Ensemble Determinization in Monte Carlo Tree Search for the Imperfect Information Card Game Magic: The Gathering

Peter I. Cowling, *Member, IEEE,* Colin D. Ward, *Member, IEEE,* and Edward J. Powley, *Member, IEEE*

***Abstract*—In this paper, we examine the use of Monte Carlo Tree Search (MCTS) for a variant of one of the most popular and profitable games in the world: the card game Magic: The Gathering (M:TG). The game tree for M:TG has a range of distinctive features, which we discuss here, and has incomplete information through the opponent's hidden cards, and randomness through card drawing from a shuffled deck. We investigate a wide range of approaches that use determinization, where all hidden and random information is assumed known to all players, alongside Monte Carlo Tree Search. We consider a number of variations to the rollout strategy using a range of levels of sophistication and expert knowledge, and decaying reward to encourage play urgency. We examine the effect of utilising various pruning strategies in order to increase the information gained from each determinization, alongside methods that increase the relevance of random choices. Additionally we deconstruct the move generation procedure into a binary yes/no decision tree and apply MCTS to this finer grained decision process. We compare our modifications to a basic MCTS approach for Magic: The Gathering using fixed decks, and show that significant improvements in playing strength can be obtained.**

***Index Terms*—Monte Carlo Tree Search, Imperfect Information, Determinization, Parallelization, Card Games, Magic: The Gathering**

I. Introduction

Monte Carlo Tree Search (MCTS) has, in recent years, provided a breakthrough in creating AI agents for games [1]. It has shown remarkable success in Go [2], [3], [4] and is being applied successfully to a wide variety of game environments [5], including Hex [6], Havannah [7], and General Game Playing [8], [9]. One of the major strengths of MCTS is that there is no requirement for a strong evaluation function and it has therefore been especially useful for games where an evaluation function is difficult to formulate, such as Go [10] and Hex [6]. In 2000, Schaeffer [11] said *"it will take many decades of research and development before world-championship-caliber Go programs exist"* and yet we have recently seen MCTS based Go players begin to challenge the best human players in the world [2]. The lack of a requirement for any specific domain knowledge has also helped MCTS to become very successful in the area of General Game Playing where there is little advance knowledge of the structure of the problem and therefore a very restricted scope in which to develop an evaluation function [9].

Removing the need to have a sophisticated evaluation function suggests the possibility of developing search-based AI game agents for much more complex games than was previously possible, and suggests an avenue for a new AI approach in video games. The video games industry is a huge and growing market: in 2009 the video game industry had sales of over $10 billion in the US alone [12] and while the graphics and visual appeal of games has progressed enormously in recent years, to the extent of mapping recognisable faces and emotional content [13], the AI being used is still largely the non-adaptive scripting approach that has always been used [14].

While MCTS has made great strides in producing strong players for perfect information games, the situation for imperfect information games is less advanced and often the use of MCTS is restricted to perfect information versions or parts of the game. For example in Settlers of Catan [15] the authors reduced the game to a perfect information variant and then applied MCTS to this perfect information system beating hand coded AI from an open source version of the game convincingly with only 10000 simulations per turn and a small amount of domain knowledge. Perfect information variants of Spades and Hearts card games have also been used to study convergence properties of UCT in a multiplayer environment [16].

Card games typically have a wealth of hidden information and provide an interesting challenge for AI. Chance actions covering all possible cards that may be drawn from a deck of cards yield a game tree with a chance node at the root which explodes combinatorially, quickly generating branching factors which may dwarf that of Go. We must also deal with the effect of hidden information e.g. the particular cards in an opponent's unseen hand. However, card and board games offer an important class of difficult decision problems for AI research, having features in common with perfect information games and video games, and a complexity somewhere between the two.

MCTS has been applied to several card games with some success. MCTS based players for Poker have started to challenge the best humans in heads-up play [17]. Advances have

Manuscript received August 13, 2011; revised February 20, 2012; revised April 13, 2012.

P.I. Cowling, C.D. Ward and E.J. Powley are currently with the Artificial Intelligence Research Centre, School of Computing, Informatics and Media, University of Bradford, UK. From September 2012 Peter Cowling and Edward Powley will be at the University of York. E-mail: {peter.cowling@york.ac.uk, c.d.ward@student.bradford.ac.uk, e.powley@bradford.ac.uk}.

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC).

DOI:

also been made in multi-player poker and Skat [18] which show promise towards challenging the best human players. Determinization, where all hidden and random information is assumed known by all players, allows recent advances in MCTS to be applied to games with incomplete information and randomness. The determinization approach is not perfect: as discussed by Frank and Basin [19], it does not handle situations where different (indistinguishable) game states suggest different optimal moves, nor situations where the opponent's influence makes certain game states more likely to occur than others. In spite of these problems, determinization has been applied successfully to several games. An MCTS-based AI agent which uses determinization has been developed that plays Klondike Solitaire [20], arguably one of the most popular computer games in the world. For the variant of the game considered, the performance of MCTS in this case exceeds human performance by a substantial margin. A determinized Monte Carlo approach to Bridge [21], which uses Monte Carlo simulations with a tree of depth one has also yielded strong play. The combination of MCTS and determinization is discussed in more detail in Section V.

In this paper we investigate MCTS approaches for the card game Magic: The Gathering (M:TG) [22]. M:TG is a strategic card game for 2 players, which shares characteristics with many other card games: hidden information in the opponent's hand and the stochastic nature of drawing cards from a shuffled deck. Where M:TG differs from other card games is that it does not use a standard deck of cards but rather cards that have been created specifically for the game. Many cards change the rules of the game in subtle ways and the interaction between the rules changes on the cards gives rise to very rich game play.

M:TG is played by over 12 million people worldwide and in 2005 the manufacturer Hasbro reported that it was their biggest selling game, outstripping Monopoly, Trivial Pursuit and Cluedo [23]. The game has a number of professional players: in 2011 the professional tour paid out almost $1 million dollars in prize money to the best players in the world. While specific sales figures are unavailable, it is estimated that more than $100 million is spent annually on the game [24].

M:TG is not only played with physical cards. In 2009 a version of the game appeared on Xbox Live Arcade that allowed players to play against a computer opponent. The details are proprietary but the game appears to use a depth-limited decision tree with static evaluation of the game state at leaf nodes [25]. The AI in the game has generally been regarded as plausible for someone who is a beginner to the game but is at a level that would not challenge an average player [26].

M:TG possesses several characteristics that we believe make it an interesting area for research into game AI:

1) M:TG does not use a standard deck of cards but instead uses cards that are specifically designed for the game. Players are free to construct their own deck using these cards, a decision problem of enormous complexity. There are currently over 9000 different cards that have been created for M:TG and more are added every year. This makes it particularly difficult to predict what cards an opponent may have in their deck and the consequent interactions between cards. It also makes M:TG arguably an exercise in general game playing and a step towards understanding generalizable approaches to intelligent game play.
2) Players are not limited to playing a single card on their turn. All cards have costs and, as the game progresses, the resources and hence options available to a player increase. A player may play any number of cards from their hand on their turn providing they can pay all the associated costs. This means that at each turn, a player can play a subset of cards in hand, giving a high branching factor.
3) The interaction between the players is highly significant, and there is substantial scope for opponent modelling and inference as to the cards the opponent holds in his hand and his deck. Inference is a critical skill in games between human players.
4) The sequence of play is not linear, and the opponent can "interrupt" the player's turn, for example to cancel the effect of playing a particular card. Hence M:TG is less rigid than most turn-based games as each player may have decision points during the opponent's turn.

The structure of this paper is as follows. In Section II we discuss Monte Carlo Tree Search; in Section III we describe the game of Magic: The Gathering and the simplified variant of the game that we have used in our trials; in Section IV we describe the rules-based players we have devised as opponents for our MCTS players; Section V surveys work on the use of parallel determinization approaches to handle uncertainty and incomplete information; our enhancements to MCTS for M:TG which use parallel determinization are presented in Section VI; Section VII presents experimental results and analysis; and Section VIII draws conclusions and provides suggestions for future work.

II. Monte Carlo Tree Search

Monte Carlo Tree Search extends ideas of bandit-based planning [27] to search *trees*. In the k-armed bandit problem, Auer et al [27] showed that it was possible to achieve best-possible logarithmic regret by selecting the arm that maximised the Upper Confidence Bound (UCB):

$$\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

where $\bar{x}_j$ is the average reward from arm j, n_j is the number of times arm j has been played so far, and n is the total number of plays so far.

Around 2006-7, several teams of researchers were investigating the application of Monte Carlo approaches to trees: Chaslot et al [28] developed the idea of Objective Monte Carlo that automatically tuned the ratio between exploration and exploitation based on the results of Monte Carlo simulations at leaf nodes in a minimax tree. Coulom [29] described a method of incrementally growing a tree based on the outcome of simulations at leaf nodes and utilising the reward from the simulated games to bias the tree growth down promising lines of play. Kocsis and Szepesvári used the UCB formula recursively as the tree was searched [30]. The resulting algorithm

is known as UCT (UCB applied to Trees), and Kocsis and Szepesvári showed that with only very limited conditions, it would produce optimal play given a very large number of simulations. The range of algorithms which use Monte Carlo simulations as an approach to heuristically building a search tree have become commonly known as Monte Carlo Tree Search (MCTS).

MCTS algorithms are generally a 4 step process that is repeated until some limit is reached, usually a limit on elapsed time or number of simulations. In this paper we have commonly used a total number of simulations as the limit in our experiments.

The steps of the algorithm, illustrated in Figure 1, are:

1) **Selection.** The algorithm uses the UCB formula (or some other approach) to select a child node of the position currently being considered, repeating this process until a leaf node is reached. Selection balances the exploitation of known good nodes with the exploration of nodes whose value is currently uncertain.
2) **Expansion.** One or more children is added to the leaf node reached in the selection step.
3) **Simulation (or Rollout).** A simulation is carried out from the new leaf node, using a random move generator or other approach at each step, until a terminal game state is reached.
4) **Backpropagation.** The reward received at the simulation step is propagated back to all nodes in the tree that were part of the selection process to update the values (e.g. number of wins/visits) in those nodes.

The algorithm has two principal advantages over conventional tree search methods such as minimax with alpha-beta pruning:

1) It is "anytime" [31]. The algorithm can be stopped at any point to yield a result which makes use of all rollout information so far. There is no need to reach a particular stage during search, before a result is obtainable, as there would be for minimax search, even with iterative deepening [32].
2) An evaluation function is not required for non-terminal game states, as simulation always reaches a terminal state. The reward for a given game state is obtained by aggregating win/lose simulation results from that state.

MCTS may utilise randomly selected moves when conducting simulations and therefore has no need of *any* specific domain knowledge, other than the moves available from each game state and the values of terminal game positions. In practice however the performance of the algorithm can usually be improved by including some domain specific considerations in the simulation and selection phases [3].

III. Magic: The Gathering

A. Game rules

In the game of Magic: The Gathering each player takes on the role of a wizard contesting a duel with their opponent. Each player's hand of cards represents the spells and resources that the wizard has available and the players play cards from their hand in order to either generate resources or play spells with which to beat their opponent.

Each player has a life total and the player whose life total is reduced to zero first loses the game. The game consists of multiple varieties of cards and multiple types of resource, consequently the possible interactions between the available cards can become extremely complex. Much of the appeal of M:TG arises through understanding and tactically exploiting the interactions between the player's cards, and between player's and opponent's cards.

The full game is very complex and difficult to model easily so we have chosen to retain the basic structure and turn order mechanics of the game but to focus on creature combat, which is the most important form of interaction between Magic cards for the majority of decks (and for essentially all decks played by beginning human players). By restricting the test environment to only land (resource) cards and creature (spell) cards we simplify encoding of the rules (which represents a significant software engineering problem in practice [33]). In our test version of the game the players have a deck of cards containing only creatures and land resource cards of a single colour.

Each *creature* card has *power* and *toughness* values denoting how good the creature is at dealing and taking damage, respectively, and a resource (or mana) *cost*. In general, more powerful creatures have a higher resource cost. Below we will refer to a creature with power P, toughness T and cost C as $P/T^{(C)}$, and omit C when it is not significant to our discussion. Each turn a player may put at most one *land* resource card into play from their hand, referred to below as L.

Over the course of the game, each player will accumulate land cards in play. On any given turn the player may expend resources equal to the total amount of land they have in play in order to meet the costs of creature cards from their hand. This allows them to play creature cards from their hand to the *in play zone* which are then available to *attack* and *defend*. These spent resources refresh at the beginning of the player's next turn. In this way, as the player controls increasing amounts of land, they can afford more expensive creatures.

Creatures may be available to defend against the opponent's attack although they are not required to do so. Creatures that have attacked on a defending player's previous turn are considered *tapped* and therefore are not available for defence. Once attacking creatures are declared, the defending player allocates each untapped defending creature (a *blocker*) to at most one attacker. Each attacking creature may have none, one or more blockers assigned to it. Blocking creatures die, and are consequently removed from play, to the *graveyard*, if the attacking creature allocates *damage* to a blocker greater than or equal to its toughness. The attacking creature dies if the corresponding blockers' total power provides damage greater than or equal to the attacking creature's toughness. In the case of an attacking creature having multiple blockers then the player controlling the attacking creature decides how to split each attacker's damage among its blockers. Creatures that are not blocked cause damage to the opponent's *life total* and a player loses the game if their life total is zero or less.

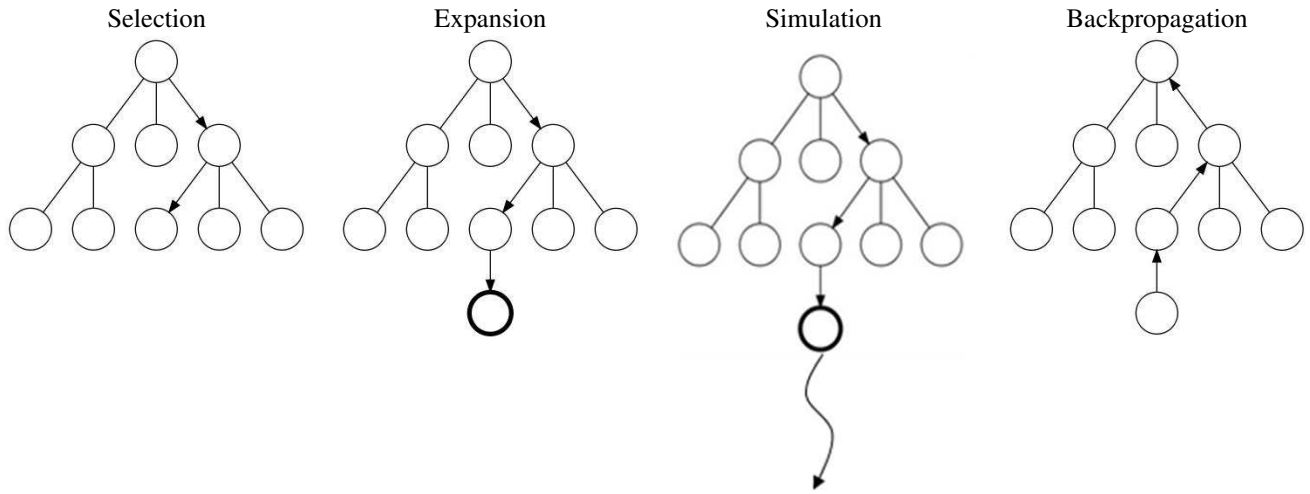

Fig. 1. The four steps of an MCTS algorithm.

See Figure 2.

B. Structure of a game turn

The players take turns. On any given turn, one player is 'active' and the other is 'non-active' and can merely respond to the actions of the active player. The sequence of events during a typical turn is:

1) The active player draws a card from their deck and adds it to their hand. If they are unable to do so (because their deck has no remaining cards) then they immediately lose the game.
2) The active player selects a subset of his creatures in play to be attackers.
3) The non-active player assigns each untapped creature he has in play to block at most one attacker.
4) Combat is resolved and any creatures taking sufficient damage are removed from play. Any unblocked attackers do damage to the non- active players life total. If the non-active player's life total falls to zero, then that player loses the game.
5) The active player may play cards from his hand. One land card may be played each turn and the accumulated land in play can be used to pay for cards to be played from his hand provided the total cost of creatures played is less than the total number of land cards in play.
6) The active and non-active players then switch roles and a new turn begins.

IV. A rule-based approach to Magic: The Gathering

Two rule-based players were created of differing play strength, as well as a purely random player, in order to provide test opponents and rollout strategies for our MCTS players. The first rule-based player had the best heuristics we were able to devise in all areas of the game, and was created using substantial human expertise. The second rule-based player had a significantly reduced set of heuristics and included elements of randomness in its decisions.

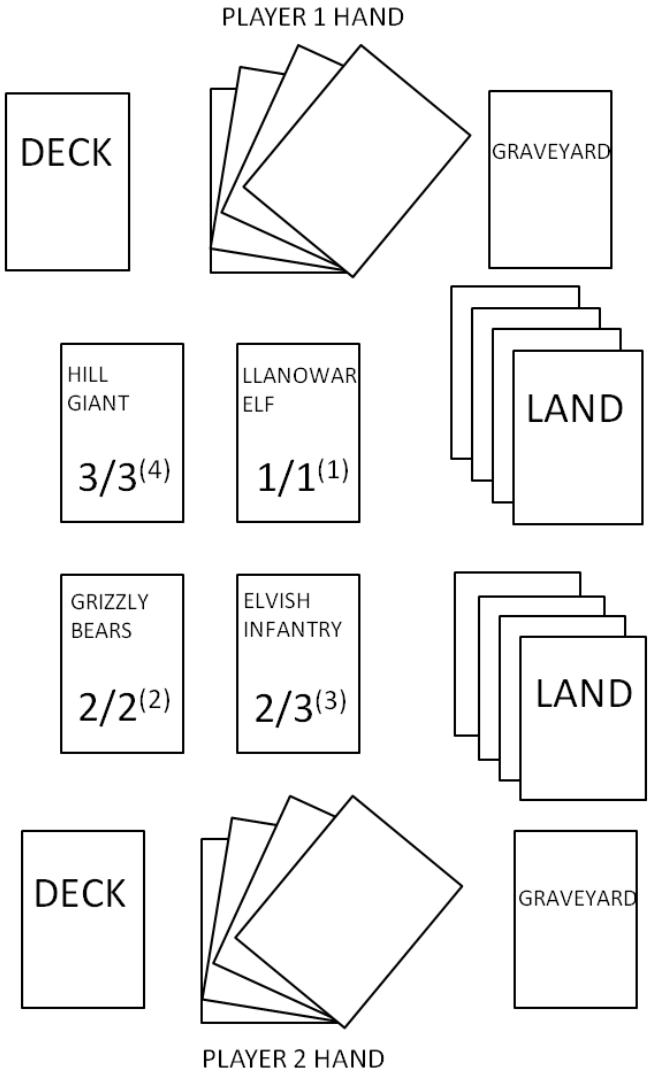

Fig. 2. Representation of the play area during a game of M:TG.

From the structure of the game turn, we can see that there are three main decisions that a player needs to make. The active player must decide which (if any) creatures will attack, and then which cards to play after the attack is resolved. The non-active player must decide how to block the attacking creatures.

Attacking and blocking decisions in M:TG are far from trivial. Multiple creatures may be selected to attack and each of those attacking creatures may be blocked by any subset of the defenders creatures (subject to the constraint that a defending creature can only block one attacking creature). There are also considerations that creatures selected to attack are unavailable for defensive duty on the next turn so the attacking player has to avoid leaving himself open to a lethal counter attack. We were fortunate to be able to call on the experience of a strong M:TG player in order to aid us in formulating some attacking and blocking heuristics.

A. 'Expert' rule-based player

Here we present the detailed description of the heuristics utilised by the expert rule-based player. There are separate heuristics for attacking, blocking and playing cards from the hand.

The CHOOSEATTACKERS function (Algorithm 1) decides which creatures from those available to the player will be selected to attack the opponent this turn. The basic approach taken is to consider each creature that could attack this turn and determine whether there is a reason that it should 'not' attack. If no such reason is found then the creature is declared as an attacker.

Lines 14–18 of CHOOSEATTACKERS (Algorithm 1) define special cases. If there are no potential attackers, then there will be no attackers (line 14). If there are no potential blockers, or the number of blockers is too small to prevent lethal damage (lines 15 and 16 respectively), then we attack with all potential attackers. $a_{\max}$ defines the maximum number of attackers to leave sufficient blockers on the next turn to prevent lethal damage. If $a_{\max}$ is zero then we will have no attackers (line 17), and if $a_{\max}$ is less than zero we will lose next turn anyway, so we attack with all possible creatures to maximise the chance that the opponent might make a mistake in blocking (line 18). In the main case we then go through possible creatures by descending power (breaking ties by descending cost) and choose to attack with a creature if there is no set of blockers that can block and kill it without any blocker being killed (line 24); no blocking combination that kills the attacker and results in only a single blocker of lower mana cost than the attacker being killed (line 25); and the attacker cannot be held back to block and kill an opposing creature of higher mana cost next turn (line 26).

The CHOOSEBLOCKERS function (Algorithm 2) is constructed by considering possible ways to block each attacking creature in descending order of attractiveness to the defending player. Ideally the attacking creature should be killed with no loss to the defender but if this is not possible then lesser outcomes are examined until ultimately, if the defending player must block because otherwise he will lose and no better outcome can be discovered, it will 'chump' block with its weakest creature. This creature will certainly die but it will prevent damage reaching the player.

Lines 14–16 of CHOOSEBLOCKERS (Algorithm 2) define special cases. If there are no attackers or no creatures available to block then no blockers need to be declared (lines 14 and 15). $b_{\min}$ defines the minimum number of attacking creatures that need to be blocked in order for the defending player to survive the attack. If this is higher than the number of potential blockers then game loss is certain and there is no point looking for blocks (line 16). In the main case, we look at each attacking creature in descending order of power (break ties by descending mana cost) and evaluate the best blocking option. These options are evaluated in a descending order of favourability for the defending player so that once an option is found whose conditions are met, we greedily assign that set of blockers and move on to the next attacking creature. Firstly we see if there is any set of blockers that would kill the attacker without any of the blockers dying. If such a set exists, we select the one that has the minimum total mana cost (line 24). Then we see if there is a single creature that would kill the attacker and has a lower mana cost than the attacker (line 26), our blocking creature would die but we would lose a less valuable creature than the attacking player. We then look for a pair of blockers that together can kill the attacker while only losing one of their number with a smaller mana cost than the attacker (for example a $4/4^{(5)}$ attacker blocked by a $2/2^{(2)}$ and a $2/3^{(3)}$) and the pair which leads to the lowest mana cost blocker being killed is chosen (line 28).

So far we have only considered blocks that are advantageous to the defending player, we then look at the neutral case where we block with a creature that will not die to the attacker but will not kill the attacker (line 30). Finally we check whether we need to look at disadvantageous blocks. If $i > k - b_{\min}$ then we must block this attacker or the player will die. First we find the lowest mana cost group that kills the attacker (line 32), or if no such group exists, we assign the lowest cost blocker still available to 'chump' block (line 33) so avoiding lethal damage to the player.

The rules for selecting cards to play are much simpler than the attacking and blocking rules. In CHOOSEMAIN (Algorithm 3). We use a greedy approach that plays land if possible (line 11) and plays out the most expensive affordable creature in the players hand greedily (line 19) until the player cannot afford any more creatures

B. 'Reduced' rule-based player

The reduced rules player utilises much simpler heuristics for its decisions and includes randomness in the decision making process. This approach is significantly weaker than the player given above but gives the possibility of choosing any attacking/blocking move, and any non-dominated move in the main phase. Our intuition suggests that this may be effective in constructing the MCTS tree and in conducting rollouts.

- CHOOSEATTACKERS: For each creature that is able to attack, the player decides with probability p whether or not to attack with that creature. For our tests $p = 0.5$,

Algorithm 1 Attacker choice for the expert rule-based player (Section IV-A).

function CHOOSEATTACKERS(P_A, P_B, l_A, l_B, m_A)
  parameters
    P_A = (potential attackers) = $\left(p_n/{t_n}^{(\alpha_n)}, p_{n-1}/{t_{n-1}}^{(\alpha_{n-1})}, \ldots, p_1/{t_1}^{(\alpha_1)}\right)$
      where $p_n \geq p_{n-1} \geq \cdots \geq p_1$ and $p_i = p_{i-1} \implies \alpha_i \geq \alpha_{i-1}$ $(i = 2, 3, \ldots, n)$
    P_B = (potential blockers) = $\left(q_m/{s_m}^{(\beta_m)}, q_{m-1}/{s_{m-1}}^{(\beta_{m-1})}, \ldots, q_1/{s_1}^{(\beta_1)}\right)$
      where $q_m \geq q_{m-1} \geq \cdots \geq q_1$
    l_A, l_B = life total for attacking and blocking player, respectively
    m_A = maximum number of creatures attacking player has enough land to play from his hand this turn
    $d = |P_A| - |P_B|$
    $a_{\max} = |P_A| + m_a - \min\{i \ : \ q_i + q_{i-1} + \cdots + q_1 \geq l_A\}$

  decision variables
    $A = \{\text{chosen attackers}\} \subseteq P_A$

  // Special cases
  if $P_A = \emptyset$ **then return** $A = \emptyset$
  else if $P_B = \emptyset$ **then return** $A = P_A$
  else if $d > 0$ and $p_d + p_{d-1} + \cdots + p_1 \geq l_B$ **then return** $A = P_A$
  else if $a_{\max} = 0$ **then return** $A = \emptyset$
  else if $a_{\max} < 0$ **then return** $A = P_A$
  end if

  // Main case
  $i \leftarrow n$; $A \leftarrow \emptyset$
  do
    if there is no $M \subseteq P_B$ with $s_j > p_i$ (for all $q_j/{s_j}^{(\beta_j)} \in M$) and $\sum_{k \in M} q_k \geq t_i$
      and there is no pair $\left(M' \subseteq P_B, q_b/{s_b}^{(\beta_b)} \in (P_B \setminus M')\right)$
        with $\beta_b < \alpha_i$ and $s_j > p_i$ (for all $q_j/{s_j}^{(\beta_j)} \in M'$) and $q_b + \sum_{k \in M'} q_k \geq t_i$
      and there is no $q_b/{s_b}^{(\beta_b)} \in P_B$ with $p_i > s_b$ and $\alpha_i < \beta_b$
    then
      $A \leftarrow A \cup \left\{p_i/{t_i}^{(\alpha_i)}\right\}$
    end if
    $i \leftarrow i - 1$
  while $|A| < a_{\max}$ and $i > 0$
  return A
end function

so that the player chooses uniformly at random from possible subsets of attackers.

- CHOOSEBLOCKERS: For each available creature that can block the player decides uniformly at random among all the available attacking creatures plus the decision not to block anything and assigns the blocking creature accordingly.
- CHOOSEMAIN: This player uses the same approach to CHOOSEMAIN as the 'expert' rules player, but with the modification that it uses an ordering of creatures in hand chosen uniformly at random from all orderings. Hence any non-dominated play can be generated. Here non-dominated means that after the main phase cards are played, there remain no more playable cards in the active player's hand.

We ran a direct comparison between our two rules based players in order to gauge their relative strength. We ran an experiment of 1000 randomly generated test games 10 times (playing 10000 games in total) in order to generate confidence interval information. The expert rules player proved to be much stronger, winning 63.7% of games with a 95% confidence interval of $\pm 0.94\%$.

C. Performance against human opponents

We also tested the ability of the Expert Rules player against a number of human opponents. A total of 114 games were played against 7 human players. Six of the human players rated themselves as strong - winning at local events and competitive within the regional/national scene, one player considered himself as a little less strong, rating himself competitive at local events. All the human players played between 10 and 25 games against the expert rules player.

Overall the expert rules player won 48 of the 114 games played for a win rate of 42.1%. The expert rules player performed slightly better when playing first in a game and won 27 out of 58 games for a win rate of 46.6%. The expert

Algorithm 2 Blocker choice for the expert rule-based player (Section IV-A).

1: **function** CHOOSEBLOCKERS(A, P_B, l_B)
2: **parameters**
3: $A = (\text{chosen attackers}) = \left(p_k/{t_k}^{(\alpha_k)}, p_{k-1}/{t_{k-1}}^{(\alpha_{k-1})}, \ldots, p_1/{t_1}^{(\alpha_1)}\right)$
where $p_k \geq p_{k-1} \geq \cdots \geq p_1$ and $p_i = p_{i-1} \implies \alpha_i \geq \alpha_{i-1}$ $(i = 2, 3, \ldots, k)$
4: $P_B = (\text{potential blockers}) = \left(q_m/{s_m}^{(\beta_m)}, q_{m-1}/{s_{m-1}}^{(\beta_{m-1})}, \ldots, q_1/{s_1}^{(\beta_1)}\right)$
where $q_m \geq q_{m-1} \geq \cdots \geq q_1$
5: $l_B = \text{life total for blocking player}$
6: $b_{\min} = \text{minimum number of blockers} = \begin{cases} \min\{i \ : \ p_i + p_{i-1} + \cdots + p_1 \geq l_B\} & \text{if } p_k + p_{k-1} + \cdots + p_1 \geq l_B \\ 0 & \text{otherwise} \end{cases}$
7:
8: **decision variables**
9: $B_{(i)} = \left\{\text{blockers chosen for attacker } p_i/{t_i}^{(\alpha_i)}\right\} \subseteq P_B$
10: $\mathscr{B} = (\text{all blocks}) = \left(B_{(1)}, B_{(2)}, \ldots, B_{(k)}\right)$
11: $B = \{\text{all blocking creatures}\} = \bigcup_i B_{(i)}$ (note $B_{(i)} \cap B_{(j)} = \emptyset$ for $i \neq j$)
12:
13: *// Special cases*
14: **if** $A = \emptyset$ **then return** $\mathscr{B} = ()$
15: **else if** $P_B = \emptyset$ **then return** $\mathscr{B} = (\emptyset, \emptyset, \ldots, \emptyset)$
16: **else if** $b_{\min} > |P_B|$ **then return** $\mathscr{B} = (\emptyset, \emptyset, \ldots, \emptyset)$
17: **end if**
18:
19: *// Main case*
20: $i \leftarrow k$
21: **do**
22: $P = P_B \setminus B$
23: $\mathscr{Q} = \left\{ Q \subseteq P \ : \ s_j > p_i \text{ for all } q/s^{(\beta)} \in Q \text{ and } \sum_{q/s^{(\beta)} \in Q} q \geq t_i \right\}$
24: **if** $\mathscr{Q} \neq \emptyset$ **then** choose $B_{(i)} \in \underset{Q \in \mathscr{Q}}{\arg\min} \sum_{q/s^{(\beta)} \in Q} \beta$; **goto** line 34
25: $\mathscr{Q}' = \left\{ q/s^{(\beta)} \in P \ : \ q \geq t_i \text{ and } \beta < \alpha_i \right\}$
26: **if** $\mathscr{Q}' \neq \emptyset$ **then** choose $B_{(i)} \in \underset{q/s^{(\beta)} \in \mathscr{Q}'}{\arg\min}\ \beta$; **goto** line 34
27: $\mathscr{Q}'' = \{ \left(q_x/{s_x}^{(\beta_x)}, q_y/{s_y}^{(\beta_y)}\right) \in P^2$
$: \ x \neq y, \beta_x \leq \beta_y, q_x + q_y \geq t_i, s_x + s_y > p_i \text{ and } \beta_j \leq \alpha_i \text{ if } s_j \leq p_i \text{ for } j \in \{x, y\} \}$
28: **if** $\mathscr{Q}'' \neq \emptyset$ **then** choose $B_{(i)} \in \underset{(q/s^{(\beta)}, q'/s'^{(\beta')}) \in \mathscr{Q}''}{\arg\min}\ \beta$; **goto** line 34
29: $\mathscr{Q}''' = \left\{ q/s^{(\beta)} \in P \ : \ s > p_i \right\}$
30: **if** $\mathscr{Q}''' \neq \emptyset$ **then** choose $B_{(i)} \in \underset{q/s^{(\beta)} \in \mathscr{Q}'''}{\arg\min}\ \beta$; **goto** line 34
31: $\mathscr{Q}'''' = \left\{ Q \subseteq P \ : \ \sum_{q/s^{(\beta)} \in Q} q \geq t_i \right\}$
32: **if** $i > k - b_{\min}$ and $\mathscr{Q}'''' \neq \emptyset$ **then** choose $B_{(i)} \in \underset{Q \in \mathscr{Q}''''}{\arg\min} \sum_{q/s^{(\beta)} \in Q} \beta$
33: **else if** $i > k - b_{\min}$ **then** choose $B_{(i)} \in \underset{q/s^{(\beta)} \in P}{\arg\min}\ \beta$
34: $P_B \leftarrow P_B \setminus B_{(i)}$
35: $i \leftarrow i - 1$
36: **while** $P_B \neq \emptyset$ and $i > 0$
37: **return** $\mathscr{B} = \left(B_{(1)}, B_{(2)}, \ldots, B_{(k)}\right)$
38: **end function**

Algorithm 3 Card choice for the expert rule-based player (Section IV-A).

function CHOOSEMAIN(L_A, C_A, m)
  parameters
    $L_A = \{$land cards in active player's hand$\}$
      $= \{L, L, \ldots, L\}$
    $C_A =$ (creature cards in active player's hand)
      $= \left(p_n/{t_n}^{(\alpha_n)}, p_{n-1}/{t_{n-1}}^{(\alpha_{n-1})}, \ldots, p_1/{t_1}^{(\alpha_1)}\right)$
      where $\alpha_n \geq \alpha_{n-1} \geq \cdots \geq \alpha_1$
    $m =$ total mana available to active player

  decision variables
    $P_A = \{$cards to play this turn$\} \subseteq L_A \cup C_A$

  // Play land
  if $L_A \neq \emptyset$ **then**
    $P_A \leftarrow P_A \cup \{L\}$
    $m \leftarrow m + 1$
  end if

  // Play creatures
  $i \leftarrow n$
  do
    if $\alpha_i \leq m$ **then**
      $P_A \leftarrow P_A \cup \left\{p_i/{t_i}^{(\alpha_i)}\right\}$
      $m \leftarrow m - \alpha_i$
    end if
    $i \leftarrow i - 1$
  while $m > 0$ and $i > 0$
  return P_A
end function

rules player performed more poorly when acting second, only winning 21 out of 56 games for a win rate of 37.5%. Comments by the human players suggested that they thought the expert rules player made good decisions generally, but was a little too cautious in its play so that they were able to win some games they believed they should have lost because the expert rules player did not act as aggressively as it might have done in some situations where it had an advantage.

V. MCTS TREES WITH DETERMINIZATION

MCTS has been applied to a range of games and puzzles and often provides good performance in cases where tree depth/width and difficulty of determining an evaluation function for nonterminal states make depth-limited minimax search ineffective. Modifications are often used to improve basic MCTS, for example by ignoring move ordering and using Rapid Action Value Estimate (RAVE) values [34] to seed values at previously unexplored nodes which share similarities with already-explored nodes, improved rollout strategies [35] or by using heuristic approaches to limit the number of children for each node [36].

Recent advances in probabilistic planning have presented the idea of *determinization* as a way to solve probabilistic problems [37]. Essentially, each stochastic state transition is determinized (i.e. fixed in advance), and then generates a plan based on the resulting deterministic problem. If the planner arrives at an unexpected state while testing its plan then it replans using the unexpected state as a starting point and a new set of determinised stochastic state transitions. This approach was extended and generalised by the technique of hindsight optimisation [38] which selects among a set of determinised problems by solving determinizations of the future states of a probabilistic problem, resulting after an AI agent's decision state.

MCTS is also making progress in dealing with large Partially Observable Markov Decision Problems (POMDPs). Silver and Veness [39] applied MCTS to POMDPs and developed a new algorithm, Partially Observable Monte-Carlo Planning (POMCP), which allowed them to deal with problems several orders of magnitude larger than was previously possible. They noted that by using MCTS they had a tool which was better able to deal with two issues that affect classic full width planning algorithms such as value iteration [40]. The *curse of dimensionality* [41] arises because in a problem with n states, value iteration reasons about an n-dimensional belief state. MCTS *samples* the state transitions instead of having to consider them all and so is able to deal with larger state spaces. The *curse of history* [41], that the number of histories is exponential in the depth, is also dealt with by sampling the histories, and heuristically choosing promising actions using the UCB formula, allowing for a much larger depth to be considered.

Using determinization as a way of dealing with uncertainty is not new. One of the approaches used in the Bridge program GIB [21] for playing out the trick taking portion of the game was to select a fixed deal, consistent with bidding and play so far, and find the play resulting in the best expected outcome in the resulting perfect information system. GIB utilised partition search [42] to greatly speed up a minimax/alpha-beta search of each determinized deal, allowing 50 simulations per play on 1990s computer hardware, and ultimately yielding play approaching the standard of human experts. It is interesting to note for GIB that using a relatively small number of determinizations is effective if they are carefully chosen.

Frank and Basin [19] provided a critique of the determinization approach, showing that it is prone to two specific problems that limit the effectiveness of the search. *Strategy fusion* is the problem that different actions are indicated when using determinization from states of the imperfect information game (actually information sets) which are indistinguishable to the player. *Non-locality* occurs since the values of nodes in an imperfect information tree are affected by decisions higher up the tree, where opponents are able to steer the game towards certain states and away from other (indistinguishable) states; this does not happen for perfect information games, nor for determinizations. In their work on Klondike Solitaire, Bjarnason et al [20] highlighted this issue, providing an example whereby the search would equally favour two moves, where one required foreknowledge of hidden information and another did not. Russell and Norvig called this kind of over optimism "averaging over clairvoyance" [43], and note that determinization is incapable of considering issues of information gathering and information hiding. Despite this, Perfect

Information Monte Carlo search (PIMC) has generated strong players in a number of game domains including Bridge [21] and Skat [18].

A recent study [44] has investigated why PIMC search gives strong results despite its theoretical limitations. By examining the particular qualities of imperfect information games and creating artificial test environments that highlighted these qualities, Long et al [44] were able to show that the potential effectiveness of a PIMC approach was highly dependent on the presence or absence of certain features in the game. They identified three features, *leaf correlation*, *bias*, and *disambiguation*. Leaf correlation measures the probability that all sibling terminal nodes in a tree having the same payoff value; bias measures the probability that the game will favour one of the players and disambiguation refers to how quickly hidden information is revealed in the course of the game. The study found that PIMC performs poorly in games where the leaf correlation is low, although it is arguable that most sample-based approaches will fail in this case. PIMC also performed poorly when disambiguation was either very high or very low. The effect of bias was small in the examples considered and largely dependent on the leaf correlation value. This correlates well with the observed performance in actual games with PIMC performing well in trick taking games such as Bridge [21] and Skat [18] where information is revealed progressively as each trick is played so that the disambiguation factor has a moderate value. The low likelihood of the outcome of the game hinging on the last trick also means that leaf correlation is fairly high.

In contrast, poker has a disambiguation factor of 0 as the hidden information (the player's hole cards) is not revealed until the end of the hand. This indicates that PIMC would not perform well at the game. Indeed, recent research in poker has been moving in a different direction using the technique of counterfactual regret minimisation (CFR) [45]. This is a method of computing a strategy profile from the game tree of an extensive form game. It has been shown that for an extensive form game it can be used to determine Nash equilibrium strategies. CFR, and its Monte Carlo sampling based variant MCCFR [46], is much more efficient than previous methods of solving extensive game trees such as linear programming [47] and has increased the size of game tree that can be analysed by two orders of magnitude [45], [46]. By collecting poker hands into a manageable number of "buckets" MCCFR can be used to produce strong players for heads up Texas Hold'Em poker [48].

M:TG is a good candidate for investigation by PIMC methods. Leaf correlation in the game is high as it is rare that games are won or lost on the basis of one move at the end of the game, it is more usual for one player to develop the upper hand and apply sufficient continuous pressure on their opponent to win the game. The progressive nature of having an initial hand, unseen by the opponent, and drawing cards from an unknown deck and playing them out into a visible play area also leads to disambiguation factor that grows slowly throughout the course of the game.

Determinization and MCTS have also been considered for probabilistic planning problems with only one "player". Bjarnason et al [20] examined the use of UCT in combination with hindsight optimisation. They compared using UCT as a method for building determinised problem sets for a Hindsight Optimisation planner and showed that it provided state of the art performance in probabilistic planning domains.

Generating multiple MCTS trees simultaneously in parallel for the same position has also been examined, usually for performance and speed reasons [49], [50]. The idea of searching several independent trees for the same position and combining the results is known as *ensemble UCT* [51], or *root parallelization* in an implementation with concurrency [49], [50], and has been shown in some situations to outperform single-tree UCT given the same total number of simulations [50].

VI. MCTS ALGORITHM DESIGN FOR M:TG

In this paper we combine the methods of ensemble UCT and determinization. We build multiple MCTS trees from the same root node and for each tree we determinize chance actions (card draws). Each tree then investigates a possible future from the state space of all possible futures (and the tree of information sets). The determinization of card draws is made as we build the MCTS tree, as late as possible. The first time we reach a state s where we would be required to create chance nodes for a card draw we sample one card draw at random as a specific action a which takes us to the new state s', thereafter whenever we visit state s in the MCTS tree we immediately transition to s' without any further sampling; this "lazy determinization" approach is also taken by the HOP-UCT algorithm of Bjarnason et al [20]. As the MCTS tree grows we effectively fix in place an ordering for the cards in each player's deck.

If our tree considers all possible outcomes for each chance node in M:TG, we may consider this as a single chance node at the top of the tree with enormous branching factor, or we may branch for each potential card drawn at each chance node. There are 60 cards in a typical M:TG deck, and one deck for each player, providing an upper bound of $(60!)^2$ on the number of deals. Since repeat copies of individual cards are allowed (and expected) there will often only be about 15 different cards, and in many games only around 20 cards will be drawn from each deck, but this still yields a combinatorial explosion of possible deals. There are typically 5-6 moves available at a decision node, so this gives a branching factor of approximately 75-90 at 1 ply, around 7000-8000 at 2 ply and approaching a million at 3 ply. The number of simulations that would be required to generate a MCTS tree capable of collecting meaningful statistics about state values, for all possible states, quickly becomes intractable with increasing depth.

A. Relevance of individual MCTS trees

When creating a determinized ordering of cards, as well as being consistent with the game play so far, it seems sensible to try to avoid bias which would make the game an easy win for one of the players. M:TG is particularly prone to this, and indeed this is one of the reasons we believe M:TG provides an interesting case study for MCTS research.

We formulate the idea of an 'interesting' card ordering: one in which the decisions of the player have an impact on the way the game progressed. We define an ordering as 'interesting' if the play 'no move' (effectively passing the turn) gives a different result to playing the move suggested by the expert rule-based player, over a number of game rollouts.

It is not necessarily straight forward to find an 'interesting' ordering for a given game state and, indeed, there may not be any ordering of cards that would qualify for our definition of 'interesting' if the game state is already heavily biased towards one of the players.

We test whether a card ordering is interesting by generating a random order of cards and carrying out two rollouts from the current game state using that ordering: one with the player making no move and one with the player making the move the expert rules player would have chosen. If the outcome of the game is different between these two rollouts then the card ordering is classified as 'interesting'. We test a small number of random rollouts for each candidate card ordering, and if any one of them yields an 'interesting' result then we accept that card order as interesting. These tests do, of course, consume CPU time and there is a limit to how much time can be sensibly spent searching for an interesting ordering. Ultimately, if we consistently fail to find an interesting ordering then we must accept that there might not be one to find, at least not within a reasonable time scale. If an interesting ordering is not found then we use an arbitrarily chosen randomly generated ordering.

An interesting card ordering could be applied to the game at several levels. Preliminary experiments considered using a fraction of the overall simulation budget to (i) find an interesting ordering for the simulations from each leaf node during MCTS; and (ii) find an interesting ordering for the whole deck at the root node only. These were found to give similar, modest improvements in playing strength, but we take option (ii) forward since option (i) significantly slows down the search time, by a factor of up to 2, whereas no slowdown is evident for (ii).

Further preliminary experiments were conducted to investigate the budget of overall simulations used to find interesting deck orderings. For the whole tree at the root node the maximum number of simulations used to find an interesting ordering was varied from 0% to 5%, with good results generally found around the 5% level. This is a maximum and an interesting ordering was usually found in a small fraction of this number of simulations.

A further preliminary investigation looked at whether it was better to use the fixed interesting ordering during simulation rollouts or to revert to the standard random rollouts. These two options were comparable, and random rollouts were chosen in later experiments.

B. Structure of the MCTS tree

We investigate two families of methods for increasing the effectiveness of search in each determinized MCTS tree.

1) Dominated move pruning: In building any search tree, limiting the nodes that are added to the tree in order to reduce the scope of the search has often been seen to provide increases in playing strength [52], [36], [35]. In this respect, MCTS is no different from any other tree searching method. How moves are pruned is generally domain dependent. We examined two levels of pruning for our restricted version of M:TG, based on incorporating limited heuristic knowledge. The first level of pruning was based around the fact that it is necessary to play land cards before any other cards can be played and that there is little strategic benefit to not playing land when you are able to do so. Non-land pruning prunes any move that does not contain a land card when the player has land in their hand, ensuring that only moves that add more land into the game are considered.

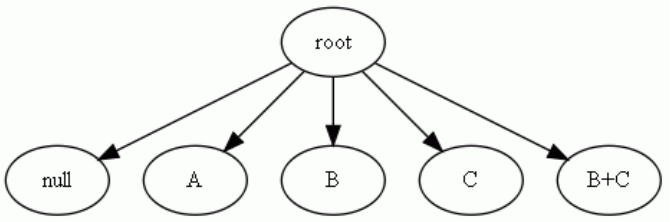

Fig. 3. Potential moves from a position where the player holds cards A, B and C, with mana costs 4, 3 and 2 respectively, and has 5 mana available.

The second, higher, level of pruning makes use of the fact that moves in M:TG are frequently comprised of multiple cards and that the player chooses a subset of the cards in their hand when they decide on a move. This level of pruning, which we called dominated move pruning, removes any move that is a proper subset of another legal move, so that a maximal set of cards is played.

In summary, the following move pruning strategies were investigated:

1) No move pruning. At this level we consider all possible moves available to each player.
2) Non-land pruning. At this level we prune any move that does not contain a land card if the same move with a land card is available.
3) Dominated move pruning. At this level we prune any move that plays a subset of the cards of another available move.

2) Binary decisions: M:TG is unusual among card games in that the moves available on a given turn in the game are a subset of all possible combinations of cards in the player's hand rather than being a single action or a single card. Moreover, the played card group remains active in play rather than being a passive group such as in a melding game such as continental rummy [53].

Consider that a player has 3 non-land cards in hand and 5 land in play. We always suppose here that if land is held, it will be played. Suppose that the cards are A, B and C, with mana costs of 4, 3 and 2 respectively. The player has 5 available moves, as shown in Figure 3.

Here we investigate the case where each node has at most 2 children, representing the decision to play a card or not. This is illustrated in Figure 4. With a fixed number of simulations per tree this will substantially increase the depth of the tree,

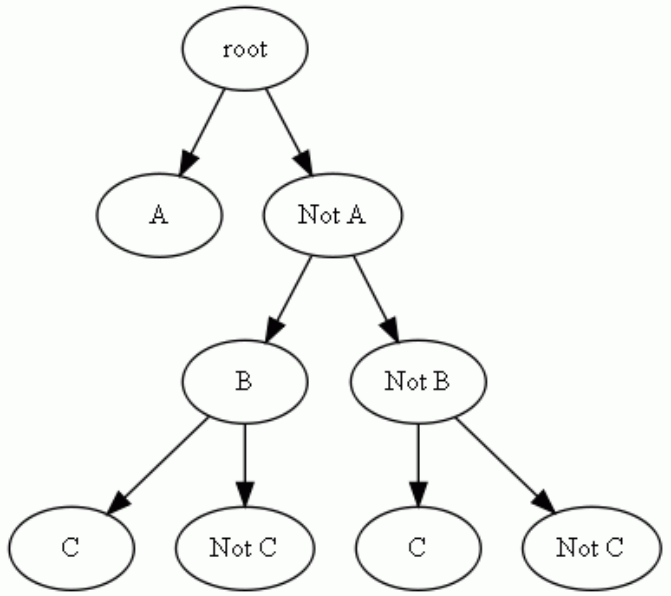

Fig. 4. Binary decision tree of moves corresponding to Figure 3.

compared to a non-binary tree which looks the same distance into the future. However, it should allow statistics for good partial decisions (i.e. on whether or not to play a card) to accumulate independently of other cards played. Hence we are able to investigate MCTS decision making on a tree which allows compound moves to be decomposed so that parts of a move can be reinforced separately. This idea, of decomposing a single decision into a sequence of smaller ones as successive levels in the MCTS tree, is similar to the move grouping approach of Childs et al [54].

We imagine this will also be useful in other applications where MCTS is used to choose a subset, for example we might use this in M:TG to select attackers and/or blockers. In this paper we investigate only the impact upon the decision of which cards to play.

When using this approach it is desirable that "important" decisions are higher in the binary tree, although it is often difficult to determine a priori a sensible importance ordering. Extensive preliminary experiments showed promise for this approach, but did not show any significant difference between using ascending/descending/random orderings based on mana cost. We use descending mana cost in the experiments in section VII-D, based on the intuition that it will often be stronger to play large creatures first.

C. Simulation strategies

While MCTS can use approaches to simulation which select randomly among all possible moves, work on MCTS approaches to computer Go suggested that using heuristics to guide the simulations provided stronger play [31], but also that a stronger playing strength used for rollouts does not necessarily yield higher playing strength when used in an MCTS framework [34], probably due in large part to the bias that this may introduce.

In our simulation rollouts, we investigate rollouts based on both of our rule-based players. The expert player provides a highly structured and completely deterministic rollout and the reduced player provides a stochastic approach with some heuristic guidance. We also (briefly) investigated an approach which chose uniformly at random among possible moves during rollouts.

D. Discounted reward

A player based on MCTS or another forward-looking tree search approach will often make weak moves when in a strong winning (or losing) position. The requirement for the search to be kept under pressure has been observed repeatedly [55]. In order to create a sense of urgency within the player we use an idea from many game tree search implementations (e.g. Kocsis and Szepesvári [30]) and discount the reward value that is propagated back up the tree from the terminal state of a simulation. If the base reward is γ and it takes t turns (counting turns for both players) to reach a terminal state from the current root state then the actual reward propagated back through the tree is $\gamma\lambda^t$ for some discount parameter λ with $0 < \lambda \leq 1$. Here we choose $\lambda = 0.99$ which yields discount factors between 0.7 and 0.5 for a typical Magic game of 40 to 60 turns.

We also compare the effects of assigning a loss a reward of 0 or -1 (a win having a reward of $+1$ in both cases). The value of -1, in combination with discounted rewards, aims to incentivise the player to put off losses for as long as possible. This can be beneficial, as extending the length of the game increases the chance of obtaining a lucky card draw.

VII. Experiments

Our empirical investigation compares MCTS players for M:TG using the approaches explained in Section VI (using parameters from Table I). In Section VII-A we present a simple experiment to show that a naïve implementation of UCT does not yield strong play. In Section VII-B we explore the effect of varying the number of determinizations for a fixed simulation budget, and show that with 10,000 simulations, around 40 determinizations, each with 250 simulations, provides good play (a similar result was found for the card game Dou Di Zhu in [56]). In Section VII-C we compare the relative performance of the approaches in Table I. In Section VII-D we evaluate the effectiveness of combinations of approaches. The baseline conditions reported in Table I are as a result of extensive preliminary experiments (some of which are reported in Section VII-B).

The cards that comprise the deck used by the players are fixed in advance and both players utilise the same deck composition. We created a selection of M:TG style creature and land cards for the decks. The decks contain 40 cards with 17 land cards and 23 creature cards. These proportions are the same as ones generally used by competitive M:TG players in tournaments as they represent the perceived wisdom of providing the best probability to draw a useful mix of land and spells throughout the game. The 23 creatures in the deck were spread among a range of combinations of power, toughness and cost from $1/1^{(1)}$ to $6/6^{(7)}$.

Short name	Description	Trees	Simulations per tree	UCT constant	Win/loss reward	Reward discount	Move pruning	Simulation strategy	Tree structure
AP	All Possible Deals / Uniform Random Rollouts	40	250	1.5	1 / 0	1	None	Uniform Random	All Possible Deals
BA	Baseline	40	250	1.5	1 / 0	0.99	Land	Reduced Rules	Unlimited Degree
IL	Interesting Simulations (Leaf. 1% of sim budget)	40	250	1.5	1 / 0	0.99	Land	**Reduced Rules / Interesting (Leaf)**	Unlimited Degree
IR	Interesting Simulations (Root. 5% of sim budget)	40	250	1.5	1 / 0	0.99	Land	**Reduced Rules / Interesting (Root)**	Unlimited Degree
NL	Negative Reward for Loss	40	250	1.5	**1 / -1**	0.99	Land	Reduced Rules	Unlimited Degree
MP	Dominated Move Pruning	40	250	1.5	1 / 0	0.99	**Dominated**	Reduced Rules	Unlimited Degree
BT	Binary Tree (Descending Mana Cost)	40	250	1.5	1 / 0	0.99	Land	Reduced Rules	**Binary**

TABLE I
SUMMARY OF EXPERIMENTAL PARAMETERS FOR SECTION VII

To provide consistency between experiments, and reduce the variance of our results, in the experiments in Sections VII-A and VII-B, we randomly generated and tested fixed deck orderings until we had 50 orderings that were not particularly biased toward either of the players. In Section VII-C we use 100 unbiased fixed orderings for each pair of players. This type of approach is used in a variety of games to reduce the variance between trials, and notably used in Bridge and Whist tournaments [57] between high-level human players. The experiments were carried out twice with the players alternating between player 1 and player 2 positions, to further reduce bias due to any advantage in going first/second.

Our experiments were carried out on a range of server machines. Broadly speaking we wanted to maintain decision times of around 1 CPU-second or less, since that would be acceptable in play versus a human player. We use number of simulations as the stopping criterion in all cases. CPU times are reported for a server with an Intel Xeon X5460 processor, and 4GB RAM, running Windows Server 2003. Code was written in C# for the Microsoft .NET framework.

A. MCTS for all possible deals

As remarked earlier, the branching factor at a chance node involving a single card draw may be 15 or higher, and since there is a chance node for each decision node in M:TG, this doubles the depth of the tree compared to determinization approaches which fix these chance nodes in advance. While we would not expect MCTS to perform well for a tree which grows so rapidly with depth, it provides an important baseline for our experiments. The approach is illustrated in Figure 5. Note that in this case as well as other experiments (unless stated otherwise), card draws were only specified at the last possible moment (i.e. at the point of drawing a card).

There are multiple methods that can be used in order to select a chance node when descending the tree. Here we select chance outcomes (card draws) uniformly at random. However, since in practice there are repeated cards in the deck, actually we only have one chance outcome per card type, and weight this according to the number of cards or that type in the deck. Another possibility, not considered here, is that one of the players in the game chooses the card to be drawn, with the active player selecting the 'best' chance action and the non-active player selecting the 'worst' chance action. In all of these cases the number of nodes in the tree increases very rapidly with each chance node, which is likely to lead to poor playing strength for MCTS.

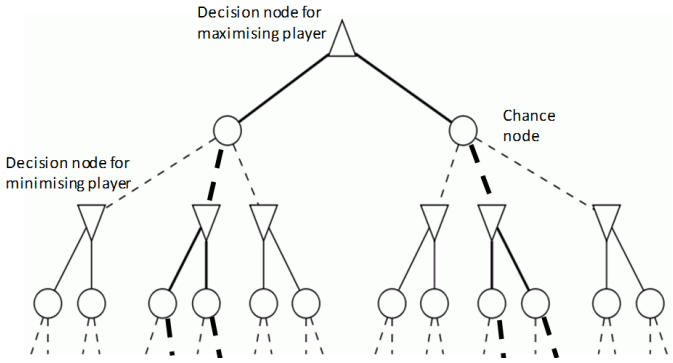

Fig. 5. An MCTS tree with chance nodes.

The All Possible Deals player was played against the expert rules and the reduced rules player, using simulation rollouts that select uniformly at random from among the available legal moves. Over 10 replications of 100 games we see that the All Possible Deals player is significantly weaker than the expert or reduced rules players, winning only 23% of games against the expert player and 38% of games against the reduced rules player. This result provides a baseline to which we can compare our other experimental results in order to determine if our adjustments to the MCTS algorithm are having a beneficial effect.

No. of trees	Expert Rules Simulations vs Reduced Rules	Expert Rules Simulations vs Expert Rules	Reduced Rules Simulations vs Reduced Rules	Reduced Rules Simulations vs Expert Rules
1	27	26	31	32
2	31	29	44	38
4	45	34	48	36
5	46	35	38	40
10	45	33	44	40
20	41	36	46	44
40	48	31	56	47
50	42	34	52	46
100	55	33	56	43
250	50	29	45	47
500	38	21	48	46
1000	15	12	41	26

TABLE II
WIN RATE (%) OF MCTS PLAYER WITH MULTIPLE DETERMINISED TREES AND 10000 SIMULATIONS IN TOTAL

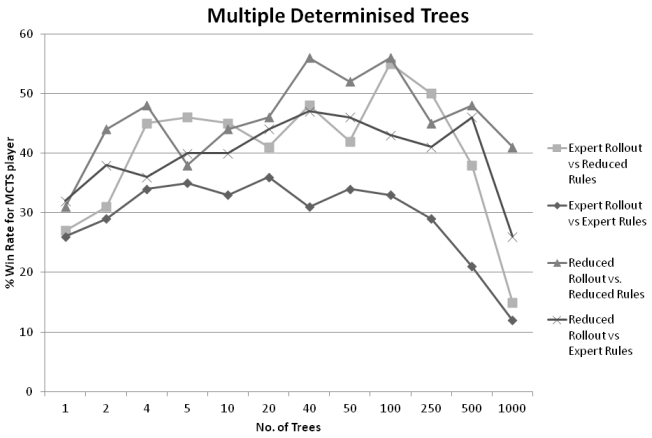

Fig. 6. Comparison of the effect of using multiple determinised trees.

B. Varying the number of determinizations

When using determinization, for a fixed budget on the total number of simulations, we trade off the number of determinization trees versus the number of simulations per tree. If the number of determinizations is too low, we may get a poor result since the small sample of determinizations is not representative of the combinatorially large set of deck orderings. If the number of simulations per tree is too small, then MCTS has not had enough time to exploit promising play lines in the tree for each determinization. We run the tests for a fixed number of total simulations on each tree and then simply add the results from all the trees together and select the move that has the most number of visits over all trees.

In Table II and Figure 6 we vary the number n of determinizations, with each determinization tree having around $10000/n$ simulations. Other experimental conditions are as for the baseline player in Table I.

The first thing we note in Table II is that using an ensemble of determinizations yields much stronger play than the naïve MCTS implementation in Section VII-A. We see also that using reduced rules simulations gives better results that using expert rules simulations, even though the reduced rules player is much weaker than the expert rules player. It seems the reduced rules player provides enough focus to make simulation results meaningful for trees of this size (compared with the results of Section VII-A) while not rigidly defining game outcomes (as for the expert player). Similar results are reported for Go in [34]. In Sections VII-C and VII-D we will consider only these more effective reduced rules simulations.

In each case we see that the best number of determinizations occurs between 20 and 100, and the best number of simulations per determinization between 500 and 100, with a total budget of 10,000 simulations. This, and results from [56] motivate us to choose 40 determinizations with 250 simulations per determinization tree in Sections VII-C and VII-D.

The CPU time used for a single move decision increases slightly as to number of trees increases, from 0.62s for a single tree to 1.12s for 50 trees. Inefficiencies in our code (and particularly the way in which trees are combined) increase the CPU time per move up to 14.01s per move for 1000 trees, although this could be significantly reduced below 1s per move through more careful design.

Similar experiments were conducted with a budget of 100,000 simulations and the number of determinizations n taking values from the set $\{1, 2, 4, 5, 10, 20, 50, 100, 500, 1000, 2000, 5000, 10000\}$, with about $100000/n$ simulations per determinization. The best number of simulations per determinization again lay in the range from 100 to 1000, suggesting that an increased simulation budget is best used in running additional determinizations rather than searching each determinization more deeply. The effects on playing strength of more simulations are analysed in table VI.

C. Comparison of MCTS enhancements

We have outlined a number of different enhancements to MCTS, all with the potential for improving the performance of the search when utilised in a game such as Magic: The Gathering. A round robin tournament was conducted with representative players from each approach (as shown in Table I), to provide a measure of comparative strength of the various enhancements.

In the tournament each player played each other player over 100 games with 50 games being played as each of player 1 and player 2. The same, fixed 50 deck orderings is used for each match, to minimise variance and provide a fair comparison. The results are shown in Table III, with average win rates for each player in Figure 7. Each player used a fixed budget of 10000 simulations; Table IV shows average CPU times per decision, from which we can generally see that BA, IR, NL and MP approaches take approximately the same amount of CPU time. The AP approach is slower, due to the overhead of generating a much wider tree than other approaches. The IL approach, which consumes extra time at every leaf node in the tree searching for an "interesting" determinization, is understandably by far the slowest method. The low branching factor of the BT approach leads to a much lower average time per move.

Player	AP	BA	IL	IR	NL	MP	BT
AP		23	26	21	26	25	18
BA	67		42	40	44	41	44
IL	74	58		46	48	53	48
IR	79	60	54		58	53	43
NL	74	56	52	42		45	44
MP	75	59	47	47	55		52
BT	82	56	52	57	56	48	

TABLE III
WIN RATE (%) OF MCTS PLAYERS IN ROUND ROBIN TOURNAMENT. TABLE SHOWS WIN RATE FOR ROW PLAYER

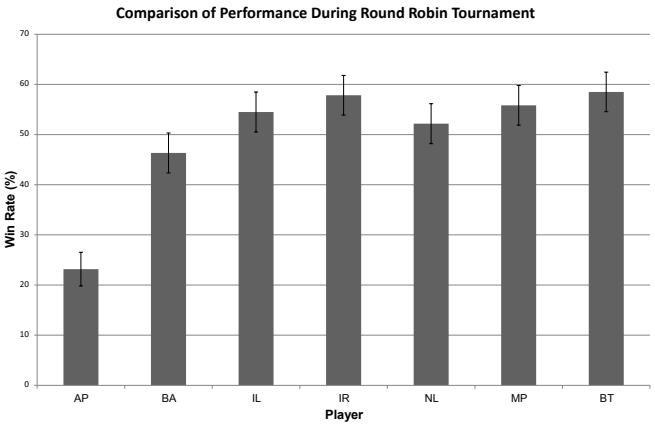

Fig. 7. Average Win Rate (%) of Players in Round Robin Tournament. Error bars show 95% confidence intervals.

Player	Average time per move (seconds)
AP	5.65
BA	0.75
IL	9.81
IR	1.00
NL	1.07
MP	1.00
BT	0.23

TABLE IV
AVERAGE CPU TIME PER MOVE FOR THE MCTS PLAYERS IN SECTION VII-C.

The benefits of ensemble determinization are clear with all other players greatly outperforming the All Possible deals (AP) player which attempts to construct the whole tree without the focussing effect of determinization. All of our enhancements to the basic ensemble determinization approach (IL, IR, NL, MP and BT) improve on the baseline (BA) approach, with the difference significant at the 95% level for all except for Negative reward for Loss (NL). Methods which maintain "pressure" on the Monte Carlo Tree Search, either by finding "interesting" determinizations (IL,IR) or by rewarding delaying tactics when behind (NL) are seen to enhance performance over the baseline player. The use of domain knowledge to prune the tree (MP) is also seen to be effective when compared to the baseline.

The IL, IR, MP and BT approaches have similar playing strength, with BT and IR slightly in front, although not significantly so. These four approaches are quite different in the way that they enhance the baseline algorithm, and the fact that they enhance different aspects of the ensemble determinization approach is further evidenced by their non-transitive performance against each other. For example, the BT approach beats the otherwise unbeaten IR approach, and IR beats MP, but MP is stronger than BT.

The Interesting simulations (Root) (IR) result is slightly better than the Interesting simulations (Leaf) (IL) result, although IR consumes significantly less CPU time than IL for a given number of simulations (1.00s per decision for IR versus 9.81s for IL; see Table IV). Hence we have evidence in support of the effectiveness of finding interesting determinizations, but it does not appear that we need the detail or computational expense of attempting to find an interesting simulation at every leaf of the tree. This observation leads us to use the IR variant in the combination experiments in the next section.

The use of binary trees (BT) is consistently strong against all players, losing only to the dominated move pruning (MP) player. This is particularly notable since the approach is more than three times as fast as any other approach. Figures 8 and 9 illustrate the difference in tree structure for the binary tree enhancement. We believe that the idea of using binary trees, in combination with domain knowledge will likely lead to further enhancements, and begin the exploration of this in the next section. However, due to the difficulty in finding appropriate domain knowledge, this is a large piece of work in itself, and we anticipate future work in this area.

D. Combinations of MCTS enhancements

We have shown in the previous section that our enhancements to the basic MCTS algorithm individually produce a stronger player than the baseline MCTS approach to using ensemble determinization. In this section we investigate the effectiveness of combinations of some of the best performing enhancements. We took four enhancements that had performed strongly in individual experiments and tested all possible

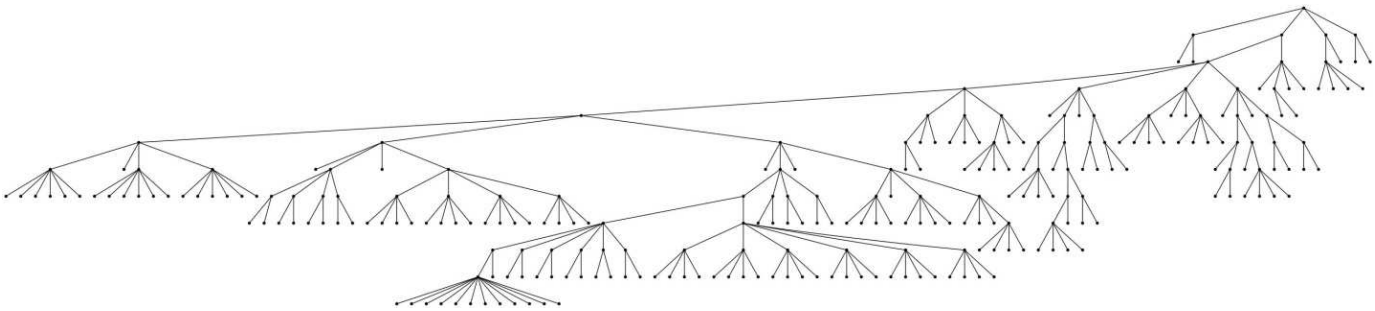

Fig. 9. A non-binary MCTS tree.

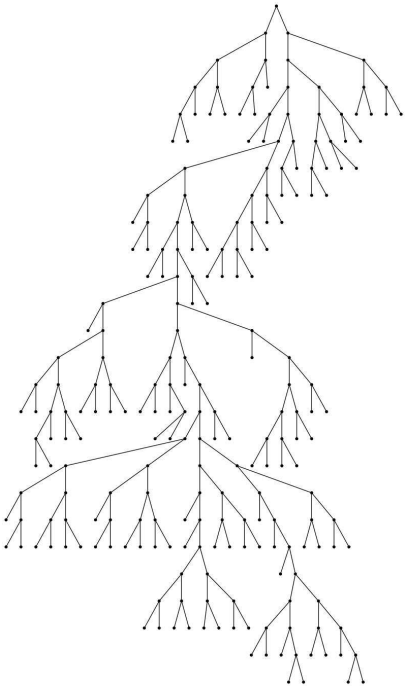

Fig. 8. A binary MCTS tree.

combinations of them. The enhancements tested were (in each case the expected stronger level is listed first):

- Binary Trees (BT/$-$): used (ordered by descending mana cost) / not used
- Move Pruning (MP/$-$): dominated move pruning / non-land pruning
- Negative Reward for Loss (NL/$-$) : -1 reward for loss / 0 reward for loss
- Interesting Simulations (Root) (IR/$-$): used (at most 5% of simulation budget used to find an interesting ordering) / not used

In the results of this section we denote each player as a 4-tuple, to denote the level of each enhancement. For example the player $(BT, MP, -, -)$ utilises Binary Trees and Dominated Move Pruning, but not the Negative Reward for Loss or Interesting Simulations (Root). These experiments are very CPU intensive, and 100 replications were conducted using a large CPU cluster.

Player	Win % vs Expert Rules Player	Average time per move (seconds)
(BT, MP, NL, IR)	49.5	0.21
$(BT, MP, NL, -)$	50.5	0.17
$(BT, MP, -, IR)$	50.4	0.27
$(BT, MP, -, -)$	50.5	0.20
$(BT, -, NL, IR)$	47.0	0.31
$(BT, -, NL, -)$	47.7	0.23
$(BT, -, -, IR)$	47.6	0.28
$(BT, -, -, -)$	47.6	0.19
$(-, MP, NL, IR)$	47.4	1.01
$(-, MP, NL, -)$	44.5	1.05
$(-, MP, -, IR)$	47.7	1.00
$(-, MP, -, -)$	46.1	1.05
$(-, -, NL, IR)$	48.3	1.00
$(-, -, NL, -)$	43.6	0.92
$(-, -, -, IR)$	47.7	1.01
$(-, -, -, -)$	43.9	0.79

TABLE V
COMBINATION EXPERIMENTS — AVERAGE WIN RATE (%) OVER 100 TRIALS

We present average performance versus the expert player in Table V. In addition to the results given in the table, we observed that reduced rules rollouts significantly outperform expert rules rollouts (by around 10% in most cases), and that all the players which use at least one enhancement significantly outperform the reduced rules player.

The results were analysed using Multiway Analysis of Variance (ANOVA) [58] using the R statistical package [59]. Multiway ANOVA showed that enhancements (BT, MP and IR) yielded performance improvements which were significant at the 99% level (i.e. that $(BT, *, *, *)$ significantly outperforms $(-, *, *, *)$ etc.). NL represented a significant improvement only at the 90% level. The following pairs of enhancements were also significant at the 99% level: BT:MP, BT:IR, and MP:IR. Only one triple of enhancements yielded significantly better performance at the 99% level: BT:MP:IR.

ANOVA analysis and the results in Table V show that our proposed enhancements do indeed improve performance of ensemble determined MCTS, in combination as well as individually. The $(BT, MP, *, *)$ players provide the strongest performance, yielding playing strength slightly better than the expert player. Achievement of a higher than 50% win rate is a substantive achievement when we consider the strength of the expert rules player against expert human opponents, and the fact that the $(BT, MP, *, *)$ players achieve this performance

Player	Win % vs Expert Rules Player	Average time per move (seconds)
(BT, MP, IR)	51.2	3.21
$(BT, MP, -)$	51.5	3.21
$(BT, -, IR)$	51.6	3.63
$(BT, -, -)$	52.0	3.36
$(-, MP, IR)$	44.3	8.38
$(-, MP, -)$	44.7	7.91
$(-, -, IR)$	47.2	6.83
$(-, -, -)$	37.3	6.45

TABLE VI
100K ROLLOUTS COMBINATION EXPERIMENTS — AVERAGE WIN RATE (%) OVER 40 TRIALS

without using the knowledge encoded in these expert rules.

The BT enhancement significantly decreases the CPU time per decision, probably as a result of the MCTS selection phase having far fewer branches to choose between at each level in the tree. MP yields a slight improvement in CPU time when coupled with BT. The other enhancements slightly increase the CPU time per decision, but not significantly so.

The results of our analysis underline the utility of all the proposed methods, the dominance of the BT:MP combination, and the complexity of the interaction between methods in yielding increased playing strength.

We carried out additional experiments in order to investigate whether increasing the number of rollouts to 100,000 would provide any significant increase in the performance of the most promising combinations. In this case we did not consider the Negative reward for Loss (NL) enhancement (using a reward for loss of zero) due to the CPU-intensive nature of these experiments and the fact that the previous results suggest that it was the least effective of the four enhancements. The results of this are shown in Table VI. Note that these experiments are very time-consuming, requiring roughly five to ten times as much CPU time per trial as those in table V.

We see here modest improvements in overall performance, when using the BT enhancement with or without other enhancements. Counterintuitively, without this enhancement performance is no better and indeed slightly worse than when using a smaller simulation budget. We have observed this phenomenon for other games of partial information [56], [60] which probably arises due to the large branching factor as we descend the tree even when determinization is used, so that the additional simulation budget is used in chasing somewhat arbitrary decision possibilities. That BT mitigates this problem suggests that this is a particularly interesting area for further study, capable of focussing search into interesting areas of the tree. BT likely improves matters here since the reduction of the degree of the tree results in a more focussed search in each determinization.

VIII. CONCLUSION

In this paper we have introduced the popular card game Magic: The Gathering. We believe M:TG is an interesting domain for Computational Intelligence and AI, and particularly Monte Carlo Tree Search, for a variety of reasons. The game is highly popular and commercially successful, and has (human) players at professional levels. It is an imperfect information game, with unique cards that provide a rich level of tactical play and provide a very high branching factor for any search based approach. Expert heuristics are difficult to formulate because of the variety and complexity of the game situations that arise and the fact that the effectiveness of many actions are highly dependent on the current game state. All of these factors suggest that M:TG would be an extremely difficult challenge for conventional evaluation based search methods.

We also feel that the structure of the game is suited to analysis by MCTS. The progressive revealing of information as players draw new cards from their decks and play them out combined with the relative unlikelihood of similar game states leading to radically different game outcomes are both features that suggest that MCTS should be able to generate strong play.

The central theme of this paper is the use of multiple determinized trees as a means of dealing with imperfect information in a MCTS search and we have shown that this approach provides significant benefits in playing strength, becoming competitive with a sophisticated expert rules player with a simulation budget of less than one CPU second on standard hardware, despite having no access to expert knowledge. In addition to that we have presented a wide variety of enhancements to the determinized trees and analysed the effect on playing strength that each enhancement offers. All of these enhancements show further improvement. We investigated a modification of the structure of the decision tree to a binary tree, well suited to M:TG where decisions amount to the choice of a subset of cards from a small set, rather than an individual card. As well as providing significant improvements in playing strength, the binary tree representation substantially reduced CPU time per move. Dominated move pruning used limited domain knowledge, of a type applicable to a wide variety of games involving subset choice, to significantly reduce the branching factor within the tree. Another promising approach maintained pressure on the Monte Carlo Tree Search algorithm by choosing "interesting" determinizations which were balanced between the two players. An enhancement which used decaying reward to encourage delaying moves when behind had some positive effect, but was not as effective as the preceding three enhancements.

The rollout strategy had a profound effect in our experiments. Applying a fully deterministic rollout strategy, as we did when using our expert rules player to handle the rollouts, provided a clearly inferior performance to utilising the reduced rules player which uses very limited domain knowledge, but incorporates some randomness within its decisions. This was true in all of our experiments and despite the fact that the expert rules player is an intrinsically stronger player than the reduced rules player. However, using a naïve rollout strategy which chose uniformly at random from all possible moves proved to be very weak.

MCTS, suitably enhanced by the range of approaches we have suggested in this paper, was able to compete with, and outperform, a strong expert rule-based player (which is in turn competitive with strong human players). Hence the paper adds to the volume of work which suggests MCTS as a powerful

algorithm for game AI, for a game of a somewhat different nature to those previously studied.

In future work we will look at increasing the complexity of the game environment by including a wider variety of M:TG cards and card types. This will increase the scope of the tactical decisions available to the player and will make it significantly harder to encode strong knowledge-based players. We also intend to look more closely at binary trees in conjunction with domain knowledge, which we believe may yield significant further improvements in playing strength.

Card and board games such as Magic: The Gathering provide excellent test beds for new artificial intelligence and computational intelligence techniques, having intermediate complexity between perfect information games such as Chess and Go, and video games. As such we believe they represent an important stepping stone towards better AI in commercial video games.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] H. J. van den Herik, "The Drosophila Revisited," *Int. Comp. Games Assoc. J.*, vol. 33, no. 2, pp. 65–66., 2010.

[2] C.-S. Lee, M.-H. Wang, G. M. J.-B. 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 Trans. Comp. Intell. AI Games*, vol. 1, no. 1, pp. 73–89, 2009.

[3] A. Rimmel, O. Teytaud, C.-S. Lee, S.-J. Yen, M.-H. Wang, and S.-R. Tsai, "Current Frontiers in Computer Go," *IEEE Trans. Comp. Intell. AI Games*, vol. 2, no. 4, pp. 229–238, 2010.

[4] C.-S. Lee, M. Müller, and O. Teytaud, "Guest Editorial: Special Issue on Monte Carlo Techniques and Computer Go," *IEEE Trans. Comp. Intell. AI Games*, vol. 2, no. 4, pp. 225–228, Dec. 2010.

[5] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Trans. Comp. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, 2012.

[6] B. Arneson, R. B. Hayward, and P. Henderson, "Monte Carlo Tree Search in Hex," *IEEE Trans. Comp. Intell. AI Games*, vol. 2, no. 4, pp. 251–258, 2010.

[7] F. Teytaud and O. Teytaud, "Creating an Upper-Confidence-Tree program for Havannah," in *Proc. Adv. Comput. Games, LNCS 6048*, Pamplona, Spain, 2010, pp. 65–74.

[8] J. Méhat and T. Cazenave, "Combining UCT and Nested Monte Carlo Search for Single-Player General Game Playing," *IEEE Trans. Comp. Intell. AI Games*, vol. 2, no. 4, pp. 271–277, 2010.

[9] Y. Björnsson and H. Finnsson, "CadiaPlayer: A Simulation-Based General Game Player," *IEEE Trans. Comp. Intell. AI Games*, vol. 1, no. 1, pp. 4–15, 2009.

[10] M. Enzenberger, M. Müller, B. Arneson, and R. B. Segal, "Fuego - An Open-Source Framework for Board Games and Go Engine Based on Monte Carlo Tree Search," *IEEE Trans. Comp. Intell. AI Games*, vol. 2, no. 4, pp. 259–270, 2010.

[11] J. Schaeffer, "The games computers (and people) play," *Adv. Comput.*, vol. 52, pp. 189–266, 2000.

[12] S. E. Siwek, "Video games in the 21st century: the 2010 report," 2010. [Online]. Available: http://www.theesa.com/facts/pdfs/VideoGames21stCentury_2010.pdf

[13] N. Ersotelos and F. Dong, "Building highly realistic facial modeling and animation: a survey," *Visual Comput.*, vol. 24, no. 1, pp. 13–30, 2008.

[14] P. Tozour, "The perils of AI scripting," in *AI Game Programming Wisdom*, S. Rabin, Ed. Charles River Media, 2002, pp. 541–547.

[15] I. Szita, G. M. J.-B. Chaslot, and P. Spronck, "Monte-Carlo Tree Search in Settlers of Catan," in *Proc. Adv. Comput. Games*, Pamplona, Spain, 2010, pp. 21–32.

[16] N. R. Sturtevant, "An Analysis of UCT in Multi-Player Games," in *Proc. Comput. and Games, LNCS 5131*, Beijing, China, 2008, pp. 37–49.

[17] G. van den Broeck, K. Driessens, and J. Ramon, "Monte-Carlo Tree Search in Poker using Expected Reward Distributions," *Adv. Mach. Learn., LNCS 5828*, no. 1, pp. 367–381, 2009.

[18] M. Buro, J. R. Long, T. Furtak, and N. R. Sturtevant, "Improving State Evaluation, Inference, and Search in Trick-Based Card Games," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, Pasadena, California, 2009, pp. 1407–1413.

[19] I. Frank and D. Basin, "Search in games with incomplete information: a case study using Bridge card play," *Artif. Intell.*, vol. 100, no. 1-2, pp. 87–123, 1998.

[20] R. Bjarnason, A. Fern, and P. Tadepalli, "Lower Bounding Klondike Solitaire with Monte-Carlo Planning," in *Proc. 19th Int. Conf. Automat. Plan. Sched.*, Thessaloniki, Greece, 2009, pp. 26–33.

[21] M. L. Ginsberg, "GIB: Imperfect Information in a Computationally Challenging Game," *J. Artif. Intell. Res.*, vol. 14, pp. 303–358, 2001.

[22] Wizards of the Coast, "Magic: The Gathering." [Online]. Available: http://www.magicthegathering.com

[23] H. Rifkind, "Magic: game that made Monopoly disappear," Jul. 2005. [Online]. Available: http://www.timesonline.co.uk/tol/life_and_style/article545389.ece?token=null&offset=0&page=1

[24] G. Giles, "House of Cards," 1995. [Online]. Available: http://www.metroactive.com/papers/sonoma/11.09.95/magic.html

[25] P. Buckland, "Duels of the Planeswalkers: All about AI," 2009. [Online]. Available: http://www.wizards.com/Magic/Magazine/Article.aspx?x=mtg/daily/feature/44

[26] Z. Mowshowitz, "Review and analysis: Duels of the Planeswalkers," 2009. [Online]. Available: http://www.top8magic.com/2009/06/review-and-analysis-duels-of-the-planeswalkers-by-zvi-mowshowitz/

[27] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[28] G. M. J.-B. Chaslot, J.-T. Saito, B. Bouzy, J. W. H. M. Uiterwijk, and H. J. van den Herik, "Monte-Carlo Strategies for Computer Go," in *Proc. BeNeLux Conf. Artif. Intell.*, Namur, Belgium, 2006, pp. 83–91.

[29] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," in *Proc. 5th Int. Conf. Comput. and Games*, Turin, Italy, 2006, pp. 72–83.

[30] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo Planning," in *Euro. Conf. Mach. Learn.*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Germany: Springer, 2006, pp. 282–293.

[31] Y. Wang and S. Gelly, "Modifications of UCT and sequence-like simulations for Monte-Carlo Go," in *Proc. IEEE Symp. Comput. Intell. Games*, Honolulu, Hawaii, 2007, pp. 175–182.

[32] R. E. Korf, "Depth-first iterative-deepening: an optimal admissible tree search," *Artif. Intell.*, vol. 27, no. 1, pp. 97–109, 1985.

[33] J. Ferraiolo, "The MODO fiasco: corporate hubris and Magic Online," 2004. [Online]. Available: http://www.starcitygames.com/php/news/article/6985.html

[34] S. Gelly and D. Silver, "Combining Online and Offline Knowledge in UCT," in *Proc. 24th Annu. Int. Conf. Mach. Learn.* Corvalis, Oregon: ACM, 2007, pp. 273–280.

[35] G. M. J.-B. Chaslot, C. Fiter, J.-B. Hoock, A. Rimmel, and O. Teytaud, "Adding Expert Knowledge and Exploration in Monte-Carlo Tree Search," in *Proc. Adv. Comput. Games, LNCS 6048*, vol. 6048, Pamplona, Spain, 2010, pp. 1–13.

[36] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. van den Herik, J. W. H. M. Uiterwijk, and B. Bouzy, "Progressive Strategies for Monte-Carlo Tree Search," *New Math. Nat. Comput.*, vol. 4, no. 3, pp. 343–357, 2008.

[37] S. Yoon, A. Fern, and R. L. Givan, "FF-Replan: A Baseline for Probabilistic Planning," in *Proc. 17th Int. Conf. Automat. Plan. Sched.*, Providence, New York, 2007, pp. 352–359.

[38] S. Yoon, A. Fern, R. L. Givan, and S. Kambhampati, "Probabilistic Planning via Determinization in Hindsight," in *Proc. Assoc. Adv. Artif. Intell.*, Chicago, Illinois, 2008, pp. 1010–1017.

[39] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Proc. Neur. Inform. Process. Sys.*, Vancouver, Canada, 2010, pp. 1–9.

[40] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1-2, pp. 99–134, 1998.

[41] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *J. Artif. Intell. Res.*, vol. 27, pp. 335–380, 2006.

[42] M. L. Ginsberg, "Partition search," in *Proc. 13th Nat. Conf. Artif. Intell. & 8th Innov. Applicat. Artif. Intell. Conf.*, 1996, pp. 228–233.

[43] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, New Jersey: Prentice Hall, 2009.
[44] J. R. Long, N. R. Sturtevant, M. Buro, and T. Furtak, "Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search," in *Proc. Assoc. Adv. Artif. Intell.*, Atlanta, Georgia, 2010, pp. 134–140.
[45] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret Minimization in Games with Incomplete Information," in *Proc. Adv. Neur. Inform. Process. Sys.*, Vancouver, Canada, 2008, pp. 1729–1736.
[46] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling, "Monte Carlo Sampling for Regret Minimization in Extensive Games," in *Proc. Adv. Neur. Inform. Process. Sys.*, Vancouver, Canada, 2009, pp. 1078–1086.
[47] D. Koller and N. Megiddo, "The complexity of two-person zero-sum games in extensive form," *Games Econ. Behav.*, vol. 4, pp. 528–552, 1992.
[48] M. Bowling, N. A. Risk, N. Bard, D. Billings, N. Burch, J. Davidson, J. Hawkin, R. Holte, M. Johanson, M. Kan, B. Paradis, J. Schaeffer, D. Schnizlein, D. Szafron, K. Waugh, and M. Zinkevich, "A Demonstration of the Polaris Poker System," in *Proc. Int. Conf. Auton. Agents Multi. Sys.*, 2009, pp. 1391–1392.
[49] T. Cazenave and N. Jouandeau, "On the Parallelization of UCT," in *Proc. Comput. Games Workshop*, Amsterdam, Netherlands, 2007, pp. 93–101.
[50] G. M. J.-B. Chaslot, M. H. M. Winands, and H. J. van den Herik, "Parallel Monte-Carlo Tree Search," in *Proc. Comput. and Games, LNCS 5131*, Beijing, China, 2008, pp. 60–71.
[51] A. Fern and P. Lewis, "Ensemble Monte-Carlo Planning: An Empirical Study," in *Proc. 21st Int. Conf. Automat. Plan. Sched.*, Freiburg, Germany, 2011, pp. 58–65.
[52] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artif. Intell.*, vol. 6, no. 4, pp. 293–326, 1975.
[53] Pagat, "Rummy." [Online]. Available: http://www.pagat.com/rummy/rummy.html
[54] B. E. Childs, J. H. Brodeur, and L. Kocsis, "Transpositions and Move Groups in Monte Carlo Tree Search," in *Proc. IEEE Symp. Comput. Intell. Games*, Perth, Australia, 2008, pp. 389–395.
[55] I. Althöfer, "On the Laziness of Monte-Carlo Game Tree Search in Non-tight Situations," Friedrich-Schiller Univ., Jena, Tech. Rep., 2008.
[56] E. J. Powley, D. Whitehouse, and P. I. Cowling, "Determinization in Monte-Carlo Tree Search for the card game Dou Di Zhu," in *Proc. Artif. Intell. Simul. Behav.*, York, United Kingdom, 2011.
[57] World Bridge Federation, "General conditions of contest," 2011. [Online]. Available: http://www.worldbridge.org/departments/rules/GeneralConditionsOfContest2011.pdf
[58] M. H. Kutner, J. Neter, C. J. Nachtsheim, and W. Wasserman, *Applied Linear Statistical Models*, 5th ed. McGraw-Hill, 2004.
[59] R Development Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2008.
[60] P. I. Cowling, E. J. Powley, and D. Whitehouse, "Information Set Monte Carlo Tree Search," *IEEE Trans. Comp. Intell. AI Games*, vol. 4, no. 2, 2012.

Peter Cowling is Professor of Computer Science and Associate Dean (Research and Knowledge Transfer) at the University of Bradford (UK), where he leads the Artificial Intelligence Research Centre. From September 2012 he will take up an Anniversary Chair at the University of York (UK) joint between the Department of Computer Science and the York Management School. He holds MA and DPhil degrees from Corpus Christi College, University of Oxford (UK). His work centres on computerised decision-making in games, scheduling and resource-constrained optimisation, where real-world situations can be modelled as constrained search problems in large directed graphs. He has a particular interest in general-purpose approaches such as hyperheuristics (where he is a pioneer) and Monte Carlo Tree Search (especially the application to games with stochastic outcomes and incomplete information). He has worked with a wide range of industrial partners, developing commercially successful systems for steel scheduling, mobile workforce planning and staff timetabling. He is a director of 2 research spin-out companies. He has published over 80 scientific papers in high-quality journals and conferences, won a range of academic prizes and "best paper" awards, and given invited talks at a wide range of Universities and conference meetings. He is a founding Associate Editor of the IEEE Transactions on Computational Intelligence and AI for Games.

Colin Ward is currently working towards a PhD at the University of Bradford (UK). He holds a BSc in Computing and Information Systems from the University of Bradford. With his background in computer science and as a competitive game player (he has been ranked among the top 150 Magic: The Gathering players in the UK) his research interests are focussed on artificial intelligence and machine learning approaches to playing games. His thesis examines game domains with incomplete information and search methods that can be applied to those domains.

Edward Powley received an MMath degree in Mathematics and Computer Science from the University of York, UK, in 2006, and was awarded the P B Kennedy Prize and the BAE Systems ATC Prize. He received a PhD in Computer Science from the University of York in 2010. He is currently a Research Fellow at the University of Bradford, where he is a member of the Artificial Intelligence Research Centre in the School of Computing, Informatics and Media. He will move to the University of York in September 2012. His current work involves investigating MCTS for games with hidden information and stochastic outcomes. His other research interests include cellular automata, and game theory for security.