Determinization and Information Set Monte Carlo Tree Search for the Card Game Dou Di Zhu

Daniel Whitehouse, Edward J. Powley, *Member, IEEE*, and Peter I. Cowling, *Member, IEEE*

Abstract*—*Determinization **is a technique for making decisions in games with stochasticity and/or imperfect information by sampling instances of the equivalent deterministic game of perfect information. *Monte-Carlo Tree Search (MCTS)* is an AI technique that has recently proved successful in the domain of deterministic games of perfect information. This paper studies the strengths and weaknesses of determinization coupled with MCTS on a game of imperfect information, the popular Chinese card game Dou Di Zhu. We compare a "cheating" agent (with access to hidden information) to an agent using determinization with random deals. We investigate the fraction of knowledge that a non-cheating agent could possibly infer about opponents' hidden cards. Furthermore, we show that an important source of error in determinization arises since this approach searches a tree that does not truly resemble the game tree for a game with stochasticity and imperfect information. Hence we introduce a novel variant of MCTS that operates directly on trees of information sets and show that our algorithm performs well in precisely those situations where determinization using random deals performs poorly.**

I. Introduction

Historically, the vast majority of research in game AI has focussed on games of *perfect information* (such as chess and checkers) in which the exact state of the game is visible to all players at all times. Work on AI for games of *imperfect information*, where the state of the game is only partially observable, is comparatively recent. One popular approach to such games is *determinization*. The game is reduced to several instances of a deterministic game of perfect information (called *determinizations*) compatible with the agent's observations of the game of imperfect information. Each of these determinizations is analysed by standard AI techniques (e.g. game tree search) and the results are combined to yield a decision for the original game. Determinization has proven successful in the domains of Bridge [1], Klondike solitaire [2] and probabilistic planning [3], among others, although there are situations where it performs poorly [4,5].

One recent innovation in AI for deterministic games of perfect information is *Monte-Carlo Tree Search (MCTS)* [6-8]. The best known example of an MCTS algorithm is *upper confidence for trees (UCT)* [6]. MCTS has proven particularly successful in games where depth-limited minimax search performs poorly, with Go [9] as a prime example.

Board and card games provide an interesting intermediate step in demonstrating the potential of MCTS for video games and real-time planning problems. Through studying games such as Dou Di Zhu, with hidden information and randomness as well as deep strategy for multiple cooperating and competing players, we aim to gain insights that may be used on still more complex domains.

In this paper we compare (for the popular Chinese card game Dou Di Zhu) the playing strength of an agent using a determinized version of UCT with that of a UCT agent that is able to "cheat" and observe the actual state of the game. It is not surprising that the latter outperforms the former. However, by considering a simplified version of the game, we argue that the majority of this apparent advantage is not attributable to the difference in information available to the two agents, but is instead a consequence of problems with the method of determinization itself.

There are three main advantages of cheating over determinization:

- Lack of *non-locality* and *strategy fusion* (two common problems with determinization that are discussed in Section II.B);
- *Inference* (knowledge of information that could conceivably be inferred from the opponent's decisions);
- *Clairvoyance* (knowledge of information that could not possibly be inferred).

Clearly the third of these advantages is insurmountable for a player that does not cheat. Our experimental evidence suggests that the effect of the second is negligible for Dou Di Zhu (this may be different for other games), and so addressing the first is the most promising avenue to obtaining a strong imperfect information player.

In light of this, we modify UCT to directly search trees for games of imperfect information, with the aim of overcoming the weaknesses of determinization. Our algorithm does not significantly outperform determinized UCT on average, but it succeeds in performing well for precisely those deals where the advantage of cheating UCT over determinized UCT is largest.

D. Whitehouse, E. J. Powley and P. I. Cowling are with the Artificial Intelligence Research Centre, School of Computing, Informatics and Media, University of Bradford, UK (e-mail: {d.whitehouse1, e.powley, p.i.cowling}@bradford.ac.uk).

This work is funded by grant EP/H049061/1 of the UK Engineering and Physical Sciences Research Council (EPSRC).

II. Background and Literature Review

A. Definitions and notation

This section introduces briefly the notions of *uncertainty* (stochasticity and/or imperfect information) in games, as well as the notation we use throughout this paper. For more details on the concepts introduced here we refer the reader to a standard textbook on game theory, e.g. [10].

A *game* can be thought of as a Markov process where, at each state, one of several agents (or *players*) is allowed to make a decision which influences the transition to the next state. A game is said to be *deterministic* if taking a particular action from a given state always leads to the same state transition. If a game is not deterministic, i.e. there is some element of randomness (or *chance*) to the transitions, the game is said to be *stochastic*. A game has *perfect information* if all players are able to observe the current state of the game. If a game does not have perfect information, i.e. the underlying Markov process is partially observable, the game is said to have *imperfect information*. For example, many card games are stochastic (because they are played with a shuffled deck) with imperfect information (because players are unable to observe opponents' cards).

In a game of imperfect information, the states as observed by each player are partitioned into *information sets*, defined as sets of states that are indistinguishable from the player's point of view. During the game, the player cannot observe the current state, but can observe the current information set.

In the notation of this paper, a game is defined by a set $\Lambda \subseteq S \times A$ of *state-action pairs* (for some sets S and A of *states* and *actions* respectively), along with a *state transition function* $f: \Lambda \to S$. The set of *legal actions* for a state $s \in S$ is denoted $A(s)$, and is defined as the set of actions $a \in A$ for which $(s, a) \in \Lambda$. The *player about to act* in a state s is denoted $\rho(s)$. If at time t the game is in state s, then player $\rho(s)$ chooses an action $a \in A(s)$, and the game transitions to state $f(s, a)$ at time $t + 1$. At a given moment, player j cannot observe the entire state s, but knows that the actual state must be in *information set* $[s]_j$. If two states are in the same information set then the player about to act must be the same in both; if the player about to act is the observer j, then the legal actions must also be the same. Thus we define $\rho([s]_j)$, and $A([s]_j)$ in the case where $\rho([s]_j) = j$, in the natural way.

B. Determinization

One approach to designing AI for games with stochasticity and/or imperfect information is *determinization*. For a stochastic game with imperfect information, a *determinization* is an instance of the equivalent deterministic game of perfect information, in which the current state is chosen from the AI agent's current information set, and the outcomes of all future chance events are fixed and known. For example, a determinization of a card game is an instance of the game where all players' cards and the shuffled deck are visible to all players. We can sample several determinizations from the current information set and for each one analyse the value of each action using AI techniques for deterministic games of perfect information. An example is Ginsberg's GIB system [1] which applies determinization to create an AI player for the card game Bridge that plays at the level of human experts. Buro et al [23] apply determinization to the card game Skat, attaining the level of a human expert player. Bjarnason [2] applies a determinized variant of UCT to the single-player card game Klondike Solitaire, resulting in an agent that achieves more than twice the estimated win rate of a human player. Determinized MCTS also shows promise in games such as Phantom Go [11] and Phantom Chess (Kriegspiel) [12], among others.

Despite these successes, determinization is not without its critics. Russell and Norvig [13] point out that "averaging over clairvoyance" will never result in plays that gather or hide information: from the point of view of the decision-making process in each determinization, all information is visible so there is nothing to gather or hide. Ginsberg [1] makes the same observations about the behaviour of GIB. Frank and Basin [4] identify two key problems with determinization:

- *Strategy fusion.* An AI agent cannot make different decisions from different states in the same information set; however, the deterministic solvers can and do make different decisions in different determinizations.
- *Non-locality.* Some determinizations may be vanishingly unlikely (rendering their solutions irrelevant to the overall decision process) due to the other players' abilities to direct play away from the corresponding states.

C. Dou Di Zhu

Dou Di Zhu [14] is a 3-player gambling card game, in the class of climbing games but also with bidding elements similar to trick taking games. Dou Di Zhu originated in China, and has increased in popularity there in recent years, particularly with internet versions of the game. One website reports 1,450,000 players per hour, and a 2008 tournament attracted 200,000 players [15].

The full rules of Dou Di Zhu are given in [14], and the variant we study is discussed in [16]. This section outlines the key features of the game from an AI perspective.

At the beginning of the game, cards are dealt to the players and one player is designated the *landlord*. Starting with the landlord, players take turns to play groups of cards from their hands. There are several *categories* of group: for example, one may play a single card, or a pair of cards of the same rank, or a sequence of five cards of consecutive rank, and so on. The first player can play any group available to them; subsequent players must play a group of the same category but of higher rank. Players also have the option of passing, regardless of whether they are able to play a group of cards. If two consecutive players pass, the third (i.e. the last player to play any cards) once again has a free choice of

group from any category. Cards are discarded once they are played, and the goal of the game is to be the first to play all cards in hand. If one of the non-landlord players achieves this then both are declared equal winners; if the landlord achieves it then he is the sole winner. Thus the non-landlord players must cooperate to compete against the landlord.

The branching factor in Dou Di Zhu varies based on the stage of the game, and particularly whether the current player must beat a previous group or has a free choice of category. For the first decision node in the game, a branching factor between 20 and 70 is typical; for subsequent nodes the branching factor is typically less than 10, and often 1 (when a player has no option but to pass).

In some categories, *kicker* cards are played along with the main cards in the group. This can result in a large branching factor: if there are n choices for the main group and m for the kickers then there are mn available moves in total. We address this with an approach similar to the move grouping approach of Childs et al [17], treating the choices of main group and kickers as two separate decisions at consecutive levels in the game tree. This increases the number of nodes and levels in the tree but decreases the branching factor at each node, and so is beneficial for tree search. Without this modification, the branching factor at the game's initial decision node can often exceed 300.

Dou Di Zhu is a game of imperfect information, as each player's cards in hand are hidden from the other two players (although the number of cards in hand is not hidden). A perfect information variant of Dou Di Zhu can be defined in the natural way, by allowing players to see each other's hands. It is worth noting that the perfect information variant retains some of the strategic depth of the original game; this is in contrast to games such as poker, which become trivial once the aspect of hidden information is removed.

D. Mini Dou Di Zhu

We introduce a simplified version of Dou Di Zhu that removes some of the complications of the full game and is small enough to be solved with exhaustive tree search techniques, while still retaining some of the strategic depth of the full game.

Mini Dou Di Zhu is a 2-player game, played with a reduced deck of 18 cards (four ranks and two jokers). Each player receives seven cards, and the four remaining cards are hidden from both players. There are four move categories, consisting of 1, 2, 3 or 4 cards of the same rank. As with the full game, the aim is to be the first player to play all cards, but unlike the full game there is no element of cooperation.

The total number of distinct deals in Mini Dou Di Zhu is 8832. The game trees for the perfect information variant are small enough that minimax search can be used to exactly determine the game theoretic value of each perfect information deal; when the non-uniform probabilities of obtaining each deal by shuffling and dealing cards are taken into account, approximately 70.7% of games are wins for player 1.

III. Design of Algorithms

The game tree for full Dou Di Zhu has a single chance node corresponding to the dealing of cards to each player. However, even after fixing the root player's own cards, the branching factor at this node is of the order 10^6, so searching this tree directly is impractical. Thus we introduce two classes of algorithm: one that uses determinization and searches each determinized game tree individually, and one that constructs and searches a tree of information sets. For comparison, we also consider "cheating" algorithms, which are allowed to observe the true state of the game and thus search the actual game tree. Each of these classes contains two algorithms: one based on exact tree search techniques (i.e. minimax or expectimax), and one based on MCTS. The former can only be applied to small games (such as Mini Dou Di Zhu), but they give us insight into the strengths and weaknesses of the latter.

A. Cheating minimax

The minimax algorithm can easily search the entire depth of the Mini Dou Di Zhu game tree. Minimax is optimal against a minimax opponent, but can occasionally make poor decisions against other types of opponent. For example, suppose that the minimax player has available two lines of play: one is a certain loss, while the other is a loss only if the opponent plays optimally from that point. Both lines have a minimax value of -1, and so minimax chooses between them arbitrarily. However, if there is any possibility that the opponent will make a "mistake" (i.e. deviate from the minimax policy, which is not unlikely if the opponent does not cheat), the second line is clearly the better choice.

To solve this problem, we equip minimax with the following tie-breaking mechanism. Each state s is assigned a value $m_\varepsilon(s)$ by

$$m_\varepsilon(s) = \max_{a \in A(s)} -m_\varepsilon\big(f(s,a)\big) + \varepsilon\left(\frac{\sum_{a \in A(s)} -m_\varepsilon\big(f(s,a)\big)}{|A(s)|}\right). \quad (1)$$

The first term is the standard negamax formulation of the minimax value; the second term is proportional to the expected value of playing a random action from state s. If two moves have the same minimax value, the tie will be broken by choosing the move that gives the opponent more opportunities to make a mistake. The constant ε must be small enough that if $m_0(s) < m_0(s')$ (where m_0 denotes the standard minimax value) then $m_\varepsilon(s) < m_\varepsilon(s')$, so that actions that maximise m_ε also maximise m; in other words, the set of actions identified as optimal by m_ε is a subset of those identified as optimal by m.

B. Cheating UCT

For consistency with determinized UCT (Section III.D), our cheating UCT agent uses multiple independent search trees: the root of each tree corresponds to the same (current)

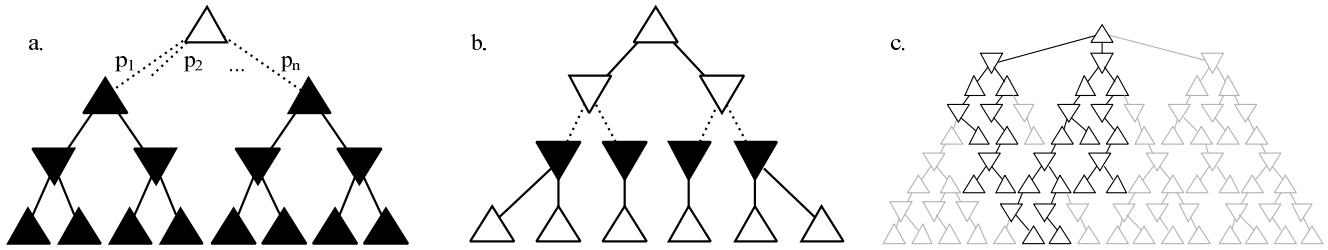

Fig. 1. Illustration of search trees for (a) determinization, (b) information set expectimax and (c) information set UCT. White triangles correspond to information sets, and black triangles to individual states. Triangles pointing upwards denote nodes belonging to the root (maximising) player, and triangles pointing downwards to the opponent (minimising) player. Solid lines denote decisions, whereas dashed lines denote decomposition of information sets into their constituent states. In (c), black nodes and edges are compatible with the current determinization and grey nodes and edges are not. All information sets are from the point of view of the root player.

state of the game, but each tree is searched by UCT independently. After the searches are completed, the numbers of visits for each action from the root are summed across all trees, and an action is chosen that maximises the total number of visits. This algorithm is equivalent to UCT with *root parallelization*, also known as *single-run parallelization* [18]. As Chaslot et al [19] observe, root parallelisation affects how UCT explores the game tree, and particularly the ability of UCT to escape local optima. For this reason, comparing determinized UCT (which uses multiple search trees) against UCT with root parallelisation is a fairer comparison than determinized UCT against UCT with a single search tree.

UCT does not appear to be as susceptible as minimax to the effects of poor tie-breaking. Exploration and random simulations in UCT are somewhat analogous to the second term in Equation (1), in that they cause suboptimal opponent actions to be factored into the evaluations of nodes.

C. Determinized minimax

We apply a determinization approach similar to the approach Ginsberg [1] uses for Bridge, searching multiple determinizations (states in the current information set) at each decision step.

In Mini Dou Di Zhu, the maximum number of states in an information set is 66, so it is feasible to iterate over all possible determinizations. We combine the results of the individual minimax searches by weighted majority voting: the "number of votes" for an action is the sum of probabilities of determinizations in which that action is chosen by minimax, and the action selected by the agent is one for which this sum is maximal.

The determinization process is illustrated in Fig. 1a. There is an initial branch for each possible determinization for the current information set, each labelled with its associated probability. Each determinization fixes the structure of the tree that is then searched by minimax.

D. Determinized UCT

Information sets in the full game of Dou Di Zhu are generally several orders of magnitude larger than those for Mini Dou Di Zhu: over the 1000 initial deals used in this paper (see Section IV.A), the maximum number of states in an information set is just over 2.6 million. Thus we must take the more conventional approach of sampling determinizations at random. So that the most likely determinizations have the greatest influence over the decision process, determinizations are sampled according to the probabilities of the states in the information set. Fixing the agent's own cards and randomly dealing the unseen cards to the opponents achieves this distribution.

As with cheating UCT, the results of the individual searches are combined by summing numbers of visits from the root.

E. Information set expectimax

Instead of searching determinized trees of states, the two algorithms below search trees of information sets. For Mini Dou Di Zhu, we use the *expectimax* search algorithm [13], which modifies the minimax algorithm to game trees containing chance nodes: the value of a chance node is the expected value of choosing one of its children at random. For trees of information sets, we treat the opponent's decision nodes as chance nodes (branching for the states in the current information set) followed by perfect information decision nodes. Each nonterminal information set $[s]_j$ is assigned a value $v_j([s]_j)$ recursively by

$$v_j([s]_j) = \begin{cases} \max\limits_{a \in A([s]_j)} v_j([f(s,a)]_j) & \text{if } \rho([s]_j) = j \\ \mathbb{E}_{s \in [s]_j} \min\limits_{a \in A(s)} v_j([f(s,a)]_j) & \text{if } \rho([s]_j) \neq j, \end{cases}$$

with terminal information sets assigned values of ± 1 for wins and losses in the usual way. The agent selects a move to maximise the value of the resulting information set. The search tree is illustrated in Fig. 1b.

For the notion of a tree of information sets to make sense, we must have a well-defined mapping from (information set, action) pairs to information sets. In other words, if states s and s' are in the same information set then we must have $f(s,a)$ in the same information set as $f(s',a)$ for all legal actions a, so that the extension of f to information sets given by $f([s]_j, a) = [f(s,a)]_j$ is well-defined. This is true for Dou Di Zhu, but is not true for games in general; extending the idea of information set search to games where this property does not hold is a subject for future work.

Define:

$\mu(N)$ = {All information sets reachable from the information set at node N}

$\mu(N, D) = \{n \in \mu(N) \mid n$ contains a state reachable from N in determinization $D\}$

$\mu_T(N) = \{n \in \mu(N) \mid n$ is in the search tree $T\}$

$\mu_T(N, D) = \{n \in \mu(N, D) \mid n$ is in the search tree $T\}$

$V(N)$ = number of visits to node N

$R(N)$ = total reward at node N

Repeat for a large number of iterations

N = root node of tree T

D = a random determinization of N

// Selection

Repeat until $\mu_T(N, D) = \emptyset$ or $\mu(N, D) \setminus \mu_T(N, D) \neq \emptyset$

Choose a child M of N for which $\frac{R(M)}{V(M)} + k\sqrt{\frac{\ln V(N)}{V(M)}}$ is maximal

Let $N = M$

// Expansion

If $\mu(N, D) \setminus \mu_T(N, D) \neq \emptyset$

Add any $M \in \mu(N, D) \setminus \mu_T(N, D)$ to T

Let $N = M$

// Simulation

r = result of performing a simulation restricted to D from N

// Backpropagation

For each ancestor M of N, including N itself and the root node

Increment $V(M)$ by 1

Increase $R(M)$ by r

Fig. 2. Pseudocode demonstrating how information set UCT constructs and searches a decision tree.

F. Information set UCT

Just as determinized (or cheating) UCT can be thought of as an analogue to determinized (or cheating) minimax, we introduce information set UCT as an analogue to information set expectimax.

In general the set of legal opponent actions from the current information set is not known, as it is a function of the (hidden) state of the game. What is known however is the set of actions that are legal in at least one state in the information set, i.e. the union of legal action sets for all states in the information set. Applying one of these actions leads to a new information set. Thus a tree of information sets can be constructed; the construction is similar to that of the information set expectimax tree, except that states are not explicitly considered at any point.

Some lines of play may be more likely than others, and some lines of play are more relevant to the decision than others. The search must strike a balance between the two when modelling decisions at opponent nodes. To gain a more realistic estimate of the outcome of the game, UCT's selection phase should generally favour a move that is likely to be available and has a good expected return over a highly unlikely move with an extremely high expected reward, although the latter cannot be ignored entirely. We achieve this by searching only one information set tree, but restricting each UCT iteration to a subset of the tree corresponding to a random determinization. Unlikely moves will be available in relatively few determinizations and so will be selected only rarely, but moves with high rewards will be exploited when they are available. This algorithm can be thought of as an extreme case of determinized UCT, where each determinization is allocated only a single UCT iteration, but these iterations operate on a single tree, rather than each on a separate tree. Hence information about the quality of actions given from different determinizations is accumulated at information set nodes so that the exploration/exploitation ideas of UCT continue to work.

The construction of the information set tree and its restriction to a determinization D are illustrated in Fig. 1c. Pseudocode for the information set UCT algorithm is given in Fig. 2.

G. Inference

In games of imperfect information, it is often possible to infer hidden information by observing the actions of the other players, according to some model of the other players' decision processes. This type of inference has frequently been applied to the game of poker [20,21], but also to other games such as Scrabble [22] and the card game Skat [23] which has similarities to Dou Di Zhu.

In Mini Dou Di Zhu, we can perform inference by applying an opponent model to all states in the current information set and comparing the observed move from the current information set with the opponent model's choice from each state: if the moves for a particular state do not match, we conclude that that state is not the true state of the game. This is Bayesian inference in the special case of pure policy opponents, where the probability of playing a move given a state is 0 or 1, similar to [22].

This type of inference requires consideration of all states in the current information set, which is infeasible for the full game of Dou Di Zhu. Developing an inference model for the full game is a subject for future work; the feature-based approach of Buro et al [23] is one possibility.

IV. Experimental results

A. Determinization for Dou Di Zhu

In [16] we study determinized UCT for Dou Di Zhu. Specifically, we investigate the effect of changing the number of randomly sampled determinizations and the number of UCT iterations performed for each determinization on the overall playing strength. We find that as long as both parameters are sufficiently large, their precise values do not have a significant impact on playing strength: 40 determinizations with 250 UCT iterations each is adequate, and so these are the parameters we use throughout this paper.

Although players' decisions have a significant effect on the outcome of a game of Dou Di Zhu, the effect is larger in some random deals than others. In an effort to reduce the variance of subsequent results and thus allow them to be compared more easily, we use a common set of 1000 Dou Di Zhu deals for all experiments in this paper. The practice of specifying deck ordering in advance is common in Bridge

TABLE 1
Playing strength of players with perfect versus imperfect information. Each row shows the win rate for the specified player(s) when they use cheating UCT or determinized UCT and all other players use determinized UCT.

	Cheating	Determinized	Difference
Player 1	49.9%	43.0%	6.9%
Player 2	65.6%	57.0%	8.6%
Player 3	68.9%	57.0%	11.9%
Players 2 & 3	78.0%	57.0%	21.0%

and Whist tournaments between human players, to minimise the effect of luck when comparing players. Details of how these deals are chosen are given in [16].

In this experiment (which also appears in [16]) we tested cheating UCT and determinized UCT against each other in various combinations. We take as a baseline the number of wins (playing one game from each of the 1000 deals identified above) when each player uses determinized UCT and measure the increase in numbers of wins when various players instead use cheating UCT. Table 1 shows the results of this experiment. These figures give us upper bounds on the performance of a non-cheating agent for Dou Di Zhu.

B. Exact algorithms for Mini Dou Di Zhu

1) Effects of cheating

To investigate the gap between cheating and determinization in Dou Di Zhu, we shift our attention to the simpler game of Mini Dou Di Zhu. This section describes an experiment to compare the playing strength of several AI agents for Mini Dou Di Zhu. Specifically, the agents are cheating minimax, determinized minimax and information set expectimax; for the latter two, we test variants with no inference model, Bayesian inference with an incorrect opponent model, and Bayesian inference with the correct opponent model. Here the "correct" opponent model uses exactly the same algorithm as the opponent, whereas the "incorrect" model uses a different, but still sensible, algorithm. The former can be considered a best case for the effectiveness of inference, whereas the latter is a more realistic test of its effectiveness against an unknown opponent.

In each case we iterate through all 8832 deals, playing a single game for each combination of agents; a single game suffices as all agents are deterministic. The measure of a particular agent's strength against some opponent is the probability that it wins a randomly dealt game, which is obtained by summing the probabilities of those deals from the 8832 that it wins. The results of this experiment are shown in Fig. 3. We see that the win rate for cheating minimax is approximately 40% greater than that of determinized minimax, although the exact value depends on the opponent type. It should come as no surprise that cheating outperforms determinization. The former has access to information that the latter does not, and so can make more informed decisions and better anticipate its opponent's responses.

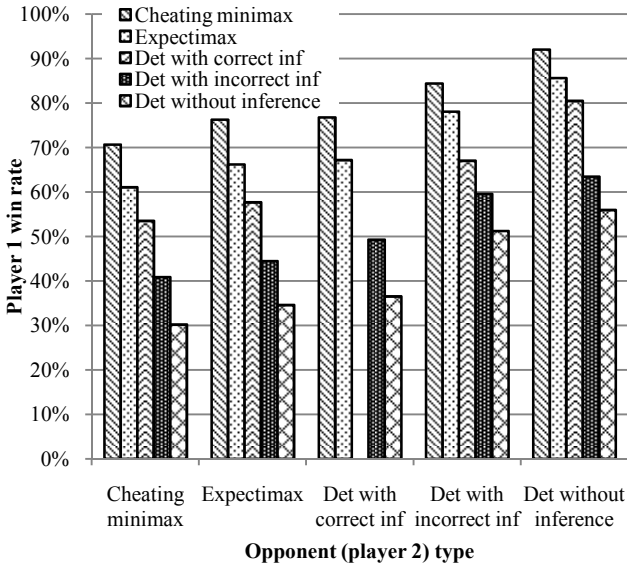

Fig. 3. Performance of several agents in Mini Dou Di Zhu. For each group of bars, player 2 uses the algorithm specified in the x-axis label; for each bar within a group, player 1 uses the algorithm specified by the legend. The bars themselves represent the win rate over all deals for player 1. We did not test determination with correct inference against itself: while possible, it is not straightforward to implement this in a way that does not require infinite nesting of opponent models.

Expectimax is a significant improvement over determinized minimax, outperforming it by around 30% and achieving a win rate around 10% lower than that of cheating minimax. This suggests that, contrary to intuition, approximately three quarters of the apparent benefit of cheating in Mini Dou Di Zhu can actually be attributed to the effects of strategy fusion and non-locality, from which expectimax does not suffer.

2) Effects of inference

Inference improves the strength of determinized minimax, increasing the win rate by around 10%, or 23% with a perfect opponent model. However, the performance of determinized minimax with inference still falls short of expectimax without inference. The determinized minimax agent with inference and the correct opponent model correctly identifies the actual state of the game (i.e. assigns the actual state probability 1 and all other states probability 0) by the fourth turn in 48% of deals against an expectimax opponent; by the end of the game, this proportion of deals increases to 74%.

Inference significantly improves the performance of determinized minimax; however, the same cannot be said for expectimax. We tested a variant of our expectimax player in which the probabilities used in calculating expected values at opponent nodes are determined by Bayesian inference. The resulting increase in playing strength is negligible: less than 0.2%. (Since these results are almost identical to those for expectimax without inference, they are omitted from Fig. 3.) One explanation could be that inferred information is

generally not helpful in Mini Dou Di Zhu, and the apparent benefit of inference in determinized minimax arises by reducing the number of determinizations to be searched, thus reducing the variance among the minimax results and ameliorating the effects of strategy fusion and non-locality.

C. Information set UCT for Mini Dou Di Zhu

To test the effectiveness of information set UCT, it was played against determinized UCT for Mini Dou Di Zhu. In all cases player 2 used determinized UCT with 20 trees and 200 UCT iterations per tree. Each of the 8832 deals was played 10 times with player 1 using determinized UCT with 20 trees and 200 UCT iterations per tree, and 10 times with player 1 using information set UCT with 4000 iterations. Information set UCT performed better with a 67.2% win rate versus 62.7% for determinized UCT. The next section investigates whether information set UCT outperforms determinized UCT in the full version of the game.

D. Information set UCT for Dou Di Zhu

Experiments were run to determine the amount of exploration that should be performed (i.e. the value of k in Fig. 2) and the number of iterations required for good performance. Each of the 1000 selected deals for Dou Di Zhu was played 5 times with the landlord player as information set UCT, an exploration constant of 0.44, and varying numbers of iterations (between 500 and 30000) against determinized UCT opponents with 40 trees and 250 iterations per tree. The results indicated that the playing strength of information set UCT increased up to 10000 iterations where it achieved a win rate of 40.8%. Increasing the number of iterations further had no significant effect on playing strength. Similarly each deal was played 5 times with the landlord player as information set UCT using 10000 iterations and varying values for the UCT exploration constant (between 0.1 and 3). The results indicated that the algorithm performs poorly with exploration less than 0.5 and achieves best performance with exploration greater than 1. Increasing the exploration beyond 1 had little effect on playing strength.

To measure the difference between cheating UCT, determinized UCT and information set UCT, each of our 1000 deals of Dou Di Zhu was played 40 times with the landlord as cheating UCT, determinized UCT and information set UCT. In all cases 40 trees with 250 iterations per tree or 10000 iterations in total were used and the exploration constant for information set UCT was chosen to be 1.0. The opponents used determinized UCT. First of all this data was used to calculate the average win rate for each player as the landlord. The results indicate that there is no significant difference in playing strength between information set UCT (42.0%) and determinized UCT (42.4%). Cheating UCT was much stronger, achieving a win rate of 54.0%.

The original aim of developing information set UCT was to overcome strategy fusion difficulties in the deals where the cheating player's advantage is largest. To investigate whether this has been achieved, the average win rate for each deal was calculated for each player type. Then for each deal the difference in win rate between cheating UCT and determinized UCT was calculated. By looking only at deals in which this difference is above a certain threshold, it is possible to compare the performance of information set UCT and determinized UCT in the deals where knowing the hidden information is most beneficial to the cheating player. Similarly the difference between information set UCT and determinized UCT can be compared for the deals in which knowing the hidden information is not beneficial. The results of this experiment are presented in Table 2.

TABLE 2
Win rates for determinized UCT and information set UCT. Each row shows win rates for a subset of the 1000 initial Dou Di Zhu deals, filtered by the difference in win rate between cheating UCT and determinized UCT ("Threshold" column).

Threshold	Determinized UCT win rate	Information set UCT win rate	Number of deals
None	42.4%	42.0%	1000
$< 0\%$	53.0%	45.2%	133
$= 0\%$	44.9%	43.3%	138
$> 0\%$	40.0%	41.1%	729
$> 25\%$	31.6%	38.4%	166
$> 50\%$	20.4%	35.0%	12
$> 75\%$	15.0%	95.0%	1

For deals where the threshold in Table 2 is less than or equal to zero, cheating UCT does not outperform determinized UCT, and the advantage of knowing the opponent's cards is not significant. In this situation, information set UCT offers no advantage and performs slightly worse than determinized UCT. When the threshold is greater than zero there is an advantage to knowing opponent cards, and information set UCT is stronger than determinized UCT. Furthermore, as the gap between cheating UCT and determinized UCT increases, the gap between information set UCT and determinized UCT increases also.

V. CONCLUSION

In this paper we compared the performance of a cheating UCT agent with that of a determinized UCT agent for Dou Di Zhu. By studying the performance of exact algorithms (cheating minimax, determinized minimax and expectimax) on a simplified version of the game, we provided evidence that a large proportion of the apparent benefit of cheating has less to do with gaining access to hidden information and more to do with overcoming the inherent shortcomings of determinization. Furthermore, the most obvious approach to closing the gap between perfect and imperfect information, namely inference, has little effect on the strength of an agent that does not use determinization. (Note that these results are specific to Mini Dou Di Zhu: inference is demonstrably beneficial in many other games.)

In light of this, we introduced a version of UCT that

operates on trees of information sets rather than trees of game states. This algorithm still uses determinization, but only as a mechanism for guiding the search during each iteration: rather than combining statistics from the root of each determinized tree after the search has ended, information set UCT combines statistics over the full depth of a single tree at every iteration of the search. Since much of the advantage of MCTS over non-tree-based Monte Carlo approaches appears to be through combining search information at all levels of the tree, information set UCT has the potential to exploit this property.

One potential problem with the algorithms presented here is that they assume the opponents have access to the player's hidden information: determinization does not randomise the player's own cards, and information set trees are built solely from the point of view of the root player. In a sense this is a worst case assumption, but it does mean that these algorithms can never exploit the opponents' lack of information. However, the solution is not as simple as merely randomising one's own cards during determinization: this assumes that the agent will not know its own cards on its next turn, making it impossible to plan more than one move ahead. Addressing this issue is particularly important for games where information hiding is a significant part of successful play.

Our experimental results show that information set UCT performs well in precisely those cases where the advantage of cheating UCT over determinized UCT is larger; unfortunately it seems that there are equally many cases where determinized UCT performs as well as cheating UCT, and slightly outperforms information set UCT, resulting in no significant increase in playing strength on average. Investigating why information set UCT fails in these cases and thus rectifying this failure is a subject for future work. If this problem can be solved, the next step is to generalise and apply the algorithm to wider classes of games with stochasticity, imperfect information and/or incomplete information.

Acknowledgements

We thank Jeff Rollason of AI Factory (www.aifactory.co.uk) for introducing us to Dou Di Zhu and for several useful and interesting conversations. We also thank the anonymous reviewers for their helpful comments.

References

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

[2] R. Bjarnason, A. Fern, and P. Tadepalli, "Lower bounding Klondike solitaire with Monte-Carlo planning," *Proc. ICAPS-2009*, 2009, p. 26–33.

[3] S. Yoon, A. Fern, and R. Givan, "FF-Replan: A Baseline for Probabilistic Planning," *Proc. ICAPS-2007*, 2007, p. 352–359.

[4] I. Frank and D. Basin, "Search in games with incomplete information: a case study using Bridge card play," *Artificial Intelligence*, 1998, pp. 87-123.

[5] J. Long, N. Sturtevant, and M. Buro, "Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search," *Proc. AAAI-10*, 2010, pp. 134-140.

[6] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," *Machine Learning: ECML 2006*, 2006, p. 282–293.

[7] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," *Proc. 5th International Conference on Computers and Games*, Springer-Verlag, 2006, p. 72–83.

[8] G. Chaslot, J.T. Saito, B. Bouzy, J. Uiterwijk, and H.J. Van Den Herik, "Monte-Carlo strategies for Computer Go," *Proc. 18th BeNeLux Conference on Artificial Intelligence*, 2006, p. 83–91.

[9] C.S. Lee, M.H. Wang, G. Chaslot, J.B. Hoock, A. Rimmel, O. Teytaud, S.R. Tsai, S.C. Hsu, and T.P. Hong, "The computational intelligence of MoGo revealed in Taiwan□s computer Go tournaments," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, 2009, p. 73–89.

[10] R.B. Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, 1997.

[11] J. Borsboom, J.-takeshi Saito, G. Chaslot, and J. Uiterwijk, "A comparison of Monte-Carlo methods for Phantom Go," *Proc. 19th Belgian–Dutch Conference on Artificial Intelligence–BNAIC*, 2007.

[12] P. Ciancarini and G.P. Favini, "Monte Carlo tree search in Kriegspiel," *Artificial Intelligence*, vol. 174, Jul. 2010, pp. 670-684.

[13] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2009.

[14] J. McLeod, "Dou Dizhu," 2010. http://www.pagat.com/climbing/doudizhu.html, accessed 28th March 2011.

[15] Wikipedia, "Dou Di Zhu," 2011. http://en.wikipedia.org/wiki/Dou_Di_Zhu, accessed 28th March 2011.

[16] E. Powley, D. Whitehouse, and P. Cowling, "Determinization in Monte-Carlo Tree Search for the card game Dou Di Zhu," *Proc. AISB 2011*, 2011.

[17] B.E. Childs, J.H. Brodeur, and L. Kocsis, "Transpositions and move groups in Monte Carlo tree search," *Proc. CIG□08*, 2008, p. 389–395.

[18] T. Cazenave and N. Jouandeau, "On the parallelization of UCT," *Proc. CGW07*, 2007, p. 93–101.

[19] G. Chaslot, M. Winands, and H. van Den Herik, "Parallel Monte-Carlo Tree Search," *Proc. CG 2008*, 2008, p. 60–71.

[20] M. Ponsen, J. Ramon, T. Croonenborghs, K. Driessens, and K. Tuyls, "Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker," *Proc. AAAI-08*, 2008, pp. 1485-1487.

[21] M. Ponsen, G. Gerritsen, and G. Chaslot, "Integrating Opponent Models with Monte-Carlo Tree Search in Poker," *Proc. Interactive Decision Theory and Game Theory Workshop at AAAI-10*, 2010, pp. 37-42.

[22] M. Richards and E. Amir, "Opponent modeling in Scrabble," *Proc. IJCAI 2007*, 2007, p. 1482–1487.

[23] M. Buro, J.R. Long, T. Furtak, and N. Sturtevant, "Improving state evaluation, inference, and search in trick-based card games," *Proc. IJCAI 2009*, 2009, pp. 1407-1413.