Opponent-Pruning Paranoid Search

Hendrik Baier
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
hendrik.baier@cwi.nl

Michael Kaisers
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
michael.kaisers@cwi.nl

ABSTRACT

This paper proposes a new search algorithm for fully observable, deterministic multiplayer games: *Opponent-Pruning Paranoid Search* (OPPS). OPPS is a generalization of a state-of-the-art technique for this class of games, Best-Reply Search (BRS+). Just like BRS+, it allows for Alpha-Beta style pruning through the paranoid assumption, and both deepens the tree and reduces the pessimism of the paranoid assumption through pruning of opponent moves. However, it introduces three parameters that allow for more fine-grained control over the resulting search. Empirically, we show the effectiveness of OPPS in Chinese Checkers variants with three, four, and six players, where it outperforms its special case BRS+ as well as classic max^n and Paranoid search. We conclude that OPPS opens a promising research direction for search in multiplayer board and video games, and beyond.

CCS CONCEPTS

• **Computing methodologies** → **Game tree search**.

KEYWORDS

game tree search, multiplayer games

ACM Reference Format:
Hendrik Baier and Michael Kaisers. 2020. Opponent-Pruning Paranoid Search. In *International Conference on the Foundations of Digital Games (FDG '20), September 15–18, 2020, Bugibba, Malta.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3402942.3402957

1 INTRODUCTION

Fully observable two-player games with perfect information have historically seen some of the greatest breakthroughs in artificial intelligence – from Deep Blue's victory over Garry Kasparov [2] to AlphaGo's victory over Ke Jie [14, 15]. To date, much less attention has been paid to game tree search for multiplayer games, i.e. games with more than two players, with their additional challenges [4, 10, 19]. Whereas multiplayer interactions are receiving significant attention by the communities of (multi-agent) learning [1, 7] and symbolic cooperative planning [22], research in game search methods has yet to re-embrace this context.

In this paper, we propose a generalization of the state-of-the-art search technique BRS+ [4]. In BRS+, which is in turn an improvement to BRS [13] and based on the paranoid assumption that all opponents of the root player form a coalition against them, only one opponent of the root player is allowed to choose among all legal moves, while the other opponents are restricted to the move ranked highest by a static move ordering function. The search process dynamically determines the player allowed to move freely as the player who can currently make the strongest move against the root player. Limiting search to branch only on one opponent at a time drastically reduces the average branching factor of the search tree, and somewhat softens the pessimism of the paranoid assumption, while still enjoying an important advantage of every paranoid algorithm: the opportunity to apply Alpha-Beta style pruning. Vanilla BRS has been shown to be very effective in several multiplayer games [6, 12, 13], and BRS+ has been shown to outperform vanilla BRS in Four-Player Chess [4]. However, the design choices of BRS+ of allowing *exactly one* opponent to choose between *all* legal moves, while all other opponents get only *exactly one* move option, are somewhat ad hoc and inflexible.

The main contribution of this paper is a new multiplayer search algorithm named *Opponent-Pruning Paranoid Search* (OPPS), which generalizes BRS+ by turning these fixed design choices into algorithm parameters. OPPS allows n_1 opponents a branching factor limit of l_1, while other opponents are restricted to a lower branching factor limit of l_2. OPPS subsumes BRS+ as a special case, extending it to a parameterized family of algorithms. We empirically show that OPPS significantly outperforms both BRS+ as well as the classic max^n [10] and Paranoid [19] multiplayer search techniques in Chinese Checkers variants with three, four, and six players. These initial results demonstrate the potential of further refining search techniques for multiplayer domains, in board games, video games, and beyond.

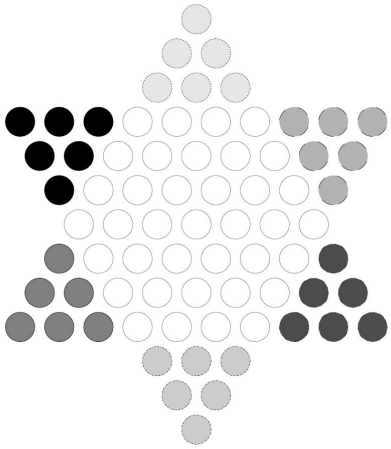
The remainder of this paper is structured as follows: Section 2 describes our test domain of Chinese Checkers. Section 3 outlines previous work and formalizes the problem. Section 4 introduces OPPS, and Section 5 shows the results of various experiments that test its performance. Finally, Section 6 concludes and discusses directions for future work.

2 CHINESE CHECKERS

Chinese Checkers is a popular fully observable and deterministic test domain for multiplayer search techniques [11, 13, 17], as well as for General Game Playing [5, 21]. It can be played on a star-shaped board by two, three, four, or six players, which allows for convenient testing of how well algorithms scale with the number of opponents. The most common Chinese Checkers board has 121 fields and is played with 10 pieces per player; in order to speed up experiments, we follow previous work [11, 17] in using a slightly

FDG '20, September 15–18, 2020, Bugibba, Malta

ACM ISBN 978-1-4503-8807-8/20/09…$15.00
https://doi.org/10.1145/3402942.3402957



checkers.JPG

Figure 1: Chinese Checkers, depicted here as the 6-player starting position of a 6-piece game variant, provides a benchmark challenge for multiplayer search.

smaller board with 73 fields and 6 pieces per player, as shown in Figure 1.

Each player starts in one of the corners of the board, and has the goal of moving all of their pieces to the opposite corner. When playing with six players, all corners are filled at the beginning; when playing with four, two opposing corners are left empty; when playing with three, every other corner is left empty. Pieces can move in all six directions to an adjacent empty field, or jump over an adjacent piece of any color if the field behind it is empty. Multiple jumps are possible in one turn, leading to much faster progress if the player can cleverly exploit the positioning of own and opponent pieces. The first player to move all six pieces into their goal corner wins the game[1]. Draws are not possible.

The search techniques discussed in this paper make use of both a *static board evaluation function*, used to assign heuristic values from the point of view of each player to leaf positions, as well as a *static move ordering function*, used to speed up search by searching more promising moves first – or restricting search to more promising moves entirely. Experiments use a slightly simplified version of a board evaluation from the literature [11], which adds up for each player the distances of each piece to the farthest field of the goal corner, and then normalizes those values so they add up to 1 over all players. The move ordering is identical to the one used in the same work [11], which orders all legal moves by how much closer they bring the moved piece to the farthest field of the goal corner. Moves are ordered by a score defined as $d_s - d_t$, where d_s is the distance before the move, and d_t is the distance after the move.

[1]We also use the modification introduced in [11] that a player has won if their goal corner is filled, regardless of the owner of the pieces, as long as the player has at least one of their own pieces in the goal corner. This avoids opponents passively preventing a player from winning by leaving one piece in their goal corner.

3 MULTIPLAYER SEARCH

This section presents the baselines for our experiments by outlining existing search algorithms for fully observable, deterministic multiplayer games. The classic search techniques max^n and Paranoid are discussed in Subsection 3.1, followed by BRS and its state-of-the-art variant BRS+ in Subsection 3.2.

3.1 Max^n and Paranoid Search

Following notation from the closest related work on BRS+ [4], we model the finite, deterministic games of perfect information relevant to this work as tuples $(N, S, Z, A, T, P, u_i, h_i, s_0)$, where $N = \{1, \ldots, n\}$ is the set of players; S is the set of game states; $Z \subseteq S$ is the set of terminal states; A is the set of moves; $T : S \times A \rightarrow S$ is the transition function mapping any state s and any move chosen from the available moves $A(s) \subseteq A$ to the successor state resulting from taking the move; $P : S \rightarrow N$ maps any state to the player to move; $u_i : Z \rightarrow [0, 1] \subseteq \mathbb{R}$ returns the utility of a terminal state for Player i; $h_i : S \rightarrow [0, 1]$ returns the *heuristic evaluation* of a state for Player i (the static board evaluation discussed in the previous section); and $s_0 \in S$ is the starting state of the game. The tuple $\mathbf{u}(s) = (u_1(s), \ldots, u_n(s))$ refers to the utility of state s from the point of view of all players; analogously for $\mathbf{h}(s)$.

The classic search algorithms for multiplayer games are max^n [10] and Paranoid [19]. Typically looking ahead to a fixed depth from a given root state, max^n is based on the idea that every player, at every internal node of the search tree, maximizes their own individual utility when it is their turn. This utility $V_d(s, a)$ of any move a in any state s with a desired lookahead of depth d is defined as follows [4]:

$$V_d(s,a) = \begin{cases} \mathbf{u}(s') & \text{if } s' \in Z \\ \mathbf{h}(s') & \text{if } s' \notin Z \text{ and } d = 0 \\ \max^{P(s)}_{a' \in A(s')} V_{d-1}(T(s', a'), a') & \text{otherwise,} \end{cases} \tag{1}$$

where max^i returns a tuple that maximizes the i^{th} value, and $s' = T(s, a)$. It has been proven that max^n, when searching the entire game tree, finds an equilibrium point of the game [10]. However, with realistic time constraints usually only a search to a small depth d can be completed, and a heuristic evaluation $\mathbf{h}(s)$ has to be employed at the leaf nodes.

Max^n has two major weaknesses [13]: First, deep pruning like in two-player Alpha-Beta search is not possible, leading to relatively shallow trees[2]. This can be especially problematic in multiplayer games, as the search might not even reach the next turn of the root player, making longer-term planning impossible. Second, max^n can be too optimistic, because it assumes all opponents are maximizing only their own outcomes, without ever forming coalitions against the root player.

Paranoid search takes the opposite approach of assuming that all opponent players have formed a coalition against the root player r. This effectively turns any multiplayer game into a two-player game between r and the coalition, with the coalition's turns consisting of sequences of moves, which makes deep Alpha-Beta style pruning

[2]Limited forms of pruning can be possible in max^n [8, 16], but not deep pruning.

possible [19]. The utility $V_d(s, a)$ for Paranoid is defined as follows:

$$V_d(s,a) = \begin{cases} \mathbf{u}(s') & \text{if } s' \in Z \\ \mathbf{h}(s') & \text{if } s' \notin Z \text{ and } d = 0 \\ \max^r_{a' \in A(s')} V_{d-1}(T(s',a'),a') & \text{if } s' \notin Z \text{ and } d > 0 \\ & \text{and } P(s) = r \\ \min^r_{a' \in A(s')} V_{d-1}(T(s',a'),a') & \text{if } s' \notin Z \text{ and } d > 0 \\ & \text{and } P(s) \neq r \end{cases} \tag{2}$$

Paranoid tends to outperform $\max^n$ in practice due to the deeper searches enabled by Alpha-Beta style pruning. However, it suffers from being overly pessimistic, which becomes more unrealistic the deeper it searches, and the more opponents it is playing against. In many games it is ultimately not possible to win if all opponents form a coalition. As a consequence, experiments show that $\max^n$ becomes relatively stronger with more players, and Paranoid performance varies depending on which ply depth is reached (see Subsection 5.2).

Several variations of $\max^n$ and Paranoid have been proposed to tackle their weaknesses, for example by modelling coalitions in different ways [9, 23], or by handling imperfect opponent models [18, 20]. In this paper, we build on a line of work that "softens the unrealistic $\max^n$ and paranoid assumptions" [13] through pruning, such as in the algorithms BRS and BRS+.

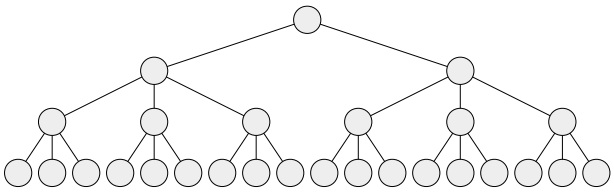
3.2 BRS and BRS+

One of the most effective search techniques based on the paranoid assumption is Best-Reply Search (BRS). Like Paranoid, BRS assumes that all opponents are playing against the root player; however, it restricts the move choices of this coalition. In vanilla BRS [13], only the opponent with the strongest move against the root player can act, while all other opponents have to pass. This means that all opponent levels of the tree are virtually combined into one, and the root player and the opponent coalition move alternatingly as if playing a two-player game.

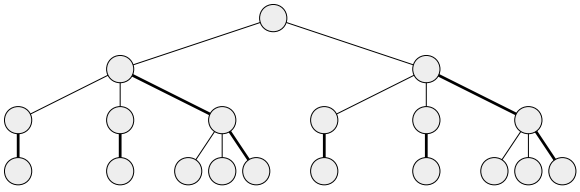
The restricted move choice of opponents in BRS fulfills two functions. First, it allows for even deeper search and more long-term planning than in vanilla Paranoid search, and Alpha-Beta pruning can also still be applied. Second, it weakens some of the negative effects of the paranoid assumption: All opponents are still playing against the root player, but their moves and therefore their strength as a coalition are limited. This places BRS somewhere between the optimism of $\max^n$, where opponents never form coalitions, and the pessimism of Paranoid, where all opponents always form a coalition playing at full strength against the root player.

BRS has been shown to work well in practice in a variety of multiplayer games [6, 12, 13]. However, it suffers from the drawback that it leads to illegal game states in the tree when the game at hand does not actually allow passing. The changed turn order also leads to the unrealistic consequence of the root player making as many moves as all other opponents combined during search. This motivated the further development of BRS into BRS+ [4]. BRS+ assumes access to a move ordering function. Its basic idea is that whenever an opponent does not have the strongest move against the root player in a given state and would be forced to pass by vanilla BRS, this opponent instead gets to play the move ranked highest by the move ordering, which maintains valid game states during search. This can be achieved without actually transforming the tree, by manipulating the sets of available moves during the search process in such a way that between two moves of the root player, exactly one opponent gets free choice of move, while the others play the highest ranked move. Subfigures 2(a) and (b) show the effect of this pruning – 2(a) is a toy example of a game tree, and 2(b) is the corresponding BRS+ search tree in which at most one of the opponents can make a free move (not ranked first by the move ordering) between two turns of the root player. The highest ranked moves are marked with bold lines in the illustration.

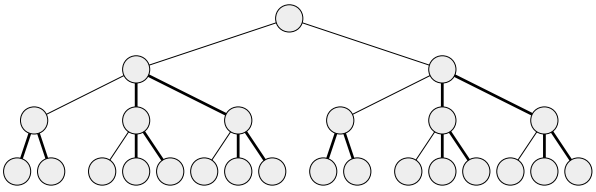
BRS+ can be considered state of the art for fully observable, deterministic multiplayer games. Problems with illegal moves and



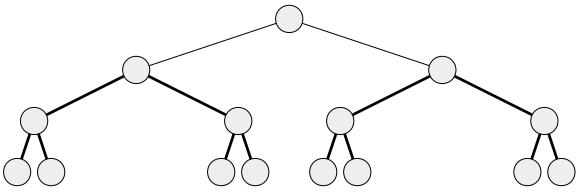
(a) The fully expanded game tree of a three-player toy game.



(b) The pruned tree as searched by BRS+ / OPPS(1,∞,1).



(c) Other parameter settings of OPPS lead to different levels of pruning – this for example is the larger tree as searched by OPPS(1,∞,2).



(d) This smaller tree is searched by OPPS(0,x,2) for any choice of x (or, in a game with 3 players, equivalently by OPPS(2,2,x) for any x).

Figure 2: Illustration of Opponent-Pruning Paranoid Search and the effects of its parameters on search tree size.

turn orders are avoided, for the price of requiring a move ordering function, and being somewhat dependent on its quality.

4 OPPONENT-PRUNING PARANOID SEARCH

Opponent-Pruning Paranoid Search (OPPS), the search technique proposed in this paper, is a natural generalization of BRS+. Between any two turns of the root player, BRS+ essentially can be seen as dividing the opponents of the root player into two groups: One group of merely one opponent who is allowed to freely choose a move – with the search process determining dynamically which opponent this is – and another group subsuming all other opponents who are required to execute their highest ranked move. OPPS generalizes this by introducing three parameters: n_1 is the number of opponents in the group with a wider move choice, with l_1 being the corresponding upper limit to their branching factor; similarly, l_2 is the upper limit to the branching factor for all other opponents, such that $l_1 \geq l_2$ w.l.o.g.[3]. Both l_1 and l_2 here refer to limits on the number of top-ranked moves as ordered by the move ordering; e.g., $l_1 = 8$ denotes that the first group chooses from the eight highest-ranked moves, and $l_2 = 2$ implies that group two only searches the two highest-ranked moves, etc. BRS+ is the special case of OPPS with $n_1 = 1$, $l_1 = \infty$ (no limit on the allowed number of moves), and $l_2 = 1$ (opponents in second group choose only their first-ranked move). By varying these three parameters, however, OPPS can more smoothly adapt both its degree of tree pruning as well as its degree of paranoia, or flexibility of the opponent coalition, to whatever is most effective in the domain at hand. We write OPPS(x, y, z) as shorthand for OPPS with $n_1 = x$, $l_1 = y$, and $l_2 = z$.

Pseudocode for OPPS is shown in Algorithm 1. It uses a variable m, counting the number of moves taken between two root player turns that would not be allowed for group two opponents – this number is restricted by the algorithm to n_1. Returning utilities and heuristic evaluation from the point of view of the root player (lines 5 and 6), as well as maximizing these values if it is the root player's turn and minimizing them otherwise (line 18), makes sure the n_1 opponents in group one are dynamically chosen in a paranoid way as the opponents who currently have the strongest moves against the root player.

Note that since OPPS is based on the paranoid assumption just like BRS and BRS+, Alpha-Beta style pruning can still be applied. As in the reference work on BRS+ [4], we have left this out of the pseudocode for the sake of simplicity and readability, but it is used in all experiments in the following section. Just like any variant of Alpha-Beta search, OPPS can also easily be modified to return the best move found together with its value.

Depending on its parameter settings, the tree searched by OPPS can be made either larger or smaller than the tree BRS+ would search. This is illustrated in Figure 2(c) and 2(d) with the trees searched by OPPS(1, ∞, 2) and OPPS(0,x,2) for any x, respectively[4]. In contrast to Subfigure 2(b), here the top *two* moves as ranked by the move ordering function are marked in bold. The small differences in tree size we can observe in this toy example can grow exponentially with tree depth, as we empirically demonstrate in Subsection 5.3.

Algorithm 1 Opponent-Pruning Paranoid Search

```
1:  n_1 ← size of group one
2:  l_1 ← branching factor limit for group one
3:  l_2 ← branching factor limit for group two
4:  function OPPS(state s, depth d, count m, root player i)
5:      if s ∈ Z then return u_i(s)
6:      else if d = 0 then return h_i(s)
7:      else
8:          A'(s) ← A(s)
9:          staticMoveSort(A'(s))
10:         Let U ← ∅ be the set of child values
11:         if P(s) = i then m ← 0
12:         else if m = n_1 then A'(s) ← first l_2 moves in A'(s)
13:         else A'(s) ← first l_1 moves in A'(s)
14:         for a ∈ A'(s) do
15:             s' ← T(s, a)
16:             m' ← if a ∉ first l_2 moves in A'(s) then m + 1 else m
17:             U ← U ∪ {OPPS(s', d − 1, m', i)}
18:         return if P(s) = i then max(U) else min(U)
19: end function
```

5 EXPERIMENTAL RESULTS

This section describes our three sets of experiments for testing the proposed Opponent-Pruning Paranoid Search in Chinese Checkers, as well as the results.

In all experiments we compare two given multiplayer search techniques. In an n-player game, there are $2^n - 2$ ways of assigning these two techniques to the n players of the game such that both techniques are present. In order to achieve a fair comparison, all of our experiments were performed in batches containing one match for each possible assignment. When we specify "a minimum of" 1000 matches played, this is therefore meant as, for each game variant, the smallest number that is a) larger than 1000 and b) a multiple of the number of player assignments in that game variant.

All experiments were conducted using a fixed time per move. The search algorithms were therefore implemented with iterative deepening, and the move returned by the last completed iteration before time ran out was used in the match. All results are reported as average winrates with 95% confidence intervals.

5.1 Performance of OPPS vs. BRS+

In our first set of experiments, OPPS was compared to the current state-of-the-art multiplayer search technique BRS+, i.e. its special case OPPS(1,∞,1). We used Chinese Checkers with three, four, and six players, and at the time controls of 50 ms, 250 ms, 1000 ms, and 5000 ms per move. OPPS' parameters n_1, l_1, and l_2 were tuned for each time control and domain by testing all combinations of $n_1 = \{0, 1, 2, 3\}$, $l_1 = \{1, 2, \ldots, 10, \infty\}$, and $l_2 = \{1, 2, \ldots, 5\}$, where $l_1 \geq l_2$. The best algorithm settings were then re-run with a minimum of 1000 matches.

[3] A further generalization to more than two groups, potentially determined by a more involved learning algorithm, seems natural and is briefly discussed as future research in Section 6. For ease of exposition we only discuss the direct extension to two parameterized groups here.

[4] The parameter l_1 is irrelevant in Figure 2(d), as there is only one group of opponents, with a branching factor of $l_2 = 2$.

Table 1: Performance of OPPS against BRS+ / OPPS(1,∞,1) in Chinese Checkers with different numbers of players and at different time settings.

players	time per move			
	50 ms	250 ms	1000 ms	5000 ms
3	62.2% ±3.0%	53.1% ±3.1%	55.7% ±3.1%	60.3% ±3.0%
4	59.6% ±3.1%	56.2% ±3.1%	58.0% ±3.1%	74.3% ±2.8%
6	60.2% ±3.0%	76.5% ±2.6%	62.0% ±3.0%	73.2% ±2.8%

The results are shown in Table 1. As a generalization of BRS+, OPPS can not be outperformed by it. In fact, it is significantly stronger ($p < 0.05$) than BRS+ in all game variants and time controls with the exception of the three-player game at 250 ms per move, where the difference did not reach significance.

We can make two interesting observations. First, OPPS is able to scale better than BRS+ with the number of opponents: Its average performance over all time controls is 57.8% for three-player Chinese Checkers, 62.0% for four players, and 68.0% for six players. This shows that taming the branching factors of larger games is increasingly effective, and is promising for future work on games with even higher numbers of players. Second, the performance of OPPS depends strongly on the time controls. A possible explanation is that spending more search time, and therefore being able to complete a search to depth $x + 1$ instead of only x, does not always give a predictable performance boost to a paranoid algorithm. In fact, when further evaluating BRS+ with depth-limited instead of time-limited searches in Chinese Checkers with four players[5], we found that searching 3 plies deep wins only 51.6% of matches against depth 2; depth 4 wins only 48.0% against depth 3; but depth 5 wins 89.3% against depth 4. Depths 3 and 4 only add an additional opponent turn, strongly pruned and evaluated by the paranoid heuristic, while depth 5 adds the next turn of the root player to the tree, which is much more important for planning. As mentioned as a strength of BRS in the literature, "more MAX [root player] nodes are visited along the search path, leading to more long-term planning" [13]. Surprisingly, searches with depth 5 even win 59.9% of games against depth 6, meaning the searches ending in a root player move are here significantly stronger. This could explain why OPPS at some time settings has a decisive advantage over BRS+ by searching to a deeper, stronger depth – while at other time settings, it may still search deeper on average, but not to a stronger depth. A detailed analysis of this phenomenon is deferred to future work.

5.2 Performance of OPPS vs. max^n and Paranoid

BRS+ has been shown to be a significant improvement over BRS in Four-Player Chess [4], and BRS in turn is generally stronger than the classic techniques max^n and Paranoid, as demonstrated in a variety of games in [11, 13]. Nevertheless, we wanted to make sure that our tuning of OPPS against BRS+ had not led to overfitting, and that OPPS was indeed still outperforming the classic techniques max^n and Paranoid as well. In our second set of experiments, we tested this at 250 ms per move in Chinese Checkers with three, four, and six players. Every comparison is based on a minimum of 1000 matches. The results of OPPS, using the same parameter settings that were found to be optimal against BRS+ in Subsection 5.1, are reported in Table 2.

Table 2: Performance of OPPS against max^n and Paranoid in Chinese Checkers with different numbers of players. 250 ms per move.

players	opponent	
	max^n	Paranoid
3	95.2% ± 1.4%	74.1% ± 2.8%
4	88.0% ± 2.1%	88.8% ± 2.0%
6	71.2% ± 2.8%	74.2% ± 2.7%

OPPS is significantly stronger than both max^n and Paranoid in all variants of the game ($p < 0.0002$). In line with previous results for max^n vs. Paranoid [11], max^n gets stronger relative to both Paranoid and OPPS the more opponents are present in the game – probably because the paranoid assumption that all players form a coalition against the root player becomes less and less realistic. It is a naturally arising but open question for future work whether max^n could outperform OPPS in games with many more opponents. The answer to this question is non-trivial, since at least given realistic computational constraints there is an intricate interplay of branching factor, self-maximisation and root-minimisation in multi-player search.

5.3 Tree Growth Comparison

The last set of experiments characterises and juxtaposes the behaviour of the multiplayer search techniques discussed in this paper by comparing how fast search trees grow with search depth. The evaluation is based on four-player Chinese Checkers, with a minimum of 100 complete games using each search depth (as deep as possible within a reasonably short timeframe) for each algorithm: max^n, Paranoid, BRS+ / OPPS(1,∞,1), and two other variants of OPPS, namely OPPS(1,5,1) and OPPS(2,8,5).

The resulting search tree sizes, averaged over all move searches in those games, are plotted in Figure 3. It shows how much smaller search trees with a given depth can get when moving from max^n to Paranoid, due to Paranoid's ability to use Alpha-Beta style pruning. At depth two and three, Paranoid searches about an order of magnitude fewer nodes. Moreover, BRS+ / OPPS(1,∞,1) provides a drastic further reduction on top of this – it searches two orders of magnitude fewer nodes than Paranoid at depth four, the highest we could test for Paranoid. The additional plots for OPPS(1,5,1) and OPPS(1,∞,2) show how the parameters of OPPS allow it to fine-tune the tree sizes to what works best in the domain at hand.

[5]These results are from a minimum of 500 matches each.

As demonstrated here, OPPS can do both more and less pruning than its special case BRS+.

In future work it would be worth studying the other salient aspect of pruning Paranoid search trees – that they not only get smaller, but less pessimistic in their move evaluations as well – and which interactions may exist between these two effects and the playing strength of OPPS.

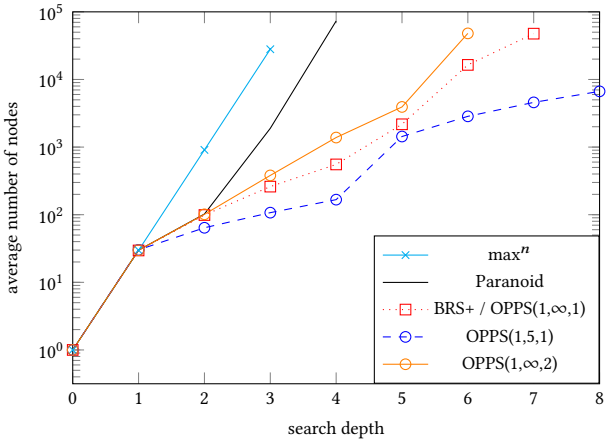


Figure 3: Empirical tree growth of different search algorithms in Chinese Checkers with 4 players.

6 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we propose a new search technique for deterministic multiplayer games with perfect information: *Opponent-Pruning Paranoid Search* (OPPS), a generalization of the current state-of-the-art technique BRS+. OPPS keeps some of the advantages of BRS+ such as Alpha-Beta style pruning, but allows for more fine-grained control over the resulting search tree. We show that OPPS outperforms its special case BRS+ as well as classic max^n and Paranoid search in Chinese Checkers with three, four, and six players.

This opens several interesting directions for future research. First, OPPS should be tested in several other multiplayer board and video games. It could also be tested together with automatically learned evaluation functions and move orderings instead of the hand-coded ones used here and in previous work. Second, coalitions could be studied in greater detail, with inspiration and baselines taken from e.g. the Comixer [9] and MP-Mix [23] approaches. Third, the idea of abstracting away some of your opponents – or some of the opponents' choices – in order to be able to focus more on your own long-term planning is also promising to explore more deeply in the context of Monte Carlo Tree Search (MCTS). In this search framework, e.g. Alpha-Beta style pruning is not possible, and the paranoid assumption has not been shown to work yet [11]; but abstraction and pruning – and unpruning – of moves have been studied in some depth, leading to interesting connections (e.g. to [3]).

Last but not least, several choices in the design of OPPS could be generalized further, even though they are already allowing for more flexibility and adaptability than in max^n, Paranoid, and BRS+. Instead of using only two groups of opponents with a fixed size and a fixed branching factor limit each, we can easily imagine a generalization to more groups, e.g. in an algorithm that makes the choice of branching factor in every node of the tree dynamically based on features of both the game state as well as the current state of the search (e.g. depth in the tree, remaining search depth, previously found "killer moves", etc). This idea could be applied to other classes of multiplayer games as well, e.g. games featuring indeterminism, simultaneous moves, or imperfect information. We see OPPS only as a first step into this promising research direction.

ACKNOWLEDGMENTS

This work is part of the project *Flexible Assets Bid Across Markets* (FABAM, project number TEUE117015), funded within the Dutch Topsector Energie / TKI Urban Energy by Rijksdienst voor Ondernemend Nederland (RvO).

REFERENCES

[1] Stefano V Albrecht and Peter Stone. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258 (2018), 66–95.
[2] Murray Campbell, A. Joseph Hoane Jr., and Feng-hsiung Hsu. 2002. Deep Blue. *Artificial Intelligence* 134, 1-2 (2002), 57–83.
[3] Guillaume M. J. B. Chaslot, Mark H. M. Winands, Jaap van den Herik, Jos W. H. M. Uiterwijk, and Bruno Bouzy. 2008. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation* 4, 03 (2008), 343–357.
[4] Markus Esser, Michael Gras, Mark H. M. Winands, Maarten P. D. Schadd, and Marc Lanctot. 2013. Improving Best-Reply Search. In *8th International Conference on Computers and Games (CG 2013) (Lecture Notes in Computer Science)*, H. Jaap van den Herik, Hiroyuki Iida, and Aske Plaat (Eds.), Vol. 8427. Springer, 125–137. https://doi.org/10.1007/978-3-319-09165-5
[5] Hilmar Finnsson. 2012. *Simulation-based General Game Playing*. Ph.D. Dissertation. School of Computer Science, Reykjavik University.
[6] Michael Gras. 2012. *Multi-Player Search in the Game of Billabong*. Master's thesis. Department of Knowledge Engineering, Maastricht University.
[7] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. 2017. A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *CoRR* abs/1707.09183 (2017). arXiv:1707.09183
[8] Richard E. Korf. 1991. Multi-Player Alpha-Beta Pruning. *Artificial Intelligence* 48, 1 (1991), 99–111.
[9] Ulf Lorenz and Tobias Tscheuschner. 2006. Player Modeling, Search Algorithms and Strategies in Multi-player Games. In *11th International Conference on Advances in Computer Games (ACG 2005) (Lecture Notes in Computer Science)*, H. Jaap van den Herik, Shun-chin Hsu, Tsan-sheng Hsu, and H. H. L. M. Donkers (Eds.), Vol. 4250. Springer, 210–224. https://doi.org/10.1007/11922155_16
[10] Carol Luckhart and Keki B. Irani. 1986. An Algorithmic Solution of N-Person Games. In *5th National Conference on Artificial Intelligence*, Tom Kehler (Ed.). Morgan Kaufmann, 158–162.
[11] Pim A. M. Nijssen. 2013. *Monte-Carlo Tree Search for Multi-Player Games*. Ph.D. Dissertation. Department of Knowledge Engineering, Maastricht University.
[12] Pim A. M. Nijssen and Mark H. M. Winands. 2013. Search Policies in Multi-Player Games. *ICGA Journal* 36, 1 (2013), 3–21.
[13] Maarten P. D. Schadd and Mark H. M. Winands. 2011. Best Reply Search for Multiplayer Games. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 1 (2011), 57–66.
[14] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (2016), 484–489.
[15] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the Game of Go without Human Knowledge. *Nature* 550, 7676 (2017), 354–359.
[16] Nathan R. Sturtevant. 2003. Last-Branch and Speculative Pruning Algorithms for Max^n. In *18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, Georg Gottlob and Toby Walsh (Eds.). Morgan Kaufmann, 669–678.
[17] Nathan R. Sturtevant. 2008. An Analysis of UCT in Multi-Player Games. *ICGA Journal* 31, 4 (2008), 195–208.

[18] Nathan R. Sturtevant and Michael H. Bowling. 2006. Robust Game Play against Unknown Opponents. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone (Eds.). ACM, 713–719. https://doi.org/10.1145/1160633.1160761

[19] Nathan R. Sturtevant and Richard E. Korf. 2000. On Pruning Techniques for Multi-Player Games. In *17th National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence*, Henry A. Kautz and Bruce W. Porter (Eds.). AAAI Press / The MIT Press, 201–207.

[20] Nathan R. Sturtevant, Martin Zinkevich, and Michael H. Bowling. 2006. Prob-Maxn: Playing N-Player Games with Opponent Models. In *The 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*. AAAI Press, 1057–1063.

[21] Mandy J. W. Tak, Mark H. M. Winands, and Yngvi Björnsson. 2012. N-Grams and the Last-Good-Reply Policy Applied in General Game Playing. *IEEE Transastions on Computational Intelligence and AI in Games* 4, 2 (2012), 73–83.

[22] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2017. Cooperative Multi-Agent Planning: A Survey. *ACM Comput. Surv.* 50, 6 (2017), 84:1–84:32.

[23] Inon Zuckerman and Ariel Felner. 2011. The MP-MIX Algorithm: Dynamic Search Strategy Selection in Multiplayer Adversarial Search. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 4 (2011), 316–331. https://doi.org/10.1109/TCIAIG.2011.2166266