CS159 Project Proposal: Hanabi AI Bot

Litwik Anand & Audrey Huang

Introduction

Rules

Hanabi is a two- to five-player card game in which players can see others' cards but not their own. Each card has both a color and number. There are five color suits: white, red, yellow, green, or blue. Each suite has three 1's, two 2's, two 3's, two 4's, and one 5. Each player begins with four to five cards (depending on the number of players), which he holds facing outwards so that other players but not he himself can see them. The remaining cards are placed face-down in a deck. There are also eight information tokens and three fuse tokens. Play then proceeds around the table. The goal of the game is to place cards down in numerical order (1-2-3-4-5) for each of the suits, similar to Solitaire. At each player's turn, he may take one of three actions.

1. **Give information:** He can point out to another player which cards are a given number (e.g. "These two cards are 1's.") OR which cards are a given color (e.g. "Your leftmost two cards are yellow.") OR if he has zero of a number or color. The information must be accurate and complete, and can only concern either a number OR color. Giving a hint uses up one of the eight information tokens.

2. **Discard a card:** He can choose a card from his hand to discard, then draw a card from the deck to replace it. This replenishes one of the eight information tokens.

3. **Play a card:** He chooses a card from his hand to add to the cards already played. He may do this if it is a "1" in a color suit not yet played, or the next number in a suit already played (e.g. a red "2" when a red "1" is already on the table). If the player plays a card but it violates one of these two conditions (e.g. playing a green "1" if a green "1" has already been played, or playing a blue "4" when only a blue "2" but not "3" has been placed), one of the three fuse tokens is consumed. If all five numbers of a suit are played, it is considered complete and an information token is replenished. The player draws a replacement card from the deck whether the card played was successful or not.

The game ends when the deck is empty or when all three fuse tokens are consumed. The players are then given a score, where the highest number card placed for each of the color suits are added up. The highest score possible is 25, which occurs when all five suits are completed up to the number "5".

Technical Challenges

Previous games such as Go considered in this course are two-player, deterministic zero-sum games. On the other hand, Hanabi occurs in a multi-player setting where players form a grand coalition. They each have partial information and must collaborate to achieve the highest overall score possible. The initial states, or the cards the players start with, will differ from game to game. In addition, there is a probabilistic aspect because of the order in which players draw new cards from the deck. Some initial states are much easier to win from than others, usually when lower-numbered cards are drawn earlier because they must be played first. Instead of having a board with moving pieces, we instead have a large decision space where each player can choose one of three types of moves, which in turn heavily affects other players' actions in a complex manner. Each decision is constrained by the available resources, in this case information. For example, giving hints uses up information tokens and discarding cards removes valuable cards from play permanently. If players don't have enough information about their cards and there aren't enough information tokens available, they will be forced to guess at a card to play or discard a card. The former option has a probability of being incorrect, which uses up flare tokens and brings the game closer to an end. Both options can be harmful because a player could discard the only "5" of a suit, instantly lowering the highest possible score by 5,

or discard a card other players need to play their own cards. The goal will be distributing and managing information well amongst players through the hinting process in order to maximize score. On a high level, this would involve giving hints which confer the most information and discarding cards with least utility while accounting for different initial states and different deck orders. To do this, we first need to be able to represent any given situation in the game as a state, each possible decision as an action, and formulate a reward, which is not trivial given the complexity of the game.

Learning the complex network of decisions will be difficult. Despite its simple setting, Hanabi gives rise to complex human interactions. A hint not only gives information of what cards a player has, but also includes negative information about what cards the player doesn't have. The hint also affects the information other players have because a player hinting another player is always with the intent of playing a card in the long-run, which gives some information to the non-hinted players of how their cards may be played. This gives rise to established meta-strategies in human players, such as "finesse", which operate on negative information. For example, if a human player doesn't have valuable information on his cards or hints to give, he will default to discarding the oldest un-played, un-hinted card. The idea is that because it was un-hinted, other players saw that it wasn't worth giving information about and can be eventually discarded. In a "finesse", imagine that a player sees that there is a white "1" on the table, the player after him has a white "2" as his rightmost card, and the third player after that has a white "3" of the same color. The player will then hint the third player of the "3". The second player will see the "1" already in play and the "3" the other player his, then assume he has the "2" as his rightmost card, which he then plays. A "finesse" has a high amount of information transferred through a hint because it affects two players, but relies on a complex network of decisions. Though this behavior would be more difficult to learn because it involves understanding and coordinating multiple players' actions, it can maximize utility gained per token used, which would lead to higher average scores achievable in a bot.

Why This Is Interesting

Hanabi is very different from most multi-player games, the classic scenario when trying to make a game AI is that the game is a two player, zero-sum, complete information and competitive game. Hanabi is none of these so extending known techniques for those kinds of games to Hanabi will broaden the scope of application for these techniques. We also have to incorporate resource management in our AI which is another uncommon aspect of this game an something which dynamically affects the decision making for each player. Primarily Hanabi is so interesting because it is a relatively simple game which is very different from the usual class of games we tackle using MCTS and if our project is successful we will increase the range of problems one can consider solving with the MCTS algorithm.

Previous Work

Recently, researchers have approached Hanabi by simulating various strategies for two-player Hanabi. [3] There was no machine learning involved in this approach as the strategies were hard-coded for the players and repeated trials were run without any change in the strategies from game to game. The main findings of the paper was that a strategy incorporating prediction and feedback were better than those without.
However due to the fact that Hanabi is not a very well known game there isn't much work done for it specifically. Apart from hobbyists making basic Hanabi bots there wasn't much more to find.

Research Questions

Our primary research goal is writing a Hanabi bot which can consistently score above some threshold, as high as the full score of 25 as possible. We will also determine how well the bot performs with different numbers of information tokens and different numbers of players, which impact the information resources and decision space available. We will start with 2 players and canonical token numbers, then extend our model to increased players and states if time allows. We will also determine how well the bot performs with different starting rules. For example, we could hard-code discarding the oldest un-hinted card as a baseline instead

of allowing the bot to learn it without any initial assumptions. It will be interesting to see if a trained bot will be able to replicate the meta-strategies employed by human players, which maximize information value per hint. We could also extend our bot to play with a 6th "rainbow" color suit (an optional extension to the game), which belongs to all other color suits simultaneously. The existence of the rainbow cards makes the game significantly harder as one cannot specifically clue rainbow cards but they are still considered to be a separate suit so a player cannot tell if any color is just the color or a rainbow.

Plan of Attack

First, though there are online versions of the game, no downloadable API exists. However, such an API would not be difficult to write in Python because the pieces of the game itself are simple, and only needs to show the color/number of the cards in each players' hands as well as the tokens. This is the first task at hand, which we will complete in the first week.

We plan to use reinforcement learning to train a neural network policy class to play Hanabi. Specifically, we will implement on-policy, model-free SARSA. This is particularly suited to our problem because we can view a game of Hanabi as a Markov decision process (MDP). The game takes states defined by the cards played, cards in the players' hands, and tokens, and an action causes a transition from one state to another. The probability of the next state depends on only the current state, action, and deck, but not the previous ones. Each action confers a rewards determined by the value of cards which can be played. Since each action impacts future actions, we need to take into account (discounted) future reward. SARSA with neural networks will allow us to choose the action at each time which maximizes discounted future reward through maximizing Q-values and back-propagating weights without constructing an estimate of the environment. [1] [4]

We apply the SARSA algorithm over the Q-learning algorithm to learn the optimal policy for the the markov decision process which governs the game. We choose to do an on-policy model because the costs of exploration can be very high in the game. A very common scenario is discarding a key card, perhaps a card we wanted to play or the last card of a certain type thus lowering the maximum possible score we can achieve. The resource management associated with the game places very strict restrictions on the freedom players have with their moves so its generally not very safe to explore. Thus we want to generally follow the expert policy and figure out the costs of exploration as we learn. [5] Going off-policy might cause the final policy to recommend discarding key cards and generally mismanaging our limited resources even though it might be the greedy option.

We can define our reinforcement learning problem as a sextuple **(S, A, P, s(0), R,** γ**)** where S is the set of states, A is the set of actions, P is the transition probability model, $s(0) \in S$ is the initial state, $R : S \times A \to \mathbb{R}$ is the reward function, and γ is the discount factor.

Once we set up the problem we initialize the Q values arbitrarily, and generate a start state. We then choose an action a by following the expert policy we created and store the reward and new state r, s'. We again choose an action a' from our policy and update $Q(s,a) = Q(s,a) + \alpha * (r + \gamma * Q(s',a') - Q(s,a))$, for some constants (α, γ) and loop until we reach an end state. [5]

Figure 1: Pieces in the Hanabi game, showcasing the deck, five color suits (plus optional rainbow suit), numbers (1 through 5), and tokens (8 information tokens, 3 fuse tokens). Players desire to place one each of the numbers 1 through 5 in order for each color suit, similar to Solitaire.

References

[1] Busoniu L, Babuska R, De Schutter B. (2008) *A Comprehensive Survey of Multiagent Reinforcement Learning* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, VOL. 38, NO. 2, MARCH 2008. `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4445757`

[2] Claus C, Boutilier C. (1998) *The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems* AAAI-98.

[3] Osawa H. (2015) *Solving Hanabi: Estimating Hands by Opponent's Actions in Cooperative Game with Incomplete Information* AAAI-15.

[4] `http://www.nervanasys.com/demystifying-deep-reinforcement-learning/`
Links:
http://www.nervanasys.com/demystifying-deep-reinforcement-learning/ http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=

[5] `http://artint.info/html/ArtInt_268.html`