## 1. Introduction

Hanabi is a cooperative card game with incomplete information. Each player is able to view every other player’s hand, but not their own. The goal of the game is to complete five separate stacks representing different coloured fireworks. A player has the ability to play/discard a card or give a hint to another player.

The inability to access all information within a game brings its unique challenges. This paper explores the possible strategies that help combat this limitation and explores the selected strategies for our own agents playing the game.

## 2. Literature Review Of Suitable Techniques

### 2.1 Incomplete Information And Its Effect On The Game

When participating in a game of Hanabi an agent is able to access every other player’s hands, but not its own. The agent has **incomplete information** when assessing what action will maximise total utility for the game.

Humans are able to alter their perception of their (inaccessible) hand and note which cards are seemingly worthless or of value based on the behaviour of others [Solving Hanabi]. An agent is limited in this respect.

An agent needs to acquire information from each action performed by every other agent in the game.

When a player gives a hint, it is important to recognise that this encounters a cost of one hint token. The information gained should be considered carefully. A hint gives two pieces of information. Those cards that *are*, and those cards that are *not*.

A discard can be interpreted that the player does not have enough information to play confidently or enough information has been given to discard a card.

Various articles offer different approaches to handle this restriction. [Solving Hanabi] offers an outline for a “self-recognition” strategy. This is built on top of a rule based agent as described below.

### 2.2 Rule Based Approach

When playing Hanabi, intuitively a number of rules become apparent when attempting to maximise the end score of the game.

When all fireworks are empty, any card with the value of “one” can be played

When all fireworks stacks are above a certain value, any cards below/equal to this value can be discarded

There is only a single five for each colour so they must not be discarded when in a player’s hand

2s, 3s and 4s all have duplicates – however if any one of these is discarded the remaining card must be treated appropriately.

These rules can be hard-code quite easily into a simple reflex agent. However, issues arise when introducing the concept of hints. Which hint will maximise the final payoff? How is an agent meant to interpret a hint? An agent is limited by the number of hint tokens available too.

The approach considered and implemented was that of a conservative/cautious agent which would follow a hierarchy of actions.

Play a card that is guaranteed to work

Give a hint to another players card which should maximise payout

Discard a card

Hints were prioritised by the amount of information that can be obtained from them. It was important that discarding was limited as it was a risk and could limit the maximum score obtained.

For every state, a utility was assigned to each card based on the information known. Therefore, the current agent was able to identify (based on the above rules) which of their cards and other players cards were important, and which were not.

Furthermore, hints to other players were checked against a memory bank so as to make sure agents weren’t wasting hint tokens on giving the same hints.

### 2.3 Self-Recognition Strategy

Since Hanabi is a cooperative game it is in each player’s own interest to play rationality. That is, to play so that each action attempts to maximise utility. To predict other agent’s actions, it is not unreasonable to simulate our own rational agent and record what actions it took. Any agents which are uncooperative will harm both everyone else’s utility and its own, an undesirable outcome.

The Self-Recognition Strategy assumes each agent has the same strategy. It also has a similar strategy to the rule-based agent with an additional step explored below.

Initially every possible hand our agent could have is generated. We then simulate the action the previous player would have played, given each of these hypothetical hands. If the hypothetical action is the same as the actual action the hand is kept in memory, otherwise it is discarded. This continues until all hypothetical hands have been exhausted. The two cards with the greatest chance of occurring within the agent’s hand is stored in the variables and . If then we estimate that this card is within the player’s hand. [Solving Hanabi] found that a threshold value () of 2.5 is optimal.

An agent with this strategy infers the state of their hand by using the assumed behaviour of others to interpret new information. This agent appears to act humanly.

### 2.4 Monte Carlo Tree Search

The Monte Carlo Tree Search (MCTS) has four major components.

The **selection** action involves traversing the tree from the root node to a leaf node (a node where no simulation has occurred). The child node which maximises the value returned by the UCB1 formula is chosen. This formula is used to balance between the *exploration* of nodes with minimal visits and *exploitation* of nodes with a high average value.

When a leaf node is reached the **expansion stage** adds all valid actions as children. The **simulation stage** involves randomly playing out one round of the game. The result of this is used to update information along all nodes from the respective node to the root. This process is known as **backpropagation**.

The MCTS seems attractive and viable for Hanabi for several reasons.

An evaluation function takes an input of a valid move and produces an estimate of the cost the move would incur on the game. Hints complicate the process of creating an efficient evaluation function. The definition of a good or bad hint varies between players since a hint requires some level of inference from the agent receiving the hint. The MCTS has no need for an evaluation function and needs only to be able to interpret the outcome of a game (What is the MCTS).

A greater runtime will expand the size of the tree the MCTS is traversing and improve the reliability of the average value of each node. However, the MCTS can be stopped at any time and it will still return the best move it has found so far.

The Monte Carlo tree grows asymmetrically, exploring those nodes which appear promising. This allows the algorithm to tackle problems with a high branching factor.

It would be ideal to run the MCTS with perfect information. Since this is not the case, what strategies can we employ to maximise the accuracy of the MCTS?

[An Intentional AI For Hanabi] outlines what is called the “Mental State Representation” for keeping track of possible identities for each of their cards. Every time a hint is given the information is translated to this table.

The below table demonstrates what a table would be initialised to. Every time the agent recognises that it does *not* have certain card it will reduce the corresponding value.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rank | Red | Green | Blue | White | Yellow |
| 1 | 3 | 3 | 3 | 3 | 3 |
| 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 2 | 2 | 2 | 2 | 2 |
| 4 | 2 | 2 | 2 | 2 | 2 |
| 5 | 1 | 1 | 1 | 1 | 1 |

From this we can determine the probability that a card is what we think it is (e.g. in the above table ). This does require multiple rounds to be have been played however, so enough information can be accumulated.

## Rationale Of Selected Technique

Ultimately, we settled on two agents. The primary agent implements a simple version of the outer-state strategy.

* Uses utility to prioritize playable cards and important other player cards to hint
  + Will prioritize playable cards in other players hands, if it finds none and has excess hint tokens it may give a hint that reveals the most information to a player
  + Else hint at random if too many hint tokens
  + Keeps memory of hints other players has received, able to give complimentary hints for other players needing information, will not give duplicate hints, only plays cards with certainty
* Self-recognition requires generating superset of hands and selecting subset of possible hands recursively, too much overhead
* Monte Carlo impractical for current implementation, cannot simulate other players hands as current players hand cannot be seen by itself due to restrictions=too much uncertainty as player is not allowed to see own hand and thus cannot simulate another player looking at the player’s hands, not much value in simulating round where one hand is absent, generalizing monte carlo breaks it
* When allowing risky moves with probability, you can only make as many mistakes as you have extra lives, which in this case is 2, so you’re not really gaining much from it, possible gain in score is limited by the shared limit of 2 mistakes per game across 3-5 players

## Implementation Description

## Validation

- Consistency between conservative and advanced agent compared to basic agent

- Substantial improvement compared to the basic agent

- Conservative + Advanced agent scored similar scores – conservative performed slightly better at 4 while advanced performed slightly better at 5 players

- Improvement in performance is down to the core basic rules

- Non-determinism (randomness) is to account for lack of perfect scores

- Advanced Agent additional rules were “fringe cases”? Conservative Agent depends on a good starting game?

Solving Hanabi: Estimating Hands By Opponent’s Actions In Cooperative Game With Incomplete Information

(ARTICLE)

An Intentional AI For Hanabi

(ARTICLE)

Monte Carlo Tree Search

<https://www.youtube.com/watch?v=UXW2yZndl7U&list=WL&index=5&t=0s>

What Is The Monte Carlo Tree Search

<https://blog.theofekfoundation.org/artificial-intelligence/2016/06/27/what-is-the-monte-carlo-tree-search/>