An Overview of Search Techniques in Multi-Player Games

J. (Pim) A. M. Nijssen and Mark H. M. Winands

Department of Knowledge Engineering, Faculty of Humanities and Sciences,
Maastricht University, Maastricht, The Netherlands
{pim.nijssen,m.winands}@maastrichtuniversity.nl

Abstract. In this paper we compare several search techniques for multi-player games. We test the performance of the minimax-based search techniques $\max^n$, paranoid search and Best-Reply Search. Furthermore, we investigate how the tree structure of each of the minimax-based techniques can be applied in MCTS. The test domain consists of four different multi-player games: Chinese Checkers, Focus, Rolit and Blokus. Based on the experimental results, we may conclude that Best-Reply Search is generally the best minimax-based search technique. Monte-Carlo Tree Search performs best with the $\max^n$ tree structure.

1 Introduction

Multi-player games are games that can be played by more than 2 players. They have several properties that makes them an interesting challenge for computers. First, contrary to 2-player games, pruning in search trees is considerably more difficult. Second, the opponents' moves are more unpredictable, as coalitions may occur.

Over the past years, several tree search techniques have been developed for playing multi-player games. In 1986, Luckhardt and Irani proposed a modification of the minimax-search technique to play multi-player games, called $\max^n$ [10]. In 2000, Sturtevant and Korf proposed the paranoid search algorithm [17]. With this technique they showed, in the trick-based card game Sergeant Major, that much more pruning is possible than in $\max^n$. However, due to the, often incorrect, assumption that all opponents cooperate against the root player, a paranoid player often plays too defensively. Trying to overcome this shortcoming of the paranoid algorithm, several techniques have been developed to make the algorithm less paranoid. In 2005, Lorenz and Tscheuschner proposed the coalition-mixer algorithm for 4-player chess [9] and in 2009, Zuckerman *et al.* proposed the MP-Mix algorithm [19]. This algorithm uses an evaluation function to determine which search technique should be used. Another algorithm was proposed by Schadd and Winands in 2011, namely Best-Reply Search (BRS) [14]. This algorithm performs significantly better than paranoid search in various multi-player games.

Over the past years, Monte-Carlo Tree Search (MCTS) [6, 8] has become a popular technique for playing multi-player games as well. MCTS is a best-first

search technique that, instead of an evaluation function, uses simulations to guide the search. This algorithm is able to compute mixed equilibria in multi-player games [16], contrary to max^n, paranoid and BRS. MCTS is used in a variety of multi-player games, such as Focus [11, 12], Chinese Checkers [11, 12, 16], Hearts [16], Spades [16], and multi-player Go [4].

In this paper we compare several search techniques that have been developed over the years. We test the performance of max^n, paranoid and BRS in four different multi-player games. Furthermore, we investigate how the tree structure of these techniques can be applied in the MCTS framework.

The paper is structured as follows. In Section 2 we give an overview of the search techniques used in this paper. Next, in Section 3 we explain the rules and applied domain knowledge of the four games. In Section 4 we describe the experiments and the results. Finally, in Section 5 we provide the conclusions and an outline of future research.

2 Search Techniques

In this section we discuss the search techniques investigated in this paper. In Subsection 2.1 we explain the three minimax-based search techniques: max^n, paranoid and BRS. In Subsection 2.2 we briefly discuss MCTS and explain how the tree structure of each of the minimax-based techniques can be applied in MCTS.

2.1 Minimax-based search techniques

The traditional algorithm for playing multi-player games is *max^n* [10]. This technique is an extension of minimax search to multi-player games. In the leaf nodes of the search tree, each player is awarded a payoff. Each player chooses the child with the highest payoff. A disadvantage of max^n is that only a limited amount of pruning is possible. Shallow pruning [17] is the easiest and safest way to achieve some cut-offs.

Paranoid search [17] assumes that all opponents have formed a coalition against the root player. Using this assumption, the game can be reduced to a 2-player game where the root player is represented in the tree by MAX nodes and the opponents by MIN nodes. The advantage of this assumption is that $\alpha\beta$-like pruning [7] is possible in the search tree, allowing deeper searches in the same amount of time. The disadvantage is that, because of the often incorrect paranoid assumption, the player may become too defensive.

In 2011, Schadd and Winands proposed a new algorithm for playing multi-player games, namely *Best Reply Search* (BRS) [14]. This technique is similar to paranoid search, but instead of allowing all opponents to make a move, only one opponent is allowed to do so. The advantage of this technique is that more MAX nodes are investigated. The disadvantage is that, if passing is not allowed, illegal positions or positions that are unreachable in the actual game are taken into account.

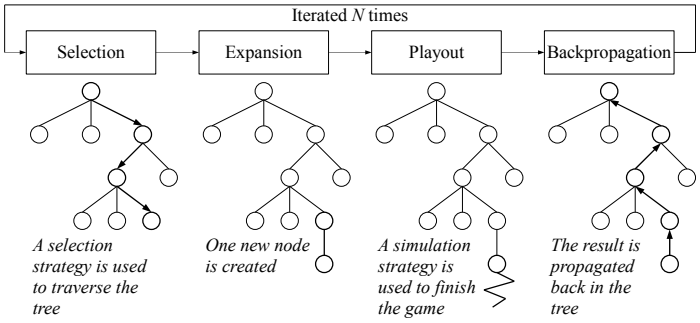


Fig. 1. Monte-Carlo Tree Search scheme (Slightly adapted from [5]).

2.2 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) [6, 8] is a search technique that gradually builds up a search tree, guided by Monte-Carlo simulations. In contrast to classic search techniques such as $\alpha\beta$-search, it does not require a heuristic evaluation function.

The MCTS algorithm consists of four phases [5]: selection, expansion, playout and backpropagation (see Fig. 1). By repeating these four phases iteratively, the search tree is constructed gradually. The tree is traversed using the Upper Confidence bounds applied to Trees (UCT) [8] selection strategy. In our program, UCT has been enhanced with Progressive History [11]. The child i with the highest score v_i in Formula 1 is selected.

$$v_i = \bar{s_i} + C\sqrt{\frac{\ln(n_p)}{n_i}} + W\frac{\bar{s_a}}{n_i(1-\bar{s_i})+1} \tag{1}$$

In this formula, $\bar{s_i}$ denotes the win rate of child i, where $\bar{s_i} \in [0, 1]$. The variables n_i and n_p denote the total number of times that child i and parent p have been visited, respectively. C is a constant that determines the exploration factor of UCT. In the Progressive History part, $\bar{s_a}$ represents the win rate of move a. W is a constant that determines the influence of Progressive History.

The basic MCTS algorithm uses a tree structure which is analogous to the $\max^n$ search tree. It is possible to apply the paranoid and BRS tree structures to MCTS as well. The idea of using a paranoid tree structure in MCTS was presented by Cazenave [4], however he did not implement or test it. When using paranoid search or BRS in MCTS, the opponents use a different UCT formula. Instead of considering their own win rate, they try to minimize the win rate of the root player. In the MIN nodes of the tree, the following modified version of Formula 1 is used.

$$v_i = (1-\bar{s_i}) + C\sqrt{\frac{\ln(n_p)}{n_i}} + W\frac{(1-\bar{s_a})}{n_i\bar{s_i}+1} \tag{2}$$

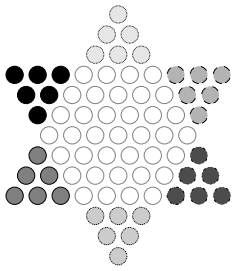


Fig. 2. A Chinese Checkers board.

3 Test Domains

The performance of the search techniques is tested in four different games: Chinese Checkers, Focus, Rolit and Blokus. In this section we briefly discuss the rules and the properties of these games in Subsections 3.2 – 3.4. In Subsection 3.5 we explain the move and board evaluators for the games.

3.1 Chinese Checkers

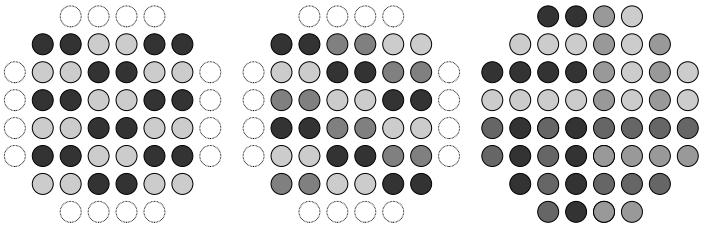
Chinese Checkers is a board game that can be played by 2 to 6 players. This game was invented in 1893 and has since then been released by various publishers under different names. Chinese Checkers is played on a star-shaped board. The most commonly used board contains 121 fields, where each player starts with 10 checkers. We decided to play on a slightly smaller board [16] (see Fig. 2). In this version, each player plays with 6 checkers. The advantage of a smaller board is that games take a shorter amount of time to complete, which means that more Monte-Carlo simulations can be performed and more experiments can be run. Also, it allows the use of a stronger evaluation function.

The goal of each player is to move all his pieces to his home base at the other side of the board. Pieces may move to one of the adjacent fields or they may jump over another piece to an empty field. It is also allowed to make multiple jumps with one piece in one turn, making it possible to create a setup that allows pieces to jump over a large distance. The first player who manages to fill his home base wins the game.

3.2 Focus

Focus is an abstract multi-player strategy board game, which was invented in 1963 by Sid Sackson [13]. This game has also been released under the name *Domination*. Focus is played on an 8×8 board where in each corner three fields are removed. It can be played by 2, 3 or 4 players. Each player starts with a number of pieces on the board. In Fig. 3, the initial board positions for the 2-, 3- and 4-player variants are given.

In Focus, pieces can be stacked on top of each other. A stack may contain up to 5 pieces. Each turn a player may move a stack orthogonally as many fields as



(a) 2 players (b) 3 players (c) 4 players

Fig. 3. Set-ups for Focus.

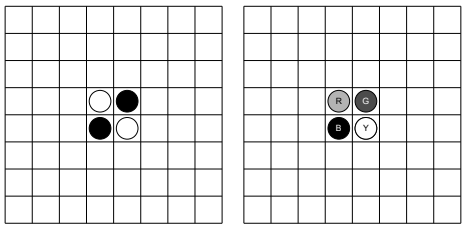
the stack is tall. A player may only move a stack of pieces if a piece of his color is on top of the stack. It is also allowed to split stacks in two smaller stacks. If a player decides to do so, then he only moves the upper stack as many fields as the number of pieces that are being moved.

If a stack lands on top of another stack, the stacks are merged. If the merged stack has a size of $n > 5$, then the bottom $n-5$ pieces are captured by the player, such that there are 5 pieces left. If a player captures one of his own pieces, he may later place one piece back on the board, instead of moving a stack. This piece may be placed either on an empty field or on top of an existing stack.

There exist two variations of the game, each with a different winning condition. In the standard version of the game, a player has won if all other players cannot make a legal move. However, such games can take a long time to finish. Therefore, we chose to use the shortened version of the game. In this version, a player has won if he has either captured certain number of pieces in total, or a number of pieces from each player. In the 2-player variant, a player wins if he has captured at least 6 pieces from the opponent. In the 3-player variant, a player has won if he has captured at least 3 pieces from both opponents or at least 10 pieces in total. In the 4-player variant, the goal is to capture at least 2 pieces from each opponent or capture at least 10 pieces in total.

3.3 Rolit

Rolit is a multi-player variant of the 2-player game Othello. This game was introduced in 1975. It is similar to a game invented around 1880, called Reversi. This game was invented by either Lewis Waterman or John W. Mollett. At the end of the 19th century it gained much popularity in England and in 1898, games publisher Ravensburger started producing the game as one of its first titles. Othello is played by 2 players, Black and White, on an 8×8 board. On this board, so-called discs are placed. Discs have two different sides: a black one and a white one. If a disc on the board has its black side faced up, it is owned by player Black and if it has its white side up, it belongs to player White. The game starts with four discs on the board, as shown in Fig. 4(a). Black always starts the game, and the players take turns alternately. When it is a player's turn he has to place a disc on the board in such a way that he captures at least one of the opponents discs. A disc is captured when it lies on a straight line between



(a) (b)

Fig. 4. Set-ups for Othello (a) and Rolit (b).

the placed disc and another disc of the player making the move. Such a straight line may not be interrupted by an empty square or a disc of the player making the move. All captured discs are flipped and the turn goes to the other player. If a player cannot make a legal move, he has to pass. If both players have to pass, the game is over. The player who owns the most discs, wins the game.

For Rolit, the rules are slightly different. Rolit can be played by up to 4 players, called Red, Yellow, Green and Blue. The initial board position is shown in Fig. 4(b). The largest difference is that if a player cannot capture any pieces, which will occur during the first few rounds of a 4-player game, he may put a piece orthogonally or diagonally adjacent to any of the pieces already on the board. Using this rule, passing does not occur and the game is finished when the entire board is filled. The scoring is similar to Othello; the player owning the most pieces wins. We remark that, contrary to Focus and Chinese Checkers, Rolit can end in a draw between several players.

3.4 Blokus

Blokus is a 4-player tile placement game developed by Bernard Tavitian in 2000. The board consists of 20×20 squares. Each player receives 21 pieces varying in size from one to five squares in all possible shapes. Alternately, the players place one of their pieces on the board. The pieces may be rotated in any way. The difficulty in this game is that any square may only be occupied by one piece and two pieces of the same player may not be orthogonally adjacent. However, they have to be adjacent diagonally to any of the player's pieces already on the board. The first pieces of the players should all be placed in one of the corners.

The game finishes when none of the players can place a piece on the board anymore. The player who has the largest number of squares on the board occupied is the winner. Note that, similar to Rolit, draws can occur. However, there is one tie breaker in this game. If more than one player manages to place all pieces on the board, the winner is the player who placed the piece of size 1 on the board during the last round.

3.5 Domain Knowledge

For the minimax-based techniques, a board evaluator is necessary to evaluate the leaf nodes of the search tree. This evaluator computes a heuristic value for each

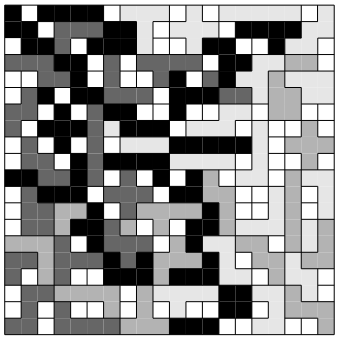


Fig. 5. A finished game of Blokus.

player, based on the current board position. A move evaluator is used for static move ordering. The evaluator assigns a value to a move, without considering the board position. The move evaluator is also used in the MCTS-framework, for determining the moves in the ϵ-greedy playouts [18].

For Chinese Checkers, the board evaluator uses a lookup table [16]. This table stores, for each possible configuration of pieces, the minimum number of moves a player should perform to get all pieces in the home base, assuming that there are no opponents' pieces on the board. For any player, the value of the board equals $28 - m$, where m is the value stored in the table which corresponds to the piece configuration of the player. We remark that 28 is the highest value stored in the table. The move evaluator of Chinese Checkers uses the function $d_s - d_t$, where d_s is the distance of the source location of the piece that is moved to the home base, and d_t the distance of the target location to the home base.

For Focus, the board evaluator is based on the minimum number of pieces each player needs to capture to win the game, r, and the number of stacks each player controls, c. For each player, the score is calculated using the formula $600 - 100r + c$. The move evaluator applies the function $10(n + t) + s$, where n is the number of pieces moved, t is the number of pieces on the target location, and s is the number of stacks the player gained. The value of s can be 1, 0, or –1.

For Rolit, the board evaluator is similar to the pattern based evaluation function used by Buro in his Othello program Logistello [3]. Over 90,000 games from the WTHOR database[1] were analyzed on 12 different patterns for 15 stages (4 moves per stage) in the game. For each pattern, the average score at the end of the game is stored. To use this pattern database in Rolit, we use the assumption that all of the opponent's pieces have the same color [14]. This reduces the accuracy, but it is unfeasible to create a pattern database for four colors. The move evaluator depends on the location of the square where the piece is placed. The values of the squares are displayed in Fig. 6.

[1] http://www.ffothello.org/info/base.php

5	2	4	3	3	4	2	5
2	1	3	3	3	3	1	2
4	3	4	4	4	4	3	4
3	3	4	4	4	4	3	3
3	3	4	4	4	4	3	3
4	3	4	4	4	4	3	4
2	1	3	3	3	3	1	2
5	2	4	3	3	4	2	5

Fig. 6. The values of the squares in Rolit.

For Blokus, the board evaluator counts the number of squares that each player has occupied. The move evaluator depends on the size of the piece that is played on the board. Large pieces are preferred over small ones.

For all board and move evaluators, a small random factor is added to the score. This factor is only to differentiate between board positions or moves that have the same value. This random factor is added to prevent the players from being deterministic.

4 Experiments

In this section, we describe the experiments performed. The program is written in Java [11]. For Formula 1, the constant C is set to 0.2 and W is set to 5. All MCTS-based players use ϵ-greedy playouts with $\epsilon = 0.05$. These values were achieved by systematic testing. All minimax-based players use a DEEP transposition table [2] and static move ordering. Furthermore, the paranoid and BRS players use killer moves [1] and the history heuristic [15]. Finally, for the $\max^n$ player, shallow pruning is applied [17], while the paranoid and BRS players use $\alpha\beta$ pruning [7].

The experiments were run on a cluster consisting of AMD64 Opteron 2.4 GHz processors. For the games, there may be an advantage regarding the order of play and the number of different players. Games where not all player types are playing are not interesting, so these are not considered. Table 1 shows in how many ways the player types can be assigned. Each assignment is played multiple times until at least 1,000 games are played and each assignment was played equally often.

Table 1. The number of ways 2 or 3 different player types can be assigned. Between brackets is the number of games that are played per match.

Number of players	2 player types	3 player types
3	6 (1050)	6 (1050)
4	14 (1050)	36 (1044)
6	62 (1054)	540 (1080)

Table 2. Results of max^n vs. paranoid vs. BRS

Game	Players	Time (ms)	Max^n Win rate (%)	Max^n Depth (ply)	Paranoid Win rate (%)	Paranoid Depth (ply)	BRS Win rate (%)	BRS Depth (ply)
Chinese Checkers	3	250	0.6±0.5	3.22	32.7±2.8	4.38	66.8±2.8	4.87
Chinese Checkers	3	1000	0.1±0.2	3.45	24.9±2.6	5.11	75.0±2.6	5.40
Chinese Checkers	3	5000	0.0±0.0	4.23	25.9±2.7	5.64	74.1±2.7	6.61
Chinese Checkers	4	250	4.5±1.3	3.09	7.2±1.6	3.64	88.3±1.9	4.39
Chinese Checkers	4	1000	3.5±1.1	3.66	21.2±2.5	4.83	75.3±2.6	5.01
Chinese Checkers	4	5000	3.9±1.1	4.23	19.3±2.4	5.38	76.8±2.6	5.73
Chinese Checkers	6	250	16.4±2.2	3.07	16.4±2.2	3.61	67.2±2.8	3.77
Chinese Checkers	6	1000	14.3±2.1	3.84	10.4±1.8	4.07	75.2±2.6	4.70
Chinese Checkers	6	5000	24.2±2.6	4.15	11.2±1.9	4.59	64.7±2.9	5.10
Focus	3	250	4.8±1.3	3.43	41.8±3.0	4.13	53.4±3.0	4.16
Focus	3	1000	4.0±1.2	3.87	34.0±2.9	4.84	62.0±2.9	4.96
Focus	3	5000	3.6±1.1	4.60	31.0±2.8	5.18	65.4±2.9	5.75
Focus	4	250	8.6±1.7	3.29	17.7±2.3	3.37	73.7±2.7	3.99
Focus	4	1000	6.0±1.4	3.69	22.7±2.5	4.48	71.3±2.7	4.82
Focus	4	5000	7.3±1.6	4.32	28.9±2.7	5.15	63.8±2.9	5.20
Rolit	3	250	0.8±0.5	4.48	41.4±3.0	6.31	57.9±3.0	6.15
Rolit	3	1000	11.0±1.9	5.26	28.3±2.7	7.74	60.7±3.0	7.39
Rolit	3	5000	0.3±0.3	6.01	66.5±2.9	8.97	33.2±2.8	8.61
Rolit	4	250	16.8±2.3	4.52	46.6±3.0	5.88	36.7±2.9	5.45
Rolit	4	1000	13.4±2.1	5.27	40.4±3.0	6.84	46.2±3.0	6.60
Rolit	4	5000	11.1±1.9	6.01	40.7±3.0	7.87	48.2±3.0	7.68
Blokus	4	250	22.2±2.5	1.89	29.0±2.8	2.37	48.8±3.0	2.46
Blokus	4	1000	19.6±2.4	2.27	25.8±2.7	2.97	54.7±3.0	3.36
Blokus	4	5000	11.8±2.0	2.76	20.8±2.5	3.38	67.3±2.9	4.03

4.1 Minimax-based techniques

In the first set of experiments we match the three basic minimax-based players against each other: max^n, paranoid and BRS. The win rates and the average search depths of the players in the different games are displayed in Table 2.

The results show that max^n is by far the weakest algorithm. In every game with any number of players and time setting, max^n has a significantly lower win rate that both paranoid and BRS. The exception is 6-player Chinese Checkers. Because paranoid also has barely any pruning, max^n plays at least as strong as paranoid. Max^n also plays relatively well in Blokus, where all players have difficulty reaching a decent search depth. Only the BRS player can reach a second level of MAX nodes. In most games, BRS is the best search technique. Overall, the BRS players can search slightly deeper than the paranoid players. The most notable exception is Rolit. In this game, the paranoid players can generally search slightly deeper. Also, with some settings paranoid outperforms BRS. This is comparable to the results achieved by Schadd and Winands [14].

4.2 MCTS variants

In the second set of experiments, we test the performance of three different MCTS players. Each player uses a different tree structure: max^n (MCTS-max^n), paranoid (MCTS-paranoid) or BRS (MCTS-BRS). The win rates and the median number of playouts per move are summarized in Table 3.

The results reveal that MCTS clearly performs best using the standard max^n tree structure. Only in Blokus, MCTS-max^n is not significantly stronger than MCTS-paranoid. Paranoid and BRS perform well in the minimax framework

Table 3. Results of MCTS-max^n vs. MCTS-paranoid vs. MCTS-BRS

Game	Players	Time (ms)	MCTS-max^n Win rate (%)	MCTS-max^n Playouts (median)	MCTS-paranoid Win rate (%)	MCTS-paranoid Playouts (median)	MCTS-BRS Win rate (%)	MCTS-BRS Playouts (median)
Chinese Checkers	3	250	40.5±3.0	1,166	35.5±2.9	1,168	24.0±2.6	1,163
Chinese Checkers	3	1000	43.7±3.0	5,004	27.5±2.7	5,008	28.8±2.7	4,968
Chinese Checkers	3	5000	54.2±3.0	26,058	19.0±2.4	25,951	26.8±2.7	25,786
Chinese Checkers	4	250	42.5±3.0	898	33.0±2.9	900	24.4±2.6	891
Chinese Checkers	4	1000	49.1±3.0	4,042	29.8±2.8	4,022	21.1±2.5	3,962
Chinese Checkers	4	5000	62.1±2.9	21,680	17.9±2.3	21,531	20.1±2.4	20,893
Chinese Checkers	6	250	45.0±3.0	711	30.2±2.7	723	24.8±2.6	713
Chinese Checkers	6	1000	51.4±3.0	3,140	25.8±2.6	3,212	22.8±2.5	3,167
Chinese Checkers	6	5000	63.6±2.9	17,155	18.9±2.3	18,113	17.5±2.3	16,780
Focus	3	250	38.1±2.9	3,370	33.0±2.8	3,389	29.9±2.8	3,473
Focus	3	1000	37.4±2.9	12,745	29.0±2.7	13,058	33.5±2.9	12,837
Focus	3	5000	38.4±2.9	57,450	27.9±2.7	60,144	33.7±2.9	57,459
Focus	4	250	40.1±3.0	2,212	37.2±2.9	2,182	22.7±2.5	2,144
Focus	4	1000	39.1±3.0	9,189	33.0±2.9	9,220	27.9±2.7	8,881
Focus	4	5000	41.6±3.0	50,260	28.3±2.7	51,442	30.1±2.8	48,206
Rolit	3	250	44.3±3.0	2,011	31.6±2.8	2,024	24.1±2.6	2,017
Rolit	3	1000	55.6±3.0	8,333	26.0±2.7	8,356	18.4±2.3	8,320
Rolit	3	5000	67.3±2.8	41,553	16.6±2.3	41,459	16.1±2.2	42,049
Rolit	4	250	41.6±3.0	2,111	33.8±2.9	2,109	24.6±2.6	2,078
Rolit	4	1000	44.9±3.0	8,510	30.2±2.8	8,483	24.9±2.6	8,331
Rolit	4	5000	56.6±3.0	42,999	23.1±2.6	42,489	20.3±2.4	41,095
Blokus	4	250	34.9±2.9	184	36.2±2.9	185	28.9±2.7	177
Blokus	4	1000	33.3±2.9	776	34.7±2.9	780	32.1±2.8	749
Blokus	4	5000	33.0±2.9	4,170	34.5±2.9	4,212	32.6±2.8	4,061

because they increase the amount of pruning. Because $\alpha\beta$-pruning does not occur in MCTS, this advantage is nonexistent in the MCTS framework. It also becomes clear that MCTS-paranoid significantly outperforms MCTS-BRS. A possible explanation for this result is that illegal positions are reached in the tree. Performing a playout from these illegal or unreachable positions apparently leads to unreliable results.

4.3 BRS versus MCTS-max^n

Based on the previous sets of experiments, we can conclude that BRS is the strongest minimax-based technique and that MCTS-max^n is the strongest MCTS technique. To determine which technique performs best in multi-player games, we let these two players play against each other in the final set of experiments. The results are displayed in Table 4.

From the results we can conclude that there is no clear winner. BRS significantly outperforms MCTS-max^n in Focus, while MCTS-max^n is stronger in Blokus and Rolit. In Chinese Checkers, the winner depends on the time settings. With a higher time setting, MCTS-max^n becomes stronger than BRS. In all games, MCTS-max^n performs relatively better with higher time settings.

5 Conclusions

Among the three minimax-based search techniques we tested, BRS turns out to be the strongest one. Overall, it reaches the highest search depth, and because of its tree structure more MAX nodes are investigated than in paranoid and max^n. BRS significantly outperforms max^n and paranoid in Chinese Checkers, Focus and Blokus. Only in Rolit, paranoid outperforms BRS with some settings.

Table 4. Results of MCTS-max^n against BRS

Game	Players	Time (ms)	MCTS-max^n Win rate (%)	MCTS-max^n Playouts (median)	BRS Win rate (%)	BRS Depth (ply)
Chinese Checkers	3	250	21.7±2.5	1,145	78.3±2.5	4.75
Chinese Checkers	3	1000	42.1±3.0	5,206	57.9±3.0	5.34
Chinese Checkers	3	5000	60.4±3.0	27,639	39.6±3.0	6.34
Chinese Checkers	4	250	26.1±2.7	874	73.9±2.7	4.22
Chinese Checkers	4	1000	55.2±3.0	4,007	44.8±3.0	4.93
Chinese Checkers	4	5000	71.5±2.7	21,603	28.5±2.7	5.52
Chinese Checkers	6	250	35.4±2.9	703	64.6±2.9	3.53
Chinese Checkers	6	1000	66.9±2.8	3,227	33.1±2.8	4.40
Chinese Checkers	6	5000	90.1±1.8	16,603	9.9±1.8	4.71
Focus	3	250	9.9±1.8	1,481	90.1±1.8	4.19
Focus	3	1000	19.4±2.4	7,449	80.6±2.4	4.88
Focus	3	5000	28.3±2.7	42,326	71.7±2.7	5.59
Focus	4	250	23.2±2.6	1,546	76.8±2.6	3.88
Focus	4	1000	29.1±2.8	6,883	70.9±2.8	4.69
Focus	4	5000	41.5±3.0	39,786	58.5±3.0	5.06
Rolit	3	250	83.6±2.2	1,954	16.4±2.2	6.19
Rolit	3	1000	93.4±1.5	8,103	6.6±1.5	7.44
Rolit	3	5000	93.6±1.5	41,397	6.4±1.5	8.60
Rolit	4	250	78.7±2.5	1,946	21.3±2.5	5.54
Rolit	4	1000	85.8±2.1	8,162	14.2±2.1	6.67
Rolit	4	5000	88.6±1.9	42,860	11.4±1.9	7.60
Blokus	4	250	47.8±3.0	165	52.2±3.0	2.46
Blokus	4	1000	68.8±2.8	818	31.2±2.8	3.25
Blokus	4	5000	83.7±2.2	4,412	16.3±2.2	3.83

In the MCTS framework, the max^n tree structure appears to perform best. The advantages of paranoid and BRS in the minimax framework do not apply in MCTS, because $\alpha\beta$-pruning is not applicable in MCTS. MCTS-paranoid outperforms MCTS-BRS and a possible reason for this is that MCTS-BRS performs playouts starting from an illegal or unreachable board position. This may lead to inaccurate results.

Finally, in a comparison between MCTS-max^n and BRS, it turns out that there is no clear winner. In Focus, BRS is considerably stronger, while in Rolit and Blokus MCTS-max^n significantly outperforms BRS. In Chinese Checkers, the winner depends on the thinking time. Overall, with higher time settings, the MCTS-based player performs relatively better.

In this research we investigated three basic tree search algorithms, i.e. max^n, paranoid and BRS. We did not consider algorithms derived from these techniques, such as the Coalition-Mixer [9] or MP-Mix [19]. They use a combination of max^n and (variations of) paranoid search. They also have numerous parameters that need to be tuned. Tuning and testing such algorithms in multi-player games is a direction of future research. Another possible future research direction is the application of paranoid search and BRS in the playout phase of MCTS. Cazenave [4] used paranoid playouts in multi-player Go, improving the performance of an MCTS player significantly.

References

1. S.G. Akl and M.M. Newborn. The Principal Continuation and the Killer Heuristic. In *Proceedings of the ACM Annual Conference*, pages 466–473, New York, NY, USA, 1977. ACM.

2. D.M. Breuker, Uiterwijk J.W.H, H, and H.J. van den Herik. Replacement Schemes and Two-Level Tables. *ICCA Journal*, 19(3):175–180, 1996.
3. M. Buro. Experiments with Multi-ProbCut and a new high-quality evaluation function for Othello. *Games in AI Research*, pages 77–96, 1997.
4. T. Cazenave. Multi-player Go. In H.J. van den Herik, X. Xu, Z. Ma, and M.H.M. Winands, editors, *Computers and Games (CG 2008)*, volume 5131 of *LNCS*, pages 50–59, Berlin, Germany, 2008. Springer.
5. G.M.J-B. Chaslot, M.H.M. Winands, J.W.H.M. Uiterwijk, H.J. van den Herik, and B. Bouzy. Progressive strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation*, 4(3):343–357, 2008.
6. R. Coulom. Efficient selectivity and backup operators in Monte-Carlo Tree Search. In H.J. van den Herik, P. Ciancarini, and H.H.L.M. Donkers, editors, *Computers and Games (CG 2006)*, volume 4630 of *LNCS*, pages 72–83, Berlin, Germany, 2007. Springer.
7. D.E. Knuth and R.W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.
8. L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *LNAI*, pages 282–293, Berlin, Germany, 2006. Springer.
9. U. Lorenz and T. Tscheuschner. Player Modeling, Search Algorithms and Strategies in Multi-player Games. In H.J. van den Herik, S.-C. Hsu, T.-S. Hsu, and H.H.L.M. Donkers, editors, *Advances in Computer Games (ACG11)*, volume 4250 of *LNCS*, pages 210–224, Berlin, Germany, 2006. Springer.
10. C. Luckhardt and K.B. Irani. An algorithmic solution of n-person games. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*, volume 1, pages 158–162, 1986.
11. J.A.M. Nijssen and M.H.M. Winands. Enhancements for Multi-Player Monte-Carlo Tree Search. In H.J. van den Herik, H. Iida, and A. Plaat, editors, *Computers and Games (CG 2010)*, volume 6515 of *LNCS*, pages 238–249, Berlin, Germany, 2011. Springer.
12. J.A.M. Nijssen and M.H.M. Winands. Playout Search for Monte-Carlo Tree Search in Multi-Player Games. In *Advances in Computer Games (ACG13)*, volume 7168 of *LNCS*, pages 72–83, Berlin, Germany, 2012.
13. S. Sackson. *A Gamut of Games*. Random House, New York, NY, USA, 1969.
14. M.P.D. Schadd and M.H.M. Winands. Best Reply Search for Multiplayer Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(1):57–66, 2011.
15. J. Schaeffer. The history heuristic. *ICCA Journal*, 6(3):16–19, 1983.
16. N.R. Sturtevant. An analysis of UCT in multi-player games. In H.J. van den Herik, X. Xu, Z. Ma, and M.H.M. Winands, editors, *Computers and Games (CG 2008)*, volume 5131 of *LNCS*, pages 37–49, Berlin, Germany, 2008. Springer.
17. N.R. Sturtevant and R.E. Korf. On pruning techniques for multi-player games. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 201–207. AAAI Press / The MIT Press, 2000.
18. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.
19. I. Zuckerman, A. Felner, and S. Kraus. Mixing Search Strategies for Multi-Player Games. In C. Boutilier, editor, *Proceedings of the Twenty-first International Joint Conferences on Artificial Intelligence (IJCAI-09)*, pages 646–651, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.