Determinization in Monte-Carlo Tree Search for the card game Dou Di Zhu

Edward Powley[1], **Daniel Whitehouse**[1], **and Peter Cowling**[1]

Abstract. *Monte-Carlo Tree Search (MCTS)* is a class of game tree search algorithms that have recently proven successful for deterministic games of perfect information, particularly the game of Go. *Determinization* is an AI technique for making decisions in stochastic games of imperfect information by analysing several instances of the equivalent deterministic game of perfect information. In this paper we combine determinization techniques with MCTS for the popular Chinese card game Dou Di Zhu. In determinized MCTS, there is a trade-off between the number of determinizations searched and the time spent searching each one; however, we show that this trade-off does not significantly affect the performance of determinized MCTS, as long as both quantities are sufficiently large. We also show that the ability to see opponents' hidden cards in Dou Di Zhu is a significant advantage, which suggests that inference techniques could potentially lead to much stronger play.

1 INTRODUCTION

From the inception of the field of game AI until relatively recently, the vast majority of research has focussed on games that are *deterministic* (do not have chance events) and have *perfect information* (all players can observe the exact state of the game). Chess [1] and checkers [2] are two well-known and well-studied examples of such games.

One recent innovation in AI for deterministic games of perfect information is *Monte-Carlo Tree Search (MCTS)*. MCTS iteratively builds a partial search tree, using random simulated games to evaluate nonterminal states. This makes it well-suited to games with long trajectories and high branching factor, as well as games where good heuristic state evaluation functions are difficult to find. The game of Go exemplifies both of these properties; MCTS has proven particularly successful for Go [3], and it is this success that has fuelled much recent interest in MCTS.

Work on AI for games that are *stochastic* (have chance events) or have *imperfect information* (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*), by randomly fixing the values of future chance events and hidden information. Each of these determinizations is analysed by AI techniques for deterministic games of perfect information and the results are combined to yield a decision for the original game. Although it is not a fool-proof approach for all games, determinization has proven successful in games such as bridge [4] and Klondike solitaire [5], as well as the domain of probabilistic planning [6].

There is an important trade-off to be made in determinization, between the number of determinizations, and the amount of effort spent on analysing each one. However, in this paper we present experimental results suggesting that this trade-off is not as important as it may appear at first: as long as both parameters are sufficiently large, adjusting the balance between them does not have a significant effect on playing strength.

It is common in games of imperfect information for certain pieces of information to be visible to some players but hidden from others. An important technique in such games is the ability to infer this information by observing the actions of other players. An upper bound can be placed on the performance gain that can be achieved by inference by measuring the performance of a player able to observe the exact state of the game, effectively modelling a player who is able to instantly and perfectly infer any hidden information. Using this method, we show that the potential benefit of inference is significant: if two cooperating players can observe the hidden information belonging to each other and to their opponent, then they can improve their win rate by around 21%.

The structure of this paper is as follows. Section 2 reviews existing work on MCTS and on AI for stochastic games of imperfect information. Section 3 introduces the card game Dou Di Zhu, which serves as our application domain for the remainder of this paper. Section 0 describes the version of determinized MCTS we use, and addresses some technical issues of how this is applied to Dou Di Zhu. Section 5 presents experimental results, including measurement of the variance in win rates for Dou Di Zhu (Section 5.1), the trade-off between number of determinizations and number of MCTS iterations per determinization (Section 5.2), and the potential increase in playing strength obtainable through inference of hidden information (Section 5.3). Finally, Section 6 gives some concluding remarks and directions for future work.

2 LITERATURE REVIEW AND BACKGROUND

2.1 Preliminaries

This section introduces briefly the notions of *uncertainty* (stochasticity and/or imperfect information) in games. For more details we refer the reader to a standard textbook on game theory, e.g. [7].

A *game* can be thought of as a multi-agent Markov decision process; that is, a Markov process where, at each state, one of the agents (or *players*) is allowed to make a decision which influences

[1]Artificial Intelligence Research Centre, School of Computing, Informatics and Media, University of Bradford, UK. Email: {e.powley, d.whitehouse1, p.i.cowling}@bradford.ac.uk.

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 no player is able to observe the cards held by the other players). Where there are stochastic state transitions, we may also regard this as a deterministic game where chance outcomes (e.g. the order of cards in a shuffled deck) are decided in advance, but are hidden from all players. This way of thinking gives rise to the determinization approach we will discuss later.

In a game of imperfect information, the states as observed by each player are partitioned into *information sets*, where an information set is defined as a set 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.

2.2 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) is a class of game tree search algorithms that make use of simulated games to evaluate non-terminal states. Simulated games select random actions until a terminal state is reached and the reward is averaged over multiple simulations to estimate the strength of each action. MCTS algorithms have gained in popularity in recent years due to their success in the field of computer Go [3]. In particular the *UCT* algorithm [8] proposed in 2006 has led to the recent upsurge of interest in MCTS algorithms. UCT enables a selective search to be performed where only the most promising areas of the tree are searched. This is particularly useful in domains with a large branching factor.

MCTS algorithms build a sub-tree of the entire decision tree where usually one new node is added after every simulation. Each node stores estimates of the rewards obtained by selecting each action and an improved estimate is available after every simulation step. In the case of the UCT algorithm, each decision in the tree is treated as a Multi-Armed Bandit problem where the arms are actions, and the rewards are the results of performing a Monte-Carlo simulation after selecting that action. The UCB algorithm [9] is used to guide the selection of actions at each decision in the tree. The UCB algorithm ensures that the tree will be selectively explored, performing a deeper search in more promising areas of the tree.

There has been some research on modifying UCT to achieve better performance for particular domains (such as Go), with the RAVE heuristic [10] being an example of such a modification. Also the algorithm can be executed in parallel [11] allowing MCTS agents to exploit modern multi-core and multi-processor hardware. UCT has been successfully applied to a wide variety of challenging domains such as General Game Playing [12], where the rules of the game are not known in advance and an AI player must use a very general approach which does not depend upon specific game knowledge. UCT requires very little domain knowledge and is therefore useful in domains where no good heuristics are known.

The structure of MCTS algorithms is generally quite similar: a discrete number of iterations are performed, after which an action from the root node is selected according to statistics collected about each action. Each simulation step performs four operations on the sub-tree built by the algorithm, namely Selection, Expansion, Simulation and Back-Propagation.

2.2.1 Selection

Starting from the root node, an action is selected at each node in the current partial game tree, according to some criterion. This mechanism is applied iteratively until it reaches either a terminal game state or a node which is not in the current partial game tree.

In the case of the UCT algorithm the mechanism used for traversing the tree is the UCB algorithm. After initially selecting every action once, the UCB algorithm selects the action which maximizes

$$\frac{\sum_{t=1}^{v_j(n)} X_{j,t}}{v_j(n)} + k\sqrt{\frac{\ln(n)}{v_j(n)}} \qquad (1)$$

where $X_{j,t}$ denotes the reward obtained by selecting arm j on trial t, n is the number of times the current node has been visited previously, $v_j(n)$ is the number of these visits in which action j was selected, and k is a constant controlling the balance between exploration and exploitation (often $k = \sqrt{2}$). The UCB algorithm balances exploration of untried moves with exploitation of known good moves. This enables UCT to find the most promising areas of the tree and search these further.

2.2.2 Expansion

If the state reached is terminal then there is no need to expand the tree or perform a simulation and the final result of the game is Back-Propagated. If the state is not terminal then it is added to the tree.

2.2.3 Simulation

Starting from the node added in the expansion phase (or the selected leaf node if no node was added), the game is played out to completion by choosing random moves for each player. The result of this simulated game is then used to update the statistics in the tree during Back-Propagation. Choosing an effective method for quickly selecting random moves which give rise to unbiased game simulations, and which are also realistic enough to be strongly correlated with the eventual winner from a position, appears crucial to strong play in MCTS, particularly for computer Go [13].

2.2.4 Back-Propagation

Each node visited during selection has its statistics updated. In the case of the UCT algorithm the number of times the node has been visited and the cumulative reward of simulations passing through that node (from the point of view of the player selecting the action at the node) are recorded. These are then used in turn by the UCB algorithm during the next selection phase.

2.3 AI for games with uncertainty

This section briefly surveys research on AI for games with stochasticity and/or imperfect information.

2.3.1 Determinization

One approach to designing AI for games with stochasticity and/or imperfect information is *determinization*, also known as *perfect information Monte Carlo (PIMC)*. For an instance of 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 then create several determinizations from the current game state, analyse each one using AI techniques for deterministic games of perfect information, and combine these decisions to yield a decision for the original game.

Ginsberg's GIB system [4] applies determinization to create an AI player for the card game Bridge which plays at the level of human experts. GIB begins by sampling a set D of card deals consistent with the current state of the game. (The question of how D is sampled is rather more subtle than it first appears, and indeed Ginsberg [4] does not give complete details). For each of these deals $d \in D$ and for each available action a, the perfect information ("double dummy") game is searched to find the score $\mu(a, d)$ resulting from playing action a in determinization d. The search uses a highly optimised exhaustive search of the double dummy Bridge game tree. Finally, GIB chooses the action for which the sum $\sum_{d \in D} \mu(a, d)$ is maximal.

A domain closely related to (single-player) stochastic games of imperfect information is that of probabilistic planning. Yoon et al's FF-Replan [6] and FF-Hindsight [14] systems apply determinization to probabilistic planning problems. Despite the simplicity of its approach, FF-Replan outperformed more complex systems to win the IPPC-04 and perform well in the IPPC-06 probabilistic planning competitions.

Bjarnason et al [5] present a variant of UCT for stochastic games, called *Sparse UCT* and apply it to the single-player card game of Klondike Solitaire. In a stochastic game, a single state-action pair can lead to multiple successor states (corresponding to the possible outcomes of the chance event); Sparse UCT handles this by allowing multiple child nodes for each action from each state. The selection phase selects actions using UCB, but the traversal to child nodes is stochastic, as is the addition of child nodes during expansion. Sparse UCT imposes an upper limit w on the number of child nodes for each state-action pair.

Bjarnason et al [5] also study an *ensemble* version of Sparse UCT, in which several search trees are constructed independently and their results (the expected rewards of actions at the root) are averaged. They find that ensemble variants of UCT often produce better results in less time than their single-tree counterparts. The ensemble case with $w = 1$, which Bjarnason et al call *HOP-UCT*, is equivalent to a straightforward application of determinization (more specifically, *hindsight optimisation* [15]) with UCT as deterministic solver, in which the determinization is constructed lazily as UCT encounters each chance event.

Bjarnason et al [5] treat Klondike Solitaire as a stochastic game of perfect information: rather than being fixed from the start of the game, the values of face down cards are determined as chance events at the moment they are revealed. This works for single-player games where the hidden information does not influence the game until it is revealed, but generally does not work for multiplayer games where the hidden information influences the other players' available and chosen actions from the beginning of the game. Hence the specific methods of Sparse UCT and lazy determinization are not immediately applicable to multiplayer games, but the general ideas are transferable.

Bjarnason et al show that Sparse UCT is able to win around 35% of Klondike Solitaire games, which more than doubles the estimated win rate for human players. Determinized MCTS also shows promise in games such as Phantom Go [16] and Phantom Chess (Kriegspiel) [17], among others.

Despite these successes, determinization is not without its critics. Russell and Norvig [18] describe it (somewhat dismissively) as "averaging over clairvoyance". They point out that determinization will never choose to make an information gathering play (i.e. a play that causes an opponent to reveal some hidden information) nor will it make an information hiding play (i.e. a play that avoids revealing some of the agent's hidden information to an opponent). Ginsberg [4] adds weight to this claim by making the same observations about GIB specifically. One explanation for this is that, from the point of view of the decision-making process in each determinization, there is no hidden information to gather or hide.

Russell and Norvig's criticisms of determinization are valid but equally valid are the experimental successes of determinization. Frank and Basin [19] identify two key problems with determinization:

1. *Strategy fusion.* An AI agent can obviously not make different decisions from different states in the same information set (since, by definition, it cannot distinguish such states); however, the deterministic solvers can and do make different decisions in different determinizations.
2. *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.

Building on the work of Frank and Basin, Long et al [20] identify three parameters of game trees and show that the effectiveness of determinization has some dependence on the game's position in this parameter space. The parameters measure the ability of a player to influence the outcome of a game in its late stages (*leaf correlation*), the bias in the game towards a particular player (*bias*) and the rate at which hidden information is revealed (*disambiguation*). Long et al [20] demonstrate how these parameters can be used to predict whether determinization is an appropriate method for a given game.

2.3.2 Other approaches

One alternative approach to tree search for stochastic games is *expectimax search* [18]. This is a modification of the well-known minimax algorithm to game trees containing chance nodes. The value of a chance node is the expected value of a randomly chosen child (i.e. the sum of the values of its children weighted by the probabilities of the corresponding chance outcomes). Unfortunately, expectimax is not well-suited to the type of card games studied in this paper, where there is a single chance node with a very large branching factor (corresponding to the initial shuffling of the deck), as the resulting game tree is too large to search even with MCTS.

As noted previously, the field of game AI focussed on deterministic games of perfect information until relatively recently. The same is not true in the field of game theory, where stochastic games of imperfect information have been well studied [7]. Thus a popular approach to AI for such games is to compute (or approximate) a Nash equilibrium strategy; examples of this approach include Gala [21] and counterfactual regret [22]. While there is undeniable appeal in finding a strategy that is provably optimal, we feel that searching for Nash equilibria is often not the

most appropriate approach to building strong game AI. The definition of Nash equilibrium requires only that the strategy is optimal against other optimal (Nash) strategies, so Nash strategies often fail to fully exploit suboptimal opponents.

2.3.3 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 [23-25], but also to other games such as Scrabble [26] and the card game Skat [27] which has similarities to the card game Dou Di Zhu which we study in this paper. Developing an inference engine for Dou Di Zhu is a subject of current and future work. Section 5.3 discusses the increase in playing strength we expect to see from a strong inference engine.

3 DOU DI ZHU

3.1 History and Popularity of Dou Di Zhu

Dou Di Zhu [28] is a 3-player gambling card game which originated in China. It falls into the class of Climbing Games but also has similar elements to Trick Taking Games. The name Dou Di Zhu translates into English as "Fight The Landlord" and is a reference to the class struggle during the Cultural Revolution in China where peasants were authorized to violate the human rights of their Landlords. In the original version of the game two players play compete together against a third player, the Landlord. There are other versions of the game involving four and five players but these are less popular.

The game was only played in a few regions of China until quite recently, when versions of the game on the Internet have led to an increase in the popularity of the game throughout the whole country. Today Dou Di Zhu is played by millions of people online, although almost exclusively in China, with one website reporting 1,450,000 players per hour. In addition there have been several major Dou Di Zhu tournaments including one in 2008 which attracted 200,000 players.Dou Di Zhu is interesting from an AI perspective as it necessitates both competition (between the Landlord and the other two players) and cooperation (between the two non-Landlord players).

3.2 Rules

Dou Di Zhu[2] uses a standard 52 card deck with the addition of a black joker and a red joker. Suit is irrelevant but the cards are ranked with 3 being the lowest rank and 2 being the highest rank (higher than Ace). The jokers are ranked higher than a two, with the red joker ranked higher than the black joker. Each player is dealt a hand of 17 cards from a shuffled deck and the remaining three cards are placed faced down on the table.

3.2.1 Bidding Phase

Each player takes turns to bid on their hand with the possible bids being 1, 2 or 3 chips. Bids must be strictly higher than the current bid but each player has the option to pass. This continues until two of the players pass consecutively. If any player bids 3 chips then the bidding phase immediately ends. If all three players initially pass, the cards are shuffled and dealt again. The winner of the bidding phase is designated as the *Landlord* and this player adds the three extra cards on the table into their hand, and plays first. The winning bid determines the *stake* for the game.

3.2.2 Card Play Phase

The goal of the game is to be the first to get rid of all cards in hand. If the Landlord wins, the other two players must each pay the stake to the Landlord. However if either of the other two players wins, the Landlord pays the stake to both opponents. This means the two non-Landlord players must cooperate to beat the Landlord. The Landlord always plays first and then play moves around the table in a fixed direction. At the end of the game the stake is doubled if a player has failed to remove any cards from their hand.

The card play takes place in a number of rounds until a player has no cards left. Whoever plays first can play any group of cards from their hand provided this group is a member of one of the legal *move categories* (see Table 1). The next player can play a group of cards from their hand provided this group is in the same category and has a higher rank than the group played by the previous player. If a player has no compatible group they must pass. This continues until two players pass, at which point the next player wins that round and may start a new round by playing a group of cards from any category.

There are a few special rules for some of the categories. If the move being played is a straight run (of singles, pairs or trios) then the next player must play a straight run of the same type and length, ending on a higher ranked card. Straight runs can contain cards of any rank from 3 to Ace, but not cards of rank 2 or jokers. A Bomb or a Nuke (also known as a Rocket) may be played at any point but doubles the stake of the game. A Bomb may be followed by another Bomb of higher rank but not a Quadplex.

Some categories allow extra *kicker* cards to be played with the group which have no effect on the category or rank of the move being played. A kicker can be any card provided it is of different rank to all the cards in the main group. If the kicker cards are single cards they must be of different rank and if the kicker cards are pairs they must be differently ranked pairs. Also a Nuke cannot be used as a kicker. If a move with kickers is played, the next player must play a move in the same category with the same number of kickers, although the ranks of the kicker cards are ignored.

Table 1 summarises the categories of moves in Dou Di Zhu.

Table 1. Description of the different move categories in Dou Di Zhu

Name	Description
Solo	Any individual card, for example A or 2. It is also possible to play runs of sequential cards with length at least 5, for example 345678 or 89TJQKA.
Pair	Any pair of identically ranked cards for example 55 or 77. It is possible to play runs of sequential pairs with length at least 3, for example 334455 or TTJJQQKK.
Trio	Any three identically ranked cards for example AAA or 888. It is possible to play runs of sequential trios of any length, for example 444555 or TTTJJJQQQ. Each trio may also have a kicker attached, for example 444555TJ or 999QQ.
Quadplex	Any four identically ranked cards with two kickers attached, for examples 4444TJ or 999955KK.
Bomb	Any four identically ranked cards, for example 5555 or 2222.
Nuke	The red joker and the black joker together.

[2] There are different versions of the rules of Dou Di Zhu. The rules used in this work were taken from [28].

3.3 Basic Strategy

Dou Di Zhu is a game which requires substantial levels of skill from expert human players. The player who wins the bidding phase usually has a high confidence they were dealt a good hand although they have three extra cards to discard. Often a hand contains lower ranked individual cards that cannot form part of a bigger group and it is possible to play these by using them as kicker cards. In general most plays should reduce the number of moves needed to get rid of all cards. Bombs and Nukes are powerful since they can be played at any point, but they double the stake of the game. For this reason playing them early in the game is risky.

Starting a new category is a good position to be in, allowing a player to choose a category where he holds multiple groups, or holds a high-ranking group that opponents are unlikely to be able to play on. Hence inference is an important aspect of Dou Di Zhu especially concerning the location of the high ranked cards in opponents' hands. The two non-landlord players also need to work together since they either both win or both lose. This can be achieved by making plays that allow the other non-Landlord player to play cards or prevent the Landlord from making further plays.

4 METHODOLOGY

In order to focus only on the card play phase, it is possible to remove the bidding phase and designate that an arbitrarily chosen player gets the extra cards and is the Landlord.

4.1 Perfect Information Dou Di Zhu

It is possible to modify Dou Di Zhu to be a game of perfect information by playing with all cards face up at all times. This removes the need to make inferences about other player's cards and allows the exact consequences of every action to be studied when searching the game tree. This version of the game we call *Perfect Information Dou Di Zhu.*

In order to compare the results of playing multiple games of Dou Di Zhu, we count numbers of wins instead of cumulative rewards, ignoring the effects of player bidding and bombs/nukes. We award a player 1 point for winning a dealt hand and 0 points for a loss. Since the UCB algorithm has been well studied with rewards in $\{0,1\}$ we can benefit from the work of others in setting the parameter k which trades off exploration and exploitation in the UCB formula (1).

4.2 Imperfect Information Dou Di Zhu

By hiding from each player the cards held in the other players' hands during a game of Perfect Information Dou Di Zhu we create the game of *Imperfect Information Dou Di Zhu.* Note that the number of cards held by each player and the history of cards played so far are still visible to all players, only the ranks of the held cards are hidden. Since there are unknown cards, the game tree has a huge branching factor for every possible combination of cards each player could hold. Instead of searching this tree we apply a determinization approach similar to the approach Ginsberg [4] uses for Bridge. This involves searching multiple determinizations of the game at each decision step. However, where Ginsberg uses minimax search, we use the UCT algorithm.

At each decision our agent samples multiple determinizations by randomly distributing the hidden cards between the other players. Each of these determinizations is a game of Perfect Information Dou Di Zhu, to which the UCT algorithm is then applied. The actions available in the first layer of the tree are the same for each determinization since the cards our agent holds are the same in each case. The visits to each action for each determinization are summed and our agent selects the action with the highest total number of visits.

5 EXPERIMENTS

In the experiments described in this section we use determinized UCT, where d denotes the number of determinizations (i.e. d independent search trees) and s denotes the number of UCT iterations per determinization.

5.1 Variation in Dou Di Zhu win rates

Although the strength of decisions made by each player has a significant effect on the outcome of a game of Dou Di Zhu, some random deals may favour one player over another, whereas others may be much more sensitive to the players' decisions. In an effort to reduce the variance of subsequent results and thus allow them to be compared more easily, we begin by choosing a set of 1000 Dou Di Zhu deals to be used for the remainder of the experiments. The practice of specifying deck ordering in advance is common in Bridge and Whist tournaments between human players, to minimise the effect of luck when comparing players. This set of deals is chosen such that when played by a set of identical AI agents (in this case determinized UCT), the number of wins for a particular player is as close as possible to the mean number of wins for 1000 random deals. In order to choose such a set, we must first determine what that mean is.

We generated 100 sets of 1000 random Dou Di Zhu deals, and for each deal we played a single game with three determinized UCT players, each with $d = 50$ determinizations and $s = 250$ UCT iterations per determinization. For each set, we recorded how many of the 1000 games were won by player 1. Figure 1 shows a histogram of these numbers of wins.

From these results we find that the number of wins appears to be normally distributed. The mean is $\mu = 433.47$; thus we choose a set of deals for which player 1 achieved exactly 433 wins in this experiment as our "representative" set for the remainder of this paper. The standard deviation is $\sigma = 16.27$ and so a 95% confidence interval for the mean number of wins for player 1 is [433.37, 433.57].

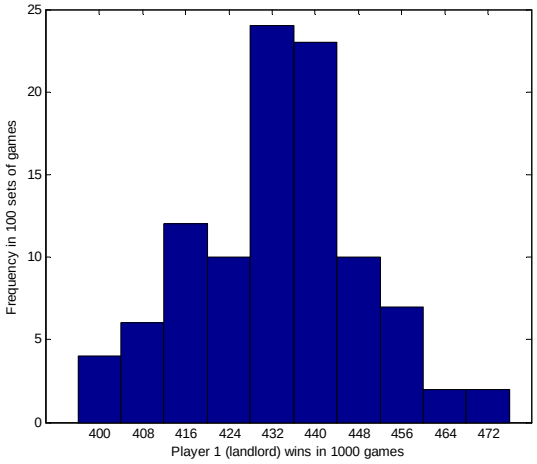

Figure 1. Histogram of win rates for the landlord player in 100 sets of 1000 Dou Di Zhu games.

5.2 Effect of d and s on playing strength

In these experiments, we played a number of games for each of our 1000 deals. In each game, players 2 and 3 used determinized UCT with 40 determinizations and 250 UCT iterations per determinization, whereas player 1 used determinized UCT with d determinizations and s iterations per determinization, each game with a different value for d and/or s. For each combination of d and s, we counted how many games out of the 1000 trials were won by player 1.

5.2.1 Varying d while s remains fixed

In this first experiment, we fixed four values for s, namely 50, 100, 250 and 500. For each of these, we used a number of values for d, ranging from 1 to 100.

Figure 2 plots the number of wins against d. We see that playing strength increases rapidly with $d < 20$, with rapidly diminishing returns for $d \geq 20$. However, there seems to be slightly more benefit to increasing the number of determinizations beyond 30 when the number of UCT iterations is low.

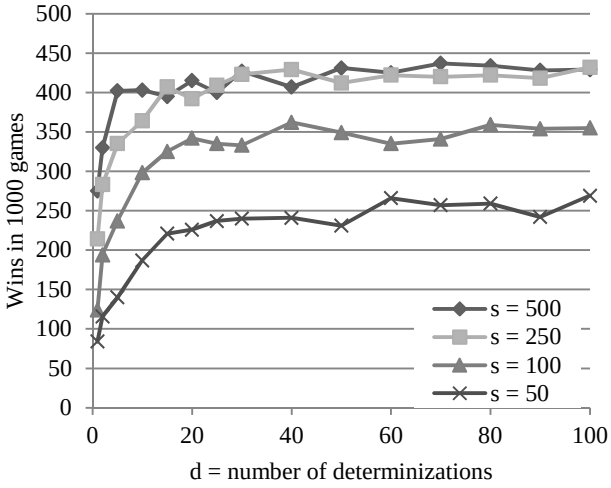

Figure 2. Plot of number of landlord (player 1) wins against number of determinizations, for fixed numbers of UCT iterations per determinization.

5.2.2 Varying s while d remains fixed

The conditions for this experiment were similar to those for the previous experiment, with the exception that we fixed four values of d (namely 5, 10, 25 and 40) and varied s (from 25 to 1000). The results are plotted in Figure 3. For $s \leq 300$ the playing strength increases logarithmically with the number of simulations, levelling off for $s > 300$.

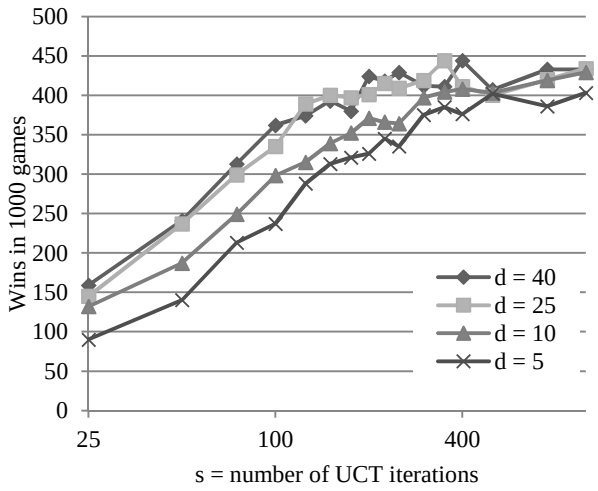

Figure 3. Plot of number of landlord (player 1) wins against number of UCT iterations per determinization, for fixed numbers of determinizations.

5.2.3 Varying d and s while ds remains fixed

Arguably the fairest comparison of the relative strength of different AI agents is to allocate them the same length of time (or the same number of CPU cycles) to make their decisions. However, this is not ideal from a scientific point of view since such measurements are inherently noisy and depend on issues of implementation and hardware. Instead, we simulate this by allocating a fixed number of UCT iterations per decision, making the (not unreasonable) assumptions that the time for a single iteration is roughly constant and the overall decision time is roughly linear in the number of iterations.

In this experiment we fixed four values of a constant Σ (namely 2500, 5000, 10000 and 15000), varied d from 1 to 100, and took $s = \left\lfloor \frac{\Sigma}{d} \right\rfloor$ for each value of d (so that $ds \approx \Sigma$). With our current implementation and hardware $\Sigma = 10000$ equates to approximately 1 CPU second of computation per move. Note that in this experiment (as in the preceding two experiments), players 2 and 3 have $ds = 40 \times 250 = 10000$.

Figure 4 plots the number of wins for player 1 against d. Given the results of the preceding experiments, it is not surprising that determinized UCT is weaker when d or s is too small, nor that its strength is somewhat independent of these parameters when both are sufficiently large.

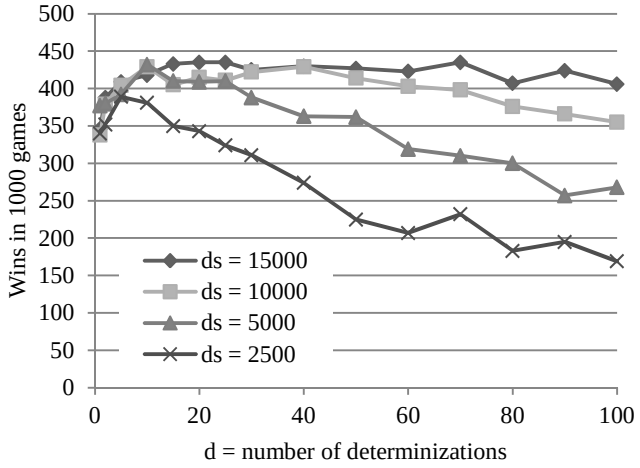

Figure 4. Plot of number of landlord (player 1) wins against number of determinizations, for a fixed number of UCT simulations divided equally among all determinizations.

5.3 Benefit of observing hidden information

As noted in Section 2.3.3, it is often useful in games of imperfect information to make inferences about the hidden information. However, it is not always clear how useful this inference may be. Richards and Amir [26] demonstrate the usefulness of inference in Scrabble by playing an AI agent with no inference engine against an AI agent with the ability to "cheat" and observe directly its opponent's hidden rack of letters. The benefit of direct access to the hidden information is an upper bound on the possible benefits of inference.

In our experiment for Dou Di Zhu, we played two types of AI agent against each other in various combinations: the determinized UCT player used in previous experiments (with $d = 40$ and $s = 250$) and a perfect information player in which the 40 determinizations are replaced by 40 copies of the actual state of the game. (This player still uses multiple UCT trees, but all trees have the same root state, which is the actual state of the game). Specifically, we take as a baseline the number of wins when each player uses determinized UCT (i.e. has only imperfect information) and measure the increase in numbers of wins when various players instead use perfect information.

Table 2 and Figure 5 show the results of this experiment. We see that knowledge of the other players' hidden cards increases the number of wins by around 7-12% (the benefit appears to be slightly less for the landlord, and slightly greater for the player preceding the landlord). Furthermore, if both non-landlord players can observe the hidden information, their win rate increases by around 21%.

When all three players have perfect information, players 2 and 3 win 12.7% more games than if all three have imperfect information. This suggests that perfect information is more beneficial to the non-landlord players than to the landlord.

Table 2. The increase in numbers of wins over 1000 games when the specified player(s) gain perfect information, and all other player(s) have imperfect information. For example, if player 2 has perfect information and players 1 and 3 have imperfect information, player 2 wins 86 more games than if all three players have imperfect information. Note that a win for player 2 is also counted as a win for player 3, and vice versa.

Player(s) with perfect information	Increase in number of wins (1000 games)
Player 1	69
Player 2	86
Player 3	119
Players 2 and 3	210

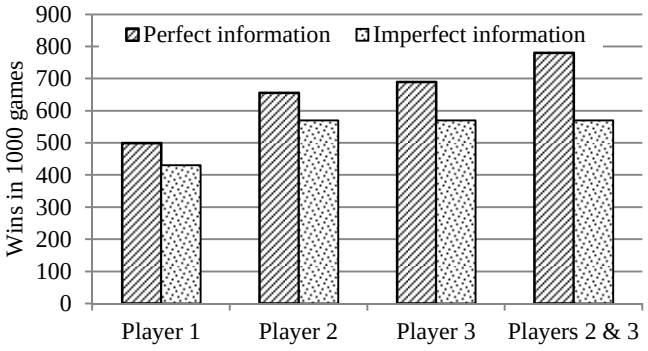

Figure 5. Number of wins for players with perfect versus imperfect information. Each pair of bars shows the numbers of wins for the specified player(s) when they have perfect or imperfect information and all other players have imperfect information.

6 CONCLUSION

In this paper we presented the hugely popular Chinese card game Dou Di Zhu as an interesting game for investigation. We applied a determinization approach using the UCT algorithm to search each determinization. Although we have yet to accurately determine the playing strength of the resulting AI agent (not least due to the lack of other strong AI for Dou Di Zhu against which to test) our informal experiments (playing against this paper's authors) suggest that it is on a par with human players. The nature of Dou Di Zhu suggests that the effects of some of the problems with determinization outlined in Section 2.3.1, are unlikely to be major: in particular, strategies such as information hiding or bluffing do not generally play an important role.

We showed that the performance of determinized UCT is not particularly sensitive to the trade-off between the number of determinizations and the number of UCT iterations per determinization. When the number of determinizations is fixed, performance scales approximately logarithmically with respect to the number of UCT iterations; when the number of simulations is fixed, performance increases rapidly with the number of determinizations until some threshold (approximately 20 in our experiments) is reached, after which increasing the number of determinizations does not significantly increase the playing strength. We do not see any obvious reason why these results should be specific to Dou Di Zhu; however we plan to repeat these experiments for other stochastic games of imperfect information to investigate this further.

We have shown that the increase in playing strength from an accurate inference engine is potentially large. One of our future

aims is to find an inference method for Dou Di Zhu that achieves some of this increase. Conceptually, it is easy to combine an inference model with determinization: instead of sampling determinizations at random, use determinizations that the inference model identifies as being close to the actual state. Our next step is to design an inference model that can identify such determinizations.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments. This work is funded by grant EP/H049061/1 of the UK Engineering and Physical Sciences Research Council (EPSRC).

REFERENCES

[1] C.E. Shannon, "Programming a computer for playing chess," *Philosophical Magazine (Series 7)*, vol. 41, 1950, p. 256–275.

[2] A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, Jul. 1959, pp. 210-229.

[3] 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.

[4] M.L. Ginsberg, "GIB: Imperfect information in a computationally challenging game," *Journal of Artificial Intelligence Research*, vol. 14, 2002, p. 313–368.

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

[6] S. Yoon, A. Fern, and R. Givan, "FF-Replan: A Baseline for Probabilistic Planning," *17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, 2007, p. 352–359.

[7] R.B. Myerson, *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

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

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, 2002, pp. 235-256.

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

[11] G. Chaslot, M. Winands, and H. van Den Herik, "Parallel Monte-Carlo tree search," *Computers and Games*, 2008, p. 60–71.

[12] H. Finnsson, "CADIA PLAYER : A Simulation-Based General Game Player," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, 2009, pp. 4-15.

[13] G. Chaslot, C. Fiter, J.B. Hoock, A. Rimmel, and O. Teytaud, "Adding expert knowledge and exploration in Monte-Carlo Tree Search," *Advances in Computer Games*, 2010, p. 1–13.

[14] S. Yoon, A. Fern, R. Givan, and S. Kambhampati, "Probabilistic Planning via Determinization in Hindsight," *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 1010-1017.

[15] E.K.P. Chong, R.L. Givan, and H. Chang, "A framework for simulation-based network control via hindsight optimization," *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000, pp. 1433-1438.

[16] 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*, Utrecht, The Netherlands, 2007.

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

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

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

[20] J. Long, N. Sturtevant, and M. Buro, "Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search," *AAAI Conference on Artificial Intelligence*, 2009.

[21] D. Koller and A. Pfeffer, "Representations and solutions for game-theoretic problems," *Artificial Intelligence*, vol. 94, 1997, p. 167–215.

[22] M. Zinkevich, M. Johanson, and M. Bowling, "Regret minimization in games with incomplete information," *21st Annual Conference on Neural Information Processing Systems (NIPS 2007)*, 2007.

[23] M. Ponsen, J. Ramon, T. Croonenborghs, K. Driessens, and K. Tuyls, "Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker," *Proceedings of the Twenty-third National Conference on Artificial Intelligence (AAAI-08)*, 2008, pp. 1485-1487.

[24] M. Ponsen, G. Gerritsen, and G. Chaslot, "Integrating Opponent Models with Monte-Carlo Tree Search in Poker," *Proceedings of Interactive Decision Theory and Game Theory Workshop at the Twenty-Fourth Conference on Artificial Intelligence (AAAI-10)*, 2010, pp. 37-42.

[25] R.J.S. Baker and P.I. Cowling, "Bayesian opponent modeling in a simple poker environment," *IEEE Symposium on Computational Intelligence and Games*, 2007, pp. 125-131.

[26] M. Richards and E. Amir, "Opponent modeling in Scrabble," *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007, p. 1482–1487.

[27] M. Buro, J.R. Long, T. Furtak, and N. Sturtevant, "Improving state evaluation, inference, and search in trick-based card games," *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, Morgan Kaufman, 2009, pp. 1407-113.

[28] J. McLeod, "Dou Dizhu," 2010. http://www.pagat.com/climbing/doudizhu.html. Accessed 4th February 2011.