Azul Project Report

Group 21:
Chen-An Fan [redacted]
Ruoqian Pan [redacted]
Yu-Ting Liu [redacted]

The link of the YouTube:
https://www.youtube.com/watch?v=VR053o_Obuk&feature=youtu.be

1 Introduction

Azul is a board game designed by Michael Kiesling, the basic game rules of Azul is that each player will draft tiles from factories or center to their own player board, Once all the tiles were taken, the score points will be calculated based on how the tiles were placed to decorate the palace. The player who gained the highest points at the end of the game will win the game. In this project, we will find one agent to play and compete in a tournament.

In this project, we chose blind search algorithms, heuristic search algorithms and Monte Carlo Tree Search algorithm as our agents, each algorithm has its advantages and disadvantages, compared every agent with each other, we choose Monte Carlo Tree Search as the final agent to join the tournament. The detailed descriptions are as followed.

2 Techniques

2.1 Monte Carol Tree Search (MCTS)

MCTS is an online planning method to solve MDP problems. The action selection is interleaved with action execution. Every time the agent encounters a new state, the MCTS is invoked. The tree is built incrementally by exploring states. And the reward of each state is based on future expectations.

When an action is executed, there may be several outcomes and each probability of outcome may be different. MCTS is good at dealing with this probabilistic uncertainty situation.

2.1.2 Reasons & Implementation

One reason why I chose MCTS is that the uncertainty of opponent's move leads to too many states in the Azul board game; this would cause too many computational efforts for blind search, heuristic search, and value or policy iteration. Therefore, the online and anytime properties of MCTS algorithm makes it more efficient and reasonable to use on Azul board game with the 1 second limit per step. The second reason is that the Azul board game has three important features which are suitable for MCTS implementation; they are:

- **Perfect Information**: States are always visible to all players.
- **Deterministic**: Execute the same action on the same game state always leads to the same outcome states
- **Discrete**: Players play alternately.
- **Equal Good Moves**: In many cases of Azul, there are usually multiple good moves. The neuron network model could not find a meaningful relation between them. However, the randomness characteristic of MCTS could help.

In figure 1, the red lines represent the available action of our agent, the blue lines are the available actions of our opponent, the red dots indicate the game states of our agent, and blue dots are opponent's game states. In my MCTS algorithm I only consider and optimize the score of our agent, therefore, only consider the red dots. As we can see in figure 1, after one available action (ex: action 1) is executed, it may lead to multiple possible outcomes (ex: B, C) due to the uncertainty of the opponent's move.

In my implementation, for each step, I expand the current node (node A in figure 1) by executing each action to get all children nodes (B,C,D,E,F,G,H). Then, I simulate all of the children nodes. I assume all children nodes have the same probability by assuming our opponent would randomly select an action. The simulation will end when the terminal state or timeout.

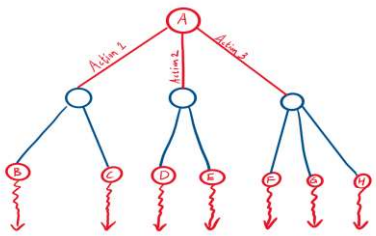

Figure 1- Game States MCTS

2.1.3 Experimental

I wrote two simulators for the experiment. One is the random move simulator which simulates the reward by executing random action. The other is the always best simulator which simulates the reward by always executes the action that can gain the most score in each move. Also, I tried different discount factors to find the best configuration.

I found that the always best simulator could outperform the random move simulator only if there is no time limitation. When there is a time limitation, the best simulator would spend too much time on finding the best action for each move, which leads to not enough time for iteration. Therefore, we only discuss the random simulator below.

I tried different discount factors and compete our agent with naive player for 100 games per configuration (Figure 2).

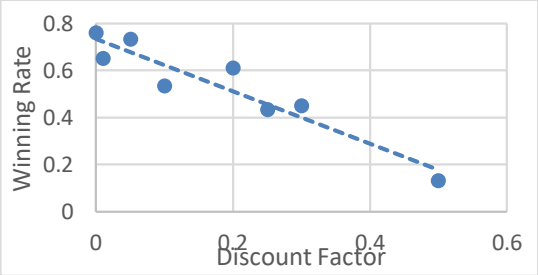

Figure 2- Discount Factor Influence

As we can see in figure 2, the larger discount factor leads to poorer performance. This tells us that we should focus on the present reward in this Azul board game to get a higher chance of winning.

2.1.4 Strengths & Weaknesses

In this section, I will discuss the strengths and weaknesses of using MCTS on our Azul board game.

Strengths:

- **Time & memory efficient**
  MCTS is an online planning algorithm, which only explores the states we may encounter.
- **Never timeout**
  MCTS is an anytime algorithm, that could stop and return the best action within a time limit.
- **Deal with uncertainty**
  MCTS algorithm can deal with the uncertainty of opponent's actions.
- **Exploit while exploring**
  MCTS could exploit the most promising states while maintaining some exploring.

Weaknesses:

- **Not learn from experience**
  The MCTS algorithms will not learn and improve when it plays more games.
- **Less robust**
  MCTS could not cover all states. Could not provide a global view.
- **Always exploring**
  In the Azul game, the probability of the same state happening is almost zero. Continuing facing new states makes the MCTS algorithm always explore and expand.

2.1.5 Improvement

My MCTS got a result of 107 wins, 6 ties, and 99 losses in the practice tournament 30-May which ranked 25 in class. This result is not good. In the future, I would not only optimize the score of our agent but also consider the opponent's score. Moreover, I would try to combine MCTS with neural networks to make my agent able to learn from experience.

2.2 Heuristic Search

2.2.1 Introduction

In this section. I choose a classical planning method heuristic search as my working agent. heuristic search algorithms are the most common search for the classical planning, the basic idea of the heuristic search is using a heuristic function to estimate remaining cost, this function is used to provide an informed way to know which node will finally lead to the goal node. By using the heuristic function, we can easily know the information about the node. In this project, we will use best-first search as our agent, best first search is an algorithm based on the bread first search, it will explore the node with choosing the node with best heuristic function value.

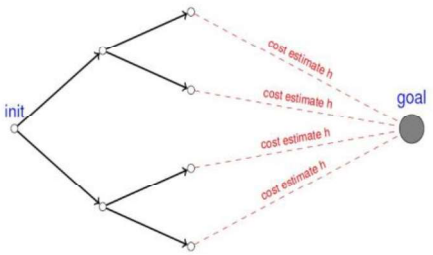

Figure 3- basic idea of heuristic search

2.2.2 Reasons & Implementation

In my opinion This game has these points that we should take into consideration when we choose the search algorithm

- **Time limit**

  For each move, the player will be given 5 seconds when a new round starts and 1 second to select the next best move, if the time runs out, the code will be terminated and a random move will be selected.

- **Generalization**

  The situation of each round of the game will be different, it is impossible to model with a fixed game situation, by the way, as Azul is a confrontation game, the other player will also try to gain the best score, the tactics of other players are different and their behavior is hard to predict, therefore, the agent must have a strong generalization ability.

- **Further consideration**

  the game's score is calculated at the end of the game, therefore, only consider one layer is not enough, the more turns we can calculate, the more accuracy the heuristic search algorithm can achieve

As for heuristic search, it doesn't need to model and train, it will explore each node while running the code, therefore, it can face each situation after the other player moves without knowing what will be faced, and it doesn't need any input, it can greatly reduce the possibility of overtime. Also, in this project, the heuristic function is easy to find, in this game, the heuristic function can be set as the sum of round score and the bonus score that each move will gain.

The heuristic function of the game will be the sum of the current score and the bonus score, when we need to choose the move, we will calculate the score,

2.2.3 Strengths & Weaknesses

In this section, I will discuss the strengths and weaknesses of using heuristic search on our Azul board game:

Strengths:

- **No Timeout**

The heuristic search will not face the situation that time is run out.

- **good accuracy**

The heuristic search win 80% races when run with the native search, and win 98% races with the random search, it means it reach a good accuracy

- **further consideration**

each node will be explored when run the heuristic search, therefore, it will provide a global review.

Weakness:

- **high space complexity**

  The heuristic search will cover all of state, that will lead to high space complexity

- **No training experiences**

  The heuristic search doesn't have any state for train and build experiences, that means, the result will only depend on the recent situation, it will reduce the accuracy.

2.2.4 Improvement

In order to improve the performance of the heuristic search, we can increase the number of layers we will search.

2.3 Blind Search

2.3.1 Introduction

Blind search algorithms use only the state, the transition function, the cost, and the goal state to solve the problem or make the planning. There is no heuristic or domain knowledge for blind search to utilize. Thus, blind search has no additional previous works on inputs. However, it's not efficient.

Breadth First Search (BFS) is one of the blind search algorithms. The structure of BFS is a search tree and it will explore the tree layer by layer. Starting from the root node, BFS will explore every child node of this root and store them into a queue (First in first out). After exploring every children of this node, BFS will pop a node from the queue and do the same procedure as mentioned above. BFS will keep doing this until the queue is empty or the goal state has been found. As a result, BFS has the property of completeness. In addition, BFS is optimal if the cost is uniform.

2.3.2 Reasons & Implementation

The reason why I choose BFS is because it's basic and has no extra work on the inputs. That is to say, it's pure and only use the information from the problem state. With

these properties, BFS is often not efficient. Thus, BFS might be suitable to become the baseline in this report.

Because states are always visible to all players. In my implementation, game states are designed as the nodes in the BFS's tree. And each layer represents one player's round because players will take turns to move. Each available move at this state for that player represents a branch connecting a child node. This child node is the game state after the player executing that specific move. In addition, the cost of each move is determined by the score change of the player. If the player is our player, the cost will become "cost – (score change)". On the contrary, if the player is not our player, the cost will become "cost + (score change)". When it comes to goal state, I have to define the goal state for BFS because there is no specific goal state in Azul. Thus, the game state with lowest total cost is defined as our goal state. The illustration of the BFS tree is shown in Figure 1.

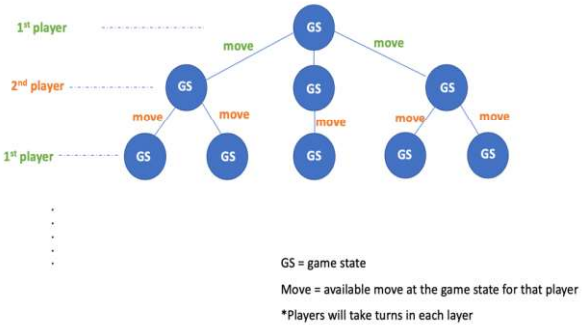

Figure 3- basic idea of blind search

2.3.3 Strengths & Weaknesses

Strength:

- Scan through all possible states because Azul has no specific goal state.

Weaknesses:

- High space complexity because our BFS will walk through all possible states.
- Timeout because our BFS will be executed once in every turn of our player which is too time consuming.
- The decision of the goal state might have some flaws because the definition of cost might not be perfect.

2.3.4 Improvement

- Set some bounds:

A. Max. 500 nodes can be explored by the BFS

B. Max. 3 layer can be explored by the BFS

These two bounds can improve a bit of the timeout issue and avoid the memory overwhelming.

3 Comparison (Evaluation)

In this part, we will compare the three agents' abilities of winning, first, every agent will against the Naïve search method for 20 rounds, From the table below, we can find that Monte Carol Tree Search (MCTS) had the highest winning scores, it won 90% of the race, Heuristics also had a good performance, it won 80% of the race and the blind search has a winning rate of 90%. All of the three algorithms are better than the basic method, that means, these three methods are all useful

Player1	Player2	Avg score	Winning rate
MCTs	BFS	60.85:38.2	95%:5%
Heuristic	BFS	44.6:40.7	65%:35%
Heuristic	MCTs	44.1:55.31	32%:64%
BFS	Naive	40.7:28.7	70%:30%
Heuristic	Naive	38.1:29.2	80%:20%
MCTs	Naive	47.9:39.6	90%:10%

Table 1- performance of the agents

Then, we let the three agents against each other, the MCTS won most of the game, it won 95% games when against BFS, and won 64% games when against heuristic search and always gained higher score, heuristic search had a better performance than the BFS, it won 65% games when against BFS. In conclusion, the MCTS is the best method among these three-search methods, it won more games and got higher scores, that's the reason why we choose MCTS as our final agent.

4 Conclusion

in this project, we used three search algorithm: blind search, heuristic search and the MCTS, by evaluating each agent, we found that the MCTS had a better performance, therefore, we chose MCTS as our final agent.

Work Allocation

MCTS: Chen-An Fan / Heuristic Search: Ruoqian Pan / Blind Search: Yu-Ting Liu

Self-Reflection

Chen-An Fan

[redacted]

In this group project, I used Monte Carlo Tree Search to implement the agent in the Azul board game and my teammates used blind search and heuristic search. In this group work, I learnt how to manage the workflows. We first held a meeting to assign jobs, then we set checkpoints to examine the progress and remind each other. I found that the checkpoints are important, this can prevent the last-minute tragedy to happen. Every week checks could also help teammates in time, boost communication, share ideas, and solve problems together. All my team members participate actively, and we conquered this task together!

Before doing this project, I only had a basic knowledge of MCTS learned from the lecture. While I did this project and implemented the MCTS, I found that I was not quite familiar with how node selection and simulate work. Therefore, I did research and even explore how the MCTS was implemented in AlhpaZero. In the lecture, we only learnt MCTS under the condition of only one agent; but after this project, I learnt how to implement it on two agents and the idea of combine neural networks with planning to let the agent learn from experience.

In my team, I managed the workflows, platform for progress sharing, the structure of our report, and most important holding the weekly check as I stated before.

As a mechanical background student, I am not familiar with python. I spent nearly one week to understand how the professor's Azul model work and what function I should call. My coding ability limited my imagination. Many of my ideas were not realized because I do not know how to write them into a program. The coding ability is what I need to improve the most in the future.

Self reflection

Ruoqian Pan

[redacted]

In this project, I used heuristic search to implement the agent in the Azul board game while my teammates used blind search and Monte Carlo Tree Search.

From this group working, I learnt several important things, first, as a group, we need to communicate frequently and always keep in touch, one meeting one week is unique, especially when the deadline is coming, we need to jointly complete the report and the video, secondly, sharing experience with other is very important, in this project, we were given a structure of a complex project, the only thing we need to do is write a agent, but we still need to understand the whole code, sometimes, I didn't know the function of a code fragment, and other group members would teach me.

This subject is my first time touch artificial intelligence in this way, it is very interesting to see AI itself can solve a question quickly without human, designing a AI and letting it solve the question is a way to make human work easier, but it is also difficult to achieve a perfect AI algorithm, we need to consider each situation, a little mistake will lead to a wrong AI.

In this project, after finishing the code, I summarized the report and write the introduction and conclusion. Also, I made part of the video recording, that's my best performance in my team.

There are many things I need to improve, I need to improve my skill of python, as a student of Mechatronics major, I never used python before this subject, therefore, it is very hard for me to write the code, at first, I wanted to use DQN search algorithm as my final algorithm, but I couldn't build the model by python, since the time is limit, I had to choose the heuristic search that I am more familiar with, that even influence the code I wrote.

Self-Reflection

Yu-Ting Liu
[redacted]

In this project, we chose three techniques such as MCTS, Heuristic search, and Blind search BFS. And I'm responsible for the BFS part.

i. What did I learn about working in a team?

As a member in this group project, I found that communication and progress arrangements are very important. This is because everyone has different aspects and working styles. With good communication, team members can grab a better understanding of each other's perspectives. As a result, we created a group chat for us to discuss ideas and progresses. When it comes to information sharing, we have a platform for progress sharing which can not only keep everyone up to date but also help us understanding others works. Last but not least, time arrangement is also critical in this project especially under COVID-19 situation. Due to COVID-19 we could only keep in touch online rather than face-to-face discussion or meeting. This made it hard for us to know other's progress. As a result, we set up strict deadline for every assigned works.

ii. What did I learn about artificial intelligence?

This is my first touch subject related to artificial intelligence. For me, it's quite interesting and exciting. When approaching the end of this semester, I have learned the concepts of classic planning, blind search, heuristic search, etc. There are many algorithms in AI and they will all perform quite differently because of their properties. Just like what we did in this project. Three techniques' performances are significantly different. Thus, choosing a suitable technique to solve the problem is important.

iii. What was the best aspect of my performance in my team?

In this project, apart from the BFS technique in our code and report, I also implemented the overall comparison of all three techniques in our project and collected the results. In addition, I contributed to the BFS part inside the video presentation.

iv. What is the area that I need to improve the most?

Several areas come up in my mind, first of all, I'm a beginner of python so my skill is not yet enough. Thus, it took me a lot of time to complete the code which might delay our team's workflow. Second, my BFS model is not efficient at all and needs to extend the time limit so that we can compare it with other techniques. I could have contributed more to the workflow arrangement.