AZUL TEAM17 PROJECT REPORT

Authors: Sejin Kim, Rajeong Moon, Nathan Batham
Presentation: https://youtu.be/QEcw861V0Ec

1. INTRODUCTION

The aim of this project was to design and implement an AI player capable of competing in a tournament based on the board game, Azul. To achieve this, three primary autonomous agents were developed and evaluated to find the best approach for submission to the competition. This report explores the design decisions and evaluation of these agents, as well as additional techniques trialled during the process, and potential improvements that could be made to the final agent.

2. TECHNIQUES EXPLORED

This section looks at the different techniques and agent models explored during this project, why they were chosen, challenges faced during their development, and their strengths and weaknesses.

2.1 Q-learning with linear function approximation

Utilising linear function approximation, the off-policy q-learning agented applied a linear combination of twelve features and weights to estimate the quality of each potential move. All features were normalised with the most heavily weighted being; the diversity of tiles in the pattern line, how many adjacent tiles on the board, and how full the destination pattern would be after the move, minus the number of tiles going to the floor. In order to store and update weights as well as the score between rounds, separate files were generated. Training was done over 75 episodes per move, with an e-greedy multi-armed bandit used to balance exploration and exploitation. During each episode a game would be simulated through to termination, with the agent using itself to simulate an opponent. At each stage the weights would be updated using the standard weight update rule, with potential-based reward shaping used to speed up training. As q-learning is an off-policy agent, during competition the agent would simply evaluate the quality of all possible next moves using the features and pre-trained weights, without any further simulation. Argmax was then used to return the highest quality move.

Justification

Given the complexity of potential move combinations, and a lack of experience with tactics for Azul, it was believed that a reinforcement learning (RL) agent would be able to better balance the importance of implemented strategies. While this approach would require training time, by using an off-policy agent the model could be pre-trained in a less time-bound environment and then simply executed during competition. In addition to this, given the large number of potential state-action combinations and that the potential moves and game states could easily be made into state-action pairs, linear function approximation was considered superior to using a q-table.

Challenges and Experimentation

While the final agent performed to an acceptable standard, experimentation was required to overcome several challenges and increase performance.

- **Features:** Feature selection was a prominent challenge that required incremental improvement during much of the early design process. By researching successful tactics for playing Azul and experimenting with trial and error, a set of twelve effective features were obtained.
- **Exploding Weights:** During early training, exploding weight values were a major challenge, resulting in complete divergence. Approaches such as weight penalties and dynamic learning rates related to simulation node depth were explored. The final solution employed a static learning rate of $\alpha = 0.001$ and an increased number of features, resulting in consistent convergence.
- **Exploration vs Exploitation:** Due to the low learning rate and to prevent overfitting, a dynamic e-greedy threshold was explored during training. It was found that by promoting exploitation early in the training there was faster convergence, and by promoting more exploration during late stage training, overfitting was reduced.

Strengths & Weaknesses

Due to the agent being off-policy, it had good time performance at competition runtime, however was quite slow to train, with weights taking a long time to adequately converge between feature changes, slowing development. In addition, the agent favoured a long-term strategy, focusing on column and set completion. This resulted in good performance against long-game agents that focused on attaining points, however inconsistent results against opponents that finished in just five rounds. Another weakness was that the agent's first move would often be scattered across the board due to the method of calculating tile adjacency, and the large weight on that feature.

2.2 Game theory

Since Azul has an extensive game form, a game theory agent was explored that would focus on maintaining a higher score than the opponent. Instead of rewarding based on the score obtained during the current turn, the agent would be rewarded based on the difference in score between players. As a result, seemingly 'sub-optimal' moves would still be considered, if they produced a greater difference in score between players.

Justification

Azul is a game played by sharing both players' status and selecting actions sequentially, and since this fits within the theory of interdependent decision making, it was believed that using game theory would be an appropriate approach for developing an agent. Being a turn-based game, the score obtained varies depending on the move of the opponent. Thus, if the agent calculated the payoff for its actions versus its opponent's subsequent actions, it could execute a strategy that consistently beat its opposition.

Challenges and Experimentation

The major challenge with the development of this agent was minimising its time complexity. Upon testing the agent on different computers, it was found that it would intermittently time out. In order to solve this, if more than forty moves were available, these would be pruned before simulating the move of the opponent. In addition, a timer was implemented that would pop the best move if the algorithm exceeded 0.9 seconds.

Strengths and weaknesses

A high score isn't required in order to win a game of Azul; therefore, a player only needs to keep one point above its opponent. As a result, the probability of winning could be increased by continuously taking actions that maintained a score gap between players. In other words, the agent did not use a strategy of filling up the pattern lines as much as possible, or to taking safe choices that minimised penalties. Instead, even if the agent itself was penalised, if the selected action increased the score gap with the opponent, the agent assumes that the action was the best action and executed it.

Since the game theory agent must find the best action for itself and that of its opponent, a large number of cased need to be considered. As such, one of its weaknesses is the large amount of computational power required, with the worst time performance occurring during the first round, where there are a large number of potential moves for both the player and opponent. Since this implementation of the Azul game imposed a one second time limit in which to make a move, it was possible that the agent would timeout. To prevent the player's move being selected at random due to a timeout, an internal timer was used to return the best move found move after 0.9 seconds. While in theory this would provide a better than random move, it is possible that a more optimal move could have been considered with greater time.

2.3 MDP with value iteration

Applying the assumption that the Azul environment was fully observable and stationary, this MDP agent was designed to find the optimal policy that maximized the expected long-term compensation. Initialising the value of each state arbitrarily, value iteration was used to update this value, based on each neighbour state. To determine the value of the next state, the difference between each player's score after the chosen movement was considered, as well as the opponent's corresponding score. To avoid timing-out, the number of moves considered was limited to 10 candidates after first being sorted based on minimising the number of tiles placed on floor line and maximising the number of tiles on the pattern line. With this approach, it was expected that the agent would discard less significant moves in favour of more valuable options.

Justification

Azul is a sequential turn-based strategy game, making it a suitable game structure for the MDP problem domain. The game's overall rules do not change; however, the best next move is based on a combination of the game state of that move and the future move of the opponent. Value iteration was therefore seen as a viable method to aid in evaluating the best action choice, as it considers the next state value when calculating the current state value. By considering the difference between the potential scores of the opponent and player, the agent could make decisions that would either maximise the point gap with the opponent when the player is winning, or minimise it if losing, leading to a higher win rate.

Challenges and Experimentation

• Time limit and data structure: The one second time-out restriction was the biggest challenge as it required developing a time-efficient agent that could still make stable and effective decisions. The current implementation limited the number of explored states after first sorting them based on the number of tiles to the pattern and floor lines. However, the number of considered states could have been increased with more efficient data structures.

• **Policy Iteration:** Policy iteration was explored as an alternative approach of finding an optimal solution. However, based on the way that the MDP agent played the game, calculating the maximum value of a move using value iteration was found to be more suitable than through policy iteration, which iterated through different policies to find the one with the maximum value.

• **Tile weighting:** Adding a weight value to similar tiles was another experiment in an attempt to increase the chance of achieving a full tile set. However, this did not significantly improve the chances of winning, and in certain circumstances resulted in undesirable moves leading to greater losses. Thus, the approach was discarded.

Strengths and weaknesses

Strength

1. If the player was losing, the agent would use the opponent's score to minimise the score difference, thus maximising the profit of an action while also minimising its risk, leading to a greater chance of winning.
2. By reordering the possible moves by their immediate potential and pruning those less favourable, the agent was still able to achieve good win rates with adequate runtime performance.

Weakness

1. Due to the timeout at one second, not all possible next states could be considered. As such, if an optimal solution existed among the discarded candidates, it could not be selected. Therefore, an optimal solution was not guaranteed.
2. Since all virtual states were not simulated until the end game, and only the current state and the next states were considered, there was no guarantee that the optimal choice of current status was linked to the maximum score that the player could obtain.

2.4 Other agents

During the process of developing the top three agents, several other techniques were explored and implemented, but ultimately abandoned in favour better performing designs.

2.4.1 Monte Carlo Tree Search

Monte Carlo Tree Search was the first agent implemented and its node structure provided a good framework upon which to build the q-learning agent. It was believed that this agent would work based on the game mechanics and the ability to; take available moves, evaluate the state and observe a reward, choose a child node, and simulate the following states through until terminal. By using an e-greedy multi-armed bandit, this would allow for an effective solution that balanced exploration and exploitation to be found and produce a result similar to that of the Minimax algorithm. This agent was eventually abandoned in favour of the q-learning agent due to its time complexity but was still used as a foundation for a q-learning approach, and as a reference against which to test future agents. If a technique such as alpha-beta pruning had been used, it is possible that the agent could have found an adequate solution within an acceptable time frame.

2.4.2 Greedy Agent (my_greedy_player)

This agent worked in greedy way, by choosing the move that would give the player maximum points after performing that move. This agent was built to use as a test case when developing the 'mdp_player'. This kind of naïve approach worked well, showing results of an 80% winning rate against the given naïve_player. Surprisingly, this agent had a higher winning ratio against naïve_player than that of 'mdp_player'. However, this agent was discard in favour of the MDP approach, as this it didn't consider the opponent's state or future possible moves when assessing the best action.

3. EVALUATING THE AGENTS

In order to conclude which agent was best suited for submission to the tournament, the performance of each agent was compared against two naive baseline agents, the test competition against the staff basic agent, and head-to-head against each other. The first baseline agent was the standard naïve_player.py, while the second baseline was a custom variant of the naïve player which simply minimised the number of tiles going to the floor. Although several versions of each agent were tested, the performance of the final agents is outlined in table 1. It can clearly be seen that the agent developed using Game Theory consistently won the most games. Although the points scored by this agent did not always have the highest average, the expected return (ER) (calculated by multiplying the percentage of wins by the average score) of the

Game Theory agent was still best in all cases. As such, it was determined that this player would be most suited for the final tournament submission.

Table 1: Win Rate, Points Average and Expected Return Over 100 Games

Agent	Naïve Player	Custom Naïve Player	Q-Learning	MDP	Game Theory	Staff Basic (5 games)
Q-Learning	67% @ 47.5 **ER = 31.83**	66% @ 46.9 **ER = 30.95**	**X**	55% @ 49.8 **ER = 27.39**	18% @ 31.2 **ER = 5.62**	60% @ 53.6 **ER = 32.16**
MDP	68% @ 40.8 **ER = 27.74**	71% @ 42.7 **ER = 30.32**	45% @ 46.9 **ER = 21.11**	**X**	21% @ 33.6 **ER = 7.06**	40% @ 46.2 **ER = 18.48**
Game Theory	90% @ 40.2 **ER = 36.18**	94% @ 44.8 **ER = 42.11**	80% @ 47.7 **ER = 38.16**	78% @ 48.0 **ER = 37.44**	**X**	80% @ 48.0 **ER = 38.40**

4. TOURNAMENT AGENT

Among the three agents developed, the game theory agent was selected as it showed the highest winning rate and expected return, achieving over 80% win rate on average. The agent would choose a strategy that prioritised the win rate over score, which performed well when winning was the main objective. However, in cases where a high score was required, the agent could underperform. In addition, the agent assumed that the opponent played greedily, always taking the move with the highest immediate reward. As a result, the payoff calculated by the agent was rarely optimal due to variances in the opponent playing style.

5. POTENTIAL IMPROVEMENTS

Since the agent heavily prioritised the winning rate, an improvement could be made by balancing this against obtaining a high score. One potential method of doing this would be to calculate the reward of a move based on maximising both points and score gap, and then if there was a tie, select the action with the median point (or score gap) value.

For example, if there were three actions which could be selected by the agent.
- If the agent selected Action 1, the agent got 5 points & the expected score gap was 1
- If the agent selected Action 2, the agent got 3 points & the expected score gap was 3,
- If the agent selected Action 3, the agent got 1 point, & the expected score gap was 5

In this situation, the current agent would have decided to take the Action 3 to increase the winning rate. However, by selecting action 2, the agent could maintain an acceptable score gap over the opponent while gaining a higher score than the current agent. Thus, obtaining both a high winning rate and score.

6. CONCLUSION

During development process, three major agents were designed using Q-learning, game theory and Markov Decision Process respectively. From the evaluations conducted and observing the test-tournaments against the teaching team's agent, the game theory agent was found to give the best performance. In game theory, the agent considers the opportunity cost and payoff of each move and the opponent's corresponding move, which was believed to be the key reason for its higher victory rate over the other two agents. During the development phase, the game theory player struggled to meet the one second time limit, which was critical to winning. However, this was handled by reducing the cases to consider and implementing an internal timer. While this agent performed well, with more improvements to its time-efficiency and ability to consider more moves, it would be expected that an even stronger agent could be built.

Self-Reflection

Student Name: Sejin Kim
Student ID: 1025560

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

The biggest thing I learned about teamwork through this group assignment was the importance of time management and cooperative communication.

The group members completed their tasks on time, and they were able to continue to update the model, develop model versions, continuously evaluate, and collaborate on difficult tasks to further develop the assignments and advance all tasks ahead of the due date.

ii. What did I learn about artificial intelligence?

Through this assignment, the lessons learned about artificial intelligence were to have more advanced knowledge of AI agents following tasks 1 and 2, and to learn about the differences and advantages of conventional heuristic search and reinforcement learning.

Another lesson learned was the flow of the decision-making process, and we learned that we can formulate more favorable strategies in the long run by formulating the actions we perform in our daily lives and its consequences.

The last thing is that there are more or less suitable algorithms depending on the domain of the given task. For example, in this task, the decision model through reinforcement learning yielded more wins than deterministic planning, and the game theory model produced a greater win than other reinforcement learning models.

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

The best aspect of my performance in my team is that I have created an agent with Markov Decision Process. I tried to adopt some different parts of other idea, and tried to upgrade the model. I always tried to communicate with group members about the agent and tried to give them feedback or adopt their feedback. I tried to give my opinion and work as a team player

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

I think I need to learn more about data structures in order to develop the learned algorithms more efficiently. And I want to upgrade my skills on using neural network libraries to get boost and build model more efficiently.

Self-Reflection

Student Name: Rajeong Moon
Student ID: 972583

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

First of all, what I learned while working as a team is that version control and exchange of opinions about collaborative work is very important. In the process of creating an agent, feedback from team members was helpful in solving unexpected errors or problems that were not discovered. In addition, we were able to further develop our code by sharing various opinions to create an agent. Also, I realized that it is very important to manage the results that team members create and modify in one place. The shared repositories allowed us to share work quickly and exchange feedback with each other to see what changed. These are helpful to complete the project successfully.

ii. What did I learn about artificial intelligence?

First of all, I was able to learn how artificial intelligence is applied in real life and what techniques are used. Before learning AI through class, I thought about artificial intelligence that mainly plays games like AlphaGo, but I realized that it was widely used in various fields such as logistic, and I became interested.

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

The best aspect of my performance in my team is that I have created an agent for use in the final tournament. I searched for the theories I dealt with in class to create a good performing agent and came up with ways to apply them. Also, in the process of developing the agent, I actively accepted the feedback from the team members and developed the agent in a good direction based on this. As a result, the results were created to satisfy the entire team. In addition, the roles and tasks to be performed as team members were completed within a given period so as not to disrupt the schedule planned with other team members. Therefore, in the next work with the team members, the work progress was not delayed.

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

Personally, I think that I should improve the mathematical part in studying the area. When studying the algorithms or contents covered in class, it took more time and a lot of effort to understand the part expressed by mathematical notation. Also, I think that more effort is needed to actually implement what I have studied in theory. In particular, I think that if I study more data structures or algorithms needed to process a large amount of computation faster, I will have better ability to understand and implement the contents covered by the AI area.

Self-Reflection

Student Name: Nathan Batham
Student ID: 762313

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

One of my biggest learnings surrounding teamwork has been how it facilitates a range of diverse ideas. By getting different perspectives and ideas on different agents, I found that I was able to improve my own work by utilising ways of doing things that I haven't previously thought of. I feel that this insight is particularly valuable as diversity of opinion is at the core of innovation. By being able to bring together and find links between different perspectives, new and creative ideas can be generated. Working in the development of medical robotics, I feel that this skill will be directly applicable to my career.

ii. What did I learn about artificial intelligence?

Coming into this subject, I had some knowledge of fundamental AI and had done some basic experience with image classification neural networks. However, I really wanted to push myself to learn more about reinforcement learning (RL), especially coming from a mechatronics background, with the potential to use reinforcement learning with robotics. Through this project I feel that I've been able to gain a solid foundation for developing basic RL agents and have the core knowledge to be able to apply these techniques to my chosen field of robotics. One example of something I learnt was how to manually select and program features to be able to get a desirable outcome using a q-learning approach with linear function approximation. While I still want to explore deep q-learning and online policies such as SARSA in more depth, I feel that this project has been an important cornerstone from which to build my practical reinforcement learning knowledge.

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

I feel that some of my most positive contributions to the team have been around organisation and project management. By making a start on the project early, putting together frameworks for the report and presentation, and organising meetings with an agenda, I feel that I have been able to provide support for my teammates while promoting consistent work on the project. Being able to further my skills in these areas is important for the future, because as I move into the workforce the ability to be organised within a team will be vital for completing work on time and to a high standard.

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

I feel that this project has highlighted two key areas that I need to work on. Firstly, it became apparent that I struggle with perfectionism when developing code, especially when a competition is involved. I found it hard to switch my focus to other tasks, such as the report, instead of continually trying to improve my agent. This insight is important as it highlights an aspect of my personality that I hadn't previously noticed and that by being aware of it, I am able to better organise my time and tasks to accommodate for it. For example, I found that by starting my day doing other work, such as the report, I was far more productive and didn't lose a whole day getting caught up in constantly working on the agent.

Secondly, I feel that I could improve on how I manage different working styles. Since completing two years of my masters, I've found that I have a tendency to get things done as early as possible, but this working style didn't necessarily suit all members of the team. As such, I pivoted to try and finish as much as I could personally do, early, then discuss realistic deadlines with the rest of the team for work that required heavy collaboration. While being on top of work is important, I believe good teamwork and project management should be able to play off the strengths of each team member's working style. As such I feel this has been an important learning in how to function effectively with different working styles, which will be vital as I move into a workforce full of diverse teams.