Partial Information Search Agents and Regret when Cooperating with Different Play Styles in Hanabi

Michael Brooks, 190735252
MSc Artificial Intelligence
Queen Mary University of London
m.j.brooks@se19.qmul.ac.uk
supervised by Dr Diego Perez-Liebana

***Abstract*—The card game Hanabi has offered interesting challenges in the field of AI research. The combination of cooperation, imperfect information and a restricted communication channel intersects several of the important challenges in the domain of creating effective game playing agents. Agents that can successfully play against unseen playing styles remains a difficult problem, as does explaining why a certain score is achieved. This work implements promising variants of forward planning tree search agents, demonstrating that they are more robust than rule-based agents when paired with different players and improve further if told the players' strategies. Also, a Hanabi simple action regret metric is introduced that offers a quantitative way to identify weaker agents in a team and the reasons behind good or bad team performance.**

I. Introduction

Hanabi is an imperfect information cooperative card game, created by game designer Antoine Bauza in 2010. It won the Spiel des Jahres and Fairplay À la carte in 2013, two highly accredited game awards.

While proposed originally as a physical card game, computer frameworks of Hanabi have since been created, allowing virtual play and simulations. The blend of imperfect information and the need to form assumptions as to how certain moves would be interpreted by other players identified the game as an excellent test-bed for game algorithms. Indeed, it has been described that "The combination of cooperative gameplay and imperfect information make Hanabi a compelling research challenge for machine learning techniques in multi-agent settings" (Bard et al. 2020).

Literature divides the problem of creating effective Hanabi agents into two categories. 'Self-play' (or mirror) studies creating an optimal set of agents where the strategy of all can be set. This differs to 'mixed-play' (or ad-hoc-play) where an agent must perform effectively with no control over the strategies other players use. While there have been advances in self-play agents that achieve almost perfect scores (Cox et al. 2015), (Hu et al. 2020), creating agents robust when playing under mixed-play remains an open problem.

The motivation is that mixed-play offers a problem likely to be encountered for game AI, collaborating with other players in ad-hoc settings. The insights from studying how different agents collaborate can be a way to create AI that can help weaker players achieve better scores. Creating AI able to improve a team's performance is a worthwhile endeavour.

II. Hanabi

Hanabi is a card game for between 2-5 players working together to build up five 'fireworks' of cards. The fireworks are runs of cards numbered 1 to 5 of the same colour: red, blue, green, white or yellow. The deck consists of 50 of such cards, each with a 'colour' (one of the five fireworks) and 'rank' (number from 1 to 5). Ranks are unevenly distributed in the deck; there are three 1s, one 5 and two of each other rank for each colour.

The game surface includes space for each firework, player hands (4-5 cards), a discard pile for cards used up throughout the game and the remainder of the cards form a draw pile. Resources in the game are an initial 8 hint tokens and 3 life tokens which are consumed through player actions.

Gameplay works by each player being dealt 5 cards (or 4 cards with 4 or 5 players). The remainder of the cards are placed down and become the draw pile. The novelty is that throughout, each player's hand is held facing away from them, visible to all other players but themselves. This dynamic gives rise to the incomplete information aspect of the game as a player by default has no knowledge of the cards they have. To contribute to the goal of building the fireworks, each player in turn has the option to perform one of four actions:

- Discard: Remove a card from own hand and dispose of, face up, to the discard pile. Take a new card from the deck. Gain a hint token.
- Play: Take a card from own hand, with intention of adding it onto the firework of its colour. If it is the next in sequence for the firework of that colour, it is added to the firework. An additional life token is gained if the action completes a firework (if the card is rank 5). If the card is not the next in the sequence, it is placed in the discard pile and a life token is removed.
- Hint Rank: Reveal to any other player all cards in their hand of a particular rank. A hint token is removed to perform this action.
- Hint Colour: Reveal to any other player all cards in their hand of a particular colour. A hint token is removed to perform this action

The hint action gives rise to the multi-agent cooperation nature of the game, as there is an information channel available to work together to achieve the common goal

The game ends under one of three conditions:

- All Fireworks Made: All sets of fireworks are formed (each colour set of cards ranked 1-5).
- Out of Life Tokens: All life tokens are used up.
- Out of Cards: When the last card is taken from draw pile, each player is allowed one more turn before the game ends.

If the game ends by Out of Like Tokens, players score 0. In all other cases, the sum of highest rank of each firework is the score and designates how well the team have done. Scores can therefore range from 0 to 25. Being the goal of Hanabi, research has predominantly focused on the creation of agents (players) that obtain the highest average scores. It is worth noting that the maximum score of 25 is not possible under certain deck shuffles, so a set of agents always achieving perfect scores is impossible.

The setup described is regarded as the 'classic' way to play Hanabi (RnRGames 2014) and is the go-to format for the majority of research. Other notable variants exist. Scoring can be less 'strict' and not revert to 0 on Out of Life Tokens. Empty hints (hinting 0 of a colour or rank) were originally allowed, but resulted in degenerate strategies that made the game less challenging. Other variants adjust the game setup such as adding an orange firework colour, adding a rainbow firework acting as a wildcard colour and being forced to identify all cards equal to or greater than a rank hinted. However, this work only focuses on the classic variant.

III. Search Methods

A. Monte Carlo Tree Search (MCTS)

Monte Carlo methods refers to a class of techniques involving empirical simulation to arrive at a probability estimate. Monte Carlo Tree Search (MCTS) was first theorised (Kocsis et al. 2006) and applied (Chaslot et al. 2008) as a formalised way of integrating Monte Carlo simulation into a best first search framework for strategic forward planning in games. The latter applied MCTS directly in their MANGO program to improve play in the ancient Chinese board game Go.

MCTS comprises of building a game tree where nodes represent states of the game (starting at the current state of the game) and edges represent actions that take the game from one state to another. Four processes, visualised in Fig.1, are repeated to change the structure of the tree and statistics contained in nodes. Statistics tracked by nodes are the total number of times visited and the cumulative backpropogated reward:

- Selection: A node of the tree is selected by choosing actions from the current state. If a node has unexplored actions, it will be selected. If all child nodes exist in the game tree, an Upper Confidence applied to Trees (UCT) heuristic (Kocsis & Szepesvári 2006) is maximised to decide which action a to take to descend a layer.

$$UCT(a) = R(a) + C\sqrt{\frac{logN}{n(a)}}$$

$R(a)$: average reward for visits to child node reached by action a. N: Number of visits to current node. $n(a)$ visits to child node reached by action a. C constant term balancing exploitation and exploration.

- Expansion: After a node is selected, a child node representing the state reached by applying a particular action is added to the game tree. This step requires a "forward model", a way of determining what a new state will look like after applying an action.
- Simulation: Steps in the game are then simulated from the state represented by this new node, either to a point where the game ends or to an action limit. In multiplayer games, this step may need assumptions about how other players will act, often referred to as an 'opponent model'.
- Backpropogation: A state will be arrived at by the end of simulation, with an associated reward. The reward represents how valuable that state is to the agent, so can be as simple as winning or losing, or a score achieved by that state. The reward is then backpropogated back up the game tree from the expanded node to the root; visit counts and cumulative rewards statistics of the nodes are incremented.

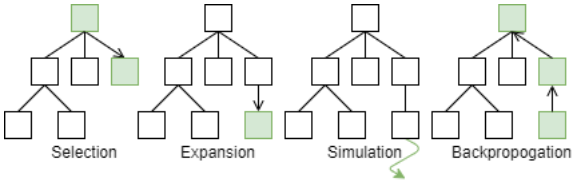

Fig. 1. Visualisation of each step of Monte Carlo Tree Search

Once completed, the action to take is the highest average reward over the immediate child nodes from the root state. There are some terminology conflicts in the literature; this paper will use 'simulation' to refer to playing out a game from a particular state and 'rollout' to mean one full iteration of the steps mentioned.

Since conception, MCTS has had widespread research in the domain of AI, for creating agents able to play games strategically. An early application was to Settlers of Catan (Szita et al. 2009), evolving a strong agent with limited domain knowledge. Further early implementations included to games like Hex (Arneson et al. 2010) , equalling performance of state of the art at the time, and Amazons (Winands et al. 2008). More recent prominent work includes playing Atari games (Guo et al. 2014) and the use of MCTS in Google DeepMind's AlphaGo program (Silver et al. 2017) that beat the current world champion Lee Sedol in a best of 5 match.

The technique has become established enough that variations and adjustments have emerged, with meta-studies published to compare these variations (Browne et al. 2012).

B. Information Set Monte Carlo Tree Search (IS-MCTS)

Imperfect information games cannot make use of the standard MCTS algorithm as a full state representation cannot be

deduced by an agent. A solution is 'determinisation', probabilistically filling in unknown information about the state at the start of each roll-out. This approach was used in Klondike (Bjarnason et al. 2009) and in the partially observable board game Magic the Gathering, developing an agent competing with or outperforming expert rule-based approaches (Cowling, Ward & Powley 2012).

A new variant, Information Set Monte Carlo Tree Search (IS-MCTS) was introduced to manage search with determinisations within a single tree. Instead of nodes in the tree representing game states, they represent a collection of possible states that could be arrived at through the sequence of actions.

It was first theorised for the game Kriegspiel, where the need to redesign the MCTS algorithm was identified: "we had to abstract the game with a model in which single states are not important, and only their perception matters" (Ciancarini & Favini 2010). The formal IS-MCTS definition was then proposed (Cowling, Powley & Whitehouse 2012) and tested on Dou Di Zhu, Phantom and Lord of the Rings: The confrontation, showing significant performance benefits compared to just determinising. More recently IS-MCTS was used to create a strong agent capable of playing the virtual card game Hearthstone (Świechowski et al. 2018).

Most relevant however, it has been applied directly to Hanabi (van den Bergh et al. 2016) and studied in greater detail (Walton-Rivers et al. 2017). IS-MCTS agents perform very poorly though. Further adjustment was needed to develop a state of the art tree search technique compatible with Hanabi.

C. Redeterminising Information Set Monte Carlo Tree Search (RIS-MCTS)

While simplifying the tree structure, there are still problems with IS-MCTS when attempting to adapt the technique to create a Hanabi playing agent. Issues with determinisation were identified and first described (Frank & Basin 1998):

- Strategy Fusion: Players must behave the same way in all possible determinations, which is often not the case when a particular determinisation is decided. The player will make use of the reveal of hidden information.
- Non-locality: Some valid possible states may never be investigated properly, in place of more immediately favourable plays under the false determinisation.

In particular there is a bias against plays that could reveal hidden information after these determinations; it was noted by Russell & Norvig (2002) that "averaging over clairvoyance will never result in plays that gather information".

Strategy fusion is the reason for poor performance of these search methods in Hanabi; the search tree is biased towards assuming player actions with high reward, but which are highly unlikely to be performed given their knowledge at that point. Information 'leaks out' from the determinisation step to the other players.

To address this, a novel extension of IS-MCTS algorithm was introduced (Goodman 2019): Redeterminising Information Set Monte Carlo Tree Search (RIS-MCTS). Here, not only is the state determinised each roll-out (player's hand randomised to another valid holding), but the state is also redeterminised at each simulation step (other player's hands are randomised to a valid holding). Information leakage is solved by disincentivising successful plays that in fact have too much uncertainty from the acting player's knowledge. After a player's turn, their hand is restored as best as possible, while allowing a played or discarded card to be retained.

D. Regret

The idea of regret was discussed in bandit-based problems, whereby an action needs to be chosen from a possible set that give different rewards (Bubeck & Cesa-Bianchi 2012). The regret is then the cumulative difference between the actions chosen and the optimal choices. The idea of simple regret was then introduced (Bubeck et al. 2009), where it is computed as the simple difference between the optimal move and the move chosen, with discussion for MCTS applications (Tolpin & Shimony 2012). Regret here is discussed in context of the goal of the selection step in MCTS to be able to guide the search to end up picking actions that minimise regret. Regret is often a theoretical concept in practice, with methods needed to approximate the regret, especially as it may not be easy to define for any given action.

IV. Hanabi Agents

There has been research since 2015 of both rule-based and search-based approaches for creating Hanabi agents. Rule-based follow logical checks and procedures, like an if else statement, whereas search-based are guided by methods such as MCTS.

A. Rule-Based

Osawa (2015) compares human play to agents they introduce. These include random actions, an inner or outer strategy retaining knowledge of previous hints about their own and other player's cards respectively and self-recognition, whereby agent estimates intention of the other player and simulates the behaviour. The results showed strategies able to estimate and simulate other player's behaviour outperformed those that didn't. The simulation used was fairly simple due to only completing one recursive simulation.

Hat principles have been used to created a set of efficient agents to build what was at the time state of the art 24.5 average score in self-play (Cox et al. 2015). The hat strategy allows hints to signal information to all players. The player set was limited (this time to 5 players and 4 cards) and inflexible to other settings. Bouzy (2017) has since generalised the concept to 3,4,5 players by .

Strategies using rules have been studied and optimised through genetic optimisation. The best agent by van den Bergh et al. (2016) (VDB) achieved a score of 15.4 average score in self-play setting.

Walton-Rivers et al. (2017) gathered rule-based agents within a rules framework and introduced three further agents: Flawed, IGGI and Piers. Flawed was for demonstration of a

bad agent, telling hints at random and playing cards with only 25% certainty. IGGI plays safe cards and offers predictability in it's discards, while Piers adapted VDB with a 'hail mary' play to risk playing cards towards the late game.

Most recently, rule-based agents have been studied under 2 player self-play and mixed-play settings (Canaan et al. n.d.), concluding that VDB and Piers were strongest agents scoring in the range 16 - 17 on average depending on which agents they were playing with.

B. Search-Based

Monte Carlo Tree search agents were investigated with reshuffling the acting player's hand, scoring 15.4 on average under self-play (van den Bergh et al. 2016). Studies have since summarised the performance of the rule-based approaches against search-based strategies (Walton-Rivers et al. 2017), concluding that agent modelling improved playing strength of tree search algorithms. An IS-MCTS agent with opponent modelling outperformed one without opponent modelling, obtaining average scores of 4 - 7 points higher on average over different numbers of players. Goodman (2019) introduced RIS-MCTS and tested it against other search variants. It was found that restricting the expansion step with branching rules, equipping a strong cooperating agents for simulation steps and using a 'playable now convention' whereby a hint about a single card should be interpreted as playable unless known otherwise, were optimal configurations achieving up to 22.0 average score in self-play settings.

C. Reinforcement Learning Based

It should be noted that the most recent state of the art techniques, particularly in self-play settings, leverage advancements in reinforcement learning. A Facebook research team developed a deep learning AI capable of state of the art self-play score of 24.61 (Lerer et al. 2020) and a study for introducing 'other-play' for the zero-shot learning task of mixed-play (Hu et al. 2020). Agents trained through Rainbow DQN architecture (a type of neural network training procedure) have been evaluated, with fairly strong self-play scores between 17 and 19 but mediocre mixed-play performance especially against rule-based agents (Canaan et al. n.d.). These results were similar to actor critic and bayesian action decoder agents, two other types of learning techniques (Bard et al. 2020).

V. Methodology

The following study investigates the performance of both rule-based and search-based agents playing against each other in 3-player self-play and mixed-play settings. The objective of the study is to see how robust search-based methods are in mixed-play, and to determine whether a metric can be created that explains why particular sets of agents are poor or effective as a team. Supporting code is publicly available (Brooks 2020). It was chosen to perform experiments using the Google Deep Mind Hanabi Learning environment (DeepMind 2018), which is a Python based wrapper for a C++ game framework, primarily designed as a reinforcement learning environment.

A. Regret

Most literature only studies scores in Hanabi, with anecdotal rather than quantitative evidence as to why a certain score is achieved. To better understand performance in games with mixed agents, an agent-specific statistic is required. A simple action regret (shortened to regret in this paper) metric is introduced for Hanabi:

$Regret = (TheoreticalMaximumScoreBeforeAction) - (TheoreticalMaximumScoreAfterAction)$

The regret is determined after each action and builds up cumulative over different turns. In all experiments, regret is tracked for an individual agent's perspective, and the Team Regret (sum of all agent regret) is also reported.

A consequence of the regret definition is that actions which are sources of regret include:

- Discard: All copies of a card are discarded. Regret is how much of the firework has been cut off from its previous maximum.
- Play: Similar to Discard, but the card was attempted to be played. Cards are discarded when not playable on a firework, so regret works the same way.
- End Game: Any failed play move when there are zero life tokens results in a score of 0. The regret in this case is simply the maximum score that was achievable before the action, as the maximum score after is 0.

A corollary of the regret definition is that the following equation holds throughout the game:

$$Score + RemainingScorePotential = 25 - TeamRegret$$

RemainingScorePotential is the number of unique cards not discarded that could still contribute to a firework (those that are not "cut off" by all of a lower rank card of the same color being discarded).

B. Rule-Based Agents

It was chosen to run experiments with rule-based agents. These are best represented in the literature and so provide a good validation of the implementation and a baseline for what can be achieved with quick, understandable agents. Most rule-based agents had already been implemented in the framework (rocanaan 2019):

- Flawed: Makes rash playing moves and random reveal moves (Walton-Rivers et al. 2017)
- Inner/Outer: Prioritises safe plays, safe discards, giving information and discard randomly. Inner only remembers hints about own hand, whereas Outer remembers hints about both own hand and others' hands (Osawa 2015).
- IGGI: Only makes safe plays and has predictable discard pattern (Walton-Rivers et al. 2017).
- Piers: Similar to VDB but equipped with 'hail mary' rule that risks late plays in the game (Walton-Rivers et al. 2017).
- van den Bergh(VDB): High performing rule-based agent formulated through genetic algorithm optimisation of rules. The rules are adapted here to consider all other

players, rather than just the next player as suggested originally (van den Bergh et al. 2016).
- Mute: Doesn't ever give hints, but otherwise plays like van den Bergh. Introduced and implemented here, although the style of an agent not able to give direct information was theorised in (Eger & Gruss 2019), but with a completely different, time-driven strategy.

C. Search-Based Agents

A RIS-MCTS agent was built from the ground up, loosely building on a skeleton MCTS structure (qpwo 2018) with the Java implementation (**?**) for reference.

1) Framework Modifications: Due to the imperfect state information shown to a player, it is necessary to be able to perform a master determinisation at the start of each rollout - converting the imperfect state to a perfect one before performing a simulation. In Hanabi, this involves being able to return a hand to the deck and redrawing random cards that are consistent with all previous information. States and the active deck are not directly modifiable in the Python wrapper of the framework, so two moves were added to the underlying C++ framework:

- Return: Allow card from any hand to be returned to the playable deck
- DealSpecific: Allow dealing a specific card from the deck into a hand

The MCTS agent was then given a version of the game environment with this added functionality to permit direct hand manipulation to use as a forward model for simulations. The option of redeterminising was implemented; all agents that act (other than the MCTS agent) have their hands returned and redrawn from the deck. After their turn, their original hand is restored as closely as possible. If a card was played or discarded, there is a check that the new hand is still viable, and if not, fills in the offending cards by: replacing with valid card or deleting card knowledge and replacing with a valid card.

It is noted in case of future work that the process of adapting the framework with the two new moves removed certain functionality, including the representation of game states in vector form, and the flexibility of specifying different rank and colour formats. This study did not require these parts of the framework.

2) MCTS: When describing an agent, this paper will use "MCTS" to refer to a very particular configuration of IS-MCTS, with the following parameters: 100 rollouts, exploration constant C = 2.5 matching that used in original paper, does not redeterminise, uses the game score for rollout state rewards, 3 maximum simulation steps, no playable now convention, simulates according to van den Bergh agent for all players. It also uses the branching rules outlined in Table I, almost identical to original (Goodman 2019), to aid performance by disregarding exploring unpromising branches.

Variations on these parameters are listed alongside MCTS and include:

TABLE I
BRANCHING RULES

Branching Rule	Description
TellMostInformation	Hint that reveals most new information
TellUseful	Hint about a playable card
Tell Dispensable	Hint about a discardable card
CompleteTellUseful	Hint to fill in single unknown infomration about playable card
CompleteTellDispensable	Hint to fill in single unknown information about discardable card
CompleteTellUnplayable	Hint to fill in single unknown information about a card that can't be played now but could in future.
PlayProbablySafe	Play card 70% certain playable
PlayProbablySafeLate	Play card if 40% certain playable and less than 5 cards left in deck
DiscardMostConfident	Discard card most certain to be not useful. Change from original.

Branching rules for search-based agents. A potential branch has to meet one of these rules to be considered in the search tree

- Random (or other agent name): Simulation assumes other players are this agent
- NoRules: Any legal action permitted in branching
- reD: Redeterminise other players' hands
- NowConvention: 'Playable now' convention added for agent actions and as an assumption in simulations.
- Regret: Rollout reward is Score - TeamRegret, where TeamRegret is accumulated only from the actions in the simulation. The idea here is to incentivise plays that balance increasing the score, but without incurring excessive regret.

D. Experiments

This research utilised Queen Mary's Apocrita HPC facility, supported by QMUL Research-IT. http://doi.org/10.5281/zenodo.438045. Experiments consisted of running and recording statistics of standard 3-player Hanabi games in the framework with different sets of agents. Statistics were:

- TeamScore: Score at end of the game
- TeamRegret: Sum of each agent's regret at the end of the game
- AgentRegret: Average for a single type of agent's regret. Always representative of regret of a single agent, and so numbers are used to indicate which agents the average was over.
- AgentRegretSource: Splits up the agent regret by the source; discarding a critical card, playing a critical card or ending the game.

All statistics are averaged over the total number of games played for the agent setup. 250 games were played for rule-based experiments and 100 for search-based experiments.

Two main classes of experiment were run:

1) Self-Play: All rule-based and several variations of search-based agents were simulated playing games against identical copies of themselves, denoted "Agent1,2,3". This serves both as a validation of the implementation against previous published results and to act as a base to compare to

the outcomes when the agents are paired with different agents in mixed-play.

2) Mixed-Play: Games run with one copy of an agent, "Agent1", paired with two copies of a different type of agent "Agent2,3". The focus is to pair one copy of varying strength rule-based agents with two copies of stronger agents to observe: what happens to the overall performance as a team, are there complementary strategies and how well strong self-play performance translates to strong mixed-play performance.

VI. Results

A. Self-Play

Table II provides results for self-play games. Roll-out number was the fixed limitation for how long to search for, but for comparison, the maximum average time to act was 3.08s. All other search experiments averaged under 2 seconds. Rule-based agents were always only a few ms, with maximum average of 3ms.

- Best search-based agent outperforms best rule-based agent by 2.76 average score
- Branching rules and VDB agent simulation drastically improves search agent performance
- Redeterminising improves performance by 1.68 without branching rules but detrimental by between 0.45 - 0.68 with branching rules

B. Mixed-Play

Table III provides results for mixed-play games. Agent Regret Sources are omitted from the table for conciseness, but available in Appendix A. It was decided to not test the NowConvention, as it is hypothesised as too strict of an assumption to be of benefit in mixed-play.

- The average score is generally between the individual self-play scores, but with notable exceptions including Mute.
- Search-based agents achieve better scores than rule-based agents with same teammates.
- Search-based agents with knowledge of teammate's strategy do marginally better than assuming a VDB agent.
- Weaker agents (by self-play score) have higher regret than the stronger agents it plays with.
- Regret search-based agents score lower, despite always succeeding in obtaining a lower TeamRegret on average.

VII. Discussion

A. Self-Play Score Performance

The 100 search-based experiments games introduce more variance into the statistics compared to 250 for rule-based, but still within a range to infer reasonable conclusions.

1) Validation Against Literature: As a sense check of the implementation, scores achieved are in line with previous results with more than one source. van Den Bergh agent's 17.12 looks on the surface anomalous, as significantly higher than both 15.4 (van den Bergh et al. 2016) and 16.06 (Canaan et al. n.d.), but it is in line with 16.95 in (Walton-Rivers et al. 2017) which, like here, extend the rules to all players involved rather than the immediate next. As another example, Inner with 10.68 is in middle of 10.97 (Osawa 2015) and 10.12. For the search-based agents, MCTS 19.15 is moderately lower than the comparable 20.2 (Goodman 2019). It is hypothesised this is due to lower number of roll outs used in the configuration here.

2) Rule vs. Search in Self-Play: The best rule-based agents perform well given their human understandable strategy and much faster evaluation (2ms vs. 2s). However the search-based agents outperform them in self-play, with the best search-based developed here (MCTS, NowConvention) obtaining an average score of 19.88, 2.76 higher than the best rule-based (VDB).

3) Redeterminising: Redeterminising is beneficial with no branching rules but detrimental when rules are used, costing between 0.45 and 0.68 average score. It is hypothesised that the crafted branching rules protect against the strategy fusion weakness of IS-MCTS. Only sensible actions considering knowledge available from the acting player are explored, so the only major difference by redeterminising is rolling out incompatible game determinations. This process pollutes tree statistics leading to slightly poorer decisions.

B. Mixed-Play Score Performance

1) Strong Self-Play $\iff$ Strong Mixed-Play?: It is observed that self-play strength generally transfers to better score in mixed-play and pairing a stronger agent with a weaker one results in a score between the self-play scores. However importantly it is not always true. Mute is a counter-example. In self-play, Mute is forced to only play and obtains a 0 average score, worst of all agents. However it becomes the second best rule-based agent with between 15.69 - 16.02 average score when paired with agents that can complement the weakness of not being able to give information. The Outer agent also scores higher than expected under mixed-play. Mute paired with Outer achieves 16.02, a score greater than either of their individual scores (0, 15.35) at a confidence outside the bounds of standard error. These agents have the seemingly rare property of complementary strategies.

2) Rule vs. Search in Mixed-Play: The MCTS agent, which does not make any adaptations or have any knowledge of the agents that it is trying to cooperate with, outperforms all rule-based agents when paired with any agent.

3) Simulation Using VDB vs. Using Known Agent: The search-based agents that know what agent they are playing with perform better than assuming VDB agent. The effect is most apparent with Flawed agent, boosting performance from 0.56 to 2.43. A conclusion is that Flawed agent needs the most direction to avoid failed games, which a search agent can plan ahead for. The score with other agents is between 0.14 - 0.33 better on average. Improvement is within standard error in these cases, so while the effect is there, knowing the opponents strategy isn't much better than the assumption they are a strong rule-based agent.

TABLE II
SELF-PLAY EXPERIMENT AVERAGED RESULTS

Players	Team		Agent Regret	Agent Regret Source		
Agent1,2,3	Score(± s.e)	Regret(± s.e)	Agent1,2,3	Discard	Play	EndGame
Mute	0.00 (.00)	25.00 (.00)	8.33	0.00	0.13	8.20
Flawed	0.00 (.00)	25.00 (.00)	8.33	0.00	0.15	8.18
Inner	10.68 (.10)	10.02 (.15)	3.34	3.34	0.00	0.00
Outer	15.35 (.10)	6.24 (.14)	2.08	2.08	0.00	0.00
IGGI	15.90 (.07)	6.36 (.12)	2.12	2.12	0.00	0.00
Piers	16.93 (.12)	5.77 (.12)	1.92	1.77	0.15	0.00
VDB	17.12 (.13)	3.95 (.13)	1.32	1.24	0.08	0.00
MCTSRandom,NoRules	2.13 (.34)	21.74 (.52)	7.25	1.75	0.10	5.40
MCTS,NoRules,Regret	5.39 (.27)	13.27 (.44)	4.42	3.88	0.14	0.40
MCTS,NoRules,Regret,reD	7.07 (.30)	12.24 (.46)	4.08	3.45	0.13	0.50
MCTS,Regret,reD	18.29 (.17)	3.08 (.19)	1.03	0.95	0.08	0.00
MCTS,reD	18.70 (.26)	3.37 (.29)	1.12	0.98	0.06	0.08
MCTS,Regret	18.97 (.17)	1.82 (.16)	0.61	0.59	0.02	0.00
MCTS	19.15 (.16)	3.44 (.19)	1.15	1.12	0.03	0.00
MCTS,NowConvention	19.88 (.17)	3.20 (.18)	1.07	1.04	0.03	0.00

Results from sets of 250 rule-based and 100 search-based, 3-player,self-play games. Team averages over all games, with standard errors. Agent Regret and Source averages over all 3 agents and over all games, so represents stats for a single agent of the team. Agents described in section V.

Fig 2 visualises the score distribution improvement from the best rule-based agent to the best search-based agent for each agent under mixed-play.

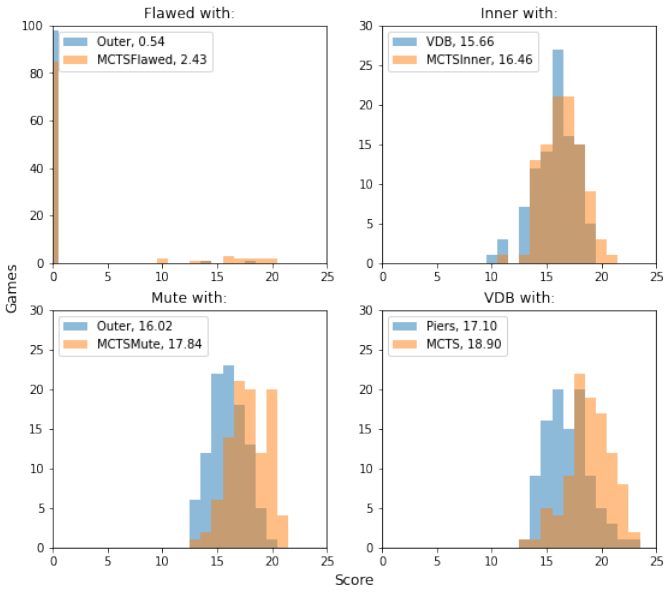

Fig. 2. Average score and distributions for single agents (Agent1) playing with pair of best rule-based and search-based agents (Agent2,3) in the mixed-play experiments

C. Regret

In general, the lower the regret the better the score.

The agent regret sources show why certain agents have poor performance. Mute and Flawed have almost exclusively regret from depleting all life tokens. It can be deduced they make too risky plays. Inner causes no regret through playing or ending the game, but is a weak agent through discarding too many critical cards because of its reluctance to play without being certain. The regret sources generally keep their relative distribution under mixed-play. Breaking down the source of regret has the benefit that the internal workings of the agent do not need to be known to deduce if and how it is making mistakes.

In mixed-play, the weaker agent from self-play almost always has the higher regret. The benefit from this insight is that self-play performance does not need to be known to deem how strong an agent is when in mixed-play. Also, both agents generally have lower individual regret than in self-play. It is hypothesised that the stronger agent helps the weaker agent make fewer mistakes, but the mistakes that are still made are by the weaker agent, which reduces the regret share for the stronger agent.

While regret seems to be an indicator of the strongest and weakest agents and scores are generally better when it is lower, it is not true that actively minimising regret causes a better score. The search-based agents demonstrate this best when searching for actions with a penalty for incurring regret. The team regret reaches as low as 1.82 on average for MCTS,Regret, a good 1.62 regret lower than the score search counterpart. However the final scores are between 0.14 - 0.73 worse on average.

The conclusion here is that while low regret is a side effect of achieving a good score, as visualised in Fig. 3, it shouldn't be the goal to minimise it directly. A hypothesis why this is true can be made by rearranging the equation in section III-D.

$$TeamRegret = 25 - Score - RemainingScorePotential$$

Minimising TeamRegret is not equivalent to maximising Score. The search-based experiments show that action regret minimisation increases the RemainingScorePotential instead, rather than the Score. In Hanabi terms, this means that minimising the direct regret from actions that cuts off fireworks has the risk of not improving the score, and instead creates "indirect" regret from ending the game with high RemainingScorePotential (still with useful cards in hands, but with no more turns left to actually complete fireworks).

TABLE III
MIXED-PLAY EXPERIMENT AVERAGED RESULTS

Players		Team		Agent Regret	
Agent1	Agent2,3	Score(± s.e)	Regret (± s.e)	Agent1	Agent2,3
Flawed	VDB	0.14 (.10)	24.84 (.11)	24.50	0.17
Flawed	Piers	0.21 (.12)	24.77 (.13)	24.25	0.26
Flawed	IGGI	0.51 (.18)	24.42 (.21)	23.85	0.28
Flawed	Outer	0.54 (.19)	24.35 (.23)	23.86	0.25
Inner	Outer	14.15 (.09)	6.92 (.13)	2.54	2.19
Inner	IGGI	14.47 (.08)	7.32 (.13)	2.70	2.31
Inner	Piers	15.63 (.11)	6.66 (.13)	2.39	2.14
Inner	VDB	15.66 (.12)	5.14 (.13)	1.95	1.60
Mute	IGGI	15.69 (.26)	6.54 (.31)	3.89	1.33
Mute	VDB	15.80 (.21)	4.85 (.26)	3.74	0.56
Mute	Piers	15.96 (.29)	6.30 (.33)	4.23	1.03
Mute	Outer	16.02 (.11)	5.09 (.15)	3.38	0.86
VDB	Outer	16.27 (.10)	5.19 (.12)	1.53	1.83
VDB	IGGI	16.52 (.09)	5.47 (.12)	1.47	2.00
VDB	Piers	17.10 (.12)	4.80 (.13)	1.22	1.79
Flawed	MCTS	0.56 (.32)	24.38 (.36)	23.26	0.56
Flawed	MCTSFlawed,Regret	2.29 (.57)	22.14 (.70)	20.55	0.80
Flawed	MCTSFlawed	2.43 (.59)	22.03 (.72)	20.55	0.74
Inner	MCTSInner Regret	15.96 (.19)	4.20 (.20)	2.09	1.06
Inner	MCTS	16.13 (.17)	5.29 (.24)	2.22	1.54
Inner	MCTSInner	16.46 (.18)	5.64 (.22)	2.62	1.51
Mute	MCTSMute,Regret	17.50 (.23)	2.74 (.20)	2.04	0.35
Mute	MCTS	17.57 (.37)	4.76 (.41)	3.39	0.68
Mute	MCTSMute	17.84 (.18)	3.98 (.19)	2.57	0.70
VDB	MCTS,Regret	18.17 (.23)	2.34 (.19)	0.56	0.89
VDB	MCTS	18.90 (.20)	3.05 (.18)	1.00	1.02

Results from sets of 250 rule-based and 100 search-based, 3-player, mixed-play games. First player is type Agent1 with the remaining two players type Agent2,3. Team averages over all games, with standard errors. Agent Regret averages over each agent and all games, so represents stats for single agents in the team. Agents described in section V.

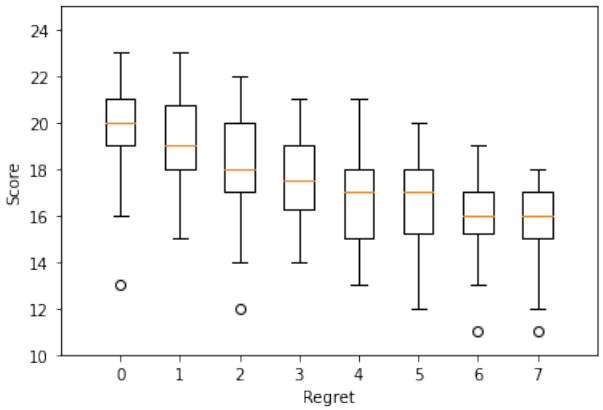

Fig. 3. Quantile summary of scores achieved in a random sample taken from 8 batches of 50 self-play and mixed-play games that ended with a particular (Team) Regret

VIII. CONCLUSIONS AND FUTURE WORK

It is shown here that search-based agents are a promising variety for robust performance under 3 player mixed-play settings in Hanabi, with performance boosted when knowing a teammate's strategy. The simple action regret introduced also offers the ability to track lost score potential, showing which agent was responsible and through what types of actions. This offers the first quantitative method for understanding why strong or weak scores are achieved, which could help fix agent weaknesses. Considering direct regret when deciding an action doesn't help to improve average score obtained. The conclusion theorised is that if actions are to be chosen based on regret, there is a need to consider both direct regret and the "indirect" regret from having useful cards in hands at the end of the game without enough turns to play them.

For future work, the regret definition could try to take into account the point in time where an agent not playing a card has removed the possibility of achieving a score. This is a difficult combinatorics problem, assuming optimal play and likely including probabilistic assumptions about the remaining deck order.

Another area of interest is creating an agent able to adapt and learn both during the game and between games to different play styles in a mixed setting. The strategy of the agent that is being played with could be modelled or inferred as it plays more and so the search agent could get better over time with a group of players. Alternatively, the agent could be shown an example set of games to infer the behaviour

Other variants of Hanabi such as 6 suit, sequential hints or fewer information tokens are underrepresented in the literature. Work could study how adaptable agents can be under these variants, and which of them are particularly challenging.

REFERENCES

Arneson, B., Hayward, R. B. & Henderson, P. (2010), 'Monte carlo tree search in hex', *IEEE Transactions on Computational Intelligence and AI in Games* **2**(4), 251–258.

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E. et al. (2020), 'The hanabi challenge: A new frontier for ai research', *Artificial Intelligence* **280**, 103216.

Bjarnason, R., Fern, A. & Tadepalli, P. (2009), Lower bounding klondike solitaire with monte-carlo planning, *in* 'Nineteenth International Conference on Automated Planning and Scheduling'.

Bouzy, B. (2017), Playing hanabi near-optimally, *in* 'Advances in Computer Games', Springer, pp. 51–62.

Brooks, M. (2020), 'mcts-hanabi github [online]'. URL: https://github.com/MBlogs/mcts-hanabi [accessed 2020-09-02],.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. & Colton, S. (2012), 'A survey of monte carlo tree search methods', *IEEE Transactions on Computational Intelligence and AI in games* **4**(1), 1–43.

Bubeck, S. & Cesa-Bianchi, N. (2012), 'Regret analysis of stochastic and nonstochastic multi-armed bandit problems', *arXiv preprint arXiv:1204.5721* .

Bubeck, S., Munos, R. & Stoltz, G. (2009), Pure exploration in multi-armed bandits problems, *in* 'International conference on Algorithmic learning theory', Springer, pp. 23–37.

Canaan, R., Gao, X., Chung, Y., Togelius, J., Nealen, A. & Menzel, S. (n.d.), 'Evaluating rl agents in hanabi with unseen partners'.

Chaslot, G. M. J., Winands, M. H., HERIK, H. J. V. D., Uiterwijk, J. W. & Bouzy, B. (2008), 'Progressive strategies for monte-carlo tree search', *New Mathematics and Natural Computation* **4**(03), 343–357.

Ciancarini, P. & Favini, G. P. (2010), 'Monte carlo tree search in kriegspiel', *Artificial Intelligence* **174**(11), 670–684.

Cowling, P. I., Powley, E. J. & Whitehouse, D. (2012), 'Information set monte carlo tree search', *IEEE Transactions on Computational Intelligence and AI in Games* **4**(2), 120–143.

Cowling, P. I., Ward, C. D. & Powley, E. J. (2012), 'Ensemble determinization in monte carlo tree search for the imperfect information card game magic: The gathering', *IEEE Transactions on Computational Intelligence and AI in Games* **4**(4), 241–257.

Cox, C., De Silva, J., Deorsey, P., Kenter, F. H., Retter, T. & Tobin, J. (2015), 'How to make the perfect fireworks display: Two strategies for hanabi', *Mathematics Magazine* **88**(5), 323–336.

DeepMind, G. (2018), 'hanabi-learning-environment github [online]'. https://github.com/deepmind/hanabi-learning-environment [accessed 2020-08-06].

Eger, M. & Gruss, D. (2019), Wait a second: playing hanabi without giving hints, *in* 'Proceedings of the 14th International Conference on the Foundations of Digital Games', pp. 1–7.

Frank, I. & Basin, D. (1998), 'Search in games with incomplete information: A case study using bridge card play', *Artificial Intelligence* **100**(1-2), 87–123.

Goodman, J. (2019), Re-determinizing mcts in hanabi, *in* '2019 IEEE Conference on Games (CoG)', IEEE, pp. 1–8.

Guo, X., Singh, S., Lee, H., Lewis, R. L. & Wang, X. (2014), Deep learning for real-time atari game play using offline monte-carlo tree search planning, *in* 'Advances in neural information processing systems', pp. 3338–3346.

Hu, H., Lerer, A., Peysakhovich, A. & Foerster, J. (2020), '" other-play" for zero-shot coordination', *arXiv preprint arXiv:2003.02979* .

Kocsis, L. & Szepesvári, C. (2006), Bandit based monte-carlo planning, *in* 'European conference on machine learning', Springer, pp. 282–293.

Kocsis, L., Szepesvári, C. & Willemson, J. (2006), 'Improved monte-carlo search', *Univ. Tartu, Estonia, Tech. Rep* **1**.

Lerer, A., Hu, H., Foerster, J. & Brown, N. (2020), Search in cooperative partially-observable games, *in* 'AAAI Conference on Artificial Intelligence'.

Osawa, H. (2015), Solving hanabi: Estimating hands by opponent's actions in cooperative game with incomplete information, *in* 'Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence'.

qpwo (2018), 'montecarlotreesearch gist [online]'. URL: https://gist.github.com/qpwo/c538c6f73727e254fdc7fab81024f6e1 [accessed 2020-08-06].

RnRGames (2014), 'Hanabi rules [online]'. URL: https://rnrgames.com/Content/RRGames/images/ProductRules/hanabiRules.PDF [accessed 2020-08-06].

rocanaan (2019), 'hanabi-ad-hoc-learning github [online]'. URL: https://github.com/rocanaan/hanabi-ad-hoc-learning [accessed 2020-08-06].

Russell, S. & Norvig, P. (2002), 'Artificial intelligence: a modern approach'.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al. (2017), 'Mastering the game of go without human knowledge', *nature* **550**(7676), 354–359.

Świechowski, M., Tajmajer, T. & Janusz, A. (2018), Improving hearthstone ai by combining mcts and supervised learning algorithms, *in* '2018 IEEE Conference on Computational Intelligence and Games (CIG)', IEEE, pp. 1–8.

Szita, I., Chaslot, G. & Spronck, P. (2009), Monte-carlo tree search in settlers of catan, *in* 'Advances in Computer Games', Springer, pp. 21–32.

Tolpin, D. & Shimony, S. E. (2012), 'Mcts based on simple regret', *arXiv preprint arXiv:1207.5536* .

van den Bergh, M. J., Hommelberg, A., Kosters, W. A. & Spieksma, F. M. (2016), Aspects of the cooperative card game hanabi, *in* 'Benelux Conference on Artificial Intelligence', Springer, pp. 93–105.

Walton-Rivers, J., Williams, P. R., Bartle, R., Perez-Liebana, D. & Lucas, S. M. (2017), Evaluating and modelling hanabi-playing agents, *in* '2017 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 1382–1389.

Winands, M. H., Björnsson, Y. & Saito, J.-T. (2008), Monte-carlo tree search solver, *in* 'International Conference on Computers and Games', Springer, pp. 25–36.

APPENDIX A
MIXED-PLAY REGRET SOURCES

TABLE IV
MIXED-PLAY EXPERIMENT AVERAGED RESULTS FOR REGRET SOURCE

Players		Team(avgs)		Agent1 Regret Source			Agent2,3 Regret Source		
Agent1	Agent2,3	Score avg	Regret avg	Discard	Play	EndGame	Discard	Play	Endgame
Flawed	VanDenBergh	0.14 (.10)	24.84 (.11)	0.04	0.31	24.15	0.17	0.00	0.00
Flawed	Piers	0.21 (.12)	24.77 (.13)	0.11	0.34	23.80	0.26	0.00	0.00
Flawed	IGGI	0.51 (.18)	24.42 (.21)	0.15	0.38	23.32	0.28	0.00	0.00
Flawed	Outer	0.54 (.19)	24.35 (.23)	0.15	0.41	23.30	0.25	0.00	0.00
Inner	Outer	14.15 (.09)	6.92 (.13)	2.54	0.00	0.00	2.19	0.00	0.00
Inner	IGGI	14.47 (.08)	7.32 (.13)	2.70	0.00	0.00	2.31	0.00	0.00
Inner	Piers	15.63 (.11)	6.66 (.13)	2.39	0.00	0.00	1.94	0.20	0.00
Inner	VanDenBergh	15.66 (.12)	5.14 (.13)	1.95	0.00	0.00	1.39	0.21	0.00
Mute	IGGI	15.69 (.26)	6.54 (.31)	2.86	0.18	0.85	1.16	0.00	0.17
Mute	VanDenBergh	15.80 (.21)	4.85 (.26)	3.10	0.12	0.52	0.43	0.13	0.00
Mute	Piers	15.96 (.29)	6.30 (.33)	2.77	0.16	1.30	0.85	0.18	0.00
Mute	Outer	16.02 (.11)	5.09 (.15)	3.14	0.24	0.00	0.86	0.00	0.00
VanDenBergh	Outer	16.27 (.10)	5.19 (.12)	1.35	0.18	0.00	1.83	0.00	0.00
VanDenBergh	IGGI	16.52 (.09)	5.47 (.12)	1.28	0.19	0.00	2.00	0.00	0.00
VanDenBergh	Piers	17.10 (.12)	4.80 (.13)	1.15	0.07	0.00	1.67	0.12	0.00
Flawed	MCTS	0.56 (.32)	24.38 (.36)	0.06	0.53	22.67	0.32	0.00	0.24
Flawed	MCTSFlawed,Regret	2.29 (.57)	22.14 (.70)	0.23	0.40	19.92	0.80	0.00	0.00
Flawed	MCTSFlawed	2.43 (.59)	22.03 (.72)	0.19	0.47	19.89	0.62	0.00	0.12
Inner	MCTSInner,Regret	15.96 (.19)	4.20 (.20)	2.09	0.00	0.00	1.01	0.05	0.00
Inner	MCTS	16.13 (.17)	5.29 (.24)	2.22	0.00	0.00	1.50	0.04	0.00
Inner	MCTSInner	16.46 (.18)	5.64 (.22)	2.62	0.00	0.00	1.48	0.03	0.00
Mute	MCTSMute,Regret	17.50 (.23)	2.74 (.20)	1.83	0.21	0.00	0.35	0.00	0.00
Mute	MCTS	17.57 (.37)	4.76 (.41)	2.52	0.18	0.69	0.66	0.02	0.00
Mute	MCTSMute	17.84 (.18)	3.98 (.19)	2.35	0.22	0.00	0.68	0.02	0.00
VDB	MCTS	18.17 (.23)	2.34 (.19)	0.44	0.12	0.00	0.86	0.03	0.00
VDB	MCTS	18.90 (.20)	3.05 (.18)	0.86	0.14	0.00	1.00	0.02	0.00

Results from sets of 250 rule-based and 100 search-based, 3-player, mixed-play games. First player is type Agent1 with the remaining two players type Agent2,3. Team averages over each game stat over all games, with standard errors. Agent Regret Source averages over each agent and all games, so represents stats for single agents in the team. Agents described in section V.