Best Reply Search for Multiplayer Games

Article *in* IEEE Transactions on Computational Intelligence and AI in Games · April 2011

DOI: 10.1109/TCIAIG.2011.2107323 · Source: IEEE Xplore

Mark H.M. Winands
Maastricht University

Best-Reply Search for Multi-Player Games

Maarten P.D. Schadd and Mark H.M. Winands

***Abstract*—This article proposes a new algorithm, called *Best-Reply Search* (BRS), for deterministic multi-player games with perfect information. In BRS, only the opponent with the strongest counter move is allowed to make a move. More turns of the root player can be searched resulting in long-term planning. We test BRS in the games of Chinese Checkers, Focus and Rolit™. In all games, BRS is superior to the max^n algorithm. We show that BRS also outperforms paranoid in Chinese Checkers and Focus. In Rolit, BRS is on equal footing with paranoid. We conclude that BRS is a promising search method for deterministic multi-player games with perfect information.**

***Index Terms*—Multi-Player Games; Best-Reply Search; Max^n; Paranoid; Chinese Checkers; Focus; Rolit**

I. Introduction

In deterministic two-player games with perfect information, the majority of research has focused on the minimax algorithm [1]. For deterministic perfect-information multi-player games, the choice of search algorithm is not as straightforward. The two main algorithms are max^n [2] and paranoid [3], each approaching the problem from a different angle. Max^n assumes that every player tries to maximize the own score, while paranoid assumes that all opponents form a coalition against the root player. Max^n and the paranoid algorithm may have conceptual drawbacks. Just changing the tie-breaking rule for the max^n algorithm can have arbitrary results for the value of the root [4]. When performing a deep search with the paranoid algorithm, the other players may dominate the root player [5].

This article proposes Best-Reply Search (BRS). This search algorithm tries to overcome the problems of the max^n and paranoid algorithms. For multi-player games, we assume that not every opponent is trying to minimize the root player's score. Instead, only one opponent is minimizing the root player's score. BRS chooses which opponent is allowed to play a counter move at a so-called MIN node in the search tree. The selected opponent is the one that has the strongest counter move against the root player. The other players have to pass their turn. By searching in this way, a significant lookahead can be achieved even with many opponents. Furthermore, the playing style is less cautious compared to the paranoid algorithm. In this article, we test BRS solely from a deterministic perfect-information standpoint. We apply BRS in three domains, Chinese Checkers, Focus, and Rolit™.

The outline of the article is as follows. First, the max^n and the paranoid algorithms are discussed in Section II. Next, BRS is introduced in Section III. Thereafter, Section IV describes the test domains, the games Chinese Checkers, Focus and Rolit. Section V presents the experiments. Finally, Section VI gives the conclusions and an outlook on future research.

Maarten Schadd and Mark Winands are members of the Games and AI Group, Department of Knowledge Engineering, Faculty of Humanities and Sciences, Maastricht University, Maastricht, The Netherlands; E-mail: {maarten.schadd, m.winands}@maastrichtuniversity.nl

II. Search Algorithms for Multi-Player Games

This section discusses two well-understood search algorithms for deterministic multi-player turn-taking games with perfect information, the max^n [2] and the paranoid algorithm [3]. We first introduce the max^n algorithm in Subsection II-A and thereafter discuss the paranoid algorithm in Subsection II-B.

A. Max^n

The max^n algorithm [2] can be used in games with any number of players. At a leaf node, an n-tuple is generated, where n is the number of players and every entry corresponds to the score a player receives. At internal nodes, a player chooses the child with the highest score for that player. An example max^n tree is depicted in Figure 1.

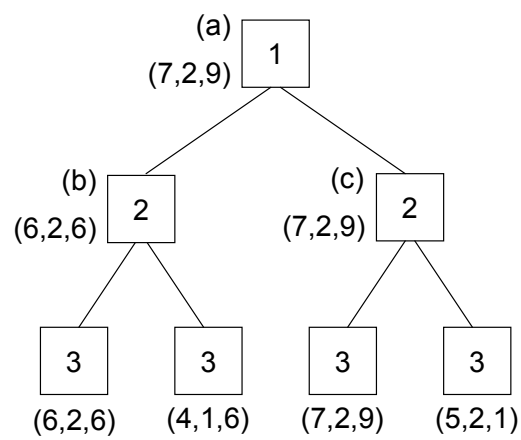


Fig. 1. An example max^n tree.

Nodes (a), (b) and (c) are internal nodes. In Node (a), Player 1 is to move. In Nodes (b) and (c), Player 2 is to move. Player 2 has the choice between (6,2,6) and (4,1,6) in Node (b) and chooses (6,2,6) to maximize the own score. In Node (c), both options are equally good for Player 2. In this example the left child is chosen and Node (c) gets value (7,2,9). Player 1 now prefers Node (c) above Node (b), because it gives a higher score for Player 1.

In multi-player games, there may exist multiple equilibrium points. It has been proven that max^n computes one equilibrium point [2]. Furthermore, if the tie-breaking rule is altered, it may arbitrarily affect the max^n value of the tree [4]. We demonstrate this by changing the tie-breaking rule in Figure 1. Instead of choosing the left-most child in case of a tie, now

the policy is used that the child which has the lowest value for the root player is chosen. In this case, Node (c) has value (5,2,1). This implies that the Player 1 prefers Node (b) over (c) and Node (a) has value (6,2,6) instead of (7,2,9).

The weakness of max^n is twofold. (1) Compared to $\alpha\beta$ search, less pruning is possible, resulting in a limited lookahead [6]. Pruning is possible in max^n if there is a lower bound on each player's score and an upper bound on the sum of scores. The basic pruning is then shallow [7]. Sturtevant [8] showed that "less-shallow" pruning is feasible, up to n ply deep. He proposed two techniques, *last-branch pruning* and *speculative pruning*. Last-branch pruning additionally prunes when intermediate players between the first and last player are all on their last branch of their search. Speculative pruning is identical to last-branch pruning, except that it does not wait until intermediate players are on their last branch. Instead, it prunes speculatively, re-searching if needed. Deep pruning in max^n is, however, not possible [7]. (2) Furthermore, the underlying assumption of max^n may be unrealistic. The max^n algorithm assumes that there is no coalition forming by the opponents. The result may be that max^n is too optimistic. To make max^n somewhat more cautious, the tie-breaking rule which assumes the worst case for the root player is used in this article. To further increase the cautiousness of max^n, a "paranoid" evaluation function (i.e., it assumes that all opponents have formed a coalition) may be considered. Several other variations of the max^n algorithm exist, which try to overcome both weaknesses. Careful max^n [9] uses a weighted-average update rule to model uncertainty if the opponent has multiple good moves. The comixer algorithm [9] considers possible coalitions against the strongest player at every node of the search tree. This may be a correct assumption if a player is ahead, but might lead to weak play if no such coalition exists. For handling imperfect opponent models, two variations have been introduced, soft-max^n [10] and prob-max^n [11].

B. Paranoid

The paranoid algorithm [3] reduces the multi-player game to a two-player game by making the "paranoid" assumption. The algorithm assumes that all opponents have formed a coalition against the root player. By doing so, regular $\alpha\beta$ pruning is possible. This leads to a larger search depth. Figure 2 depicts an example of the paranoid algorithm. It is the same tree as in Figure 1, but now the leaf nodes are evaluated in a paranoid way. Here, the sum of the evaluation scores of Player 2 and 3 are subtracted from the evaluation score of Player 1 [6]. A second possibility is that Players 2 and 3 ignore their own score, and only minimize Player 1's score [4]. In this article we apply the first approach. In Node (b), the right child is preferred with value -3. After finding -4 as value of the first child of Node (c), all the remaining children can be safely pruned according to the standard $\alpha\beta$ rule. The root node (a) receives value -3.

In the best case, $O(b^{d/2})$ nodes are expanded for two-player games [1], where b is the average branching factor and d the search depth. Sturtevant and Korf [4] showed that the paranoid algorithm expands $O\left(b^{d\times(n-1)/n}\right)$ nodes in the best case for multi-player games, which is a generalization of the best case for two-player games. Paranoid may outperform max^n due to the larger lookahead (e.g., for Chinese Checkers or Hearts) [6].

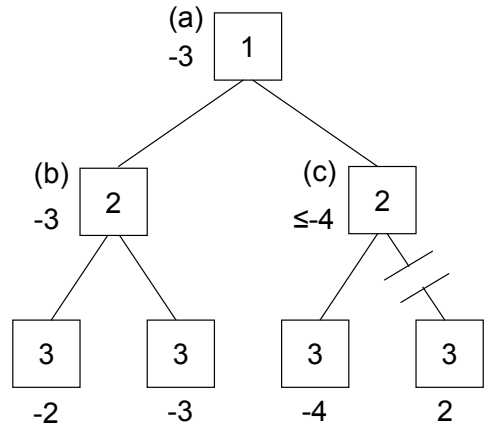


Fig. 2. An example paranoid tree.

Because of the unrealistic paranoid assumption, suboptimal play can occur [3]. Furthermore, if an infinite amount of time would be available, the root player might assume that all moves are losing, leading to poor play. For the game of Rolit^{TM}, Saito and Winands [5] showed that for three players on the 6×6 board, the first and second player cannot gain any points under the paranoid assumption (and the third player only 1 point because he places the last stone). Usually, it is not possible to win when all opponents form a coalition. As a general observation, the deeper the search goes, the more pessimistic the value of the root becomes.

III. Best-Reply Search

Sturtevant proposed a table-based evaluation function for Chinese Checkers which assumes solitary play [6]. Although this assumption is unrealistic, this evaluation function proved to work well. Inspired by this result, we investigate in this section whether this idea may be transferred from the evaluation function to the search tree. We propose Best-Reply Search (BRS) to overcome some weaknesses of the max^n and paranoid algorithms.

Subsection III-A presents the underlying idea behind BRS. The pseudo code of BRS is given in Subsection III-B. In Subsection III-C the best-case analysis of BRS is given. Finally, Subsection III-D discusses strengths and weaknesses of BRS.

A. Idea

Traditional search methods allow every player to make a move, resulting in large search trees. In BRS, not every opponent is allowed to make a move. Only the opponent with the strongest move against the root player may move. At a MIN node, all moves for all opponents are searched. It means that an opponent is allowed to make a move even if it is not its turn. At the following MAX node, it is again the root player's turn. BRS achieves long-term planning because more

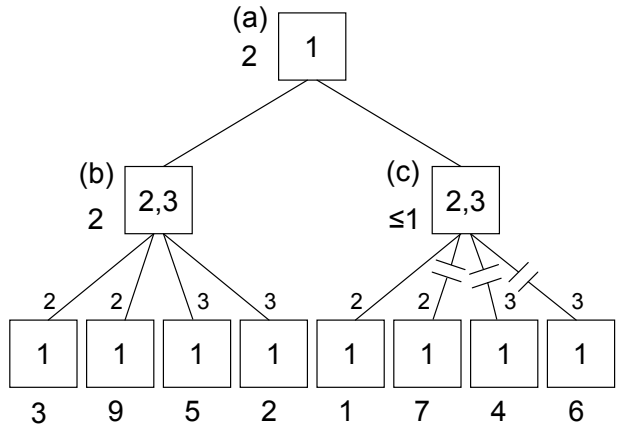


Fig. 3. An example BRS tree.

MAX nodes are visited along the search path while being at the same time less paranoid. When iterative deepening is applied in BRS, a MIN node always contains all players. When comparing this to the standard search depth of the paranoid algorithm, the search depth is increased irregularly. For instance with 4 players, the search depth of BRS is increased to depth 1, 4, 5, 8, 9, etc., with respect to the standard depth. An example of BRS is depicted in Figure 3.

Nodes (b) and (c) are labeled '2,3', which represents that one of the opponents (i.e., Player 2 or 3) is allowed to make a counter move. The labels near the edges underneath Nodes (b) and (c) indicate which player's move is played. At the next ply, it is again Player 1's turn to play. Node (b) is assigned value 2, because Player 3 has the strongest counter move. When searching the first child of Node (c), which is a move by Player 2, a regular $\alpha\beta$ pruning occurs. In this case, the remaining moves of Player 2 and all moves of Player 3 are pruned.

B. Pseudo Code

Algorithm 1 shows the pseudo code for BRS. The first change to the standard $\alpha\beta$ algorithm for the NegaMax framework is shown in lines 7−15. If the current node is a MAX node, the moves are generated as usual (Line 8). If it is a MIN node the moves for all opponents are generated (Lines 11−13). Before performing the traversal of all moves (Line 17), the type of node should have been altered at every ply (Lines 9 and 14).

C. Best-Case Analysis of BRS

If pruning is not feasible, max^n has to examine the complete search tree. With average branching factor b and search depth d, max^n searches $O\left(b^d\right)$ nodes. Sturtevant and Korf [4] showed that the paranoid algorithm explores $O\left(b^{d\times(n-1)/n}\right)$ nodes in the best case, where n is the number of players. Analogous to their proof, we can prove the best case of BRS.

Theorem. *Best-Reply Search explores in the best case* $O\left((b\times(n-1))^{\lceil\frac{2\times d}{n}\rceil/2}\right)$ *nodes.*

Algorithm 1 Best-Reply Search.

```
1: BRS(alpha, beta, depth, turn)
2:
3: if depth ≤ 0 then
4:    return eval()
5: end if
6:
7: if turn == MAX then
8:    Moves = GenerateMoves(MaxPlayer);
9:    turn = MIN;
10: else
11:    for all Opponents o do
12:       Moves += GenerateMoves(o);
13:    end for
14:    turn = MAX;
15: end if
16:
17: for all Moves m do
18:    doMove(m);
19:    v = −BRS(−beta, −alpha, depth−1, turn);
20:    undoMove(m);
21:
22:    if v ≥ beta then
23:       return v;
24:    end if
25:    alpha = max(alpha, v);
26: end for
27:
28: return alpha;
```

Proof: Assume a uniform tree is searched until depth d. In BRS, this search depth is reduced to $\lceil\frac{2\times d}{n}\rceil$ because the layers of n successive players is reduced to 2 layers. The branching factor b is increased to $b\times(n-1)$ at MIN nodes, assuming that the opponent moves do not interact with each other. To calculate the minimum number of nodes that have to be examined within the game tree, we need a strategy for the MAX and MIN player. For finding a strategy for the MAX player, 1 move has to be searched at a MAX node, and $b\times(n-1)$ moves at a MIN node, resulting in $(b\times(n-1))^{\lceil\frac{2\times d}{n}\rceil/2}$ nodes. For finding a strategy for the MIN player, the collection of all opponents, 1 move has to be searched at a MIN node and b moves at a MAX node, resulting in $b^{\lceil\frac{2\times d}{n}\rceil/2}$ nodes. Therefore, the total number of nodes by both the MAX and MIN player is $(b\times(n-1))^{\lceil\frac{2\times d}{n}\rceil/2}+b^{\lceil\frac{2\times d}{n}\rceil/2}$ nodes. Thus, BRS explores in the best case $O\left((b\times(n-1))^{\lceil\frac{2\times d}{n}\rceil/2}\right)$ nodes. ■

We remark that for two players, the best case of BRS is identical to the best case of $\alpha\beta$, which is $O\left(b^{d/2}\right)$.

D. Strengths and Weaknesses of BRS

We point out two advantages of BRS over the max^n and paranoid algorithms. (1) More MAX nodes are visited along the search path, leading to more long-term planning. (2) It softens the unrealistic max^n and paranoid assumptions. Max^n assumes that there are no coalitions, while paranoid assumes

that all opponents form a coalition against the root player. An additional advantage over max^n is that BRS may be able to prune parts of the tree.

We point out two drawbacks of BRS as well. (1) Not all players are allowed to make a move, leading to illegal positions. (2) Opponent moves which are beneficial for the root player might not be considered.

For trick-based card games, such as Hearts and Spades, BRS is not an appropriate method. The first problem is that the first player to play a card in a trick determines which suit is played. If BRS would be applied here, a suit may be played which the first player does not have, creating an illegal position that is considerably different from a legal position. The second problem occurs when not all opponents play a card during a trick. This causes that players have a different number of cards in their hands, and it is not defined what happens at the end of the game. A third problem in the game of Hearts is that playing the Ace of Hearts card may result in only gaining 2 points instead of 4 when applying BRS. Problems such as these make BRS not applicable to trick-based card games.

IV. Test Domain

To test whether BRS works well, we have decided to use three games, Chinese Checkers, Focus, and Rolit^{TM}. Chinese Checkers is a race game and the rules are given in Subsection IV-A. Focus is a material-based game and is explained in Subsection IV-B. In Subsection IV-C the rules of the territorial-based game Rolit are given. Rolit is the multi-player version of Othello.[1] The game engines are described at the end of each subsection.

A. Chinese Checkers

Chinese Checkers is a board game that can be played by two to six players. It 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 common board contains 121 fields, where each player starts with 10 pieces. We decided to play on a slightly smaller board [12] (see Figure 4). In this version, each player plays with 6 pieces. The advantage of a smaller board is that it allows us to use a strong evaluation function [6].

The goal of each player is to move the own pieces to the own base at the other side of the board. Pieces may move to one of the adjacent squares or they may jump over another piece to an empty field. A player may also make multiple jumps with one piece in one turn. It is possible to create a setup that allows pieces to jump over a large distance. The first player who manages to fill the home base wins the game. To avoid blocking behavior, the player wins the game when the home base is filled *and* the player owns at least one of the pieces in the home base.

Engine: To evaluate a board position, we use a lookup table which stores the number of moves a single player would require to finish the game [13]. This number does not take into account opponent pieces, which results in erroneous evaluation

[1]Also known as Reversi.

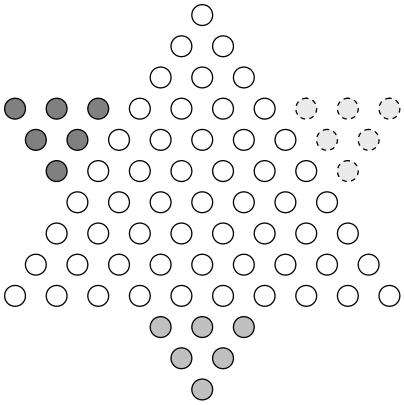


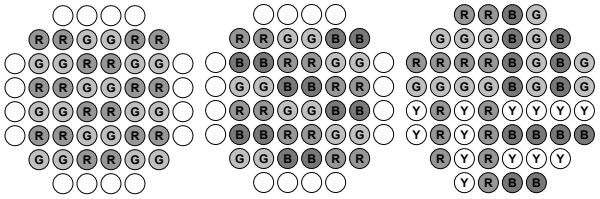
Fig. 4. A three-player Chinese Checkers board [12].

in the midgame. In the endgame, this lookup table allows perfect play. The value found in the table was multiplied by 1,000. Additionally, a random factor of 5 points was included to prevent games from being repeated (due to deterministic play) in the experiments.

The moves are ordered statically such that moves which approach the home base the most are investigated first (e.g., long jumping moves towards the home base).

B. Focus

Focus is an abstract multi-player strategy board game, invented in 1963 by Sid Sackson [14]. This game has also been released under the name Domination. Focus is played on an 8×8 board where in each corner 3 squares are removed. It can be played by two, three or four players. Each player starts with a number of pieces on the board. In Figure 5, the initial board positions for the two-, three- and four-player variants are given. The letters R, G, B, and Y correspond to the piece colors of the game: red, green, blue, and yellow, respectively.



(a) Two players (b) Three players (c) Four players

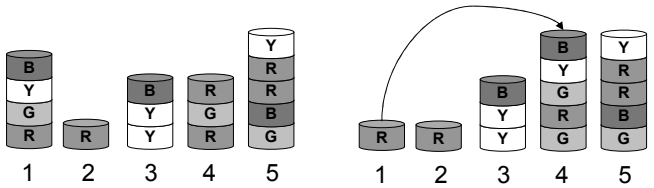
Fig. 5. Setups for Focus [15].

In contrast to many other games, pieces in Focus may be stacked on top of each other. Each turn a player may move a stack, which contains one or more pieces, orthogonally as many squares as the stack is tall. The player may only move a stack of pieces if a piece of their color is on top of the stack. Players are also allowed to split stacks in two smaller stacks. If they decide to do so, then they only move the upper stack as many squares as the number of pieces in that stack.

If a stack is moved onto another stack, then 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 the own pieces, the player may later choose to place an own piece back on the board, instead of moving a stack.

An example move is depicted in Figure 6. Here, Blue chooses to move three pieces of Stack 1, three positions to the right. By this move, the control of Stack 1 is transferred to Red, which owns the highest piece of Stack 1 after Blue has moved. Stack 4 would contain 6 pieces after Blue has moved, indicating that a capture shall take place. Only the bottom red piece is captured because 5 pieces are allowed per stack.



(a) Blue to move. (b) Blue has captured a piece of Red.

Fig. 6. Example move for Focus.

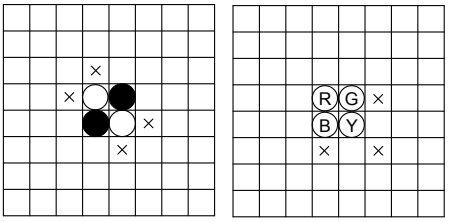
There exist two variations of the game, each with a different winning condition. In the standard version of the game, a player has won if all other players cannot perform a legal move. However, these games may take a long time to finish. Therefore, in this article the shortened version of the game is used. In this version, a player has won if either a certain number of pieces or a certain number of pieces from each opponent have been captured. In the two-player variant, a player wins if at least 6 pieces from the opponent are captured. In the three-player variant, a player has won if either at least 10 pieces in total or at least 3 pieces from each opponent are captured. Finally, in the four-player variant, the goal is to either capture 10 opponent pieces or at least 2 pieces from each opponent. If the game is not decided after 300 moves, it is scored as a draw.

Engine: The evaluation function consists of two parts. (1) The first term is the minimum number of pieces needed to finish the game for either finish condition. This number is multiplied by 1,000. (2) The second term is the position of pieces in a stack. The higher the piece on a stack of pieces, the more points it is worth. Being on top of a stack gives control of the stack, and this is especially valuable if the stack is tall. Furthermore, the higher a piece, the more difficult it is to capture it. For every own piece its height is squared and added to the total score. A small random factor of 5 points is included to prevent repetition.

The static move ordering consists of two parts. (1) Moves which involve a large number of pieces (pieces of the moved stack plus pieces of the target stack). (2) Moves which increase the number of stacks a player controls. The first term is the dominant one.

C. Rolit

Rolit™ is a multi-player variant of the well-known game Othello. Therefore, we start with a description of this game before turning to Rolit. Othello is a deterministic two-player game with perfect information played on an 8×8 board. The players are called Black and White and their objective is to maximize their number of pieces on the board. The initial position is shown in Figure 7. A player may only place a piece as part of a flipping move and a player has to pass if no such move is available. A *flipping move* places a piece at the end of a flipping line. A *flipping line* is a straight line of squares on the board such that four conditions are met: (1) The squares on the line have to be all vertically, all horizontally, or all diagonally connected. (2) All squares on the line are occupied. (3) Both ends of the line must contain a piece of the player to move. (4) All pieces between are of the opponent. If a flipping move is played, the opponent pieces between the ends of the flipping line are flipped over to its own color.[2] If a move enables more than one flipping line, all flipping lines are executed. The game ends when the board is completely occupied or one player has no pieces left. The player with more pieces on the board wins the game.



(a) Othello (b) Rolit

Fig. 7. Setups for Othello and Rolit. Legal moves are marked with ×.

Rolit is the multi-player generalization of Othello and has only minor differences in the rules. It may be played with two, three and four players, called Red, Green, Yellow and Blue. The initial position, which is identical for the two-, three- and four-player variant, is depicted in Figure 7. This implies that when playing with two or three players, there initially exist pieces on the board which do not belong to any of the players. Red starts the game and immediately has the opportunity to eliminate an opponent. In order to allow a player who has no piece left on the board to come back into the game, the rules are changed compared to Othello. When no flipping move is available (either because a player is eliminated or because there are just no flipping moves available) a player is allowed to place a piece on an empty field. The empty field has to be horizontally, vertically, or diagonally adjacent to a piece on the board. The game is scored analogously to Othello.

Engine: Our evaluation function is pattern based, inspired by the work of Buro [16]. Almost 100,000 games of the WTHOR database[3] were analyzed on 10 different patterns for 12 stages of the game. These patterns are the orthogonal lines of size 8 (4 lines due to symmetry), the entire diagonals of at least size 4 (5 diagonals due to symmetry) and the 2×4 corner region [16]. These add up to a total number of 511,272 patterns. For each pattern the average score at the end of the game was computed. All the pattern scores are averaged for

[2]The pieces are black one side, and white on the other.

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

evaluating a position, and a random factor of 0.1 point is added. These patterns are based on two players, while we need to evaluate positions for three and four players in Rolit. To bridge the gap, we assume that all opponent pieces have the same color when looking up pattern values in the database. It would be more accurate to create a pattern database for three and four players, but we abandoned this idea due to the combinatorial explosion and the unavailability of a Rolit database. The pattern-based approach, though, was superior to our original evaluation function that employed standard Othello features.

Our static move ordering prefers moves on squares which are known to be good. For example, the most preferred ones are the corners. The squares adjacent to the corners are the least preferred.

V. Experiments

In this section, the experiments with BRS are presented for the games of Chinese Checkers, Focus, and Rolit. The performance of BRS is tested with three different time settings (250, 1000, 5000 ms) and with three, four and/or six players against one type of opponent (max^n or paranoid). In a three-player game there are $2^3 = 8$ different player-type assignments. Games where only one type of algorithm is playing are not interesting, leaving 6 ways to assign player types. For four players, there are $2^4 - 2 = 14$ assignments, and for six players, there are $2^6 - 2 = 62$ assignments [4]. Each assignment is played multiple times until at least 1,000 games are reached and each assignment was played equally often. The random factor in each evaluation function prevented board repetition. All experiments were performed on an AMD64 2.4 GHz computer.

The following techniques were used if not mentioned otherwise. All algorithms used Two-Deep Transposition Tables [17], [18] and iterative deepening until the available time was depleted. Furthermore, Paranoid and BRS used the History Heuristic [19] and Killer Moves [20]. In all columns labeled 'Win Ratio' a 95% confidence interval is applied.

Advanced pruning techniques were not used during this research. For max^n, speculative pruning is able to significantly increase the playing strength [6], [8]. However, the tight upper bounds that are required are not present for our evaluation function. For the paranoid algorithm and BRS, well-known forward-pruning techniques could be used as well. These include the null-move heuristic [21], [22], Multi-Cut [23] and ProbCut [24]. In order to successfully apply these forward-pruning techniques, the preconditions and parameters need to be tuned rather well. We did not enable any of these techniques because the effect of each of them is domain dependent.

In Subsection V-A, we present the experiments to validate the implementation. The average search depth that the different algorithms are able to achieve is shown in Subsection V-B. Subsection V-C presents the performance of BRS against max^n. In Subsection V-D, the experiments of BRS against paranoid are discussed. Subsection V-E shows experiments with three players, where there is 1 BRS player, 1 paranoid player and 1 max^n player.

TABLE I
Winning statistics for Paranoid vs. Max^n for Chinese Checkers with 250 ms per move.

Two-player	Paranoid	Max^n	Win Ratio
Basic Settings	955	45	95.5% ± 1.3%
Advanced Settings	917	83	91.7% ± 1.7%
Three-player	**Paranoid**	**Max^n**	**Win Ratio**
Basic Settings	350	250	58.3% ± 3.9%
Advanced Settings	474	126	79.0% ± 3.3%

TABLE II
Winning statistics for Paranoid vs. BRS for two-players with 250 ms per move.

Chinese Checkers	Paranoid	BRS	Win Ratio
Basic Settings	527	473	52.7% ± 3.1%
Advanced Settings	475	525	47.5% ± 3.1%
Focus	**Paranoid**	**BRS**	**Win Ratio**
Basic Settings	529	471	52.9% ± 3.1%
Advanced Settings	506	494	50.6% ± 3.1%
Rolit	**Paranoid**	**BRS**	**Win Ratio**
Basic Settings	476.5	523.5	47.7% ± 3.1%
Advanced Settings	487.5	512.5	48.8% ± 3.1%

A. Validation

To check whether the implementation of max^n and paranoid is similar to Sturtevant [4], we reconstructed his experiments. Paranoid played 600 three-player Chinese Checkers games against max^n using 250 ms (approximately 250k nodes). The results are presented in the lower part of Table I. With the basic settings, max^n and paranoid did not use any additional techniques. In the advanced settings, max^n and paranoid used Transposition Tables, and paranoid was furthermore allowed to use the History Heuristic and Killer Moves. The results of the basic setting confirm the results by Sturtevant [4]. He reports a win ratio of 60.6% for his paranoid program. Out of curiosity, we also performed this experiment for the two-player version of Chinese Checkers. The results of 1,000 games are given in the upper part of Table I. Here it is clear that paranoid, which is now a regular $\alpha\beta$ search, outperforms max^n due to larger lookahead. The reason for the advanced setting winning fewer games in the two-player variant might be that it plays more defensive and takes less risk.

If BRS is applied to a two-player game, it should behave identical to the paranoid algorithm. For verification purposes, paranoid was matched against BRS for two-player Chinese Checkers, Focus and Rolit. There may be a slight overhead in BRS due to the move generation, but a win ratio of 50% should be expected. 1,000 games were played for each setting and game. Table II shows that for the two-player version of all three games, paranoid and BRS are equally strong.

B. Average Search Depth

The average search depth, which max^n, paranoid and BRS can achieve in Chinese Checkers, Focus and Rolit with different time settings and different number of players, is shown in Table III.

Here we see that in all games paranoid is always able to search deeper than max^n. In Chinese Checkers, paranoid performs close to the best case, or even better (e.g., the best-case depth for four players with 5 seconds thinking time is

TABLE III
AVERAGE SEARCH DEPTH.

Chinese Checkers				
Players	Time (ms)	Max^n	Paranoid	BRS
3	250	3.0	4.9	4.7
3	1,000	3.2	5.0	5.0
3	5,000	4.0	5.7	6.0
4	250	3.0	4.8	4.5
4	1,000	3.6	5.1	5.0
4	5,000	4.0	5.9	5.5
6	250	3.0	3.9	3.9
6	1,000	3.5	4.2	4.9
6	5,000	4.0	4.9	5.0
Focus				
Players	Time (ms)	Max^n	Paranoid	BRS
3	250	3.0	4.5	4.6
3	1,000	3.2	5.0	5.0
3	5,000	3.9	5.5	5.8
4	250	3.0	4.1	4.4
4	1,000	3.3	4.9	5.1
4	5,000	4.0	5.5	5.6
Rolit				
Players	Time (ms)	Max^n	Paranoid	BRS
3	250	3.3	5.0	5.1
3	1,000	4.1	5.7	5.9
3	5,000	4.6	7.0	7.3
4	250	3.4	5.0	4.8
4	1,000	4.2	5.9	5.4
4	5,000	4.9	6.6	6.7

approximately $4.0 \times \frac{4}{3} \approx 5.3$). This performance is due to the effective move ordering. For instance, 1.1 moves are searched in CUT nodes [25] on average for three players, and only 1.05 for four players. Moreover, there are two explanations how it is possible to perform better than the theoretical best case, $O(b^{d(n-1)/n})$. (1) Because only complete plies are counted, the data is coarse-grained. (2) The paranoid algorithm can perform better than $O(b^{d(n-1)/n})$ if the domain-dependent move ordering prefers slim subtrees above large subtrees, taking advantage of the non-uniform nature of the game tree (i.e., a variable branching factor and search depth [26]). For Chinese Checkers these are moves which enter the own goal area, because once entered, pieces are not allowed to leave anymore. In the games of Focus and Rolit, paranoid is close to the best case.

For all games BRS achieves a similar search depth as paranoid for every setting. However, these numbers are not strictly comparable due to the different search approaches. One conclusion we may draw is that BRS visits along the search path at least as many MAX nodes as paranoid. For example, for six players and 5 seconds thinking time, paranoid and BRS search approximately 5 ply. Due to the large number of players, paranoid only visits 1 MAX node, which is the root node. BRS is able to visit 3 MAX nodes in this case.

C. BRS against Max^n

Table IV shows the winning performance of BRS against max^n for Chinese Checkers, Focus, and Rolit. In Chinese Checkers there are no draws possible. For Focus, draws are counted as $\frac{1}{3}$ or $\frac{1}{4}$ point for the three- and four-player variants, respectively. In Rolit, a draw may be shared between a subset of players. For example, if there are 2 BRS players and 1 paranoid player, both BRS players may share the highest score. The winning players receive the corresponding fraction of a point.

TABLE IV
WINNING STATISTICS FOR BRS VS. MAX^n.

Chinese Checkers				
Players	Time (ms)	BRS	Max^n	Win Ratio
3	250	818	184	81.6% ± 2.4%
3	1,000	882	120	88.0% ± 2.0%
3	5,000	871	131	86.9% ± 2.1%
4	250	730	278	72.4% ± 2.8%
4	1,000	857	151	85.0% ± 2.2%
4	5,000	846	162	83.9% ± 2.3%
6	250	735	319	72.9% ± 2.7%
6	1,000	793	261	78.7% ± 2.5%
6	5,000	832	222	82.5% ± 2.3%
Focus				
Players	Time (ms)	BRS	Max^n	Win Ratio
3	250	940.7	61.3	93.9% ± 1.5%
3	1,000	952.0	50.0	95.0% ± 1.3%
3	5,000	868.0	134.0	86.6% ± 2.1%
4	250	841.0	167.0	83.4% ± 2.3%
4	1,000	823.3	184.8	81.7% ± 2.4%
4	5,000	818.5	189.5	81.2% ± 2.4%
Rolit				
Players	Time (ms)	BRS	Max^n	Win Ratio
3	250	637.5	364.5	63.6% ± 3.0%
3	1,000	661.5	340.5	66.0% ± 2.9%
3	5,000	650.5	351.5	64.9% ± 3.0%
4	250	690.5	317.5	68.5% ± 2.9%
4	1,000	696.9	311.1	69.1% ± 2.9%
4	5,000	664.5	343.5	65.9% ± 2.9%

For Chinese Checkers we see that max^n is outperformed by BRS. In the worst case, a win ratio of 72.4% ± 2.8% is still achieved. In most cases, the performance is approximately 80%. In the best case, a win ratio of 88.0% ± 2.0% is achieved with 1,000 ms per move. We observe that in general, BRS gets stronger with more thinking time. In the material-based game Focus, we observe that BRS is outperforming max^n easily. In the worst case, a win ratio of 81.2% ± 2.4% is still achieved. The max^n algorithm plays the strongest in Rolit. However, it is still outperformed by BRS with a win ratio between 65% and 70%.

D. BRS against Paranoid

Table V shows the performance of BRS against paranoid in Chinese Checkers, Focus, and Rolit. Draws are addressed in a similar manner as in Table IV. For the 5,000 ms experiment of Rolit we played the double number of games to reach statistical significance (2,004 games for three players and 2,016 for four players).

For three-player Chinese Checkers, BRS wins above 70% of the games against paranoid. For four and six players, a win ratio of approximately 60% is achieved. In this game, we do not observe a clear performance trend when increasing the thinking time. As expected, paranoid performs better against BRS than max^n did in Chinese Checkers.

In the experiments of BRS against paranoid in Focus, we see that a relatively stable win ratio between 58.0% and 68.2% is achieved. We again may conclude that BRS plays stronger than paranoid.

TABLE V
WINNING STATISTICS FOR BRS VS. PARANOID.

Chinese Checkers				
Players	Time (ms)	BRS	Paranoid	Win Ratio
3	250	713	289	71.2% ± 2.8%
3	1,000	763	239	76.1% ± 2.6%
3	5,000	722	280	72.1% ± 2.8%
4	250	594	414	58.9% ± 3.0%
4	1,000	593	415	58.8% ± 3.0%
4	5,000	572	436	56.7% ± 3.1%
6	250	610	444	60.5% ± 3.0%
6	1,000	611	443	60.6% ± 3.0%
6	5,000	596	458	59.1% ± 3.0%
Focus				
Players	Time (ms)	BRS	Paranoid	Win Ratio
3	250	581.3	420.7	58.0% ± 3.1%
3	1,000	683.7	318.3	68.2% ± 2.9%
3	5,000	673.0	329.0	67.2% ± 2.9%
4	250	654.3	353.8	64.9% ± 2.9%
4	1,000	659.3	348.8	65.4% ± 2.9%
4	5,000	609.5	398.5	60.5% ± 3.0%
Rolit				
Players	Time (ms)	BRS	Paranoid	Win Ratio
3	250	293.5	708.5	29.3% ± 2.8%
3	1,000	580.0	422.0	57.9% ± 3.1%
3	5,000	992.5	1011.5	49.5% ± 2.2%
4	250	490.0	518.0	48.6% ± 3.1%
4	1,000	580.2	427.8	57.6% ± 3.1%
4	5,000	1055.5	960.5	52.4% ± 2.2%

In Rolit, we see that BRS is weaker in a short time setting for three and four players when playing against paranoid. A possible reason for this is that due to the short thinking time, the paranoid player is not too paranoid yet. With a deeper search, paranoid may become too careful. With 1,000 ms per move, BRS wins about 57% of the games. With 5,000 ms thinking time, the performance of BRS is dropping again to 49.5% ± 2.2% for three players and 52.4% ± 2.2% for four players. A possible explanation for this is that a move in Rolit changes the board significantly and the illegal states have a large influence on this time setting.

E. BRS vs. Paranoid vs. Max^n

When competing against one type of opponent, one can win the game if the opponent's weakness is discovered. When playing against different kinds of opponents, the game dynamics change. Every opponent has different weak spots which have to be exploited at the same time. Therefore, we matched BRS, paranoid, and max^n against each other in the three-player variant of Chinese Checkers, Focus, and Rolit. The results are shown in Table VI. Draws are addressed in a similar manner as before. If all algorithms would be equally strong, a win ratio of 33.3% would be expected.

The results show that max^n clearly is the weakest algorithm of the three, having its best performance in Rolit. For Chinese Checkers and Focus, BRS is clearly the best algorithm. In Rolit, the performance of BRS and paranoid are comparable with 250 and 1,000 ms per move. For 5,000 ms, paranoid is the best algorithm.

VI. CONCLUSIONS AND FUTURE RESEARCH

In this article we proposed a new search algorithm called Best-Reply Search (BRS) for deterministic multi-player games with perfect information. The algorithm allows only one opponent to play a counter move. This opponent is the one with the strongest move against the root player. The other players have to pass their turn. Using this approach, more turns of the root player can be searched, resulting in long-term planning. At the same time, some sort of cautiousness is preserved by searching the strongest opponent move.

TABLE VI
TOURNAMENT RESULTS.

Chinese Checkers				
Time (ms)	BRS	Paranoid	Max^n	BRS Win ratio
250	582	333	87	58.1% ± 3.1%
1,000	600	324	78	59.9% ± 3.0%
5,000	661	193	163	66.0% ± 2.9%
Focus				
Time (ms)	BRS	Paranoid	Max^n	BRS Win ratio
250	507.7	431.7	62.7	50.7% ± 3.1%
1,000	619.7	319.7	62.7	61.8% ± 3.0%
5,000	616.7	265.7	119.7	61.5% ± 3.0%
Rolit				
Time (ms)	BRS	Paranoid	Max^n	BRS Win ratio
250	416.0	393.5	192.5	41.5% ± 3.1%
1,000	399.0	417.5	185.5	39.8% ± 3.0%
5,000	372.0	453.0	177.0	37.1% ± 3.0%

The first conclusion we may draw is that BRS is able to significantly outperform max^n in Chinese Checkers, Focus, and Rolit. In Chinese Checkers, BRS wins between 72% and 88% of the games. For Focus, BRS wins more than 80% of the games. An impressive 95% win ratio for three players with 1,000 ms per move was achieved. In Rolit, BRS wins approximately 65% of all games.

Our second conclusion is that against paranoid, BRS is significantly stronger in Chinese Checkers and Focus. In these games BRS won approximately 60% across all experiments. In Rolit, BRS did not perform well with 250 ms of thinking time. However, BRS was stronger than paranoid with 1,000 ms, and on equal footing with 5,000 ms of thinking time. A possible reason why BRS is not outperforming paranoid in Rolit is that the board changes significantly with every move.

Our third conclusion is that when playing different kind of opponents, BRS is the strongest algorithm in Chinese Checkers and Focus. In Rolit, BRS was somewhat behind paranoid.

The fourth conclusion we may draw is that increasing the search time generally does not have a negative effect on the performance of BRS (in Chinese Checkers and Focus). This implies that searching illegal positions, which are generated by forcing opponents to pass, does not have a large influence. The possible negative effect is outbalanced by the larger lookahead. In spite of this negative effect being visible in Rolit, BRS was still competitive with paranoid.

The first direction of future research is the application of Monte-Carlo algorithms. Over the past years, Monte-Carlo Tree Search (MCTS) [27], [28] has become increasingly popular for letting computers play games. It has been applied successfully in quite some two-player games (e.g., Go [27], [29], [30], Amazons [31], [32], Lines of Action [33], Hex [34] and Kriegspiel [35]). Moreover, Cazenave [36] applied MCTS successfully for multi-player Go. Sturtevant [12] showed that MCTS outperforms max^n and paranoid in Chinese Checkers,

when given enough time. An interesting experiment would be to compare the playing strength of BRS against an MCTS program in Chinese Checkers and Focus. Furthermore, the BRS principle can be used in MCTS programs as well. This might lead to an improvement in playing strength.

The second direction of future research is variable-depth search [37]. Forward-pruning techniques prune unpromising branches in advance with only a small risk. The most prominent techniques are null moves [21], [22], ProbCut [24] and Multi-Cut [23]. A prerequisite of successfully applying these techniques is to have a strong evaluation function. This function should be a good predictor and not suffer from the horizon or odd-even effects. Using these techniques, an even larger lookahead would be possible. Because of the direct succession of MAX and MIN nodes in BRS, we expect that forward-pruning techniques are more effective in BRS than in paranoid. This could give BRS an edge over paranoid.

The third to seventh directions for future research are as follows. (3) Searching illegal positions is not necessary for BRS. Instead, the opponents who are not selected for the counter move could be allowed to play the first move from the static move ordering. This may make BRS applicable to the game of Hearts. (4) BRS should be tested for more domains, such as four-player chess [9]. (5) Because lookahead is important, it would be interesting to test how an algorithm performs which only searches moves by the root player (making it a one-player game, as in the evaluation function for Chinese Checkers by Sturtevant [6]). (6) The MP-Mixed algorithm chooses a search method based on the current situation of the game [38]. BRS may be able to improve the strength of this technique as well. (7) It should be investigated what the applicability of BRS is in non-deterministic games or games with imperfect information.

Acknowledgment

This work is funded by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project TACTICS, grant number 612.000.525. Moreover, the authors would like to thank Jos Uiterwijk and the anonymous reviewers for their helpful comments.

References

[1] D. E. Knuth and R. W. Moore, "An Analysis of Alpha-Beta Pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[2] C. A. Luckhardt and K. B. Irani, "An Algorithmic Solution of N-Person Games," in *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*. Menlo Park, CA, USA: AAAI Press, 1986, pp. 158–162.

[3] N. R. Sturtevant and R. Korf, "On Pruning Techniques for Multi-Player Games," in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*. Menlo Park, CA, USA: AAAI Press, 2000, pp. 201–207.

[4] N. R. Sturtevant, "A Comparison of Algorithms for Multi-Player Games," in *Computers and Games (CG 2002)*, ser. Lecture Notes in Computer Science (LNCS), J. Schaeffer, M. Müller, and Y. Björnsson, Eds., vol. 2883. Berlin, Germany: Springer-Verlag, 2003, pp. 108–122.

[5] J.-T. Saito and M. H. M. Winands, "Paranoid Proof-Number Search," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games (CIG 2010)*, G. N. Yannakakis and J. Togelius, Eds. Piscataway, NJ, USA: IEEE press, 2010, pp. 203–210.

[6] N. R. Sturtevant, "Multi-Player Games: Algorithms and Approaches," Ph.D. dissertation, Computer Science Department, University of California, Los Angeles, CA, USA, 2003.

[7] R. E. Korf, "Multi-Player Alpha-Beta Pruning," *Artificial Intelligence*, vol. 48, no. 1, pp. 99–111, 1991.

[8] N. R. Sturtevant, "Last-Branch and Speculative Pruning Algorithms for Max^n," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, G. Gottlob and T. Walsh, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 669–678.

[9] U. Lorenz and T. Tscheuschner, "Player Modeling, Search Algorithms and Strategies in Multi Player Games," in *Advances in Computer Games (ACG 2005)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik, S.-C.. Hsu, T.-S.. Hsu, and H. H. L. M. Donkers, Eds., vol. 4250. Berlin, Germany: Springer-Verlag, 2006, pp. 210–224.

[10] N. R. Sturtevant and M. H. Bowling, "Robust Game Play Against Unknown Opponents," in *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, Eds. Hakodate, Japan: ACM, 2006, pp. 713–719.

[11] N. R. Sturtevant, M. Zinkevich, and M. H. Bowling, "Prob-Max^n: Playing N-player Games with Opponent Models," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*. Menlo Park, CA, USA: AAAI Press, 2006, pp. 1057–1063.

[12] N. R. Sturtevant, "An Analysis of UCT in Multi-Player Games," in *Computers and Games (CG 2008)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, Eds., vol. 5131. Berlin, Germany: Springer-Verlag, 2008, pp. 37–49.

[13] ——, "An Analysis of UCT in Multi-Player Games," *ICGA Journal*, vol. 31, no. 4, pp. 195–208, 2008.

[14] S. Sackson, *A Gamut of Games*. New York, NY, USA: Random House, 1969.

[15] J. A. M. Nijssen and M. H. M. Winands, "Enhancements for Multi-Player Monte-Carlo Tree Search," in *Computers and Games (CG 2010)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik, H. Iida, and A. Plaat, Eds., vol. 6515. Berlin, Germany: Springer-Verlag, 2011, pp. 238–249.

[16] M. Buro, "The Evolution of Strong Othello Programs," in *Entertainment Computing - Technology and Applications*, ser. IFIP Advances in Information and Communication Technology, R. Nakatsu and J. Hoshino, Eds., vol. 112. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2003, pp. 81–88.

[17] R. D. Greenblatt, D. E. Eastlake, and S. D. Crocker, "The Greenblatt Chess Program," in *Proceedings of the AFIPS Fall Joint Computer Conference 31*, 1967, pp. 801–810, Reprinted (1988) in *Computer Chess Compendium* (ed. D. N. L. Levy), pp. 56–66. B. T. Batsford Ltd., London, United Kingdom.

[18] D. M. Breuker, "Memory versus Search in Games," Ph.D. dissertation, Department of Computer Science, Maastricht University, Maastricht, The Netherlands, 1998.

[19] J. Schaeffer, "The History Heuristic," *ICCA Journal*, vol. 6, no. 3, pp. 16–19, 1983.

[20] S. G. Akl and M. M. Newborn, "The Principal Continuation and the Killer Heuristic," in *1977 ACM Annual Conference Proceedings*. New York, NY, USA: ACM Press, 1977, pp. 466–473.

[21] D. F. Beal, "Experiments with the Null Move," in *Advances in Computer Chess 5*, D. F. Beal, Ed. Amsterdam, The Netherlands: Elsevier Science Publishers, 1989, pp. 65–89.

[22] G. Goetsch and M. S. Campell, "Experiments with the Null-move Heuristic," in *Computers, Chess, and Cognition*, T. A. Marsland and J. Schaeffer, Eds. New York, NY, USA: Springer-Verlag, 1990, pp. 159–168.

[23] Y. Björnsson and T. A. Marsland, "Multi-Cut $\alpha\beta$-Pruning in Game-Tree Search," *Theoretical Computer Science*, vol. 252, no. 1–2, pp. 177–196, 2001.

[24] M. Buro, "ProbCut: An Effective Selective Extension of the Alpha-Beta Algorithm," *ICCA Journal*, vol. 18, no. 2, pp. 71–76, 1995.

[25] T. A. Marsland and F. Popowich, "Parallel Game-Tree Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI–7, no. 4, pp. 442–452, 1985.

[26] A. Plaat, "Research Re: Search & Re-Search," Ph.D. dissertation, Tinbergen Institute and Department of Computer Science, Erasmus University Rotterdam, Rotterdam, The Netherlands, 1996.

[27] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," in *Computers and Games (CG 2006)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, Eds., vol. 4630. Heidelberg, Germany: Springer-Verlag, 2007, pp. 72–83.

[28] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo Planning," in *Proceedings of the 17th European Conference on Machine Learning (ECML 2006)*, ser. Lecture Notes in Computer Science (LNCS), J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., vol. 4212. Berlin Heidelberg, Germany: Springer-Verlag, 2006, pp. 282–293.

[29] 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*, vol. 4, no. 3, pp. 343–357, 2008.

[30] S. Gelly and D. Silver, "Combining Online and Offline Knowledge in UCT," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, Z. Ghahramani, Ed. New York, NY, USA: ACM Press, 2007, pp. 273–280.

[31] J. Kloetzer, H. Iida, and B. Bouzy, "Playing Amazons Endgames," *ICGA Journal*, vol. 32, no. 3, pp. 140–148, 2009.

[32] R. J. Lorentz, "Amazons Discover Monte-Carlo," in *Computers and Games (CG 2008)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, Eds., vol. 5131. Berlin, Germany: Springer-Verlag, 2008, pp. 13–24.

[33] M. H. M. Winands and Y. Björnsson, "Evaluation Function based Monte-Carlo LOA," in *Advances in Computer Games (ACG 2009)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik and P. Spronck, Eds., vol. 6048. Berlin, Germany: Springer-Verlag, 2010, pp. 33–44.

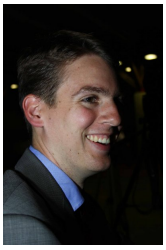
[34] T. Cazenave and A. Saffidine, "Utilisation de la Recherche Arborescente Monte-Carlo au Hex," *Revue d'Intelligence Artificielle*, vol. 23, no. 2–3, pp. 183–202, 2009, in French.

[35] P. Ciancarini and G. P. Favini, "Monte Carlo Tree Search Techniques in the Game of Kriegspiel," in *Proceedings of the Twenty-first International Joint Conferences on Artificial Intelligence (IJCAI-09)*, C. Boutilier, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 474–479.

[36] T. Cazenave, "Multi-player Go," in *Computers and Games (CG 2008)*, ser. Lecture Notes in Computer Science (LNCS), H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, Eds., vol. 5131. Berlin, Germany: Springer-Verlag, 2008, pp. 50–59.

[37] T. A. Marsland and Y. Björnsson, "Variable-Depth Search," in *Advances in Computer Games (ACG 9)*, H. J. van den Herik and B. Monien, Eds. Maastricht, The Netherlands: Universiteit Maastricht, 2001, pp. 9–24.

[38] I. Zuckerman, A. Felner, and S. Kraus, "Mixing Search Strategies for Multi-Player Games," in *Proceedings of the Twenty-first International Joint Conferences on Artificial Intelligence (IJCAI-09)*, C. Boutilier, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 646–651.



Maarten Schadd received the M.Sc. degree in Artificial Intelligence from the Maastricht ICT Competence Centre, Maastricht University, Maastricht, The Netherlands, in 2004. Currently, he is working on his Ph.D. degree in Artificial Intelligence at the Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands. His research covers various classes of games, from one-player games to multi-player games.

Mark Winands received the Ph.D. degree in Artificial Intelligence from the Department of Computer Science, Maastricht University, Maastricht, The Netherlands, in 2004. Currently, he is an Assistant Professor at the Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands. His research interests include heuristic search, machine learning and games. Dr. Winands regularly serves on program committees of major AI and computer games conferences. Since 2009 he is a member of the editorial board of the ICGA Journal.