Yasha Ektefaie, Ian Galbraith, Joseph Palakapilly, Nirmaan Shanker
November 20, 2018

1 Abstract

The Settlers of Catan board game has a goal of earning 10 "victory points" as fast as possible by building a network of settlements, cities, and roads and buying victory points. In our implementation, a threshold based system was created, which segments which actions the player can play depending on the stage of the board game. Additionally, our agent's strategies are based off maximizing utility for their given task by conducting a first price auction system to determine the current optimal action. This implementation was able to complete a one-player game in 96.25 turns as well as compete with the Staff Bot with a 99% win rate.

2 Strategy

Settlers of Catan has a vast number of actions a player can take at each turn and thus suggests the use of heuristics for our agent. Our overall bot is an evaluator for the different possible actions we can take at a given board state.

2.1 Plan Board

At some point in the game, we will want to build settlements. Since settlements (and cities) are the only way to passively obtain resources, the placement of these buildings is key to our overall strategy. Thus, a scoring scheme was developed to determine the best locations on the board, where factors such as variety of resources at a vertex, scarcity of resources across the board, as well as the dice roll values on each resource. To find the score of a vertex, we first started by looking at the expected resources that a player would gain per turn by placing a settlement on that vertex. This is equivalent to looking at the adjacent tiles and summing the probabilities that their dice labels get rolled. However, we do not do a simple sum of the expected resources. We multiply the expectation of each resource x, E_x, by R_x, shown in Table 1.

R_{wood}	R_{brick}	R_{grain}
1.2	1.2	1

Table 1: Weights for resources

S_{wood}	S_{brick}	S_{grain}
$\frac{T_{wood}}{T}$	$\frac{T_{brick}}{T}$	$\frac{T_{grain}}{T}$

Table 2: Scarcity Multipliers

Scarcity was accounted for as follows. Let T_x be the number of tiles of resource x on the board and let T be the number of resource tiles on the board (not counting the desert). Then, the scarcity

weightings are shown in Table 2. Now, the expectation combined with these weights gives us a score, S, for each vertex on the board:

$$S = \sum_{x \in \{W,B,G\}} E_x R_x T_x$$

We return a mapping of vertex location to S which we use to plan which settlement to build next in-game.

2.2 Action Policy

Our bot's action policy is based on the decision that in a game of Catan, the most advantageous position is a network of nodes rather than the raw benefit of a single node. These optimal vertices will allow us to have access to the maximum amount of resources, which is optimal. We defined our agent by having it evaluate each possible action (city, settlement, card, and road). Each action has a utility function dependent on the expected time until the action can be completed multiplied by weights for each action. (See Table 3 for predetermined weights). To calculate the expected time for a given action, the agent will first calculate the number of resources needed for that action and using the expected value of resources gained per turn and accounting for possible trades the agent can determine how long it will take to complete its task. We conduct a first price auction then based on the scores the agent returns and choose whichever action has the largest utility value, where utility value is defined as the inverse of the expected number of turns needed to acquire the resources to complete the task scaled by the appropriate weight, and will follow the agent's action to completion, regardless if other moves can be made in the meantime.

City	Settlement	Card	Road
$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{5}$	1

Table 3: Weights for actions

However, despite the player always having four available actions, the player should not always be deciding between all four actions. For example, at the beginning stage of the game it is advantageous to focus on building more settlements and cities, as building more settlements will allow the player to increase the expected number of resources acquired per turn. To determine which actions are active we then define a set of thresholds for these actions, based on our current points, settlements, and roads we have in our current game. (See Table 4 for the predetermined states). These thresholds were determined empirically and were found to be optimal. Our action policy will finally at the end of each turn trade proactively, by trading excess resources in its hand in exchange for resources needed to complete the current action.

0,0	0,1	1,0	1,1	2,0
road		road		
city	city	city	city	city
	settlement		settlement	
		card	card	card

Table 4: Thresholds

2.3 Dump Policy

The dump policy is based upon what the next action (road, settlement, city, and card) the player is trying to accomplish. The dump policy is executed as follows:

- Determine the resources that are extraneous to building the next action. For example, if the player is trying to buy a city (which requires [0, 3, 3], where $[a,b,c]$ corresponds to [brick, wood, grain] and the player's hand is [2, 4, 2], the extraneous resources are $[2,1,0]$.
- Keep discarding from the extraneous pile based on the scarcity of resource relative to the player. The scarcity is determined by the expected number of each resource the player receives per turn.
- If more cards need to be discarded, discard from the remaining hand based on the scarcity of resource relative to the player.

The rationale for the above dump policy was to ensure that the player would still be able to best complete its action. Since our action policy is not flexible (i.e. the player cannot change its next action until it completes its current action), we want to ensure that the player keeps the cards that allows it to best complete its action.

3 Results

We tested our bot first on 1000 boards with 100 trials each and averaged 96.25 turns. We then rigorously tested the bot against itself in a similar fashion and found the bots beat their respective counterparts half of the time as expected. We then tested the bots against teach other on very scarce boards (boards with only one of each resource) and found that they can successfully finish games in around 130 turns. We then tested our bot against the staff bot on Gradescope and found we beat the staff bot 99 percent of the time.

4 Conclusion

Using the heuristics described above we have an effective bot that is able to operate on a wide variety of boards while not sacrificing optimality. Our heuristic based approach was able to operate effectively in a large state space where other AI techniques would struggle. This has been shown through its ability to not only finish on 1000s of randomly generated board, but also its above 99 percent win rate against the staff bot. Further improvements could be made through optimizing our weights and observing the global strategy of our bot in a game.