|  |
| --- |
|  |
| CITS3001 Project |
| Hanabi Agents |

|  |
| --- |
| Joshua Milambo 21985993 &  [Date] |

**Introduction**

The 2018 CITS3001 assignment requires us to research and implement relevant algorithms/methods to create two agents that can play the Hanabi card game. This report will document the design choices made for the two agents, including research into suitable techniques, implementation and validation.

**The game**

Hanabi is a cooperative card game where players work as pyrotechnicians and aim to hastily create fireworks for an upcoming festival. The game is played with coloured cards ranging from numbers 1-5 with colours of white, red, blue, yellow and green. The goal is to build 5 fireworks, where each firework is a set of each colour in correct numerical order (1,2,3,4,5). The catch of this game is that no one can look at their own cards, only everyone else’s.

A player has a choice of three options a turn:

* Give a clue
* Discard a card
* Play a card

Give a clue

If a player decides to give a clue to another player, it costs a blue counter (of which there is 8 initially). There are two types of clues allowed:

* A colour clue
  + Where you can identify the cards that have a certain colour
* A value clue
  + Where you can identify the cards that have a certain value

When giving a clue all the information is given, i.e. if a player has two white cards and another player gives a hint about the first player’s white cards, then both must be identified and not just one of them.

Discard a card

If a player decides to discard a card then a blue counter is gained, a card from their hand is put in the discard pile face up and a new unknown card is added to their hand. This action cannot be undertaken if there are already 8 blue counters available.

Play a card

If a player decides to play a card, they place it face up in front of them. If the card can successfully be added to one of the fireworks (is the number next in line for the stack), then it can be placed on that stack or the card is discarded, and a red counter is gained.

The Score

The score of the game is calculated by adding all the values of the top cards on each of the firework stacks. Using the ‘International Federation of Master Pyrotechnicians’ scale, you can then work out your Quality of Display.

**Literature Review**

The main aspects of Hanabi that need to be considered when researching implementation options is that it is a cooperative game of incomplete information. This means the agents must work together and will not be able to distinguish what the state will be like within the next several turns. This makes a competitive game search algorithm like Minmax redundant. And, makes brute force tree searches that look several turns into the future less effective because they lack information to determine the next best move (unless there is an effective enough heuristic). With this considered let’s examine possible implementation methods.

**Algorithmic Implementations**

**Heuristics**

Algorithms like A\* require heuristics to determine the next best turn to pick. They are valued on an “effective branching factor” b\*, which is number of successors generated by a “typical” node for a given search problem (Ozark Hendrix, 2018). Possible heuristics for Hanabi would be:

* Current score
* Number of useless cards in the discard pile
* Number of certainly playable cards in your hand
* Number of useless cards in your hand
* Number of Blue Counters
* Number of Red Counters

By creating an evaluation function that uses these values you can rank a set of turns.

**A\* Search**

The A\* algorithm is used to find the best path via tree traversal and is widely used in game playing agents. It is an informed search, meaning it selects the next best node depending on its estimated total distance to the goal state. It calculates this estimated total distance by adding the distance from the current node R to the next node C and the estimated distance from node C to the goal node G. For example, in Hanabi the current distance from R->C could be the number of safely playable cards + the number of safely discard-able cards. And, a heuristic value from C->G could be the how close the score is from C->G the best score of the game.

The advantage of A\* is that if the heuristic being used is admissible and the costs over a given path are monotonic then the traversal will be optimal(lecture). Which essentially means the heuristic never over estimates and the heuristic doesn’t breach the triangle inequality equation. While this is the case it may be hard to find heuristics with strong branching factors. Although likely that this algorithm will not be used, it is still an option, and could be viable with the right heuristics.

**Monte Carlo Tree Based Search**

The Monte Carlo Tree Search algorithm loops 4 steps over a given time frame to select the next best move for a finite set of moves. These steps are selection, expansion, simulation and backpropagation (FullStack Academy, 2017).

Starting from root node R, where each node is a game state (with R being the current game state), the algorithm chooses a node from the child nodes of R and expands until a leaf node is reached (an unexpanded node).When selecting, the algorithm is more likely to select the most successful child node, allowing for a better analysis of the favourable option. If the selected node is not going to bring the game to an end, expand the node and create the child nodes. Select one of these nodes and play random moves until an end game is reached record the outcome as either a win, loss or draw. This technique is also known as a playout. Use this outcome to update the parent nodes “win tally” from the leaf node back to the root node and repeat the cycle. When a time limit is reached, you select the node with the best win/loss ratio.

An advantage of this algorithm is that it will always have a result. This is advantageous for the agent because you can cap the tree search to one second to meet the game conditions of a turn, ensuring the most researched result is used. Another advantage is that it is domain dependent. This means it does not require an evaluation function (heuristic), since it instead uses the number of “victory paths” (win, loss or draw) to value each child from the root node. While this algorithm can be very efficient for some games, I can also be quite slow for others. Games with a larger number of options will create larger trees that require more time to search. This can create a suboptimal result that takes much longer to reach.

**Rule Based Implementation**

Rule-Based implementation is an effective implementation for incomplete games because it doesn’t require knowledge of future turns. There are many examples of Hanabi agents that perform just as well and sometimes better than a Monte Carlo Tree Based Search, i.e. Osawa who implemented an agent that learns more about it’s hand from how other agents play and not just hints(Osawa, 2015). The agent would recreate each possible hand, and assuming the other player has the same strategy, it would remove all the hands that are not possible for it to have based on their move. It does so by creating a hypothetical game state from the other player’s view, and since it knows the strategy (same as its own) it can make assumptions about its own hand. It could then determine the best move with this new information by following a set of rules(Osawa, 2015):

* If the player has a playable card, it plays the card.
* If the player has a discard-able card, it discards the card.
* If the other player has a playable card, and there is an information token, the player states one of the attributes (colour or number) of the card.
* If there is an information token, the player states one of the attributes (colour or number) of a card randomly selected from the other player’s hand.
* The player randomly selects one of its cards and discards it.

Another rule-based implementation was the Van den Bergh Rule, a rule set made by a genetic algorithm. Van den Bergh didn’t use probabilities to predict one’s cards like Osawa, but instead used a simpler set of rules with probabilities for actions. They were as follows (University of Essex, 2017):

* If there is more than 1 life remaining, then play a probably safe card with a 60% chance. Else play a safe card.
* Discard a probably useless card with 100% chance.
* Tell anyone about a useful card
* Tell anyone about a useless card
* Tell most information
* Discard a probably useless card with a 0% chance.

There are clear advantages to a rule-based implementation, most notably is the speed. As opposed to searching a tree, most of the time each rule only needs to search their own cards or the cards of the other players. Even with the maximum of 5 players this is significantly faster and easier to implement. These two rule-based implementations also show how you can use probabilities to get more data in an incomplete information game/take advantage of a high-risk high reward playstyle. A drawback to some rule-based implementations is that they require certain assumptions that aren’t true. For example, Osawa’s agent assumes the opposing agent will have the same strategy as itself. This will usually not be the case and will make Osawa’s agent less efficient in a game with differing agents.

**Implementation**

**Validation**

**Bibliography**

University of Essex. 2017. Evaluating and Modelling Hanabi-Playing Agents. [ONLINE] Available at: <http://teaching.csse.uwa.edu.au/units/CITS3001/project/2018/papers/MCTSHanabi.pdf>. [Accessed 15 October 2018].

Jean-Francois Baffier, Man-Kwun Chiu, Yago Diez, Matias Korman, Valia Mitsou, Andr´e van Renssen, Marcel Roeloffzen, Yushi Uno. 2018. Hanabi is NP-complete, Even for Cheaters who Look at Their Cards. [ONLINE] Available at: <http://teaching.csse.uwa.edu.au/units/CITS3001/project/2018/papers/HanabiNP.pdf>. [Accessed 15 October 2018].

Mark J.H. van den Bergh, Walter A. Kosters, Flora M. Spieksma. [UNKNOWN]. Aspects of the Cooperative Card Game Hanabi. [ONLINE] Available at: <http://teaching.csse.uwa.edu.au/units/CITS3001/project/2018/papers/hanabi-strategy.pdf>. [Accessed 15 October 2018].

YouTube. 2017. Monte Carlo Tree Search (MCTS) Tutorial - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=Fbs4lnGLS8M>. [Accessed 17 October 2018].

ozark.hendrix.edu. 2018. *No page title*. [ONLINE] Available at: <http://ozark.hendrix.edu/~ferrer/courses/335/f11/lectures/effective-branching.html>. [Accessed 16 October 2018].

Hirotaka Osawa. 2015. Solving Hanabi: Estimating Hands By Opponent's Actions in Cooperative Game with Incomplete Information. [ONLINE] Available at: <https://pdfs.semanticscholar.org/d7a7/b4158ceaa20756e9b2f577654d2da1789bc4.pdf>. [Accessed 18 October 2018].