Azul Group Project Report

Coconut_Opener
Pengbo Yan (952253)
Yingming Ma (1008102)
Jieyu Wang (1066616)

I. YouTube Video

https://youtu.be/JNOQ37g5gx0

II. Overview Explaining of Techniques

Introduction

The purpose of our project is to use different techniques to implement an autonomous agent for game Azul. Through online games on Azee and the overview of the rules of Azul, we had a basic understanding of the game. Our final goal of the game is to move tiles from our complete pattern lines over to our walls in order to get scores as much as possible. There are several scoring policies for different patterns of tiles on the wall. We found that trying to figure out how to place on the space matching its color and lines with directly adjacent is the key to success.

Depth First Search(DFS)

At the beginning of discovering, we tried DFS, one of blind search methods, as the main technique. This searching method requires us to traverse all states of the game but the time is not enough to do so. This algorithm is easy to implement and suitable for the endgame because we thought the search space would be small in the endgame. However, the deep copy of the game states takes more time than we expected. Even if we searched for the last round, DFS still cannot finish in 1 second. So we came up with two other strategies to figure it out. One is to add a Heuristic to estimate the profit of each move. The other is to restrict the depth of the DFS and do a Minmax backpropagation. These will be explained in the following parts.

Heuristic

Then we introduced a heuristic to estimate the profit of each move (*naive_player* also has a very simple heuristic). With a Heuristic, our agent stops searching with a depth whatever it reaches the end and then chooses a best move based on Heuristic. However, Azul is a game played by 2 agents. If we assume that our foes will choose the move randomly, then our agent may make some bad choices because in fact our foes will probably choose a good move for it.

Game Theory

To get a better assumption about our foe, we considered Game Theory and introduced the Minmax backpropagation. This strategy assumes that our foe will choose the best move based on the same Heuristic as ours. This assumption is still not true but it is far better than the assumption of random

choice. After we combined the DFS, Heuristic and Game Theory, our agent can make some very reasonable choices.

Monte Carlo Tree Search (MCTs)

MCTs is an anytime algorithm. It can be terminated at any time and still give the best answer searched in a pre-defined time period. It guarantees to output a valid move within the time limit for every move. Its reward could be calculated and the simulation is easy to implement.

First of all, when a tree structure was built, every node recorded its game_state, playerid, the move executed from last game_state, parent node, child nodes, number of visits, total reward, available moves for current game_state, whether the node is fully expanded, as well as whether the current game_state is the end of a round.The framework of MCTs has four stages, selection, expansion, simulation and backpropagation. Selection is to select a single node in the tree that is not fully expanded. In this case, it means at least one of its children is not yet explored. Expansion means to expand the selected node by applying one available move from the node. Simulation is from the expanded node.We simulate the game until the round comes to the end by random select valid moves for both players. Backpropagation represents the number of visits and total reward of the node are back propagated to the root node, updating the values of each parent node on the way back to the root. To balance the exploration-exploitation trade-off, we select the best move based on calculated UCB values for nodes. The formula is:

$$\text{exploration parameter } C = 1/\sqrt{2}$$

$$UCB = \frac{totalreward}{numberofvisits} + C \times \sqrt{2 \times \frac{\ln(parentnode.numberofvisits)}{numberofvisits}}$$

One of the advantages of MCTs in Azul projects is that the search can be terminated at any time, and it could return the best action searched in that time period. MCTs do not need a deep understanding of the game rules, the reward can be calculated only based on the game results and with the development of reward calculation, the performance of MCTs can be improved. The tree in MCTs is constructed asymmetrically because the algorithm prefers to expand the tree node with promising results, therefore it requires smaller space to store the tree than classic search algorithms like blind search and have more time to explore more promising tree nodes. One of the drawbacks is it needs a large amount of iterations to fully take advantage of the algorithm, which is not quite possible in this project since the time limit for one move is only one second. Another disadvantage is that the search may not output reasonable moves when playing against expert agents in some cases. The reason behind is that simulation moves are randomly selected and there are chances the optimal move is not simulated, especially at the beginning of the round, since there are too many combinations to search in one second.

The first challenge we experienced is how to transform the implementation of MCTs to search more effectively in Azul games. Another challenge is how to improve the reward system based on the game state and rules. We tried to play with each other to figure out a reward system based on human

experience. Rather than only calculate the difference score between two players as a reward, we managed to set each explored node a default reward based on the game state, and it works well according to the results.

In terms of future improvements of MCTs, reducing the number of valid moves to explore at the start of the round and next a few steps would boost the performance of the agent. Some other strategies should also be explored and applied with MCTs to overcome the drawbacks of different models, such as we could utilize MCTs, blind search, game theory, and reinforcement learning in different game stages (beginning, middle, end of a round) into one agent. A description of sensible improvements they would make to the agent given additional time. Since MCTs is an anytime algorithm, if additional time is given to the agent, it means the more possible moves will be simulated, and the possibility to output the optimal move will be increased.

III. Final Tournament Agent

Overview

The final version combined the MCTs, Blind Search, Heuristic and Game Theory. First of all, before the agent knows that the game will end in the current round, it will select moves by a MinMax search with a heuristic. The depth is dynamic, depending on the size of available moves. Also, in some cases the agent will cut some moves with low heuristic value so that the depth can increase. The settings of the depth and cutting number are based on the experiment to satisfy the 1-second time limit.Under this structure, the heuristic will influence the performance significantly. The initial heuristic was constructed by our intuition and understanding of Azul. Then we tried to change the parameters in the heuristic function to improve the performance. We also looked at the replays between our agent and the staff_team_top agent to find the drawbacks of our agent so that we could change them. Finally, the heuristic function became very complicated but it did well.

Additionally, after the agent knows that the current round is the last round by finding one row of one player will be filled at the end of this round, the agent changes to MCTs mode. This is because we find that the above algorithm is not suitable for the endgame while the MCT works very well at this stage. This is reasonable because MCT will try to search the ends and it probably covers almost all the ends at this stage, while the heuristic only considers the effect of moves but not the result of the whole game.The final tournament agent had a very good performance. The detailed evaluations show in Part IV. The rate of winning when it plays with naive_player is around 98% and the last ranking on the test context is 6th.

Weakness and Future Work

The main weakness of our agent is that it could fill the floor line and get a negative score in some rounds, especially when it played with the staff_team_top. Modification of the heuristic may solve this problem, but it will make the average performance lower (when playing with many different agents) so we abandon this idea. Although our ranking (and winning number) is better than the staff_team_top, our agent always loses when it plays with the staff_team_top. This fact shows that there are different strategies of playing this game and these strategies have a Rock-Paper-Scissors relation.However, it is possible to make an agent who could choose the right strategy depending on its foe’s strategy. This needs to know the kinds of

strategies, the relation between them, and how to judge the foe's strategy. If there is more time to do some deeper work on this AI, we believe this is an interesting direction.

IV. Final Evaluation

	Naive	Heu_GT_1	Heu_GT_DFS	MCT	Final Version
Naive_Player	BaseLine	-	-	-	-
Heu_GT_1	91 - 2 - 7 47.05 - 21.40	Consider 1 step for each player	-	-	-
Heu_GT_DFS	98 - 1 - 1 51.73 - 18.47	84 - 1 - 15 41.10 - 24.68	Heuristic + Minmax	-	-
MCT	94 - 2 - 4 49.19 - 19.55	66 - 2 - 32 41.65 - 31.38	41 - 2 - 58 37.55 - 41.08	Monte Carlo Tree Search	-
Final Version	97 - 1 - 2 52.87 - 20.23	81 - 1 - 18 40.83 - 28.32	57 - 0 - 43 43.46 - 39.08	58 - 1 - 40 43.94 - 39.70	Final version

Table 1- The performances of our techniques against each other. (Win - Tie - Lost / Average Score)

Based on the table1 shown above, we can see our final version has the highest average winning ratio of 4 versions we have. Our first version is Heu_GT_1 which is considered the first step for each player. This version has a good performance against baseline. The winning ratio is 91%. But when we combined the Minmax and DFS method with the heuristic method, we found that the winning ratio against baseline had a great progress which is 98%. Meanwhile, we also have the experiment of Monte Carlo Tree Search. It also did well against baseline with 94% winning ratio, but not as well as Heu_GT_1. So we decided to keep deeper discovering based on the Heu_GT_DFS version.Finally, the final version has a 97% chance of winning against the naive player. Actually, both the Heu_GT_DFS and Final version performed well against the naive player, even Heu_GT_DFS has the higher winning ratio. But it could not win the final version. Considering that we need to play with many other groups, we choose the Final Version as our final tournament agent.

We have 5 competitions with other formal practice Tournaments(table2). The highest position is 2ed by using the Heu_GT_DFS version. There are lots of reasons that Final Version did not have the same position. The winning ratio of five competitions is 73%-72%-76%-82%-77% which are always stable for the practice competitions. However, the formula of the final score is "Scores + 50 * Win + 20 * Tie". Thus, winning is not the only goal. Some agents may win fewer times but have higher average scores so that they could have higher final scores.

In conclusion, to find a balance between getting more scores and winning is also an interesting topic. This balance will heavily depend on the environment and it's hard to get a general conclusion.

V. Self-reflection

Yingming Ma

Teamwork is an activity where several people work towards a common objective and it is vital to develop teamwork skills to work efficiently in a group environment. The first thing I learned from the project is communication is the most important ability to access success in teamwork. Despite being affected by coronavirus, we were not able to communicate face to face, we managed to discuss the Azul project via instant message application WeChat and video chat Zoom. During video chat, with respect and the eager to win, we brainstorm a lot of ideas and construct the foundation of our agent. Support is also a significant teamwork skill to help the whole team to get through troubles when team members felt frustrated.

Artificial intelligence is the computer science that makes computers make decisions from a given state like a human. In this project, we have multiple techniques and algorithms to choose to implement our agent, and we chose several techniques to implement to explore their performance on Azul tasks. I was assigned to implement Monte Carlo Tree Search and it is a great experience to deeply explore the underlying principle of these algorithms. It helps me to understand the advantages and disadvantages of MCTs when applying it in real projects.

The best aspect of my performance in my team is I explored Monte Carlo Tree Search and implemented the algorithm in an agent. The performance of the agent using Monte Carlo Tree Search had a 92% winning rate against the naïve player in Azul. Due to the limited time for every move, it is not possible to fully take advantage of MCTs in the beginning of the round, so we used it to select moves when the round is about to be ended, and it outperformed other algorithms in the experiments.

Reinforcement learning is also a kind of machine learning like supervised and unsupervised, and deep reinforcement learning is becoming a cutting-edge technique in artificial intelligence. The area that I need to improve the most is Reinforcement learning because I did not have much experience of implementing any model of reinforcement learning such as Q-learning and SARSA (State-Action-Reward-State-Action) and they were not used in our agent.

Jieyu Wang

Good teamwork requires us to have a good collaboration on splitting tasks and communication. In this project, we did a very good job as a team. It is necessary for every member in a group to get a design thinking committee to build trust, respect and highly efficient teamwork. In a group, we need to pay more attention on how to contribute to a group project with focusing on our unique skills. Also, communication is really important between each other. Before starting the project, we brainstorm the main goal and game rules. During the communication process, we find out some different ideas and that makes the whole project go well.

Artificial intelligence is really difficult for me to get the main points with some algorithms. However, this project offered us the chance to experiment different algorithm strategies to the real tasks. I find out for every algorithm; we need to transfer the real task to a simple traversing nodes problem. Different algorithms have different emphases, so they can achieve different functions. Especially before this project, I actually don't have much insight into heuristic algorithms.But after communicating with group members, I do have new ideas about heuristic algorithms. Moreover, this project helps me to learn how to combine different algorithms to achieve our final goal.

I am actually not good at coding because of trans-major from undergraduate school. So I just implement a simple version of dfs. However, I always have some insights about the game rules and strategies to win the game. I think that can help a lot for the project. Also, I like to record what we discussed during the meeting and put those thoughts on our paper. I also did the presentation videos for the project.

I do need to improve the coding skills and transfer what we learned about algorithms to the real task. How to realize my idea is what I need to consider in the future study. It is better to ask my group members and learn from them to achieve the coding part.

Pengbo Yan

Working in a team requests the members to split the works and combine them then. It is better to split the works with few duplicated parts and make sure the works are easy to combine, especially for the coding. Also, it is also important to split the right work to the right person, depending on the interest and the preference of each person.

After I studied a brunch of algorithms of AI, I felt that they have a very similar kernel. Basically, all the algorithm needs to search the states, but generally the number of states is almost infinite, and it is totally impossible to do a full search. Thus, there must be a method to cut the searching set so that the algorithm could end in a reasonably short time but also return some valuable results. The classic algorithms use heuristic to do this. The heuristic is defined by humans or conducted automatically within some algorithm. The advanced algorithm, whatever the MCT, Q learning, or other algorithms, also try to construct a similar function to reflect the importance of a state. The difference is that these algorithms could construct the function during searching. Moreover, this also makes me understand the human's mind, because when we play games, we also do some small searches in our mind and we also only consider some 'important' moves and expand them. There is also some 'heuristic' in our mind. When we play a game more and more, that heuristic also improves a lot. Understanding this similarity between different searches (even between humans and AI) made me very happy for studying this subject.

I am very good at programming, debugging, and solving problems. I wrote half of the codes in the final agent and helped my teammates to debug sometime. I also introduced many ideas to improve our agent.

Compared to programming, I am not good at writing reports and presentations. I think this is the area that I need to improve the most