Monte Carlo Search Applied to Card Selection in Magic: The Gathering

C. D. Ward *Member, IEEE*, P. I. Cowling *Member, IEEE*

***Abstract*—We present the card game Magic: The Gathering as an interesting test bed for AI research. We believe that the complexity of the game offers new challenges in areas such as search in imperfect information domains and opponent modelling. Since there are a thousands of possible cards, and many cards change the rules to some extent, to successfully build AI for Magic: The Gathering ultimately requires a rather general form of game intelligence (although we only consider a small subset of these cards in this paper). We create a range of players based on stochastic, rule-based and Monte Carlo approaches and investigate Monte Carlo search with and without the use of a sophisticated rule-based approach to generate game rollouts. We also examine the effect of increasing numbers of Monte Carlo simulations on playing strength and investigate whether Monte Carlo simulations can enable an otherwise weak player to overcome a stronger rule-based player. Overall, we show that Monte Carlo search is a promising avenue for generating a strong AI player for Magic: The Gathering.**

I. INTRODUCTION

Magic: The Gathering (M:TG) [1] is a strategic card game for 2 players. In 1994 it was awarded the Mensa Select award at the annual Mind Games Competition [2] and in 2005 the manufacturer Hasbro reported that M:TG was its biggest selling game, outstripping Monopoly, Trivial Pursuit and Cluedo [3]. Currently M:TG has over 215,000 registered tournament players worldwide and many more casual players. The game supports a pro tour which pays out hundreds of thousands of dollars in prize money to the best players in the world [4]. Specific sales figures are unavailable but it is estimated that more that $100 million is spent annually on the game [5].

Like some of the older, more well known card games such as Poker and Bridge, M:TG is both stochastic and a game of imperfect information. Where it differs from other card games is that it does not use a standard deck of cards but rather uses cards that have been created specifically for the game. The cards change the rules of the game in subtle ways and the interaction between the rules changes on the cards gives rise to much of the game play. In *constructed* M:TG, which we consider here, players are allowed to build a deck of 60 cards from any of the over 9000 cards that have so far been created with more being added every year. This breadth of possible interactions makes creating a strong AI player for M:TG somewhat akin to creating a player for general game playing [6] due to the great differences between games played with different decks (and hence different rules).

M:TG is not only played with physical cards, In 2002 an online version of the game was released [7]. The online game does not provide any AI game players, but provides a framework that enforces the rules of the game when (human) players are playing. The flexibility this provides in finding opponents to play has further increased the popularity of the game.

M:TG is a superficially simple game to play. The complexity of the game arises from the many different types of potential interactions between the individual cards. We believe that the complexity and popularity of M:TG make it an interesting subject for AI research in the areas of search in imperfect information domains, opponent modelling and general game playing.

In this paper we will be looking at the use of Monte Carlo search for selecting which cards to play. We will compare the use of Monte Carlo simulations with a strong rule-based player (developed by an expert player of M:TG), investigating how many Monte Carlo simulations are required in order to significantly improve playing strength. We also investigate the use of Monte Carlo simulations with a weak (random) player in order to examine whether with sufficient simulated games the weak player can overcome an otherwise stronger (rule-based) player. In this work, we use UCB rather than the more powerful UCT based Monte Carlo approaches for card selection. This is due to the nature of the game, and we discuss the use of UCT for M:TG as a topic for further research.

II. RELATED WORK

Card games typify imperfect information stochastic games. The combination of hidden information (the other player's hand of cards), plus the element of randomness (from deck shuffling) make for a very challenging domain for human players, and for developing AI players [8]. Two of the more popular cards games for AI research have been Poker and Bridge [9].

Poker is a complex card game for multiple players, usually a maximum of 8 at any one time. The game involves aspects of probabilistic analysis, opponent modelling and potential deception on the part of the players [10] [11]. Good play at poker requires the ability to evaluate hand strength correctly and bet accordingly. Work on AI players in this area has focused on using sequence simulations in order to play out the hand multiple times and judge how likely it is to win the pot [12]. Another important area is the ability to work out how an opponent is playing so that the player's strategy can be

C.D. Ward and P.I. Cowling are with the AI Research Group, School of Computing, Informatics and Media, University of Bradford BD7 1DP. E-Mail C.D.Ward or P.I.Cowling@bradford.ac.uk

adjusted accordingly. Work on opponent modelling has shown that Bayesian analysis can used to predict what tactic a player will use based on their past performance and can also help to identify when an opponent might be changing their strategy [13] [14].

Bridge is a strategic card game for 4 players in 2 partnerships. Following suggestions that an AI player would be able to use Monte Carlo search techniques to play Bridge to an expert level [15], Ginsberg developed the bridge playing program GIB [16] which utilises a variety of techniques including Monte Carlo cardplay algorithms, partition search [17] and squeaky wheel optimization in order to provide a layered approach to the whole problem. This has provided a strong AI player with GIB being the first bridge playing program that is able to compete at a master level with humans.

Monte Carlo search has proven effective in games where conventional minimax search approaches fail. Go is a game of perfect information but unlike other perfect information games such as Chess [18] and Checkers [19], Go has resisted the use of brute force search techniques in developing a strong player. With the failure of conventional search approaches for Go. A Monte Carlo approach was suggested [20]. Several Monte Carlo based search strategies have been presented [21] including combining Monte Carlo and tactical search [22] and Monte Carlo Tree Search [23]. All of the approaches utilise the basic premise of Monte Carlo search: rather than examining all the possible moves that are available at every step, we consider all the moves that are available at the first step and then play out the game to the end multiple times using the position resulting from each of those first moves and using either a random move selector or pattern generator to select further moves. The idea is that while we cannot examine every possible move, the results of simulated games will be statistically representative of the strength of each move.

An interesting recent development has been that of bandit based planning [24]. This takes its name from the idea that given a number of one armed bandit slot machines, how do you decided which one to play next in order to maximise the reward over time, given that the results of any individual bandit trial is stochastic. Kocsis et al [24] took an algorithm that had been developed to maximise reward over multiple bandit games and applied it to a tree search in order to find good moves. An algorithm was presented called UCT (Upper Confidence Bounds for Trees) that samples each of the available actions once and then based on the rewards generated, re-samples selected actions with a probability designed to maximise the expected reward. The overall effect is a good representation of the exploration/exploitation problem in that high reward actions are sampled frequently but other actions that have had a low reward in the past are still sampled occasionally. This planning system has been applied to the game of Go with significant success [25] [26]. Indeed the results achieved against strong human players point to UCT representing something of a breakthrough for Computer Go.

UCT has also proven successful in the development of General Game Playing agents [27]. The central components of many AI game players are search and evaluation. Search allows the ability to forecast where the game is going and evaluation assesses the strength of intermediate game positions found during search. In general game playing, there is no prior domain knowledge to describe the strength of a game position, this knowledge has to be derived during the course of the game [28]. The use of Monte Carlo search and UCT in general game playing is showing success [29] in part because it relies much less on heuristic based evaluation functions. CADIAPLAYER [6], a general game playing agent based on UCT approaches was the winner in both the 2007 and 2008 Association for the Advancement of Artificial Intelligence General Game Playing competitions [30]

We believe that Monte Carlo search techniques may yield strong players for the problem of card selection in M:TG, due to their success in other games, and particularly their effectiveness in the general game playing domain. The biggest obstacle in selecting which cards to play is the unknown information (cards in the opponent's hand) and the stochastic information (what cards will be drawn by the player and the opponent). Given that the opponent could have any of a large number of different cards in their hand, any straightforward search technique would have to deal with an enormous branching factor (and note that the branching factor is an important reason why "standard" search techniques did not yield strong AI Go players). It is very difficult to evaluate unfinished positions of M:TG (other than by carrying out game rollouts) which is similar to the problems encountered in both general game playing and Go (and where Monte Carlo techniques have shown their effectiveness).

The rest of this paper is structured as follows. Section 3 describes our test environment, which uses a limited selection of the available card types, and describes the rules of the game. Section 4 describes the variety of AI players we created to play in the test environment and Section 5 gives more detail on the Monte Carlo algorithm used. Section 6 presents our experimental results. Our conclusions are presented in Section 7 and Section 8 identifies some areas of future work.

III. Creature Combat: A subset of M:TG

In the game of Magic: The Gathering each player takes on the role of a wizard contesting a duel with their opponent. The 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 kill 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 Magic arises through understanding and tactically exploiting the interactions between your cards, and between your cards and the opponent's cards. For our research we have chosen to retain the basic structure and turn order mechanics of the game but to focus on creature combat as the way players interact with each other. Creature combat 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 [31]). In our test version of the game players have a deck of cards containing only creatures and land resource cards (of the same, single, colour). The creature cards in the deck have a range of power and toughness values denoting how good they at dealing and taking damage, respectively, and a range of resource costs with, in general, more powerful creatures having a higher cost. Each turn a player may put at most one land into play from their hand. Accumulating sufficient land allows the player to 'cast' a creature card from their hand to the play zone which is then available to attack and defend. Creatures the opponent controls may be available to defend against this attack although they are not required to do so, however, creatures that have attacked on the defending player's previous turn are considered 'tapped' and therefore are not available for defensive duties. Creatures can die, and are consequently removed from play, if they take sufficient damage from an opponent's creatures. Creatures that are not defended against cause damage to the opponent's life total and a player loses the game if their life total reaches zero.

Structure of a game turn

```
AP = ActivePlayer;
OP = OpponentPlayer;

Begin Loop

AP.DrawCard();

//The active player decides which creatures to
//attack with

foreach (PotentialAttacker in AP.PlayZone)
  if (IsGoodAttacker(PotentialAttacker))
    AddAttacker(PotentialAttacker);

//If there are some attackers, the opponent decides
//whether they will block each attacker and which of
//their creatures they will use. Combat Encounters
//are then generated for each attacker/blockers
//combination

if(Attackers.Count > 0)
  foreach (PotentialBlocker in OP.PlayZone)
    att = PotentialBlocker.ChooseAttacker();
    if (att ≠ null)
      att.AddBlocker (PotentialBocker)

//Each Combat Encounter is then resolved,
//unblocked creatures do damage to the player,
//and blocked creatures do damage to each other. The
//active player decides how to apportion an
//attacking creature's damage amongst the creatures
//blocking it

foreach(AttackingCreature)
  if(blockers.count == 0)
    OP.TakeDamage(AttackingCreature.power);
  else
    AP.AssignDamage(AttackingCreature, blockers)
    foreach (blocker in blockers)
      if (blocker.damage >= blocker.toughness)
        MovetoGraveyard(blocker)
    if(blockers.SumOfPower >=
AttackingCreature.toughness)
      MovetoGraveyard(AttackingCreature)

//After combat is resolved see if the Opponent
//Player has died.

if (OP.life <= 0)
  AP.WinGame();
  STOP;

//Then the active player decides which cards to play
//from their hand

AP.SelectCardsForPlayZone(AP.Hand);

//Once cards have been played the Active Player and
//Opponent are switched around and a new turn
//begins.

SwitchActivePlayer();

End Loop
```

Fig. 1: Pseudocode of actions in a typical turn of M:TG.

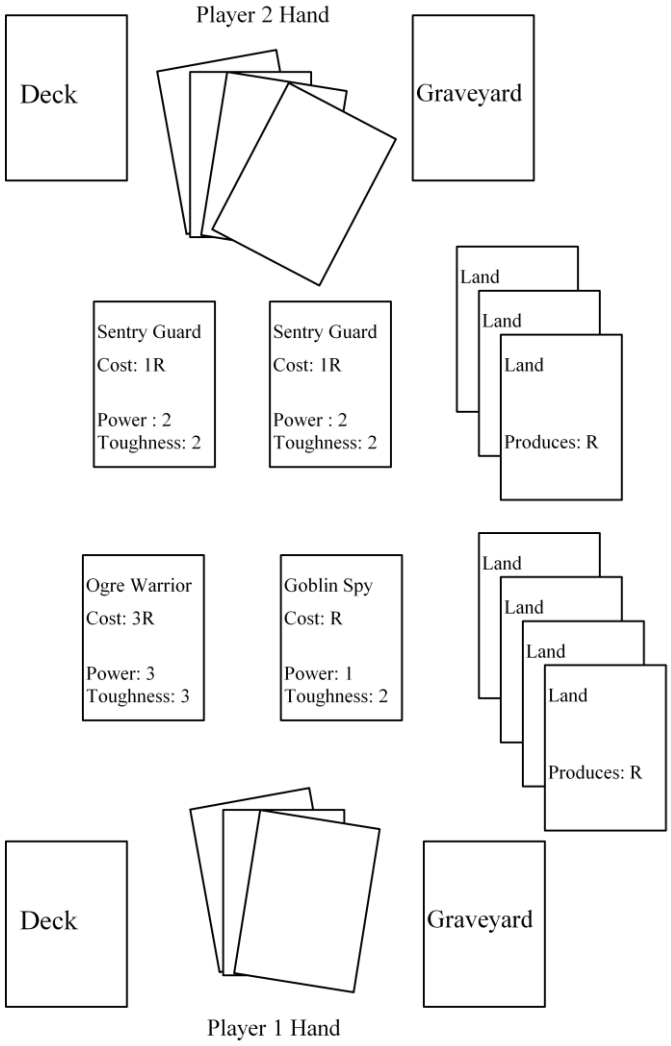

Fig. 2: Layout of the game area during a typical game

Figure 2 shows a representation of the game area during play. Each player has a deck of cards, a graveyard where destroyed cards are placed and a hand of cards that they may play. The play zone between the two players contains those cards that have already been played: land cards that provide resources and creature cards that can be used to attack and defend.

IV. AI PLAYERS

All our AI players use a deck comprising the same type and number of cards. This was to focus the experiments on the effect of the different AI decision processes on player performance rather than any effects gained from one deck being inherently stronger than another. The composition of the deck was chosen to give a good range of creatures ranging from cheap and small up to expensive and large and with sufficient land cards that the players would have a good chance of drawing enough of them to be able to play all of their creatures throughout the course of a game.

Number of Cards	Name	Type	Cost	Power	Toughness
2	Craw_Wurm	creature	4RR	6	4
3	Briarthorn	creature	3R	3	3
3	Elvish_Infantry	creature	1RR	2	3
3	Grizzly_Bear	creature	1R	2	2
3	Jackel_Pup	creature	1R	2	1
2	Cavern_Thoctar	creature	5R	5	5
3	Llanowar_elf	creature	R	1	1
3	Loxodon_Hierarch	creature	3RR	4	4
1	Spiritmonger	creature	4RRR	6	6
17	Mountain (produces R)	land	NA	NA	NA

Table 1: Composition of decks used during testing.

The cost of a creature is comprised of a combination of numbers and letters denoting restrictions on what land cards are required to pay for it. For example a Craw Wurm which costs 4RR requires a total of 6 land, 2 of which must specifically produce R ("red mana") and 4 more that can produce any type of resource ("mana").

Given the structure of the test environment that we have created, there are three areas where the player is required to make a decision

1. Deciding with which creatures to attack the opponent.
2. After having been attacked, deciding how to block the attacking creatures.
3. Deciding which cards to play from their hand.

Decisions about attacking and blocking are essentially perfect information decisions (the immediate result of these decisions can be seen, although their longer term impact is unclear). All the information about how the opponent can block is available to the attacker and the defender sees all of the attacking creatures before making any decisions about how to block them. In the full game of M:TG both players would have the opportunity to play cards which affect the outcome of the combat once attackers and blockers have been decided but before those decisions are resolved. This ability to play 'tricks' was not considered during these tests as it increases enormously the complexity of the decision making process, although we will consider this in future work.

Decisions about which cards to play are imperfect information decisions. The unseen decks of cards are randomly ordered, and players also have no information about what might be in the opponent's hand and deck. In normal play there will usually be multiple options for the player about which cards to play. Whether or not to play a land card, whether to spend resources to play one expensive creature or whether to play multiple cheaper creatures. In some occasions in actual expert play, it is also better to do nothing and wait and see how the game develops.

The branching factor of the amount of potential plays that may be possible from a hand of cards is very much dependant on what stage the game has reached. In the early game the number of possible decisions will be small as it will usually be limited to playing a land and/or maybe playing a single small creature. After about the fourth turn, the number of potential plays increases rapidly (to around 30) as the player will be able to play multiple creatures from their hand and exactly which combination of creatures to play for best effect becomes an interesting decision. In the late game the number of potential plays reduces again as the player will likely have emptied their hand of cards and will be limited to playing the one card that they draw each turn.

For each of the attack, block and play decisions we implemented both a random decision process and one that was based on a knowledge-based approach, capturing the playing style of an expert human player using a collection of rules.

For attacking the rules use the number of creatures controlled by each player and a comparison of the relative power and toughness of those creatures in order to decide which creatures to attack with. The goal is to try and inflict as much damage on the opponent as possible without sacrificing creatures rashly by attacking with weak creatures into strong defenders.

For blocking the rules attempt to calculate whether it is more advantageous to not block because the player will do more damage on the next turn with an attack using all creatures. If this is not the case then the rules attempt to make the most profitable blocks in order to minimise both the amount of damage taken and the number/strength of creatures lost to combat. The rules attempt to avoid situations where more defending creatures are destroyed than attacking ones. The attacking and blocking strategies are somewhat involved, and rules are not given for reasons of space.

For selecting cards to play there are several potential strategies that could have been adopted. Some of the options are

- Maximise the number of creatures the player controls
- Maximise the total power and / or toughness of creatures the player controls
- Always play the most expensive creature the player can afford on any given turn.

The final approach was a combination of these, as follows:

```
1. If there is a land card in the player's hand then
play that land.

2. Play creatures in descending cost order (i.e.
using a "greedy knapsack" heuristic).
```

V. MONTE CARLO SEARCH

In addition to random and rule-based, a third approach for the decision of which cards to play was to implement a bandit based Monte Carlo search in order to find the best move. Our approach utilized the UCB1 algorithm [32]. This algorithm was motivated by the multi-armed bandit problem, to provide

a way of maximising the reward of multiple plays of a set of stochastic bandit machines.

The basic algorithm is as follows

Initialisation: Play each machine once.

Loop: Play machine *j* that maximizes

$$\overline{x_j} + C\sqrt{\frac{\ln n}{n_j}}$$

where $\overline{x_j}$ is the average reward from machine j, *C* is a constant in the range 0-1 that controls how diverse the search is. Larger values giving a more uniform search and smaller values giving a more selective search, n_j is the number of times machine *j* has been played so far and $n=\Sigma_i\, n_i$ is the overall number of plays done so far.

Motivated by ideas from research into Monte Carlo Go [25] where the researchers had carried out multiple simulated games for potential moves in order to try and select the best move, we created a Monte Carlo approach to M:TG:

1: From the players hand determine all possible legal combinations of cards that can be played.

2: Play a simulated game to completion for each legal play using a randomised version of both the players and the opponent's deck in order to represent the unknown information. Record whether the player wins or loses that game.

3: Using the UCB algorithm to select which legal play to investigate, play out further simulated games to completion randomising both players' decks each time.

4: After some arbitrary limit is reached, return the play that has the greatest number of simulations.

We also considered utilising UCT as alternative to UCB in order to compare the effectiveness of both approaches. However, implementation of the UCT algorithm for M:TG is not straightforward. As part of the game of M:TG, players draw an unknown card from their deck at the beginning of each turn. These chance nodes greatly increase the branching factor of the M:TG game tree in the UCT algorithm. At each node beyond the first the tree would have to consider the possibility of drawing any one of the remaining cards in the player's deck. A similar issue arises in backgammon where search beyond the current move has to consider the stochastic roll of the dice and all possible moves that can be made as a result [33]. In M:TG there is no upper limit to how big a player's deck can be, the only restriction is that it must contain a minimum of 40 cards.

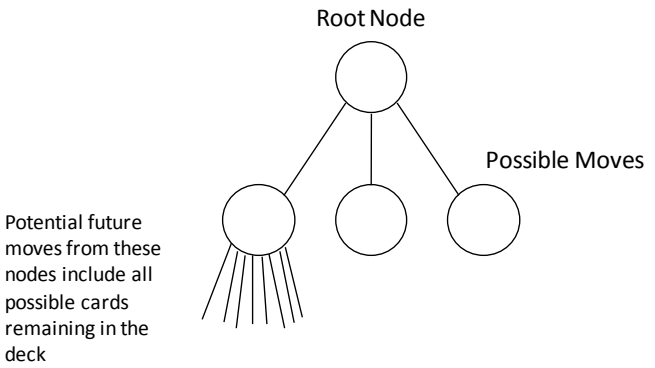

Fig. 3: Illustration of the large branching factor in building a UCT game tree.

A branching factor of the order of 40 for every node in the game tree quickly becomes unrealistic to search in practice and that is the best case scenario, larger decks only make the problem more acute.

Utilising optimisation algorithms such as partition search [17] would allow us to reduce the size of the branching factor in the game tree and this is an avenue we shall explore in future work.

VI. Experimental Results

For the experiments, twelve AI players were created. These players covered all the possible combinations of random play, rule-based play and Monte Carlo search for the various decision points in the game.

AI Players

AI Name	Attack Strategy	Blocking Strategy	Card Play Selection
RA_RB_RP	Random	Random	Random
RA_RB_SP	Random	Random	Rule-based
RA_RB_MC	Random	Random	Monte Carlo
RA_SB_RP	Random	Rule-based	Random
RA_SB_SP	Random	Rule-based	Rule-based
RA_SB_MC	Random	Rule-based	Monte Carlo
SA_RB_RP	Rule-based	Random	Random
SA_RB_SP	Rule-based	Random	Rule-based
SA_RB_MC	Rule-based	Random	Monte Carlo
SA_SB_RP	Rule-based	Rule-based	Random
SA_SB_SP	Rule-based	Rule-based	Rule-based
SA_SB_MC	Rule-based	Rule-based	Monte Carlo

Table 2: Play strategies for AI players

The Monte Carlo based players were allowed 350 simulated games in order decide upon the best move to play, following initial trials. We were able to carry out this many simulations in about one CPU-second. We will see later that this is a suitable compromise between playing strength and CPU time.

Experiment 1: Determining whether it is advantageous to play first or second.

This experiment was carried out in order to determine if there was any inbuilt advantage within the game to playing first or second. At the beginning of a game of M:TG a random method is used to select one of the players. That player then has the choice of acting as either player 1 or player 2 in the game. On the first turn of the game player 1 does not draw a

	RA_RB_RM	RA_RB_SM	RA_SB_RM	RA_SB_SM	SA_RB_RM	SA_RB_SM	SA_SB_RM	SA_SB_SM	RA_RB_MC	RA_SB_MC	SA_RB_MC	SA_SB_MC
RA_RB_RM	100	155	59	117	117	171	99	170	152	110	164	165
RA_RB_SM	45	100	29	69	48	120	49	109	93	65	118	117
RA_SB_RM	141	171	100	146	167	189	172	181	187	152	192	190
RA_SB_SM	83	131	54	100	108	169	109	166	153	117	181	173
SA_RB_RM	83	152	33	92	100	161	76	140	132	81	154	135
SA_RB_SM	29	80	11	31	39	100	29	81	73	49	97	83
SA_SB_RM	101	151	28	91	124	171	100	133	163	79	184	176
SA_SB_SM	30	91	19	34	60	119	67	100	104	62	157	164
RA_RB_MC	48	107	13	47	68	127	37	96	100	39	108	90
RA_SB_MC	90	135	48	83	119	151	121	138	161	100	179	169
SA_RB_MC	36	82	8	19	46	103	16	43	92	21	100	41
SA_SB_MC	35	83	10	27	65	117	24	36	110	31	159	100
Avg No. of wins	68	120	34	71	88	142	75	116	127	76	149	134

Table 4: No. of wins for each AI player after 100 test games as first/second player. Results show the combined number of wins (/200) for the column player. The table is colour coded with increasing whiteness representing better results.

card from their deck, to offset the tempo advantage gained by going first. Despite this, the accepted wisdom among players of the game is that, except in fairly unusual circumstances, it is in the player's best interest to play first.

In the experiment illustrated in table 3, each AI player played 100 games against itself with both players utilising the same deck of cards and the number of wins for each player was recorded. Overall player 1 wins 52% of the games and player 2 wins 48%. This is not significant enough to suggest that there is any particular advantage within the game to either playing position when playing with these decks, and that the extra card which player 2 has offsets the tempo advantage of going first in this case, at least for the deck that we are using. There is a perceived wisdom among M:TG players that certain types of decks do benefit from acting from particular playing positions, specifically that decks that contain a higher preponderance of cheap cards and allow for multiple cards to be played on earlier turns of the game tend to benefit from acting first and that decks which contain a higher ratio of more expensive cards benefit more from playing second.

	Player 1 wins	Player 2 wins
RA_RB_RP	53	47
RA_RB_SP	53	47
RA_SB_RP	48	52
RA_SB_SP	44	56
SA_RB_RP	63	37
SA_RB_SP	44	56
SA_SB_RP	63	37
SA_SB_SP	49	51
RA_RB_MC	48	52
RA_SB_MC	49	51
SA_RB_MC	54	46
SA_SB_MC	53	47

Table 3: No. of wins out of 100 test games for each player when the same AI plays as both players

Experiment 2: Do Monte Carlo simulations provide an advantage in playing strength?

This experiment was carried out in order determine whether using the Monte Carlo simulations to select which cards to play does provide any advantage in playing the game. Each AI played 100 test games against every other AI both as player 1 and player 2.

	% of Random wins	% of Rule-based wins	% of MC wins	% Improvement by MC approaches over Rule-based approaches
RA_RB_X	34	60	64	6.7
RA_SB_X	17	36	38	5.6
SA_RB_X	44	71	75	5.6
SA_SB_X	38	64	67	4.7

Table 5: Comparison of % wins for each type of player, where X represents random (RM), rule-based (SM) and Monte Carlo (MC).

The results of tables 4 and 5 show that there is an advantage for the Monte Carlo based players over their rule-based counterparts. The AI players using Monte Carlo approaches consistently score more game wins than the equivalent rule based player.

As expected the random players in general perform more poorly than players using both rule-based and Monte Carlo approaches. In particular the players using the random attack algorithm tend to have much poorer scores as the AI player will attack with creatures that have no chance of damaging the opponent and will just be destroyed to no effect. However, the rule based blocking players tend to perform less well than their random blocking counterparts. We believe that the expert rules used for the blocking algorithm are too cautious and that the random blocker, which will tend to block more often even if the block is not that advantageous will protect the players life total to a greater extent, enabling the game to go longer and providing more opportunities to gain a victory.

Experiment 3: Effect of number of simulated games on Monte Carlo AI performance.

The purpose of this experiment was to investigate the effect of the Monte Carlo algorithm on AI player's performance and to examine how many Monte Carlo simulations are required before a significant effect on the strength of the player is seen. For this experiment the AI using rule-based heuristics on all aspects of the game (SA_SB_SP) was tested against the AI using heuristics for attacking and blocking and Monte Carlo selection for the card selection (SA_SB_MC). Rule-based players were chosen to remove as much randomness from the test as possible. Trials were carried out with the Monte Carlo player allowed varying numbers of simulated games in order to select which cards to play. Note that when the Monte Carlo based player has multiple moves that have generated the same

reward, it plays one of those moves at random. As the number of Monte Carlo rollouts tends towards zero simulated games we expect similar performance to that of the SA_SB_RM player, which selects a move at random from the legal moves available to it

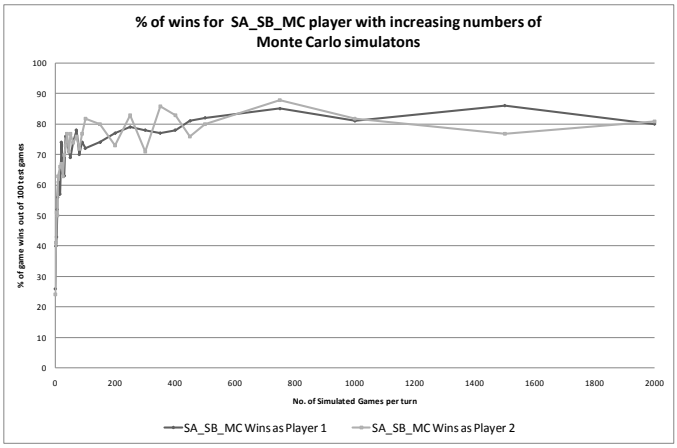

Fig. 4: Comparison of Monte Carlo player wins with increasing numbers of simulated games.

From earlier tests the average number of game wins for the SA_SB_RM player against the SA_SB_SM player is 33%., similar to the number of wins for the SA_SB_MC with 0 simulations at 25%. However it takes very few simulated game rollouts in order to achieve a significant increase in performance. With only 10 simulated games the Monte Carlo based player wins 61% of its games. Increasing the number of simulations available to the Monte Carlo player continues to improve its performance with the performance reaching a plateau of approximately 80% games wins at around 600 simulated games per turn. The results are similar with the Monte Carlo player acting as either player although there is significantly more variation in individual results when the Monte Carlo player is acting as player 2 suggesting that the player 2 positions may be a bit more susceptible to fluctuations due to the stochastic nature of the game.

Experiment 4: Can Monte Carlo simulations enable a weak random player to overcome a stronger rule based player.

The purpose of this experiment was to see whether the use of Monte Carlo simulations based on a weak (random) player strategy can enable the weak AI player to beat a stronger rule based player. AI player RA_RB_MC was tested against AI player SA_SB_SM in trials of 100 test games with the Monte Carlo player allowed increasing numbers of simulations in order to find the best move

It's clear from the graph that the use of Monte Carlo simulations does allow the player, using only random simulation rollouts to increase its performance to a level where it beats a more sophisticated player based upon expert rules. With zero simulations the random player wins only 16% of its games. This win ratio increases rapidly as more simulations are allowed approximately doubling to a 35% win ratio at only 10 simulations. At approx. 200 simulations and above the Monte Carlo player based on random simulation game roll outs achieves a win rate over 50%. Improvement plateaus at around that point suggesting that the limit of improvement has been reached.

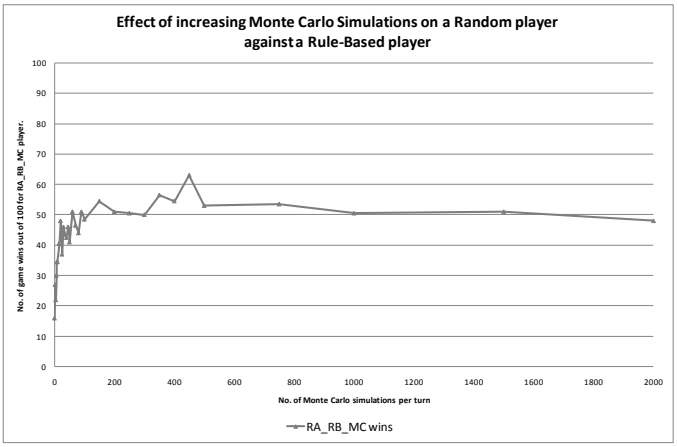

Fig. 5: RA_RB_MC player wins in 100 test games with increasing numbers of simulated games.

VII. Conclusion

In this paper we have advanced reasons why we believe Magic: The Gathering is an interesting test bed for AI games research. The work we have carried out shows that bandit based Monte Carlo search can be applied to the problem of card selection in Magic: The Gathering with some success. The addition of a Monte Carlo search algorithm to a rule-based AI player yields a significant increase in playing strength. We have shown that only a small amount of Monte Carlo simulations are required to achieve a significant increase in playing strength and also that Monte Carlo simulations can be used to enable an otherwise weak random player to overcome a stronger rule based player.

VIII. Further Work

Some areas for future work have already been suggested above, in particular the use of a wider variety of decks and cards from the very wide selection available. Using these cards yields research questions related to questions of producing AI for General Game Playing, especially when the cards in the opponent's deck are unknown.

We discussed above the reasons why the UCT implementations which have proven so successful for Go need further investigation before they can be used for Magic: The Gathering. This is the most pressing question which we will investigate next.

References

[1] Wizards of the Coast. (2009, June) Wizards of the Coast. [Online]. http://www.wizards.com/Magic/Multiverse

[2] Mensa. (1994) Mensa Mindgames. [Online]. http://mindgames.us.mensa.org/AM/Template.cfm?Section=Winning_Games&Template=/customsource/mindgames/winners_list.cfm

[3] Hugo Rifkind. (2005, July) Times Online. [Online]. http://www.timesonline.co.uk/tol/life_and_style/article545389.ece?token=null&offset=0&page=1

[4] Wizards of the Coast. (2009, June) Magic: The

Gathering Pro Tour Honolulu Information. [Online]. http://www.wizards.com/Magic/TCG/Events.aspx?x=mtgcom/protour/honolulu09-format

[5] G. Giles. (1995) Metroactive - Sonoma County Independant. [Online]. http://www.metroactive.com/papers/sonoma/11.09.95/magic.html

[6] Y. Bjornsson, H. Filmar, "CadiaPlayer: A Simulation-Based General Game Player," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 4-15, March 2009.

[7] Wizards of the Coast. (2009, June) Magic: The Gathering Online. [Online]. http://www.wizards.com/Magic/Digital/MagicOnline.aspx

[8] M. Bowling, M. Johanson, N. Burch and D. Szafron, "Strategy Evaluation in Extensive Games with Importance Sampling," in *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008, pp. 72-80.

[9] J. Schaeffer, "A Gamut of Games," *AI Magazine*, vol. 22, pp. 29-46, 2001.

[10] D. Billings, N. Burch, A. Davidson, J. Schaeffer, T. Schauenberg, D. Szafron, "Approximating Game-Theoretic Optimal Strategies for Full-Scale Poker," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003, pp. 661-668.

[11] D. Billings, A. Davidson, J. Schaeffer, D. Szafron, "The Challenge of Poker," *Artificial Intelligence Journal*, vol. 134 (1-2), pp. 201-240, 2002.

[12] D. Billings, A.Davidson, T. Schauenberg, N. Burch, M. Bowling, R. Holte, J. Schaeffer, D. Szafron, "Game Tree Search with Adaptation in Stochastic Imperfect Information Games," in *4th International Conference on Computer and Games*, vol. 3846, 2006, pp. 21-34.

[13] R.J.S. Baker, P.I. Cowling, "Bayesian Opponent Modeling in a Simple Poker Environment," in *IEEE Symposium on Computational Intelligence and Games, 2007.*, 2007, pp. 125-131.

[14] R.J.S. Baker, P. I. Cowling, T. Randall, P. Jiang, "Can Opponent Models Aid Poker Player Evolution?," in *IEEE Symposium on Computational Intelligence and Games*, 2008, pp. 23-30.

[15] M. Ginsberg, "How Computers will play Bridge," *The Bridge World Magazine*, pp. 3-7, June 1996.

[16] M. Ginsberg, "Gib: Imperfect Information in a Computationally Challenging Game," *Journal of Artificial Intelligence Research*, vol. 14, pp. 303-358, 2001.

[17] M. Ginsberg, "Partition Search," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, vol. 2, 1996, pp. 228-233.

[18] M. Campbell, A. Joseph Hoane Jr. , F. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, pp. 57-83, 2002.

[19] J. Schaeffer, *One Jump Ahead: Computer Perfection at Checkers.*: Springer, 2008.

[20] B. Brugmann, "Monte Carlo Go," White Paper 1993.

[21] G. Chaslot, J.T. Saito, J. Uiterwijk, B. Bouzy, H.J. Van der Herik, "Monte Carlo Strategies for Computer Go," in *Proceedings of the 18th Benelux Conference on Artificial Intelligence*, 2006, pp. 83-90.

[22] T. Cazenave, B. Helmstetter, "Combining Tactical Search and Monte-Carlo in the Game of Go," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2005, pp. 171 - 175.

[23] G. Chaslot, M. Winands, J Uiterwijk, H. J. Van der Herik, "Progressive Strategies for Monte Carlo Tree Search," in *Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007)*, 2007, pp. 655-661.

[24] L. Kocsis, C. Szepesvari, "Bandit Based Monte Carlo Planning," *Machine Learning: ECML*, vol. 4212, pp. 282-293, 2006.

[25] S. Gelly Y. Wang, "Modifications of UCT and Sequence-like Simulations for Monte-Carlo Go.," in *Proceedings of the 2007 IEEE Conference on Computational Intelligence and Games*, 2007, pp. 175 - 182.

[26] Chang-Shing Lee, Mei-Hui Wang, G. Chaslot, J-B. Hoock, A. Rimmel, O. Teytaud, Shang-Rong Tsai, Shun-Chin Hsu, Tzung-Pei Hong, "The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 73 - 89, March 2009.

[27] H. Finnsson, Y. Bjornsson, "Simulation-based approach to general game playing," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 259-264.

[28] J. Clune, "Heuristic evaluation functions for general game playing," in *Proceedings of the 22nd AAAI Conference on Artifical Intelligence*, 2007, pp. 1134 - 1139.

[29] S. Sharma, Z. Kobti, S. Goodwin, "Knowledge Generation for Improving Simulations in UCT for General Game Playing," in *AI 2008: Advances in Artificial Intelligence.*: Springer, 2008, pp. 49-55.

[30] M. Genesereth, N. Love, B. Pell, "General Game Playing: Overview of the AAAI Competition," *AI Magazine*, vol. 26, no. 2, pp. 62-72, March 2005.

[31] Jim Ferraiolo. (2004, March) Star City Games. [Online]. http://www.starcitygames.com/php/news/article/6985.html

[32] P. Auer, N. Cesa-Bianchi, P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, pp. 235-256, 2002.

[33] G. Tesauro, "Temporal Difference Learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58-68, March 1995.