Reinforcement Learning Approach for Settlers of Catan

Abby Tonkinson and Evan Gray

Abstract

Catan is a strategic board game with complex properties, including multiple players, an intricate state space, and a large action space. Because of the complexity of the game, it is difficult to build reinforcement learning agents that can fully play Catan. In this paper, we explore Monte Carlo tree search agents, in addition to two different greedy agents and a random agent, to handle the wide range of actions required of the game and to show how the learning agents compare to more human-like strategies.

1 Introduction

We ran experiments on agents learning the board game Settlers of Catan (referred to as "Catan"). The central goal of Catan is to gain 10 victory points by means of building settlements, cities and other board based achievements. In a standard game, 2 to 4 players place settlements on the vertices of numbered hexagonal tiles and take turns rolling a pair of dice. If any player has a settlement adjacent to a tile with the number shown on the dice, they pick up a material/resource card corresponding to the picture on the tile. These cards can be used to build roads and settlements, upgrade settlements into cities and purchase development cards which provide potential victory points, materials and additional actions impacting other players. There is a token on the board referred to as the "Robber". This piece is initially located on a barren tile, one with no probability which yields no materials. When the Robber is moved to a different tile, the player moving the Robber may take a random card from any player with a settlement on the target tile. For the duration of the Robber being located on a given tile, no materials granted by the tile can be collected by any player who has a settlement adjacent to the tile. When a 7 is rolled, all players with more than 7 cards must discard half of their cards. Also, the Robber is moved according to the desire of the current player. The Robber may also be moved if the current player has and uses a knight development card. The first player to play 3 knights will gain the possession of the Largest Army achievement granting 2 victory points. This achievement can be taken by the first person to play the next largest number of knights. The victory points are moved to the next player with the Largest Army (2 points taken from the previous player and 2 points given to the new player). There is also an achievement worth 2 points called Longest Road. This achievement is given to the first player to build 5 contiguous roads. Similar to Largest Army, the Longest Road achievement is moved based on the current largest number of contiguous roads. Catan also has ports in set locations on the edges of the board. Players who have settlements adjacent to ports can perform trade at the printed rate during their turn. The rate is 2:1 for specified ports and 3:1 for general ports. Specified ports have a specific material of which the player must trade 2 to the board to get 1 material of their choice. General ports have no specified material so a player can trade 3 of the same material of their choice to get 1 material of their choice.

There are multiple paths to victory so based upon prior experience a player may choose to rapidly spend resources, focus on buying development cards or save resources in the hopes of building or upgrading in the future. Players may plan far in the future or choose to spend their resources as soon as they are able. Due to the risk of losing half the materials upon the rolling of a 7, many players elect to spend sooner rather than later.

We hope to evaluate the effectiveness of strategies with transfer learning by training on one board setup and testing on a new random assortment of tiles.

We aim to find an optimal strategy for winning the game, given players who have different decision processes.

2 Background Related Work

While solidifying our project idea, we found three strong pieces of related literature: Deep Reinforcement Learning in Strategic Board Game Environments by Konstantia, Xenou and Chalkiadakis, Georgios and Afantenos [1], Stergos and Playing Catan with Cross-Dimensional Neural Network by Gendre, Quentin, and Tomoyuki Kaneko [2], and Introduction to Monte Carlo Tree Search by Jeff

Bradberry.

This first paper discusses their development of a deep learning algorithm that uses action-dependent state features to approximate Q-value locally [1]. They also implemented a deep neural network with Long Short Term Memory components, so computations occur simultaneously. The algorithm trains itself during game play, so it does not require prior training experiences. The researchers tested their algorithm on Settlers of Catan to confirm their effectiveness. Although our project does not implement the same techniques as this paper, it has been very helpful in the formation of our project. Within the paper, the domain and agents they used for Catan is very clearly laid out and described. This information has helped us clearly understand the requirements of this project. The researchers used jSettler agents that use business negotiation strategies for evaluation offers and trading, and while our agents are not as advanced, it helps us understand what was necessary from our agents.

The second paper utilizes a cross-dimensional neural network, which handles a wide range of outputs and information sources [2]. Whereas the first paper uses jSettler, which is known to be the best heuristic agent available, the researchers in this paper have found that their RL agent outperforms jSettler. A big question we had encountered was how to structure the board given the hexagonal tiles and overall shape. Although this paper uses a neural network, they very clearly explain how they broke down the tiles to include spaces for possible roads and settlements/cities. This breakdown of tiles in the paper was very helpful for developing our environment. We ultimately used hexadecimal coordinates to represent our nodes and edges, but this research was very crucial in our development.

The third and final piece of related work heavily influenced our use of the Monte Carlo tree search. This resource carefully walks through the steps of the MCTS and how to implement it. The author walks through the construction of a Monte Carlo tree search agent for simple games such as Connect 4 and Checkers so we adapted the code for our implementation of the data structures and processes of Catan. We did not have a very solidified understanding of MCTS prior to this research, so it has been a strong basis for our MCTS agent.

Overall, the related work that we found has been mainly useful in the formation of our project idea. They heavily contributed to our general understanding of what was required when handling this problem. As we worked on the program, we did find other resources, which have been cited below, but they served as more of a reference point rather than a strong influence in our project.

3 Theoretical Framework and Problem Formulation

In our technical approach we built a framework of the game including board and cards such that between 2 and 4 agents of any type may take part in the program. We built the rules into the structure of the game such that illegal actions may not be taken. The overall layout of the project involves an environment object, agent objects and a main function to run the program.

3.1 The Environment

As a high level description of the environment we hope to establish the board as a randomized set of material tiles. Each board has the same number of each type of tile as included in a typical game. As a lower level description of the board it has been constructed as a hexgrid [4]. Each node and edge is represented by a hexadecimal coordinate which allows traversing between adjacent nodes/edges very easy, as seen below. The tiles are represented by the six hexadecimal node coordinates which form the tile itself. Tile information includes the probability, number and type of resource. We also have included the ports, which includes the presence of a port, material type and the trade rate. Our game is held in a board dictionary which has all of the players as keys. This dictionary hold all information about a player such as the resource cards it has, where their roads/settlements/cities are, their development cards and well as their victory points. This allows the information to be easily updated and accessible.

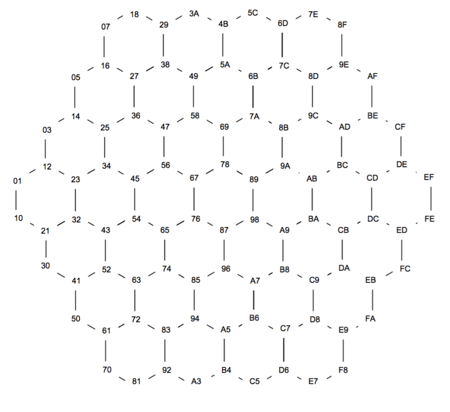


Figure A.2 Node Coordinates

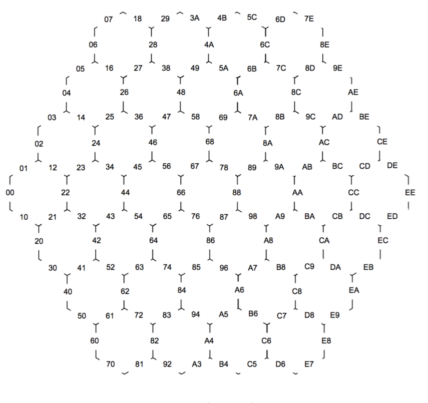


Figure A.3 Edge Coordinates

3.2 The Agents

The agents are limited to actions allowed by the game and can make decisions that a human player can make. Victory points will act as a reward so the agents should try to maximize victory points. One of our agents is Monte Carlo Tree Search which is able to plan and make decisions [3]. Monte Carlo Tree Search learns from repeated plays of the game to infer a strategy best for winning. It learns using the Upper Confidence Bound method to value each action in back-propagation. We have a greedy agent that focuses on building; cities get more victory points than settlements, and settlements more than roads. So, this greedy agent prioritizes building cities over settlements over roads. Our other greedy agent prioritizes trading and development cards. Essentially they focus on collecting resources in order to gain victory points through development cards. Finally, we have a random agent that selects a random action from the list of playable actions.

3.3 The Board

Our hope was to construct a visual of the board and create a graphical user interface to allow a human to watch the game being played or allow a human to play the agent or agents. We were unable to achieve this with the time constraint, but, would hope to implement this if we had more time. We had to make slight adjustments to the functionality of the board to ease decision processes and speed up games for the agents. We omitted player-to-player trading, differing resources in the year of plenty development card and the seven being rolled mechanic. The longest road achievement is granted to total overall roads rather than consecutive roads. We randomized the placement of numbers on tiles. We provided each agent with initialized settlement placements on the board and gave each agent 2 of every resource at the start of each game.

3.4 Problem Formulation

Our state space is the entire board as it houses all of the information needed for making decisions. Within the MCTS, the state history is used to run the simulation in order for the agent to make a decision. This is essentially a copy of the board and its states.

Our action space consists of every possible decision a player could make. This includes all buying. building, upgrading and trading. For certain actions such as ports and development cards, an agent also has to select a resource type to acquire. These decisions are also a part of the action space.

Victory points act as a natural reward system since the goal of the game is to attain 10 victory points. Since the board space is finite and games can end, we modeled Catan as an episodic task.

4 Evaluation

The evaluation for our project is the comparison of the outcome of the four agents at the completion of the game. The victory points dictate who wins in a game of Catan, so we compared the number of victory points for each of our agents. Victory points are acquired by having the longest road, having the largest army, a development card, and by building settlements and cities.

The environment in which the agents are existing is the board setup of Catan. This includes the different material tiles, built roads, the ports that allow for trade, as well as the knight. A key aspect of the environment is knowing which material tiles touch each other as well as the settlements and cities. This dictates which materials are acquired by each agent.

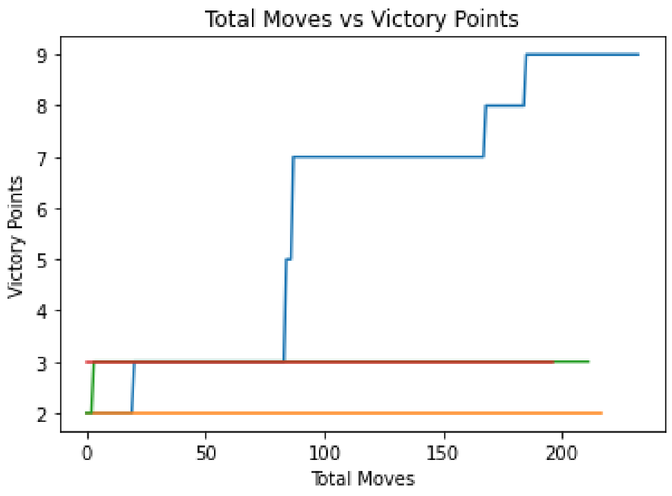
We hypothesized that the greedy agent that prioritizes building would win the game because victory points are easily acquired in building. Our greedy and random agents are more hard coded and represent more human strategies since we were unable to include an actual human player. Without a human-like comparison, we would have insight on the relative performances of the agents, but not truly understand how the agents compare to a real world game of Settlers of Catan.

5 Experiments

To test the effectiveness of the agents, we ran 100 simulated mini-games. Each player was allowed up to 5 moves (trade, build, etc.) before their turn was forced to be over. Games would be played until a winner was found or 100 total moves occurred. After each game, the board would be reset with a different random state and the agents would play from the start. We collected the number of victory points of each player after each turn.

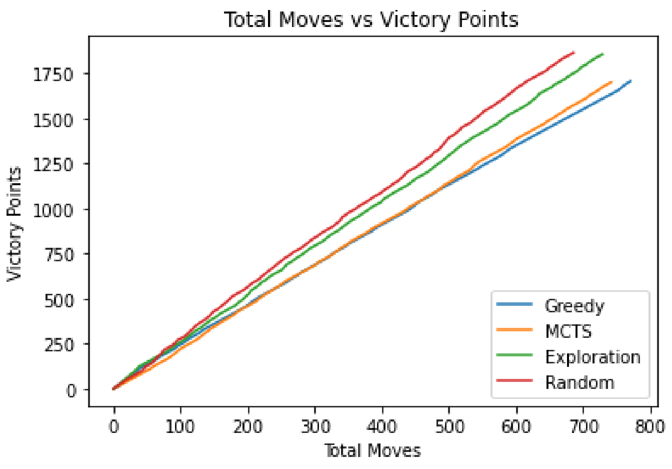
6 Results

6.1 Victory Points in One Game



This is an example of an early, full game. The greedy-building agent outperformed the other agents significantly. As you can see, it took roughly 100 moves until the greedy agent starting acquiring significantly more victory points. Given that this agent prioritizes building cities, it takes time to collect enough cards for a city. Similarly, since this agent also prioritizes building roads, it is likely that it also has the longest road card.

6.2 Cumulative Victory Points



Provided are the accrued victory points over 100 games representing the learning curve of the MCTS agent relative to the success of the other agents. It should be noted that turns are variable between 1 and 5 moves so not all agents played the same number of moves though they participated in the same games. The exploration (agent that prioritizes ports and development cards) and random agents outperformed the other two. This is a stark contrast to the results of the early game from above. We believe this great difference is because the agents learn throughout the games, and the early game is not the best representation of the agents.

7 Conclusion

The results were surprising to our suspicions as we believed the greedy or MCTS agent to be superior in total victory points. The greedy agent has a hierarchy of moves to make based on victory points so we thought this agent would perform the best. We thought the random strategy would perform the worst because of the complex nature of Catan. It can be seen that the MCTS agent starts off as the worst agent and steadily rises to be better than the greedy agent, possibly indicating successful learning.

The somewhat lackluster performance of the Monte Carlo Tree Search agent is likely due to the broad state space of the game. There are often many possible paths for the agent to travel at a given turn and this complexity is compounded by the turn based nature of the game. The agent must traverse many possible paths where victory is not guaranteed. Since the initial play-throughs of the game treated the MCTS agent as essentially a random agent, the MCTS agent had to learn itself which moves rewarded victory points. Also, the large state space meant the effect of state-action relationships was minimal. These are likely the reasons MCTS learned so slowly.

Based on the results of our experiments, more work would need to be done to improve the Monte Carlo tree search implementation. A successful agent should perform better than random and a human with minimal domain knowledge is very likely to beat all agents displayed. Although the Monte Carlo tree search implementation simulates future turns, the state space we implemented is too large to have a significant effect on learning. Humans would have a better perspective of planning such as building consecutive roads to reach a port or specific resource while the agents use the victory point aspect of settlements as a reference. Overall, humans have a detailed thought process to promote planning while agents assign estimated value to plan or play.

8 Future Work

There are a few areas of our project that we would want to improve in the future. The biggest change would be visualizing the game in some way. As mentioned, we tried making pyGames and different GUIs work, but had no luck. If we were able to spend more time on this assignment, our main focus would be to find an existing visualization compatible with our code and agents, or attempt to make our own GameMaker. We feel that having the visualization of the agents playing the game would not only be more interesting, but also provide more context for those that do not understand Catan very well. We find it easy to understand the game just from how well we know the game, but it is difficult to interpret the code output into the context of the board.

Another area for improvement is our agents. As of now, our only true learning agent is the Monte Carlo tree search. If we were able to expand, we would be very eager to use a neural network or Q-learning for an additional agent and compare that to MCTS as well as the more human-like players. We feel that this would give us more insight into the functionality of the learning agents and would only further our understanding of their strengths and weaknesses.

We would also aim to include trading between players in the future. With the time constraints we had, we were unable to allocate time to figure out trading between agents. This is a large part of the game, but it is more complex than having an agent simply make a decision. We need to have the agents actually "communicate" with each other not only to ask for certain cards, but decide whether they want to trade. If we had more time, we would really aim to understand how to achieve this trading.

Finally, we would hope to include an actual human player in the future. Although some of our agents follow human strategy, it is still not the same comparison as an actual human player would be. If we were to have the actual visualization of the game, having a human player would be much easier given the location of settlements, roads, and cities would be represented on the board rather than just coordinates.

9 References

[1] Konstantia, Xenou and Chalkiadakis, Georgios and Afantenos, Stergos Deep Reinforcement Learning in Strategic Board Game Environments. (2019) In: 16th European Conference on Multi-Agent Systems (EUMAS 2018), 6 December 2018 - 7 December 2018 (Bergen, Norway).

[2] Gendre, Quentin, and Tomoyuki Kaneko. "Playing Catan with Cross-Dimensional Neural Network." International Conference on Neural Information Processing. Springer, Cham, 2020.

[3] Guillaume Chaslot, Sander Bakkes, Istvan Szita and Pieter Spronck. "Monte-Carlo Tree Search: A New Framework for Game AI." Fourth Artificial Intelligence and Interactive Digital Entertainment Conference. October 22-24, 2008

[4] Bradberry, Jeff. ”Introduction to Monte Carlo Tree Search.” September 7, 2015

[5] Hamish, Ross. ”HexGrid”, https://github.com/rosshamish/hexgrid.

[6] Collazo, Bryan. ”Catanatron”, https://github.com/bcollazo/catanatron.

[7] ”Settlers of Catan Board” https://fathergeek.com/the-settlers-of-catan/