The 2018 *Hanabi* competition

Joseph Walton-Rivers
School of Computer Science and Electronic Engineering
University of Essex, Colchester
CO4 3SQ, UK
jwalto@essex.ac.uk

Piers R. Williams
School of Computer Science and Electronic Engineering
University of Essex, Colchester
CO4 3SQ, UK
prwilliams@gmail.com

Richard Bartle
School of Computer Science and Electronic Engineering
University of Essex, Colchester
CO4 3SQ, UK
rabartle@essex.ac.uk

***Abstract*—This paper outlines the *Hanabi* competition, first run at CIG 2018, and returning for COG 2019. *Hanabi* presents a useful domain for game agents which must function in a co-operative environment. The paper presents the results of the two tracks which formed the 2018 competition and introduces the learning track, a new track for 2019 which allows the agents to collect statistics across multiple games.**

I. Introduction

Hanabi is a co-operative, partially observable card game designed by Antoine Bauza[1] and first published in 2010. The card game won the prestigious 2013 Spiel des Jahres[1] award and has received recent interest from academics as a domain for games research.

The *Hanabi* AI competition was held at CIG 2018. The competition consisted of two tracks: the mirror track where all agents were following the same strategy, and the mixed track where the agents did not know the identities of the other agents they are paired with. This second track has interesting implications for co-operative team-based games, where players are expected to form teams on an ad-hoc basis.

There are several properties which make the game an interesting domain for game AI research:

Firstly, the game is turn-based, so a generous limit on the time available to make a move can easily be arranged.

Secondly, the game features partial observability — a characteristic present in many games. The nature of *Hanabi*'s partial observability also introduces a requirement for the agents to co-operate, as they cannot make informed decisions about their actions without the co-operation of the other agents.

Thirdly, the means for co-operation is explicit and well-defined as part of the game mechanics. The rules of the game restrict out-of-band communication and therefore save agents from having to understand complex communications and researchers from having to invent an agent-specific communication mechanism.

[1]https://www.spiel-des-jahres.com/en

Lastly, the game's rules are fairly simple and well-documented, which means that researchers can focus on creating better co-operative agents rather than on creating general agents capable of dealing with complex game features or unknown domains. Furthermore, the rules are particularly conducive to creating agents that possess a theory of mind.

A. Knowledge

Hanabi is a co-operative game in which players do not have full information about the game state. They can see what cards are in the hands of other players but do not know what cards are in their own hand; nor do they know the order of the cards remaining in the deck. The players gain limited knowledge of what the unknown cards are through the communication actions undertaken by other players.

Such partial observability is a property of many games. Algorithms that can cope with gaps in knowledge are therefore required for the creation of better game-playing agents.

B. Believable Agents

Hanabi players focus on *why* other players take the actions they take. This modelling of other players' intentions is an important feature of play, and one of the reasons human players rate the game so highly (it won the prestigious Spiele des Jahres award in 2013). As most *Hanabi* strategies focus on understanding how the other players will interpret what we tell them, an understanding of how our actions are perceived is important for the creation of *Hanabi* playing agents. The authors hope that the techniques developed for *Hanabi* playing agents, therefore, might be generalisable to other domains that feature this requirement.

From an AI perspective, the game requires that the agents (players) model the effects of a given action on the other players. This is a common feature of co-operative domains, but is not typically a strong component of game AI; in particular, the agents involved are often criticised for lacking social believability [2]. Eger et al. found that players who played a 2 player online variant of *Hanabi* preferred to play with agents that exhibited intentional behaviour [3].

Agents within *Hanabi* must collaborate in order to complete their common objective. The work presented in [4] performs well when a model of the other agents is available, but models of the other agents might not be available. This could be because another agent's decision-making processes are opaque (as with human players), or because the agent's identity is unknown (as with networked bot players), or because the other agent's decision-making processes are known but are too computationally-expensive to be practical (as with Monte Carlo Tree Search (MCTS)).

The remainder of this paper will be structured as follows: Firstly, section II will provide a background of both research on *Hanabi* and other relevant co-operative games competitions. Next, section III will outline the game rules. This will be followed by a description of the competition (section IV) and a description of the submitted agents by entrants (section V) and the scores they obtained (VII). The paper will then outline the changes made for the 2019 competition (VIII). Finally, section IX will provide conclusions and future work.

II. Recent Research

A. *Hanabi*

There have been many research papers published on *Hanabi* as a game and agents for playing the game. Some mathematical analysis of the game has been conducted, and the game has been shown to be NP-Complete by Baffier, Chiu, Diez, *et al.* [5], this is also the case if all hidden information is removed.

There have been multiple attempts to create rule-based agents. Osawa created a set of agents for the two-player variant of the game, including agents that include an agent that only considers their internal state, an agent that includes what the other agent knows to avoid providing duplicate information and an agent that attempts to figure out information about the game state based on previous moves made by the other player. The work by Bergh, Hommelberg, Kosters, *et al.* [7] explores a set of rules to create agents. They perform experiments on the 3 player variant of *Hanabi* and analysed the effectiveness of their created rules. They found that taking some risks when playing cards had better results than playing cautiously.

Cox, De Silva, Deorsey, *et al.* created agents that used a 'hat-guessing' strategy to play *Hanabi* near perfectly. The agents can achieve consistently high scores, but this strategy assumes that all players are playing the same strategy. This is limiting for co-operative games research, and especially problematic when human players are involved. The strategy only works with 5 player games due to the encoding system used. Bouzy [9] relaxed the rules to allow players to point out a lack of cards of a given colour or rank. This allowed an improvement in the scores obtained by Cox, De Silva, Deorsey, *et al.* They also combined the hat guessing agent with a tree search technique to further improve the scores.

There has also been research into using *Hanabi* playing agents with human players. Eger, Martens, and Córdoba [3] used a rule-based agent based on the work of Osawa to create an intentional agent that attempts to provide intelligent hints based on information that will be useful to their human counterpart. They found that humans preferred to play with agents that exhibited intentional behaviour.

More recently, Bard, Foerster, Chandar, *et al.* has proposed the use of *Hanabi* as a domain for agents based on machine learning techniques [10]. They reference some of the same reasons outlined in the motivation section (creating agents that possess a theory of mind). Their research cites not only the existing agents from academic literature but also rule-based techniques sourced from public source code repositories. Within the field of machine learning algorithms, Foerster, Song, Hughes, *et al.* has created a reinforcement learning technique that is capable of dealing with multi-agent environments that feature communication and has applied it to *hanabi* [11].

B. *Co-operative Competitions*

The General Video Game AI (GVG-AI) competition [12] recently introduced a two-player track [13]. This features a mix of both co-operative and competitive two-player games, but most of them are competitive games. Within this competition each agent is paired with another, unknown, agent and must play a series of games (some of which are co-operative and some of which are competitive); the aim is to obtain the highest possible score across all agents. The agents must, therefore, be competent both at competitive and at co-operative games. The agents must also be able to function across a range of unknown games; they can evaluate the effects of an action on a given state by simulating the action using a forward model. The agents do not know if the game they are playing is cooperative or competitive and as a majority of the games are competitive, it is better to attempt to minimise opponent's scores rather than maximise them. The framework is also based on simultaneous move games in which the controllers are expected to return a move within 40ms. While general game playing agents are useful, the competition's focus is not on creating agents that are meant to co-operate with each other. The focus on *Hanabi* allows for rule-based agents that take domain knowledge into account as well as knowledge of existing strategies into the agent's design.

Other game domains that look at co-operative game AI include the geometry friends competition [14], in which two agents with asymmetric action sets attempt to solve physics-based puzzles together. The agents that compete in this domain must be able to cope with both co-operation and real-time physics. The competition organises a separate single player track were agents focus on attempting to solve the physics-based puzzles without the complexity of dealing with co-operative actions. This results in efforts being split between creating agents that can correctly function in the physics-based environment and focusing on co-operative gameplay strategies. The turn-based nature of *Hanabi* allows the agents to focus more on the interaction between agents rather than the game rules themselves.

The Ms. Pac-Man vs Ghost Team competition [15] pits a single controller (Ms. Pac-Man) against a team of four possibly non-identical ghost-controllers. The agents have limited visibility of the game world centred on their location

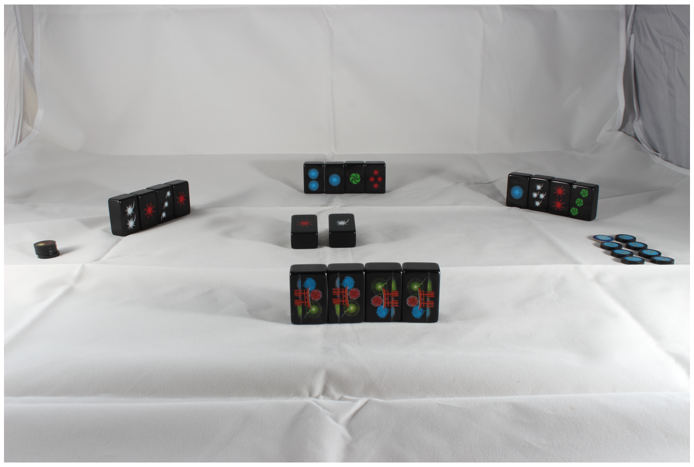
Fig. 1. A mid-game *Hanabi* state. We can see the other players cards but not our own. The life tokens are located in a stack on the left of the image, the information tokens on the right. The centre shows that a red 1 and a white 1 have been played

and as a result, must coordinate with each other in order to track and trap Ms. Pac-Man. The ghosts can make use of a communication channel, which allows them to exchange location information about Ms. Pac-Man (with a delay). The ghost agents are submitted as a team, which means the ghosts only need to work as a fixed team of agents and do not need to work with unknown agents.

III. Domain Description

Hanabi is a co-operative card game that can be played by 2-5 players. The game consists of a deck of cards containing 5 coloured suits (red, yellow, green, blue and white), each suit contains (three 1s, two 2s, two 3s, two 4s and one 5). The game also contains two types of tokens: information tokens (maximum of 8) and life tokens (maximum of 3). The players always start with the maximum number of both tokens.

The objective of the game is for the players to form sets of cards of each suit, in rank order from the lowest value (1) to the highest value (5). For completing a suit, the players also get an additional information token (if they have less than the maximum 8 tokens).

Each player has a hand consisting of a set of cards. The player cannot observe their cards, but the cards in other player's hands are visible to them. The number of cards in each player's hand depends on the number of players, for 2 and 3 player games each player has 5 cards, for 4 and 5 player cards, they have 4 cards each.

The game is played in turns, with each player performing a single action during their turn. Three types of action that can be performed:

- **Tell Information**: Spend an information token to tell another player about cards in that player's hand (pointing out all cards of a given rank or suit). A tell action must always be complete (players must identify all cards or a given rank or value in that player's hand, not a subset of them) and the information must be accurate (players cannot lie to another player about their cards). The player must also be able to point to at least 1 card (pointing out 'you have no red cards' is not permitted).
- **Discard a card**: Remove a card from their hand, placing it in the discard pile and gaining one information token (up to the maximum of 8). The player then draws a replacement card from the deck. If the team already has 8 information tokens, discards are not permitted.
- **Play a card**: Play a card from their hand, if the card is the next in sequence for that suit (based on the current card of that suit on the table), the card is placed on the table. Otherwise, the players lose a life token, and the card is placed in the discard pile. The player then draws a replacement card from the deck.

When the deck runs out of cards, each player gets 1 last action. The score is then calculated. The game also ends if the players run out of life tokens or the maximum score of has been reached. As every action either causes a card to be drawn from the deck or spends an information token (of which there are a finite amount) the game is guaranteed to end.

A. Scoring

The game ends when one of the following conditions are met:

- the players have completed every suit (thus obtaining a perfect score of 25)
- If the players are unable to do this, the game ends when either they run out of cards (as described above)
- the team runs out of life tokens[2]

The score obtained is the total of the top cards of each suit on the table (i.e. the highest value of each suit that was correctly played). For example, if the table at the end of the game had the following cards displayed:

Red	Yellow	Green	Blue	White
5	3	1	4	2

Then the final score obtained would be 15 $(5+3+1+4+2)$.

IV. The 2018 Competition

The competition is built using the *Hanabi* implementation presented in [4]. The framework is written in Java and designed around a client-server model to allow for network play and greater isolation between agents. The 'server' scores a fully-observable copy of the game, during its turn, an agent is queried for its move, and then the effects of that move are sent to all of the agents. Events that should not be observed by a particular agent (for example, telling the current player what card they drew) are not transmitted to that agent.

The framework has been made publicly available (under the GNU General Public Licence version 3) and is available (along with source code) from Maven Central. For each game played, the total score obtained, the number of lives remaining

[2]In our framework, we have interpreted losing all of lives to count as the score obtained so far, it has been suggested that this should award a score of zero. For consistency with last year's competition, we have left this rule in place

at the end of the game and the total number of turns taken are recorded.

As the deck order can affect the maximum possible score, it is important that agents are evaluated using the same random-number seeds. The seeds for each run of the agents are fixed ahead of time. Note that many seeds can correspond to the same deck ordering, although this is extremely rare, we filtered the seeds to remove duplicate deck orderings.

Note also that deck orderings can be different but similar, for example, the red and blue cards in one deck could be the blue and red cards in another. Such similarities could be detected and removed, but because agent strategies can legitimately privilege one suit over another (preferentially telling about red cards for example), we do not check for similar deck-orderings.

Agents can be time-limited using a configurable time limit present in the framework. For the competition, agents are expected to return a move within 40 milliseconds; failing to do so results in agent disqualification (no further moves are permitted) and the agents obtain the score that they have obtained up until that point. For network play, this restriction is relaxed to prevent agents from being disqualified due to network lag. For our experiments, we run the agents within the same JVM and so can avoid potential network-latency problems.

A. Competition Tracks

The 2018 competition consisted of two tracks. In both, the agents play 250 different deck orderings, determined by random seeds. For each deck ordering the agents play the 2, 3, 4 and 5 player variants of the game. Agents were re-initialised before each game.

- **Mixed** In the mixed track, the evaluated agent is paired with $N-1$ copies of each strategy listed in section VI. The position of the evaluated agent is determined by the random seed and will be consistent for a given set of parameters (paired agent, deck ordering and the number of players). Each agent played 7,000 games in total (7 paired agents, 4 different number of players, 250 deck orderings).
- **Mirror** In the mirror track the agents are paired with copies of themselves. Each agent played 1,000 games in total for this track (250 deck orderings, 4 different number of players)

The agents for the mixed track are drawn from both the agents that are published with the framework and new agents created specifically for the competition. The exact agents that were used were not made public during the competition.

As the agents for the mirror track are all sharing the same strategy, it is important that the agents do not share any global state. This was stated in the competition rules but it is difficult to enforce pragmatically if all agents are in the same JVM.

B. Competition Framework

The agents are written in Java and must implement the Agent interface (fig. 2). This consists of a single compulsory method, doMove which must return the action the agent wishes to execute this turn. The agent is provided with a partially observable copy of the game state and the current agent ID as arguments to this method.

```
package com.fossgalaxy.games.fireworks.ai;

@FunctionalInterface
public interface Agent {

  /**
   * Standardised interface for game-playing agents.
   *
   * @param agentID the ID of this agent
   * @param state   the current state of the game
   * @return the move this agent would like to make
   */
   Action doMove(int agentID, GameState state);
}
```

Fig. 2. The agent interface

The game state provides access to information about the game state: such as the discarded cards, the current cards on the table, the number of information tokens currently available, the number of live tokens remaining, the cards the player can currently observe and the information each player has been told about their cards. The state can be copied and modified to allow the agent to reason about possible states. The framework also provides utility methods to find the legal actions for a given state and allows applying of actions to states that have been modified to be fully observable. The agent cannot easily tell the cards that are in their hand from the cards in the deck. The deck will therefore also contain cards that are located within the player's hand.

The GameState object provides successors for properties of the game, such as:

- the cards contained in the discard pile
- the information about each player's hand, including:
  - the order in which the cards were drawn
  - what the agent has been told about the cards
  - (for other agents) the cards contained in their hand
- the cards remaining in the deck
- the current information tokens
- the current lives remaining
- the value of each suit on the table
- the ability to detect if the game is over
- the ability to create a copy of the state
- the ability to modify the copy of the state, allowing the agent to make the state fully observable
- the ability to access the previous moves made, and the effects of these actions (from our perspective).

Agents can query if an action is legal in a given state by using the action's isLegal method. A full list of possible actions (and of the subset comprised of those actions which are legal) can be obtained via a utility method. If an agent attempts to make an invalid move, that agent will be allowed to reattempt a move before being disqualified by the game runner; the exact number of reattempts allowed is configurable, but the default is zero.

Disqualification results in the game engine reporting the current score obtained, but it will note that the agent was disqualified. For the competition, the score obtained until that point is used; this means that agents which are disqualified will tend to obtain lower scores.

C. Online Evaluation

As well as providing the ability to evaluate the agents offline using the framework, the competition provided a web service which allowed the evaluation of the agents under the same conditions as the final ranking. This allowed the participants to test their agents worked correctly.

The agents used for the online evaluation were fixed but different from those used for the final results. Every half an hour new seeds were generated and the agents evaluated against them. The average score was displayed on the web page for the agent.

D. Rule-based framework

Our *Hanabi* framework includes a range of controllers distributed with it. These are drawn from existing literature and our own experience.

This allows the creation of policies which consist of a sequence of rules, which will be evaluated in order until the condition for the rule matches. For a given state the rule will always report the same result when querying if it will execute for a given state. The effects of the rule may be non-deterministic (for example, choosing a random card to discard). A web-based agent creator [16] can be used to build agents using the rules present in the framework. This allows easy creation of strategies for testing.

V. Competition Agents

A. MonteCarloNN

The MonteCarloNN[17] agent is a modified version of IS-MCTS [18]. To decrease the execution time required per evaluation, the agent incorporates a learned evaluation function which is used to evaluate the game state. A variant of this agent, MonteCarloOppNN, includes a Bayesian opponent model based on policies published with the framework which attempts to detect the paired agent based on the current agent's observations.

B. NYUGameLab

NYUGameLab agent, created by Canaan, Shen, Torrado, *et al.* [19]. The agent is a combination of the rule-based agent framework supplied with the framework combined with a genetic algorithm. The genetic algorithm is used to evolve a fixed policy that was used for the competition. They also added new rules as part of their agent.

C. thunder

The thunder agent is a renamed version of the random controller. This has been left in the analysis to allow comparisons with random play.

VI. Paired Controllers

For the 'mixed' games, the controllers were paired with a set of unknown controllers. These were drawn from the set of controllers which are distrusted with the framework as well as new unknown agents. The agents that were used for the mixed track are:

A. Internal

The internal agent, originally proposed by [6]. This agent represents a reasonable but limited strategy.

1) PlaySafeCard
2) OsawaDiscard
3) TellPlayableCard
4) TellRandomly
5) DiscardRandomly

The OsawaDiscard rule implements the discard rule described in [6]. Namely, discard cards that are no longer required. This can occur either because the card is a duplicate of one that has already been played or because it is no longer possible to play the card (for example, a red 4 of both red 3s have been discarded). This rule describes a relatively intelligent discard strategy and so the rule is used for most of the rule-based agents.

B. Piers

Piers is a modified version of IGGI which was presented in [4]. The agent obtained the highest score for a rule-based agent in the original paper.

1) IfRule (lives $> 1 \wedge \neg$deck.hasCardsLeft) Then (PlayProbablySafeCard(0.0))
2) PlaySafeCard
3) IfRule (lives > 1) Then (PlayProbablySafeCard(0.6))
4) TellAnyoneAboutUsefulCard
5) IfRule (information < 4) Then (TellDispensable)
6) OsawaDiscard
7) DiscardOldestFirst
8) TellRandomly
9) DiscardRandomly

The value passed to the PlayProbablySafeCard rule determines what the rule considers an acceptable risk. The risk is calculated by taking all possible cards that could occupy a given position and calculating the subset of these which are playable. This is then represented as a value between 0 and 1 (playable cards divided by possible cards). If this value is greater than or equal to the threshold, the card is played.

C. VDB-paper

This our agent is our implementation of the best rule-based agent described in [7].

1) IfRule (lives > 1) Then (PlayProbablySafeCard(.6)) Else (PlaySafeCard)
2) DiscardProbablyUselessCard(1.0)
3) TellAnyoneAboutUsefulCard
4) TellAnyoneAboutUselessCard
5) TellMostInformation

6) DiscardProbablyUselessCard(0.0)

The DiscardProbablyUselessCard rule works under the same principle as PlayProbablySafeCard, but rather than calculating the playable cards, instead it calculates the cards that can be safely discarded.

D. Flawed

Flawed represents a player who has a 'flawed' strategy, the agent will play the most useful card even if this card is unlikely to be playable in the current situation. Another agent can prevent flawed losing lives by informing it of cards which are playable, so it prefers to play these cards.

1) PlaySafeCard
2) PlayProbablySafeCard(0.25)
3) TellRandomly
4) OsawaDiscard
5) DiscardOldestFirst
6) DiscardRandomly

E. IGGI

The IGGI agent represents a competent known player. This strategy is provided with the framework and was previously described in [4].

1) PlayIfCertain
2) PlaySafeCard
3) TellAnyoneAboutUsefulCard
4) OsawaDiscard
5) DiscardOldestFirst

F. Handcrafted Agent

This agent is a previously unknown agent created using the web interface[16]. The agent is a rule-based agent that consisted of the following rules:

1) Tell Fives
2) Tell Playable Card Outer
3) Play Finesse
4) Discard Probably Useless Card (0.7)
5) Discard Oldest No Info First
6) Discard Randomly
7) Tell Randomly

The agent is designed to represent a relatively competent strategy but has a flaw; the play rule used is designed to be used as part of a 'Finesse' move (as described in [20]. This agent was designed to represent an extreme case and scores very poorly in mirror games.

G. Evolved Agent

This agent was a previously unknown agent that was created by performing evolution on the rule-set provided with the engine. This agent is meant to represent a competent but unknown strategy. The agent is built using existing rules from the framework and prioritises playing useful cards, and discarding cards which it knows are no longer needed, followed by telling players about useful cards.

VII. 2018 Competition Results & Analysis

The results for the competition will be presented using overall mean scores (rounded to two decimal places) and then present the results broken down by the number of players per game. For the mixed track, the performance per paired agent is also shown.

A. Mirror Track

TABLE I
Overall Results (Mirror Games)

Agent	*Score*
MonteCarloNN	20.57
NYUGameLab	17.52
thunder	1.27

TABLE II
Table Type Styles (Mirror Games)

	Number of Players			
	2	*3*	*4*	*5*
MonteCarloNN	20.63	20.98	20.89	19.79
NYUGameLab	18.50	17.89	17.40	16.31
thunder	1.18	1.28	1.32	1.28

The results from the mirror track show a fairly large overall difference between the two best performing agents (table I). When the results are split by the number of players (table II in the game, MonteCarloNN performs relatively consistently, regardless of the number of players, with a minor drop-off towards a higher number of players. The NYUGameLab agent shows a more significant drop in performance in larger games. This is consistent with the results presented in their competition paper [19].

B. Mixed Track

TABLE III
Average scores over all played games (Mixed Track)

Agent	*Score*
MonteCarloOppNN	13.24
NYUGameLab	12.85
Thunder	5.12

TABLE IV
Average score split by number of players (Mixed Track)

	Number of Players			
	2	*3*	*4*	*5*
MonteCarloOppNN	13.99	14.13	12.83	12.00
NYUGameLab	13.40	13.66	12.58	11.77
thunder	1.79	4.71	6.50	7.45

The results from the mixed track show a much smaller difference in agent performance, table III shows that although MonteCarloOppNN performed slightly better, there is less than

TABLE V
AVERAGE SCORE SPLIT BY PAIRED AGENT (MIXED TRACK)

	Submitted Agent		
	MonteCarloOppNN	***NYUGameLab***	***thunder***
Flawed	3.48	3.31	1.71
IGGI	17.15	17.20	7.08
Internal	13.10	13.72	4.52
Evolved	18.71	18.06	7.68
Handcrafted	4.61	3.01	1.00
Piers	18.09	17.57	6.89
VDB-paper	17.54	17.10	6.93

a point between the agents. The thunder performed better on this track than this mirror track. This is probably due to the presence of relatively strong agents in the paired set. This is also indicated by table IV, where the score obtained in 4 and 5 player games for thunder is much higher than the score obtained in two player games. The two 'intelligent' agents show relatively steady scores across the number of players.

When the results are split by paired agent (table V), the results show that MonteCarloOppNN performed slightly better than NYUGameLab with the unseen agents (Handcrafted and evolved). The hand-crafted agent performs extremely poorly with both agents, indicating that the agent may not be capable of effective play. All three of the agents performed poorly when paired with flawed. The ordering of the other agents (IGGI, internal, piers and vdb-paper) are consistent with the ordering present in the earlier research using the framework [4].

VIII. CHANGES FOR 2019

The 2019 competition features some changes based on feedback from last year's entrants and the results presented in this paper.

A. Sample Agent Selection

The sample agents being used for the 2019 competition are the rule-based controllers presented in our first *Hanabi* paper. These more accurately reflect the range of abilities represented by the framework rules.

B. The learning track

As well as the two existing tracks, the 2019 competition features a new track, named the 'learning' track. This track provides the agents with the opportunity to adapt to the paired agent's strategies over multiple games. The track is similar to the mixed track in that agents are paired against other strategies.

A set of policies are placed into a pool, each policy is assigned a label (for example, a single letter). There may be duplicates of the a given policy within the pool, (for example, both A and C might be IGGI). Before a game starts, the evaluated agent is told the labels for each agent currently in the game (for example, player 0 is A, player 1 is D, and so on). These labels are consistent though-out the experiment. The pool of agents, their selections and the deck orderings used will be the same for all evaluated agents.

The strategies for the pool of agents will be sourced from the framework as well as previously unseen agents. The evaluated agents will not be re-created between games and therefore will be able to collect information about agent playing patterns between games.

This allows the evaluated agent more opportunities to learn from player behaviours (as the agent will be able to collect statistics over multiple games). The score for the agent will be the mean score of all games played. This track is designed to be similar to how human players learn to play the game (after playing multiple games with the same individual, you begin to learn how they play and can adjust your strategy accordingly).

C. Longer decision durations

The 2018 competition used turn durations of 40ms. This was based on other AI competitions such as the GVG-AI competition [12]. As *Hanabi* is turn-based, and real-time moves are not required, the time per move has been increased to 1 second to allow for more complex strategies. Longer times were considered, but increasing the time budget per move further would reduce the number of games that could be evaluated given the time constraints of the competition and available compute power.

IX. CONCLUSIONS AND FUTURE WORK

This paper outlined the format and motivation for the *Hanabi* competition and outlined some of the existing work in the area. Showing research both in rule-based techniques and search-based strategies. The new 'learning' track provides a greater opportunity for adapting to player styles over multiple games.

The results from the 2018 competition show that the difference between the two best performing agents is more pronounced in mirror games than mixed games. In the mirror games, the IS-MCTS search based agent was able to outperform a rule-based agent created using an evolutionary strategy. The mixed track showed less of a difference between the performance of the agents. Although the MonteCarloOppNN agent features a basic form of player modelling and the rule-based agent, this does not seem to have increased performance when compared to the other entrants. This could be due selection of the paired agents or due to the modelling techniques employed. The 2019 competition will use a wider range of agents in the paired agent set.

There are multiple areas for future work; firstly, the work presented by Bard, Foerster, Chandar, *et al.* [10] on the creation of a framework for learning agents is an exciting area of research. The creation of a 'bridge' to allow these agents to be used in the competition framework would allow for learning agents to be compared to search-based techniques. Given that the competition framework's architecture is designed around message passing, this should be possible.

The *Hanabi* rule book contains three variations around the use of a 6^{th} suit (the multi-coloured suit). The suit can be simply added to the game as another suit (meaning the game now contains 6 complete suits), increasing the maximum score to 30. In another variant, only one card of each rank in the suit is added to the deck. This means the team cannot allow any multi-coloured cards to be discarded if they wish to obtain a perfect score. Finally, the multi-coloured suit can act as 'wild' when executing a tell suit action. This means fully identifying a multi-coloured card requires 3 tell actions (one to tell value and two contradictory tell suit actions). The cards still from their own deck on the table when played. Introducing these would show the adaptability and robustness of the agents; however, this is not currently possible without changes to the engine.

Another possible variation on the learning track would be to allow offline learning by providing game logs from the paired agents ahead of time. This would be implemented in a similar way to the learning track (agents being given labels rather than their names). This would give the agents more time to process the behaviours of the players and could allow for more complex modelling techniques.

Acknowledgment

This work was funded by the EPSRC Center for Doctoral Training in Intelligent Games & Games Intelligence (IGGI) [EP/L015846/1]

References

[1] A. Bauza, *Hanabi*, boardgame, 2010.

[2] M. Johansson, M. Eladhari, J. McCoy, and H. Verhagen, "Social believability in games," in *Proceedings of DIGRA*, 2013, pp. 216–228.

[3] M. Eger, C. Martens, and M. A. Córdoba, "An intentional ai for hanabi," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, IEEE, 2017, pp. 68–75.

[4] J. Walton-Rivers, P. R. Williams, R. Bartle, D. Perez-Liebana, and S. M. Lucas, "Evaluating and modelling hanabi-playing agents," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1382–1389. DOI: 10.1109/CEC.2017.7969465.

[5] J.-F. Baffier, M.-K. Chiu, Y. Diez, M. Korman, V. Mitsou, A. van Renssen, M. Roeloffzen, and Y. Uno, "Hanabi is np-hard, even for cheaters who look at their cards," *Theoretical Computer Science*, vol. 675, pp. 43–55, 2017, ISSN: 0304-3975. DOI: 10.1016/j.tcs.2017.02.024. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2017.02.024.

[6] H. Osawa, "Solving hanabi: Estimating hands by opponent's actions in cooperative game with incomplete information.," in *AAAI workshop: Computer Poker and Imperfect Information*, 2015, pp. 37–43.

[7] M. J. van den Bergh, A. Hommelberg, W. A. Kosters, and F. M. Spieksma, "Aspects of the cooperative card game hanabi," in *Benelux Conference on Artificial Intelligence*, Springer, 2016, pp. 93–105.

[8] C. Cox, J. De Silva, P. Deorsey, F. H. Kenter, T. Retter, and J. Tobin, "How to make the perfect fireworks display: Two strategies for hanabi," *Mathematics Magazine*, vol. 88, no. 5, pp. 323–336, 2015.

[9] B. Bouzy, "Playing hanabi near-optimally," in *Advances in Computer Games*, Springer, 2017, pp. 51–62.

[10] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, and M. Bowling, *The hanabi challenge: A new frontier for ai research*, 2019. arXiv: 1902.00506 `[cs.LG]`.

[11] J. N. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling, *Bayesian action decoder for deep multi-agent reinforcement learning*, 2018. arXiv: 1811.01458 `[cs.MA]`.

[12] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C.-U. Lim, and T. Thompson, "The 2014 general video game playing competition," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 229–243, 2016.

[13] R. D. Gaina, D. Pérez-Liébana, and S. M. Lucas, "General video game for 2 players: Framework and competition," in *Computer Science and Electronic Engineering (CEEC), 2016 8th*, IEEE, 2016, pp. 186–191.

[14] R. Prada, P. Lopes, J. Catarino, J. Quiterio, and F. S. Melo, "The geometry friends game ai competition," in *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, IEEE, 2015, pp. 431–438.

[15] P. R. Williams, D. Perez-Liebana, and S. M. Lucas, "Ms. pac-man versus ghost team cig 2016 competition," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, IEEE, 2016, pp. 1–8.

[16] J. Walton-Rivers and P. Williams. (2017). Hanabi agent builder, [Online]. Available: https://hanabi.aiclash.com/builder.html (visited on Feb. 15, 2019).

[17] J. Goodman, *Re-determinizing information set monte carlo tree search in hanabi*, 2019. arXiv: 1902.06075 `[cs.AI]`. (visited on Jun. 24, 2019).

[18] P. I. Cowling, E. J. Powley, and D. Whitehouse, "Information set monte carlo tree search," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 120–143, 2012.

[19] R. Canaan, H. Shen, R. Torrado, J. Togelius, A. Nealen, and S. Menzel, "Evolving agents for the hanabi 2018 cig competition," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 2018, pp. 1–8. DOI: 10.1109/CIG.2018.8490449.

[20] B. Small. (2017). Finesse, bluff, reverse finesse - explained, [Online]. Available: https://boardgamegeek.com/thread/1309490/finesse-bluff-reverse-finesse-explained (visited on Jun. 24, 2019).