Re-determinizing MCTS in Hanabi

James Goodman
james@janigo.co.uk

***Abstract*—We introduce Re-determinizing IS-MCTS, a novel extension of Information Set Monte Carlo Tree Search (IS-MCTS) that prevents a leakage of hidden information into opponent models that can occur in MCTS, and is particularly severe in Hanabi. When combined with a learned evaluation function to estimate leaf node values and avoid full simulations during MCTS this won the Hanabi competition at the Computation Intelligence in Games (CIG) 2018.**

***Index Terms*—Hanabi, Monte Carlo Tree Search, Opponent Modelling, Imperfect Information**

I. Introduction

Hanabi [1] is a co-operative game for 2-5 players that has attracted attention in games research due to the role of hidden information, a restricted communication channel and need to model one's fellow players, most recently [2]. The Hanabi competition at the Computational Intelligence in Games (CIG) 2018 conference had two tracks. In the Mirror track all players in a game use the same agent. This makes modelling of other players straightforward, but requires a strategy for communicating hidden information. In the Mixed track a random set of unknown agents play each game, and it is necessary to model the strategies used by the others.

Information Set Monte Carlo Tree Search (IS-MCTS) [3] performs very poorly in Hanabi, regardless of time budget. We trace this to the specific form of hidden information in Hanabi, and address it with a new variant of IS-MCTS that we term Re-determinizing IS-MCTS (RIS-MCTS) that avoids leakage of hidden information known to the acting player into the modelling of other players in the tree search. This is done by re-determinizing hidden information from the perspective of the acting player at each node in the tree search.

A good standard of play requires amendments to vanilla MCTS to reduce the branching factor via rule heuristics, and learn an evaluation function from the result of off-line RIS-MCTS games. The final competition entry avoids full simulations during search, and evaluates the expanded node(s) in the tree using the learned function.

We offer two specific contributions. First the Re-determinizing IS-MCTS algorithm, as the information leakage problem identified in Hanabi may be present in other environments. Second we introduce a method of training evaluation functions from offline MCTS games that uses more of the data in an MCTS tree than just the root node. All code is available at https://github.com/hopshackle/fireworks. The Mixed Track entry does not use RIS-MCTS, and is described in [4].

Sections II and III cover background and previous work in Hanabi and MCTS. We then describe the RIS-MCTS algorithm (Section IV), game-specific improvements needed for good performance in Hanabi (SectionV), and how to learn an evaluation function from off-line MCTS games to make a decision in 40ms (Section VI). Section VII concludes, and discusses further work.

Fig. 1. A game of Hanabi in progress. The player at the camera's perspective can see the other players' cards, but not their own. The current score in the game is 12, from the sum of the top cards in each suit in the tableau.

II. Background

A. Hanabi

Hanabi has a deck of 50 cards. There are five suits (red, blue, green, white, yellow), and five possible face values (one through five). Each suit has 10 cards - one 'five' card, three 'one' cards, and two of each other face value. The goal is to play out cards to a shared tableau with each suit played in order, from the 'one' card to the 'five'. The final score is the sum of the top face-value cards in each suit, for a maximum of 25. The game is co-operative with all players trying to obtain the best final score. Each player in Hanabi is dealt a hand of cards, but holds the cards to face the other players without looking at them. Each player therefore knows the cards held by their fellow players, but not their own. The remaining cards form a face-down deck.

Play proceeds clockwise round the table, and on their turn a player may do one of three things:

- Play a card from their hand to the tableau;
- Discard a card from their hand;
- Hint to another player by touching all of that player's cards of a stated colour, or with a stated face value.

If a card played to the tableau is not in sequence, then one life is lost. After three lives are lost the game is over. When a

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

card is played or discarded, the player draws a new card from the deck. Once the last card is drawn from the deck, each player has one last turn and then the game is also over. There are 8 hint tokens. When a hint is given, one of these is used up; every time a card is discarded or a 'five' card successfully played, then a hint token is regenerated. If no hint token is left, a player must either Play or Discard a card. A hint must refer to a card colour or face value that the player possesses - e.g. it is not possible to tell a player that they have no red cards in their hand.

Baffier et al. [5] show that Hanabi is NP-complete, and not all games can achieve a perfect score of 25 even if players could see all the cards and the deck ordering. The challenge in the game is to make efficient use of the hint tokens, and model the thinking of other players, to predict their future actions and infer from their hints what one's own cards are.

B. MCTS

Monte Carlo Tree Search (MCTS) is surveyed in [6], and has been used successfully in many games, such as Go [7], Settlers of Catan [8], Magic: The Gathering [9], Scotland Yard [10] and Hearthstone [11]. It is an *anytime* algorithm that uses a time budget to stochastically search the game tree from the current state. Each iteration has four steps:

1) Selection. Select an action to take from the current state. If all actions have been selected at least once then the 'best' one is picked (usually, as in this work, using an Upper Confidence Bound (1) [12]), and this is repeated down the tree of game states until a state is reached with previously untried actions.

$$J(a) = V(a) + C\sqrt{\frac{\log N}{n(a)}} \tag{1}$$

   In (1), the action with largest $J(a)$ is selected at each tree node. N is the total number of visits to (iterations through) the node; $n(a)$ is the number of those visits that then took action a; $V(a)$ is the mean score for all visits to the node that took action a; C is a parameter that controls the trade-off between exploitation using the empirical score $V(a)$, and exploration choosing actions with few visits so far.
2) Expansion. Pick one of the untried actions at random, and expand this, creating a new node in the game tree.
3) Simulation. From the expanded node, simulate a complete game to obtain a final score, for example by taking moves at random. During the simulation step, domain knowledge is commonly embodied in a simulation policy rather than using random moves, e.g. [13] [8].
4) Back-propagation. Back-propagate this final score up the game tree. Each node records the mean score of all iterations that take a given action from that node as $V(a)$ in (1), that will affect future Selection steps.

Once the time budget has been used up the action at the root node with the highest score is executed in the actual game environment.

III. Previous Work

A. MCTS with hidden information

MCTS requires a forward model of the game, so that when an action is selected at a node, the game state can be rolled forward to a hypothetical 'what-if' game state at the child node. In a game with hidden information we cannot roll forward to a single game state, as this depends on unknown information. One approach is to sample a possible set of hidden information as a game 'determinization', and proceed as if this information is known. Determinizations are sampled from the player's information set (the set of all game states that are indistinguishable to the player) and MCTS applied independently to each one as a perfect information game. A final action is chosen by averaging across the statistics at the root nodes for each determinized game. This Perfect Information Monte Carlo (PIMC) has been used in Klondike Solitaire [14] and Bridge [15].

The major problems with determinization are *strategy fusion* and *non-locality* [16]. Strategy fusion occurs when an agent makes different decisions from the same information set due to implicit or explicit use of hidden information. For example, if we use MCTS on several determinizations of a game, then in each case MCTS will determine the single best action to take, and these will vary with the determinization despite the player being in the same information set. We *hope* that averaging over these incompatible suggestions will give a good answer, but the best move might be one that gathers more information about the opponent's hand so we can make a more informed decision later; PIMC will never find this.

Non-locality arises because the likely values of hidden information will depend on historic moves in the game - this is not a problem in perfect information environments, which are sub-game perfect and can be decomposed into the solution of their sub-games [17]. For example, in Whist it is unusual to lead away from an Ace; so if an opponent leads a low heart, we would expect their hand (the hidden information) to probably not include the Ace of Hearts.

Monte Carlo Search in Partially Observable Markov Decision Processes (POMDPs) with a single-player [18] addresses strategy fusion with a particle filter of possible determinizations constructed at the root, sampled from the current information set. Each iteration uses a single determinization from this pool, and maintains a set of possible determinizations at each node in the tree. This also addresses non-locality; after each action in the real game the set of determinizations at the child node is used to seed the particle filter for the next action, defining a non-uniform distribution for the new information set and incorporating information from historic moves.

Information Set MCTS [3] is introduced in two forms. Single Observer (SO-ISMCTS) and Multiple Observer (MO-ISMCTS). Both sample a different possible determinization at the root of the tree for each MCTS iteration, and maintain a node for each information set from the perspective of the current (root) player. SO-ISMCTS opponents make random moves consistent with the current determinization. MO-

ISMCTS improves opponent modelling by using IS-MCTS for opponent moves, and constructs a tree for each player. Decisions made by other players are made with respect to determinizations from the root player's perspective, and non-locality is not addressed. Cowling et al. [19] extend MO-ISMCTS to the bluffing game *The Resistance* with improved opponent modelling. This algorithm (MT-ISMCTS) maintains one tree per opponent per information set that they could be in, and these are updated using determinizations that the root player knows to be incorrect, but which *could* be from the perspective of the other players. This produces a much better opponent model. In Resistance there are only six possible information sets for a given player. In Hanabi the number of information sets that another player could be in is defined by the cards of the current player, which at the start of the game is $\sim 10^5$ with 4 unknown cards, making this approach of one tree per information set less tractable.

Semi-Determinized MCTS (SDMCTS) [17] extends IS-MCTS to include inference on which individual states in an information set are more likely and address non-locality. This determinizes the opponent's previous move, and runs IS-MCTS separately for each possibility. An opponent model pre-learned from human play traces is used to predict a distribution over the opponent's previous move and calculate an expected best response. Nijssen et al. [10] bias the determinizations in MCTS using statistics from self-play games to similarly address non-locality

B. Hanabi

Ozawa et al. 2015 [20] investigate 2-player Hanabi, and compare some heuristic rules with an approach that models the other player's likely next action assuming they are using the same strategy.

Van den Bergh et al. 2016 [21] investigate heuristic rules in 3-player Hanabi, and search over 48 combinations to find the best-scoring. They try MCTS with 500 iterations, but find that the average performance here is lower (14.5 vs 15.4) than for the simple heuristic rule. While not using IS-MCTS the authors note that care must be taken to avoid leaking hidden information to later players. They re-shuffle the hand of the acting player at each move in tree selection to one that is compatible with the hints given so far (IS-MCTS only shuffles once per tree iteration to determinize the hand of the root player). They do not use complete game simulations, but record the number of points gained over the next 5 moves, assuming infinite lives with a random simulation policy.

Cox et al. 2015 [22] have a different approach that uses hat-guessing to pass on information to other players. This assigns a value (the 'hat') to each hand that instructs the player which card is playable or discardable to give a number between 0 and 2C - 1 (where C is the number of cards in a hand). The sum of this over all hands, modulus the number of players, is used via a pre-agreed lookup table to define a Hint in terms of player, colour and face value. After one round of Hints, each player can calculate their own 'hat', and which card in their hand to play or discard. However this only works with 5-players. Bouzy 2017 [23] extends hat-guessing to different numbers of players and obtains excellent results, but drops the rule that forbids a hint to touch no cards. Without this relaxation of the rules it can be impossible for a required hint to be made, and the hat-guessing approach fails.

As well as the hat-guessing approach, [23] uses 1-ply expectimax search with 1000 different determinizations of the current state to approximate the hidden information distribution, and simulates all of these for each possible action using heuristic simulation policies. They find the best result uses a 'confidence' simulation policy that assumes a card hinted by a previous player is playable.

Walton-Rivers et al. [24] find that pure MO-ISMCTS gives poor results, and explicitly model the other players instead. They use an IS-MCTS agent that assumes the other players use known heuristic rules. This means that all tree nodes are only from the perspective of the acting player, with the actions of the other players subsumed into the environment.

Eger et al. 2017 [25] look at Hanabi from the perspective of designing an AI agent to play with humans. They find that human players prefer playing with an agent that gives hints that are immediately relevant to play, such as hinting an immediately playable or discardable card, rather than those that maximise information in the longer term.

Canaan et al. 2018 [26] use genetic algorithms to evolve a heuristic constructed from an ordering of rules inspired by those of [24], and using the same framework. They evolve rules that surpass previous work (excluding the hat-guessing of [22]), and came second in the CIG 2018 Hanabi competition.

Most recently Bard et al. 2018 [27], Foerster et al. 2019 [2] use Deep Reinforcement Learning to and achieve state of the art performance in 2-player Hanabi. They document hand-coded bots from outside the academic literature that are state of the art with 3 or more players (see Table III), and use a strictly harder Hanabi variant in which losing all lives gives zero points and not the current tableau score.

IV. Re-determinizing Information Set MCTS

A. Why MO-ISMCTS fails in Hanabi

MO-ISMCTS determinizes at the root node for each iteration, randomising the hand of the current player as all other players' hands are known. This avoids strategy fusion occurring for the current (root) player, but not when deciding the actions of other players. For example if the root player is A, and B has a playable Red Two (R2) in their hand, then regardless of what A does, when we reach B in their search tree playing that card will always be R2 and increase the score, despite the fact that B cannot possibly know this when making their decision. This renders any hints by A meaningless, as they have no impact either on the available actions or the action consequences for B. This problem is inherent in MO-ISMCTS given a single determinization at the root of all the player trees, and information leaks out to inform the moves made by opponents - we model their behaviour as if they know what we know (which they don't). The node for B represents an information set that is reachable from many very different

hypothetical games, in most of which the given card is *not* R2. MO-ISMCTS does not account for this and suggests different actions in different games in the same information set for B: strategy fusion.

B. The RIS-MCTS algorithm

Algorithm 1 RIS-MCTS Algorithm outline. The changes to MO-ISMCTS are the functions ENTERNODE, and EXITNODE. REDETERMINIZE shuffles the player's hand and deck in line with their current information set.

```
1:  function RIS-MCTS(root)
2:      while timeAvailable do
3:          root.hand ← REDETERMINIZE(root)
4:          rootNode ← node ← emptyNode
5:          player ← root
6:          while node fully expanded do
7:              newNode ← SELECTUCT(node)
8:              EXITNODE(player, rootPlayer)
9:              player ← newNode.player
10:             node ← newNode
11:             ENTERNODE(player, rootPlayer)
12:         end while
13:         node ← EXPAND(node)
14:         score ← SIMULATION(node)
15:         BACKPROPAGATE(score, node)
16:     end while
17:     return rootNode.bestAction
18: end function
19:
20: procedure ENTERNODE(player, root)
21:     if player ≠ root then
22:         savedHand ← player.hand
23:         player.hand ← REDETERMINIZE(player)
24:     end if
25: end procedure
26:
27: procedure EXITNODE(player, root)
28:     if player ≠ root then
29:         player.hand ← savedHand
30:         REMOVEINCOMPATIBLECARDS(player.hand)
31:     end if
32: end procedure
```

One approach with multiple trees is to determinize the state for each player independently, and play down each tree using this determinization. Since all such determinizations cannot be mutually compatible (unless they are identical), we also need to account for what happens if a player makes a move that is illegal from the perspective of another.

MT-ISMCTS [19] addresses this by having a separate tree for each possible information set each other player could be in, but this is not feasible in Hanabi. Van den Bergh et al. [21] re-determinize the hand of the active player at each node in the tree, and bypass the illegality problem by assuming infinite lives and restricting any rollout to just 5 moves. This means that losing lives and ending the game is not penalised in a simulation. This is not ideal.

Our solution to this problem, which we term Re-determinizing IS-MCTS (RIS-MCTS) is to re-determinize the game-state at every node. When any player other than the root player has to make a decision in their tree, we first re-determinize their hidden information (randomize their hand to a valid set of cards given the information they have). If they play a card, then the card played in-game is the randomly sampled one, and *not* the card that the other players know they actually have. This resolves information leakage - if the sampled card is not playable, then a life is lost, and it is now worthwhile for others to hint that the card is a 'two'. The illegality problem remains, and determinizations may be incompatible with the root player's information set. Say that B played a card in the above scenario, and this was sampled to be Yellow Five (Y5), then the Y5 card is now publicly in the discard pile, and a new card drawn into the slot. This contradicts the root knowledge that R2 was in the slot, and possibly even that Y5 is elsewhere in B's hand. Regardless, we keep Y5 as the card discarded. From the perspective of the *other* players this iteration has moved into a different game. We continue playing down the tree in this new game with the other players moved into new information sets.

This makes sense if we consider the node for B to be a node in many hypothetical game trees in the same information set for B, only one of which is the game currently being played. What we have done is sampled one of these many games (that A knows to be false), and determinized to that, even though the game-state is now inconsistent with the information sets of the other players in the original game. When we exit a node we restore the player's hand and game state to the known values in so far as we can. This is always possible if they chose a Hint action, but not if they chose to Play or Discard a card that was different in their re-determinized hand. Apart from this re-determinization at each node, the algorithm is MO-ISMCTS as used in [24]. Algorithm 1 highlights the changes made to the outline of MO-ISMCTS.

The possible Hanabi score range 0-25 is standardized to 0-1, and a C-value of 0.1 is used, as this gave the best results in initial experiments with $C \in \{0.01, 0.03, 0.1, 0.3, 1, 3\}$. The same value of C was found to be optimal for both RIS-MCTS and MO-ISMCTS. All experiments were run on a single Google cloud n1 virtual CPU.

The IS-MCTS implementation here and in [24] uses an Open Loop approach [28], and defines information sets by the historic actions taken by all players; the new card drawn does not change the information set. This would be formally correct, but lead to a much larger branching factor, as after each Play or Discard up to 25 different cards could be drawn, generating 25 distinct branches. These would differ by one card and share all other information. We treat these as a single node in the tree, and share statistics; actions that are not possible at a node for the current determinization are ignored.

The top-half of Table I shows the improvement with a random simulation policy. Vanilla MO-ISMCTS performs very

TABLE I
MO-ISMCTS AND RIS-MCTS IN 4-PLAYER HANABI WITH RANDOM AND VAN DEN BERGH (THE BEST POLICY IN [21]) SIMULATION POLICIES BY TIME BUDGET. ALL PLAYERS USE THE SAME AGENT. STANDARD ERROR AS $\pm$ AND 500 GAMES RUN FOR EACH SETTING. SECOND SECTION HAS RESULTS FOR RIS-MCTS WITH RULE-CONSTRAINED TREE OPTIONS; '+ C' INDICATES USE OF THE 'PLAYABLE NOW' CONVENTION.

Algorithm	Simulation Policy	100ms	300ms	1000ms	3000ms
MO-ISMCTS	Random	3.87 ± 0.09	3.94 ± 0.10	3.88 ± 0.10	3.94 ± 0.10
MO-ISMCTS	Van Den Bergh	11.40 ± 0.13	14.53 ± 0.09	16.79 ± 0.08	18.36 ± 0.08
RIS-MCTS	Random	4.28 ± 0.15	4.21 ± 0.14	4.61 ± 0.15	5.21 ± 0.14
RIS-MCTS	Van Den Bergh	11.90 ± 0.12	14.85 ± 0.10	16.85 ± 0.08	18.48 ± 0.07
MO-ISMCTS	Random	9.49 ± 0.11	10.09 ± 0.11	10.54 ± 0.10	10.67 ± 0.10
MO-ISMCTS	Van Den Bergh	7.12 ± 0.09	7.85 ± 0.09	8.30 ± 0.09	8.97 ± 0.10
RIS-MCTS	Random	17.43 ± 0.10	17.93 ± 0.08	18.06 ± 0.08	18.14 ± 0.07
RIS-MCTS	Van Den Bergh	17.41 ± 0.07	18.31 ± 0.07	19.41 ± 0.06	19.84 ± 0.06
RIS-MCTS + C	Random	19.40 ± 0.07	19.67 ± 0.08	19.84 ± 0.07	19.76 ± 0.07
RIS-MCTS + C	Van Den Bergh	17.86 ± 0.07	19.11 ± 0.06	20.20 ± 0.06	20.81 ± 0.06

poorly and fails to give any improvement with additional search time - scoring 3.9 for any time budget between 100ms and 3000ms. Any hint given *cannot* affect the results of another player's action, as these use the known values of their cards independently of any hints given. If we use a heuristic simulation policy, hints will still not directly affect play in the selection phase - but *can* now affect the results of the simulation if the heuristic policy makes use of them. This explains why using the Van Den Bergh simulation policy is better and improves with time. However the raw Van Den Bergh policy achieves a raw score of 17.20 ± 0.08 by itself with $<$ 1ms per move, and MO-ISMCTS with 1000ms time budget is still significantly worse than this.

With RIS-MCTS a random simulation policy increases its performance as the time budget increases, from 4.3 at 100ms to 5.2 at 3000ms. Only the tree selection phase can use hints, and vanilla RIS-MCTS is still a poor player. Using hints in the simulation policy is more important, and while RIS-MCTS with a heuristic simulation policy is better with less than 1 second of budget, beyond that it only reaches the same level of performance as MO-ISMCTS.

V. FURTHER IMPROVEMENTS

We now consider further improvements specific to Hanabi that are required to get competition-winning performance from RIS-MCTS. We restrict the branching factor of the tree in V-A, and apply a convention that commonly emerges in human play of the game in V-B.

A. Action space restriction with Rules

One reason for the poor performance of RIS-MCTS is the large branching factor. In a four-player game, each player has 4 cards for 4 Discard actions, 4 Play actions, and up to 24 Hint actions (one per other player and card, for either colour or face value), for a branching factor of 32. Hence the search tree does not go very deep. RIS-MCTS reaches a decision depth (the mean depth of the deepest node) in the tree of 3.2 at 100ms, increasing to 5.2 at 3000ms. With a simulation policy, the mean depths are 2.5 and 3.8 due to the computational overhead of the simulation policy. This means it is only exploring its first move, and the immediate moves of up to the next three players.

TABLE II
RULES USED TO LIMIT THE BRANCHING FACTOR IN HANABI

1	Hint the most information (not previously given) to any other player
2	Hint about a playable card
3	Hint about a card that can be discarded
4	Hint missing information about a playable card
5	Hint full information about a discardable card
6	Hint full information about an unplayable (but not discardable) card
7	Play a card if we are at least 70% confident it is playable
8	Play a card is we are at least 40% confident it is playable, and we have 5 or fewer cards left in the deck
9	Discard the card that the player is most confident is discardable

A human player does not consider all possible moves. They will rarely play a card about which they have no information, and consider only a subset of all possible hints, for example currently playable or discardable cards. To construct a 'sensible' subset of actions, we use the Rules implemented in the Hanabi competition framework (see [24]). The Rules we use are listed in Table II. Each rule defines 0 (if invalid) or 1 action, and these define the only possibilities within tree search.

This cuts the maximum branching factor in a 4-player game from 32 to 9. The second section of Table I shows this dramatically improves the playing strength of the agent, and outclasses the Van Den Bergh heuristic with 100ms of thinking time. Some rules require a probability of a card being playable or discardable. This is calculated by considering the possible cards it could be, taking into account hints given and all cards visible to the player.

The results of an ablative study on the relative impact of restricting the action space via rule heuristics, and the RIS-MCTS changes are shown in Table I. With rule-based action restriction in MO-ISMCTS, hints made within the tree can now affect later actions, as the rules will use them to determine which actions are available later in the tree (the random simulation policy is not rule constrained). This helps the random simulation policy, but the Van Den Bergh policy gives worse results than with the full action space. RIS-MCTS now provides a benefit of 8-11 points over MO-ISMCTS, and beats the Van Den Bergh benchmark of 17.2 even with 100ms budget. The decision depth increases to 11.0 with a random simulation policy and 3000ms, and 7.7 with the Van Den

Bergh policy.

Other methods have been used to restrict the explored action space. Progressive un-pruning [29] uses a domain-specific heuristic to initially reduce the actions in the selection step so that only the best ones (according to the heuristic) are available, and actions are added to this as the number of visits to the node increases. This focuses the search on actions valued by the heuristic, but investigates all actions as the time-budget increases. Here we use Rules of Table II to provide the heuristic, and do not un-prune.

B. The Power of Convention

With a restricted communication channel to pass hidden information to team-members, Hanabi has some similarities to Bridge with a single team instead of two. As in Bridge, when a group of humans play Hanabi conventions form. A common example is the convention that a hint will tend to highlight an immediately playable card. If A tells B that they have a single blue card, then in the absence of other information B will often play this card, despite formally having no information about its face value. A convention overloads the hint with additional information that 'this card is playable now'. This means that some hints - e.g. hinting a card is red, when it is not currently playable - cannot be given if the convention is in use. A convention is useful if the hints that it prevents would rarely be useful, and increases the effective communication capacity. Foerster et al. [2] find that conventions learned by their agents contribute 40% of the information in Hints, with 60% from the 'grounded' rules of the game.

This 'playable now' convention is used by [25] in designing an 'intentional' AI for Hanabi, that plays nicely with humans; in the 'confidence' method of Bouzy 2017 [23] and is one of the rules that Canaan et al. [26] find helpful in their evolutionary search. We adopt a convention that if a player hints about a *single* card to the *next* player, then this card is immediately playable, unless this is impossible with information the player has). This affects the calculated probability that a card is playable or discardable. For example, if A hints to B about a single Red card they hold, and B has no other information about this card, then B will know this is 100% playable - unless all the playable red cards are visible elsewhere, in which case they know it is 100% discardable. We modify the rules to avoid giving hints that would be false under the convention. This increases the mean performance of our RIS-MCTS agents by up to 2 points (Table I as 'RIS-MCTS + C').

Table III compares the results of previous published work on Hanabi with the performance of our 'RIS-MCTS + C' algorithm. The results for Van den Bergh [21] use the re-implementation of their rules in the Hanabi CIG 2018 framework (see [24]), as the original paper only reports for 3-player games. The scores for 'Bouzy 2017' use that paper's 'Confidence' heuristic. Bouzy's Hat-guessing algorithm makes extensive use of the illegal Hints mentioned in III-B, and we exclude it. The use of 1s and 10s budgets for the RIS-MCTS algorithms is motivated by 1 second being an appropriate time for play with humans, and 10s being the budget per move that Bouzy's Expectimax Search algorithm uses in [23].

TABLE III
RESULTS FOR 2 TO 5 PLAYER HANABI GAMES. BEST PUBLISHED RESULTS IN BOLD. STANDARD ERROR ON RIS-MCTS IS ± 0.2 FOR 1S (200 GAMES) AND ± 0.3 FOR 10S (100 GAMES). '+ C' USES THE 'PLAYABLE NOW' CONVENTION; '+ VDB' USES VAN DEN BERGH SIMULATION.

Algorithm	2-P	3-P	4-P	5-P
Ozawa et al. 2015 [20]	15.9	-	-	-
Van den Bergh 2015 [21]	13.8	17.7	17.2	16.3
Cox et al. 2015 [22]	-	-	-	24.7
Bouzy 2017 [23]	15.9	17.9	19.7	19.2
Bouzy Expectimax 2017 [23]	19.0	20.4	21.1	20.4
Canaan et al. 2018 [26]	20.1	19.6	19.4	18.3
Foerster et al. 2018 [27]	**23.9**	-	-	-
Bard et al. 2019 [2]	22.7	20.2	21.6	16.8
RIS-MCTS (1s)	17.7	18.6	18.1	17.0
RIS-MCTS (10s)	17.9	18.9	18.2	17.1
RIS-MCTS + C (1s)	20.4	19.9	19.8	18.8
RIS-MCTS + C (10s)	20.6	19.8	19.7	18.5
RIS-MCTS + vdb (1s)	18.3	20.2	19.4	18.4
RIS-MCTS + vdb (10s)	20.0	21.0	20.2	19.3
RIS-MCTS + vdb + C (1s)	19.6	20.8	20.2	19.3
RIS-MCTS + vdb + C (10s)	20.5	22.0	21.3	20.0
WTFWThat [2]	19.5	**24.2**	**24.8**	**24.9**
SmartBot [2]	23.0	23.1	22.2	20.3

Table III shows no significant difference between 1s and 10s of computation time for RIS-MCTS when using a random simulation policy. With a Van Den Bergh simulation policy RIS-MCTS continues to improve with time budget and beats the 1-ply Expectimax search [23] that uses a similar time budget and convention. Overall RIS-MCTS with the 'playable now' convention yields what were, at the time of CIG 2018, state-of-the-art results in Hanabi. Two papers since then [2], [27] using Deep Reinforcement Learning provide a new benchmark and use two bots from outside the academic literature that beat RIS-MCTS. These are shown in the last two lines of Table III; *WTFWThat* uses a hat-guessing variant that is legal under standard Hanabi rules, and SmartBot uses a number of hand-crafted conventions modelled on high-level human play.

VI. MCTS IN 40 MILLISECONDS

The CIG 2018 Hanabi competition had a time limit of 40ms per move; in Table III, algorithms that would breach this limit are show in *italics*. To meet the time budget use ideas from Silver et al. [7] and Anthony et al. [30] and run RIS-MCTS games offline, then use their output to train an evaluation function; a similar approach was used for Monte Carlo search in Skat [31]. We use this trained function to make decisions within the 40ms time limit.

We learn a function approximator for the action-value function $Q(s, a)$, which predicts the end-game score after taking action a from state s. We record a feature representation for each move that combines the game state and action chosen. The 38 features used to summarise a state/action pair are detailed in the associated technical report [4].

Each move in an RIS-MCTS game generates one input-output pair for each action considered from the root during tree search. The output target for each of these is the mean

TABLE IV
HANABI RESULTS WITH 40MS LIMIT, FOR 2, 3, 4 AND 5 PLAYERS. THE STANDARD ERROR ON ALL RIS-MCTS FIGURES IS ± 0.1.

Algorithm	2-P	3-P	4-P	5-P
Van den Bergh 2015 [21]	13.8	17.7	17.2	16.3
Cox et al. 2015 [22]	-	-	-	24.7
Bouzy 2017 [23]	15.9	17.9	19.7	19.2
Canaan et al. 2018 [26]	20.1	19.6	19.4	18.3
Foerster et al. 2018 [27]	**23.9**	-	-	-
Bard et al. 2019 [2]	22.7	20.2	21.6	16.8
WTFWThat [2]	19.5	**24.2**	**24.8**	**24.9**
EvalFn (1)	20.0	19.5	19.0	17.8
RIS-MTCS-NR (1)	20.0	20.1	19.8	19.0
EvalFn (2)	20.0	19.9	19.6	18.5
RIS-MTCS-NR (2)	21.0	20.5	20.0	19.1
EvalFn (3)	18.8	20.0	20.1	19.2
RIS-MTCS-NR (3)	20.5	21.0	20.9	19.7

score of the child node. We use a shallow neural network to learn a mapping from input to output. The network has 38 input neurons and a single output neuron for the value of $Q(s, a)$. There is a single hidden layer with 30 neurons. To use this function approximator, we apply it separately to each of the possible actions implied by the rules in Table II and select the action that gives the highest predicted end-game score.

All Neural Networks were trained using DL4J [32], with Rectified Linear activation at the hidden layer, Rectified Tanh activations at the output, squared error loss, momentum of 0.9, a learning rate of 1e-4 and L1 normalization of 1e-5. The default Adam optimiser was used with a batch size of 16. All inputs were normalized to a mean of zero and variance of one.

Rather than just use the statistics at the root node at each move, as in previous work [7] [11], we also use statistics from deeper nodes in the tree. Each of these deeper nodes represents a game-state explored by the training game, and can be used to generate input-output training pairs in the same way as the root node. We exclude nodes with fewer than 50 visits to avoid noise from nodes not fully explored.

These deeper states form a penumbra of exploration away from the game actually played. If we restrict training to game-states that were actually encountered in the training games then we risk our evaluation function being fragile in play, as it will be sensitive to out-of-sample game states in which it may play badly. This is the insight behind the DAGGER algorithm in imitation learning [33], and [34] show that this fragility from training on only the states encountered by the expert that we are learning to imitate leads in general to a final error that increases as $O(T^2)$ in the worst case, where T is the number of sequential moves made. Adjusting the training data to sample from the distribution the function will encounter on test data reduces this dependence to $O(T)$, and our 'penumbra of exploration' makes a step in this direction.

The resultant networks can be used to play Hanabi directly. This is much faster than the 40ms time limit, so to make productive use of this time we use the same approach as DeepMind's AlphaZero [7]. Within the time budget we conduct MCTS with no simulation step, and estimate the value of new leaf nodes using the learned $Q(s, a)$. We initialise expanded nodes with $n(a) = 1$ and $V(a)$ equal to the estimate by the trained evaluation function. At a leaf node in the tree, all the child nodes are expanded at the same time instead of just one, as in [29], and the value of the highest estimate is used directly in back-propagation. Swiechowski et al. [11] use a similar approach in Hearthstone.

Table IV summarises the results. 'EvalFn (1)' is the value function trained on the results of 300 games using RIS-MCTS with random policy - all with 5 players and 3 seconds per move.'EvalFn (2)' is trained using 500 games using RIS-MCTS and 'EvalFn (1)' as the simulation policy, with 10 seconds per move. Finally 'EvalFn (3)' was trained using the output of 200 games, with 30 seconds per move, but with $C = 3$ instead of $C = 0.1$ used in the previous iterations. This change was designed to sample states from a broader penumbra of exploration around the base game trajectory. This broadening was necessary to obtain a performance improvement in the third iteration. 'RIS-MCTS-NR (n)' uses RIS-MCTS with 30ms per move, the 'playable now' convention and the leaf node valued by 'EvalFn (n)' with no further rollout (NR). A similar interleaving of imitation learning and data generation is used in [30].

Table IV shows an increase in performance as we iterate on the evaluation function. Using the trained evaluation function to define a greedy action plays Hanabi at least as well as the RIS-MCTS algorithm that generated the training data, and often much better. Using a 30ms search budget improves on the raw EvalFn, and Table I shows a 30ms budget is as effective as 1000ms in full RIS-MCTS.

VII. CONCLUSIONS AND FUTURE WORK

MO-ISMCTS overcomes the problem of strategy fusion for the acting player, but not for the other players in tree search. As detailed in Section IV-A, as the number of iterations of MO-ISMCTS increases, the actions of other players are predicted using perfect knowledge of the information known to the root player. Information leaks from the root player into its opponent models and causes strategy fusion in the opponents' decisions in the tree. RIS-MCTS re-determinizes at each node and avoids information known by the root player leaking into the opponent model within tree search.

This information leakage in IS-MCTS is particularly relevant to Hanabi because the unusual form of information hiding means we have knowledge about the other player's position that they do not. This means the leakage affects the next player's move directly, at depth two in the search tree. In the classic case where we have perfect information about our own position but not of the other players, this information leakage will be much weaker as it can only affect opponent decisions after our *next* move much deeper in the tree.

Cowling et al. [19], introduce 'fake' determinizations into opponent trees with the MT-ISMCTS algorithm to address this and the problem of non-locality. This better models the opponents' lack of full knowledge and incentivises bluffing. In a co-operative game like Hanabi we wish to impart genuine information to the other players ('signalling' rather than

'bluffing'), but the principle remains that there is no incentive to pass on information in vanilla MO-ISMCTS. Counterfactual Regret methods [17] sample game tree trajectories from the start of the game that do not lead to the current root game state. In all cases, sampling of game-states from outside the current game sub-tree is essential to address non-locality: what might have happened earlier in the game, but didn't, affects the optimal decision now. RIS-MCTS does not address the non-locality problem in any new way.

By re-determinizing at each node, RIS-MCTS creates determinized game states that are incompatible with the root information set, and introduces 'fake' determinizations as in MT-ISMCTS. The differences are that we do not maintain one opponent IS-MCTS tree for each of their possible information sets and these incompatible determinizations are created during each MCTS iteration as mutations from the root determinization. The introduction of non-compatible determinizations at opponent nodes in RIS-MCTS avoids strategy fusion, but back-propagates impossible game results to the root node. As Cowling et al. note [19], use of invalid determinizations in this way pollutes the tree statistics, and may lead to poorer decisions. This is a theoretical weakness in RIS-MCTS as described here, since the result of a determinized game should only be back-propagated to nodes which that game could actually have originated. We intend to address this aspect of RIS-MCTS, and investigate its performance in games with more classic forms of hidden information.

REFERENCES

[1] A. Bauza. (2010) Hanabi. [Online]. Available: "https://boardgamegeek.com/boardgame/98778/hanabi"

[2] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra *et al.*, "The hanabi challenge: A new frontier for ai research," *arXiv preprint arXiv:1902.00506*, 2019.

[3] 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.

[4] J. Goodman, "Re-determinizing information set monte carlo tree search in hanabi," *arXiv preprint arXiv:1902.06075*, 2019.

[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.

[6] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez *et al.*, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[7] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[8] I. Szita, G. Chaslot, and P. Spronck, "Monte-carlo tree search in settlers of catan," in *Advances in Computer Games*. Springer, 2009, pp. 21–32.

[9] P. I. Cowling, C. D. Ward, and E. J. Powley, "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*, vol. 4, no. 4, pp. 241–257, 2012.

[10] P. Nijssen and M. H. Winands, "Monte carlo tree search for the hide-and-seek game Scotland Yard," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 4, p. 282–294, 2012.

[11] M. Świechowski, T. Tajmajer, and A. Janusz, "Improving hearthstone ai by combining mcts and supervised learning algorithms," in *Computational Intelligence and Games (CIG), 2018 IEEE Conference on*. IEEE, 2018, pp. 445–452.

[12] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[13] S. Gelly and D. Silver, "Combining online and offline knowledge in uct," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 273–280.

[14] R. Bjarnason, A. Fern, and P. Tadepalli, "Lower bounding klondike solitaire with monte-carlo planning." in *ICAPS*, 2009.

[15] M. L. Ginsberg, "Gib: Imperfect information in a computationally challenging game," *Journal of Artificial Intelligence Research*, vol. 14, pp. 303–358, 2001.

[16] I. Frank and D. Basin, "Search in games with incomplete information: A case study using bridge card play," *Artificial Intelligence*, vol. 100, no. 1-2, pp. 87–123, 1998.

[17] M. Bitan and S. Kraus, "Combining prediction of human decisions with ismcts in imperfect information games," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1874–1876.

[18] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems*, 2010.

[19] P. I. Cowling, D. Whitehouse, and E. J. Powley, "Emergent bluffing and inference with monte carlo tree search," in *2015 IEEE Conference on Computational Intelligence and Games*. IEEE, 2015, pp. 114–121.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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*. IEEE, 2018, pp. 409–416.

[27] 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," *arXiv preprint arXiv:1811.01458*, 2018.

[28] D. Perez Liebana, J. Dieskau, M. Hunermund, S. Mostaghim, and S. Lucas, "Open loop search for general video game playing," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 337–344.

[29] G. Chaslot, M. H. Winands, H. van den Herik, J. Uiterwijk, and B. Bouzy, "Progressive strategies for monte-carlo tree search," *New Mathematics and Natural Computation*, vol. 4, no. 03, 2008.

[30] T. Anthony, Z. Tian, and D. Barber, "Thinking fast and slow with deep learning and tree search," in *Advances in Neural Information Processing Systems*, 2017, pp. 5360–5370.

[31] M. Buro, J. R. Long, T. Furtak, and N. Sturtevant, "Improving state evaluation, inference, and search in trick-based card games," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[32] Eclipse deeplearning4j development team. deeplearning4j: Open-source distributed deep learning for the jvm, apache software foundation license 2.0. [Online]. Available: "http://deeplearning4j.org"

[33] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[34] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 661–668.