Also each major milestone of creating a certain pickaxe has specific subtasks that must be completed before the creation of the pickaxe.

The AI is able to accomplish many different tasks. Our AI is able to find and mine wooden logs, turn that into planks and sticks and then create a wooden pickaxe. After the creation of the wooden pickaxe it is able to find a golden block and create a gold pickaxe. Finally after completing both those tasks it can find a diamond block and create a diamond pickaxe. It also is able to avoid water spots so as to not die from falling into it.

1 Introduction

Our project is about the Malmo Platform for Minecraft. We created an AI that can do various tasks within minecraft that a normal player might do during their time playing the game. This includes mining wooden blocks, gold blocks and diamond blocks. This also includes creating items in the game like wood planks, sticks and pickaxes of various materials like wood, gold, and diamond. The AI uses the wood blocks to create new items like the wood planks, sticks, and wooden pickaxe. The AI would use the wooden pickaxe to ultimately obtain a golden block and then use that to create a golden pickaxe. Then the AI would then use the gold pickaxe to obtain a diamond block and craft a diamond pickaxe.

1.1 Problems Introduction

We have multiple problems that we are addressing. The first is how the environment around our AI should be created to maximize the learning that takes place. If the map is too big then it may wander into unnecessary spots often and run inefficiently. We fixed this by creating a smaller but complex map. The second problem is trying to stop our agent from endlessly digging downwards and basically leaving the area of the environment. To combat this we created a specific environment with special blocks and two separate action sets, one with hitting blocks and movement and one with just movement. This allowed us to stop the endless, inefficient hitting of the agent while allowing the

Preprint. Under review.

agent to still collect blocks. The problem was interesting for us in that, had we not found any way to change the environment/digging for our AI we would have had a very hard time completing our tasks in any sort of timely fashion. By creating the exact environment we need, we are able to test our AI efficiently and have the AI learn and grow. Below is a picture of our environment.

Figure 1: The environment our AI uses.

1.2 Map Size Experimentation

We ran multiple different experiments throughout our time testing. We even changed the number of trials multiple times from 300 to 500 to 1000. Each trial we gave our AI 30 seconds as well.

Our first big experiment was the size of the map and its impact on learning. We found that our map was too big in the beginning, which caused inefficient learning. This led to a shrinking of the map which helped the efficiency and running time of our AI.

1.3 Turning and Digging Experimentation

Our second experiment was based on the turning and digging abilities of the agent. As previously mentioned the endless digging was a problem where our agent may dig completely out of the area of the task. We experimented by adding turning and jumping to help the agent possibly turn and jump out of the holes it was digging itself but that did not help as we wasted many moves jumping/turning for no reason.

1.4 Reward Experimentation

Our third and most important experiment was the rewards. We needed to find the sweet spot for our rewards, which proved difficult. Too high of a negative reward may cause the AI to just move back and forth or not at all to avoid losing reward score. Too low of a negative reward and the AI wont avoid dumb mistakes. After tweaking we found the right combination of rewards, both negative and positive, such as -1000 for falling in water and +100 for stepping onto the "indicator" block. Also

our experimentation of tweaking rewards actually helped us lead to the aforementioned "indicator" blocks where the AI digs only on a special block and thus telling the agent it is ok to dig.

1.5 Learning Rate Experimentation

Our fourth and final experiment was the learning rate where we tweaked that value from 0.1 to 1.0 trying to find the best one. Eventually we found that 0.4 - 0.6 was the best and we settled to 0.5.

1.6 Results Introduction

Our results were that our AI agent is able to find wood logs, then craft a wooden pickaxe from those 2 logs, then find a gold block and create a golden pickaxe and then lastly frind the diamond block and craft a diamond pickaxe. We also found that our learning rate was best at at 0.5 with discount rate 0.8.

2 Background

Firstly our main platforms were Malmo, Minecraft and MalmoPython as without these three platforms we would have no porject at all. Secondly we are using many normal python libraries including json, sys, os, time, logging, random, with our most important libraries being json and random (json for reading the observations and random so we can have the agent sometimes randomly move instead of doing the best move). We also used the TKinter library to draw out our Q-table and that function in general was provided by the Malmo installation.

Our Agent's main algorithm is a Tabular-Q learning based algorithm. Our Q-Learning agent has two different action sets, one with only 4 movement directions and one with those same movement directions + digging if on a "indicator" block. The agent also has 51 regular map states and 6 special states. The 6 special states are the "indicator" blocks and the different materials the agent needs to collect. Our tabular Q agent uses the Bellman Equation to update the Q-Table and policy which can be found in the Methods Section. Our XML file is mostly composed by us using the XML Schema Documentation to help find commands and syntax.

3 Problem Statement

The first problem we had is the size of our environment. We try to make the environment big to increase the complexity for our agent. However, after several steps, our agent starts to act very slowly. The second problem is due to the setting of Malmo. Our agent will dig everything even the bedrock. So it not only makes the environment hard to control but also the learning inefficient. Therefore, we need to think some method to prevent our agent from keep digging down without limit and dig everything.

We also have some problem regarding how to set our rewards. We want to make our agent first learn to collect the log, then the gold and finally the diamond. So we may want to change the reward after our agent collect some number of logs so that it will learn try to collect gold instead of keep collecting logs even it has learned the best way of collecting log. We also set some other negative rewards: -2 for every movement and -1000 for falling into water.

Our minimum milestone is to find and acquire two wooden logs by hitting them and collecting them. We tried to add turning and jumping actions to make our agent more agile but that always led to the a lot of redundant steps. After collecting the logs the second part of the minimum milestone was to create a wood pickaxe. Also we removed the turn and jump actions. Our next milestone was to collect a golden block and then turn that into a golden pickaxe. Our final milestone was to find a diamond block and turn that into a diamond piackaxe. To do these steps our agents need to first try to collect enough of each specific resource before making each pickaxe.

4 Methods

The model class we use in our project to find the optimal policy is the tabular Q-learning. It takes action A on state S, notes the reward and the next state S', then it chooses the max Q-Value in state S'

then use all these info to update Q(S, A), then moves to S' and executes epsilon greedy action which does not necessarily result in taking action that has the max Q-Value in state S'.(**?**)

The formula for our one-step tabular Q-learning is shown in Figure 1. We set the learning rate alpha 0.1 and the reward discount factor gamma 0.8 at first.

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

Figure 2: Our Q-Learning Formula.

5 Experiments

First experiment we did was how the size of the map influence the efficiency of the learning. In the beginning we were testing the map of size 11x20 with turning abilities. The result came out poorly since the map is too big and normally takes 500+ rounds to even just finish the first milestone which is to craft a wood pickaxe. So we decrease the size of the map and see if the result varies. After several experiments we found that if we decrease the size of the map to 9x14 the agent would achieve the most optimal sweat spot of the size, however, the most important elements that are influencing the efficiency is the turning function and the digging function.

The second experiment we were focusing on is the turning and digging abilities of the agent. Like in the problem statement, the digging ability was not optimal due to the fact that the agent would drop into the hole after digging and started to dig further downwards which makes the experiment and the learning curve hard to proceed. We tried to experiment adding a jump ability when the agent drop down to a hole or getting negative rewards or even death after dropping down. But the result was not ideal after all the testings so in the end we decided to create a "indicator" block so that the agent would only dig when it walks on the indicator block.

The third experiment we did the most amount of testing was the rewards. It is hard to measure what is the most optimal reward for each action. For example, we set a negative 100 reward for whenever the agent dig a dirt just to prevent the agent from digging too much dirt over the actual items that are set to goals. However, this negative reward tend to make the agent undecided and even overtime stop moving at all since there are only negative rewards in the surrounding. (the digging was acted after every moving motion) So we reconsidered the environment and tend to imitate the agent to a real person. A gamer would not dig every block to see if it is wood but to visually see a tree and then went straight forward and dig. With this mindset, we instead created a "indicator" block for the agent and the agent's goal is to find the indicator block then star digging which prevents the digging problem and the lack of efficiency.

The last experiment we were working on is the learning rate which stated at 1. After many days of testing we found that the learning rate in between 0.4 - 0.6 is the most optimal which takes the least amount of time.

6 Results

After running the aforementioned experiments, we find that our agent performances best in learning rate 0.5. The average reward of our agent in the 0.5 learning rate is about 705 which is much higher when the learning rate is 0.1 and 1. Figure 3, which plots the learning curve of our agent (mean return per episode over 10) with learning rate 0.5, also shows that our agent performances better with more and more training than with learning rate 0.1. Also the agent is able to find the materials for and craft a wooden, golden, and diamond pickaxe.

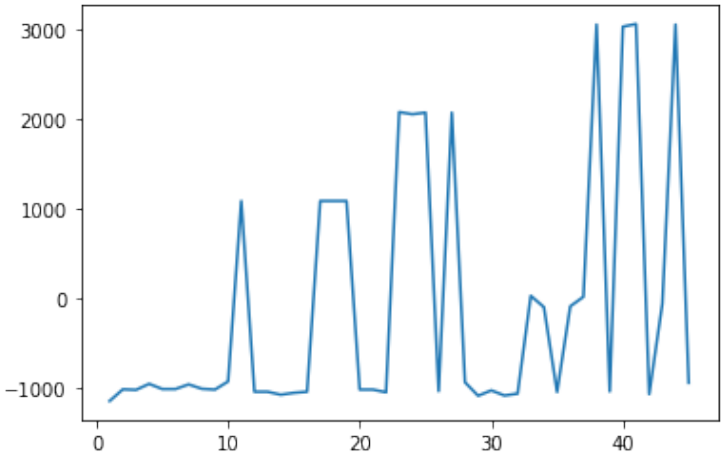


Figure 3: Our agent learning curve with learning rate 0.5.

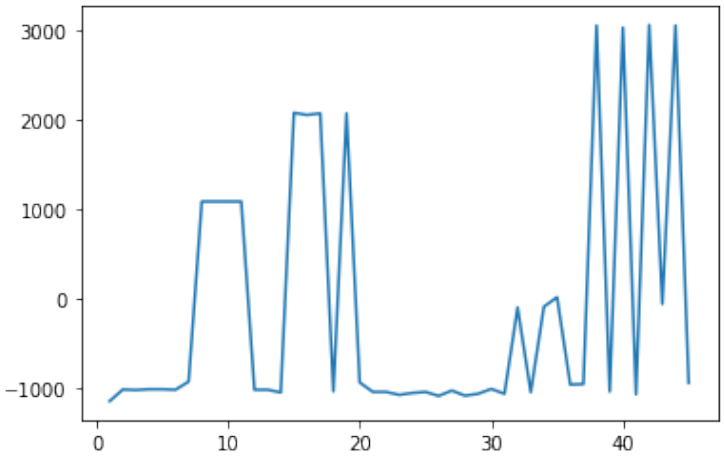
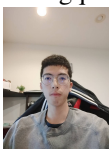


Figure 4: Our agent learning curve with learning rate 0.1.

7 Contribution

Changqi Chen



Implement the method to update Q-table using the aforementioned formula. Do the second experiment which add jump and turn actions and add the "indicator" block in our environment. Testing the best learning rate for our agent. Writing the method and problem statement parts.

Michael Douglass

Implementing crafting that the agent can do: Crafting sticks, wood planks, wood pickaxes, gold ingots, gold pickaxes, diamonds, and diamond pickaxes. Setting xml for inventory observations and allowing crafting/inventory commands. Writing Abstract, introductions and Background. Website.

Mark Liu

Giving the idea of the crafting agent. Setting up the xml environment for the project including rewards, terrain, traps, etc. Testing optimal rate for each rewards. Writing the Experiment and Result parts.