CS182 Final Project Proposal October 30, 2017

Juan C. Perdomo
Matt Goldberg
Nicholas Larus-Stone

1. Overview of the Problem

For our final project we intend to create an agent designed to play the popular card game Presidents[1]. We will have to encode the state of the card game and make sure it is playable by computer agents--however, this is not the focus of our project. The key parts of the strategy of this game involve deciding when to pass as well as which card to play. We chose the card game Presidents because it is an interesting example of a multi-player, incomplete information game. Each agent can see the cards that have been played (and by whom they were played), but it is impossible to know what cards each other agent has (as long as there are more than 2 players). This leads to a lot of uncertainty as well as a large state space--both interesting problems for us to solve.

Initially, we plan on simply on using game playing algorithms for games of incomplete information in order to design our agent. A very basic algorithm that we could implement would be minimax with a heuristic that evaluates the current strength of one's hand (similar to how one might perform minimax on a chess game). A more complicated algorithm that we plan to implement is Monte Carlo Tree Search (see Rubin). Since our state space is fairly large and the search tree has a pretty high branching factor, MCTS would be good for exploring the search tree randomly and then examining further only the moves that seem promising, as opposed to a brute-force tree search using expectimax. Having multiple types of agents will allow us to play them against each other and see which ones perform best in different situations.

After successfully designing our game-playing agents, we will create a new agent built using reinforcement learning and train it by having it play games against our game-playing agents. Doing so will allow us to compare the performance of both approaches and hopefully provide us with some insights as to how each of the individual agent can be improved. Much of our work on the RL agent will be informed by what we have talked about in class regarding RL. However, we plan to fine tune our approach to the RL agent based on the paper by Fujita, Matsuno, and Ishii. They describe their approach to building a RL agent for Hearts, and many of the same problems we are facing apply to Hearts as well. Our RL agent will view the world as a partially-observable Markov decision process and attempt to maximize its reward based on the information it has and what it expects the world to look like.

2. Expected Behavior

The expected behavior of our game-playing agents are that are able to play the game competitively; that is, if we deal the agent a hand and have it play the game, it makes rational

[1] Descriptions of the rules: http://www.pagat.com/climbing/president.html, https://en.wikipedia.org/wiki/President_(card_game)

decisions for what cards to play and when to pass, and generally performs well. In particular, we hope that our agents will be able to successfully convert strong hands into winning outcomes. The algorithms we will be implementing are intended to handle game-playing problems.

3. Issues to Focus On

While we will certainly have to spend some time designing our state representation, the most interesting parts of the project will be in investigating game-playing algorithms for selecting actions, specifying a reward function, and in evaluating performance. First, we will have to investigate different algorithms (those discussed in the papers below) for playing multi-agent, incomplete information games, weighing their pros and cons, and evaluating their performances. These include the paranoid algorithm (which treats other players as a coalition against the player making the move, reducing an n-player game to a 2-player game), the max^n algorithm (which generalizes minimax/expectimax to n players), Monte Carlo tree search (which randomly explores the game tree and then explores further the moves that are more promising), and reinforcement learning. A critical part of any of these algorithms is how they handle the uncertainty involved in playing against other players when we don’t know their hands. Another related issue that we will focus on is the choice of heuristics in the event that we do not have the computing power to search the entire tree and need to limit the search’s depth.

The second issue we will focus on is how to specify the reward function and how that affects the behavior of our game-playing agents. Since Presidents is inherently a repeated game, it sometimes makes sense to play for a sub-optimal outcome in the current game in order to put oneself in a better position for the next game. How we weight the different outcomes and different points in the game can have a great impact on how willing our agents will be to take risks; for example, if we were to only give a reward if the agent became the president for the next round, then it will be willing to risk everything to get president, whereas if we give a slightly smaller but still positive reward for vice-president, then the agent may play for vice-president instead of president if it knows it is unlikely to get the presidency (i.e., it will maximize its expected reward).

The final issue we will focus on is how we will evaluate the relative performance of different agents. For example, if we have two different agents, we may compare them by playing them against each other and seeing who wins the majority of games; however, because the initialization of the first game can have a great impact on the outcomes of repetitions of the game, we will have to be careful when we evaluate performance this way by introducing random restarts and by inspecting initial hand strengths at each game. Moreover, we can analyze the consistency of an agent by looking at whether better hands generally lead to better outcomes; for example, if an agent has a better hand than an opponent, it should be able to outperform that opponent in most cases.

The exact way we will divide our tasks is still to be determined, but generally as we do more research on different algorithms, natural preferences will arise and we will implement the different algorithms, heuristics, and systems, for the most part, independently (but according to a common interface). We have experience working on group projects together (from our CS51 final project), so we do not anticipate having any trouble working together and dividing work evenly.

4. Research and Further Readings

- Gilpin - “Algorithms for abstracting and solving imperfect information games”
- Sandholm - “The State of Solving Large Incomplete-Information Games, and Application to Poker”
- Fujita - “A reinforcement learning scheme for a multi-agent card game”
- Rubin - “Computer Poker, a review”
- Sturtevant - “Multi-Player Games: Algorithms and Approaches”