Playout Search for Monte-Carlo Tree Search in Multi-Player Games [1]

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

Department of Knowledge Engineering, Faculty of Humanities and Sciences
Maastricht University, Maastricht, The Netherlands

Over the past years, Monte-Carlo Tree Search (MCTS) [2, 3] has become a popular technique for playing deterministic perfect-information multi-player games. MCTS is a best-first search technique that instead of an evaluation function uses simulations to guide the search. For MCTS, a tradeoff between search and knowledge has to be made. The more knowledge is added, the slower each playout gets. The trend seems to favor fast simulations with computationally light knowledge, although recently, adding more heuristic knowledge at the cost of slowing down the playouts has proven beneficial in some games [8].

In this abstract we propose Playout Search for MCTS in multi-player games. Instead of playing random moves biased by computationally light knowledge in the playout phase, domain knowledge can be incorporated by performing small searches. These searches employ more expensive evaluation functions to assess the leaf nodes of non-terminal positions. This reduces the number of playouts per second significantly, but it improves the reliability of the playouts. When selecting a move in the playout phase, one of the following three search techniques is used to choose a move.

Two-ply max^n [4]. A two-ply max^n search tree is built where the current player is the root player and the first opponent plays at the second ply. Both the root player and the first opponent try to maximize their own score. $\alpha\beta$-pruning in a two-ply max^n search tree is not possible.

Two-ply Paranoid [7]. Similar to max^n, a two-ply search tree is built where the current player is the root player and the first opponent plays at the second ply. The root player tries to maximize its own score, while the first opponent tries to minimize the root player's score. Contrary to max^n, $\alpha\beta$-pruning is possible.

Two-ply Best Reply Search (BRS) [5]. The tree structure of BRS is similar to that of Paranoid search. The difference is that at the second ply, not only the moves of the first opponent are considered, but the moves of all opponents are investigated. Similar to paranoid, $\alpha\beta$-pruning is possible.

The major disadvantage of incorporating search in the playout phase of MCTS is the reduction of the number of playouts per second [8]. In order to prevent this reduction from outweighing the benefit of the quality of the playouts, enhancements should be implemented to speed up the search and keep the reduction of the number of playouts to a minimum. The number of searches can be reduced by using *ϵ-greedy playouts* [6]. With a probability of ϵ, a move is chosen uniform randomly. Otherwise, the selected search technique is used to select the best move. The amount of $\alpha\beta$-pruning in a tree can be increased by using *move ordering*. When using move ordering, a player's moves are sorted using a static move evaluator. Another move ordering technique is applying *killer moves* [1]. In each search, two killer moves are always tried first. These are the two last moves that were best or caused a cutoff, at the current depth. Moreover, if the search is completed, the killer moves for that specific level in the playout are stored, such that they can be used during the next MCTS iterations. Killer moves are only used with search techniques where $\alpha\beta$-pruning is possible, i.e., Paranoid and BRS search. The size of the tree can be further reduced by using *k-best pruning*. Only the k best moves are investigated. This reduces the branching factor of the tree from b to k.

To test the performance of Playout Search, we performed several round-robin tournaments where each participating player uses a different playout strategy. These playout strategies include 2-ply max^n, 2-ply Paranoid and 2-ply BRS. Additionally, we include players with one-ply and the static move evaluator as reference players. The tournaments were run for 3-player and 4-player Chinese Checkers and 3-player and

[1] The full version of this paper is published in: *Advances in Computer Games (ACG13)*, LNCS 7168, pp. 72–83, 2012.

4-player Focus. In each game, two different player types participate. If one player wins, a score of 1 is added to the total score of the corresponding player type.

In the first set of experiments, all players were allowed to perform 5000 playouts per move. For 3-player Chinese Checkers, BRS is the best technique. It performs slightly better than max^n and Paranoid. BRS wins 53.4% of the games against max^n and 50.9% against Paranoid. These three techniques perform significantly better than one-ply and the move evaluator. In the 4-player variant, max^n, Paranoid and BRS remain the best techniques, where BRS performs slightly better than the other two. BRS wins 53.8% of the games against Paranoid and 51.9% against max^n. For 3-player Focus, the best technique is BRS, winning 54.8% against max^n and 55.5% against Paranoid. Max^n and Paranoid are equally strong. BRS is also the best technique in 4-player Focus, though it is closely followed by max^n and Paranoid. BRS wins 51.5% of the games against max^n and 51.8% against Paranoid.

In the second set of experiments, we gave each player 5 seconds per move. In 3-player Chinese Checkers, one-ply and Paranoid are the best techniques. Paranoid wins 49.2% of the games against one-ply and 68.5% against the move evaluator. BRS ranks third, and the move evaluator and max^n are the weakest techniques. In 4-player Chinese Checkers, one-ply is the best technique, closely followed by Paranoid. Paranoid wins 46.3% of the games against one-ply. Paranoid is still stronger than the move evaluator, winning 64.6% of the games. BRS comes in third place, outperforming max^n and the move evaluator. One-ply also performs the best in 3-player Focus. Paranoid plays slightly stronger than the move evaluator, with Paranoid winning 51.9% of the games against the move evaluator and 46.1% against one-ply. The move evaluator and Paranoid perform better than BRS and max^n. In 4-player Focus, Paranoid performs better than in the 3-player version and slightly outperforms one-ply. Paranoid wins 51.7% of the games against one-ply and 59.9% against the move evaluator. Max^n also performs significantly better than in the 3-player version. It is as strong as one-ply and better than the move evaluator.

In the final set of experiments, we gave the players 30 seconds per move. Because these games take quite some time to finish, only the one-ply player and the Paranoid player were matched against each other. In the previous set of experiments, these two techniques turned out to be the strongest. Paranoid appears to perform slightly better when the players receive 30 seconds per move compared to 5 seconds per move. In 3-player Chinese Checkers, Paranoid wins 53.9% of the games, compared to 49.2% with 5 seconds. In 4-player Chinese Checkers, 48.3% of the games are won by Paranoid, compared to 46.3% with 5 seconds. In 3-player Focus, the win rate of Paranoid increases from 46.1% with 5 seconds to 50.7% with 30 seconds and in 4-player Focus from 51.7% to 54.1%.

The results show that Playout Search significantly improves the quality of the playouts in MCTS. This benefit is countered by a reduction of the number of playouts per second. Especially BRS and max^n suffer from this effect. Based on the experimental results we may conclude that Playout Search for multi-player games might be beneficial if the players receive sufficient thinking time and Paranoid search is employed. Under these conditions, Playout Search outperforms playouts using light heuristic knowledge in the 4-player variant of Focus and the 3-player variant of Chinese Checkers.

References

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

[2] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In H.J. van den Herik, P. Ciancarini, and H.H.L.M. Donkers, editors, *Computers and Games (CG 2006)*, volume 4630 of *LNCS*, pages 72–83, Berlin, Germany, 2007. Springer.

[3] L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 282–293, Berlin, Germany, 2006. Springer.

[4] C. Luckhardt and K.B. Irani. An Algorithmic Solution of N-Person Games. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*, volume 1, pages 158–162, 1986.

[5] M.P.D. Schadd and M.H.M. Winands. Best Reply Search for Multiplayer Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(1):57–66, 2011.

[6] N.R. Sturtevant. An Analysis of UCT in Multi-player Games. In H.J. van den Herik, X. Xu, Z. Ma, and M.H.M. Winands, editors, *Computers and Games (CG 2008)*, volume 5131 of *LNCS*, pages 37–49, Berlin, Germany, 2008. Springer.

[7] N.R. Sturtevant and R.E. Korf. On Pruning Techniques for Multi-Player Games. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 201–207. AAAI Press / The MIT Press, 2000.

[8] M.H.M. Winands and Y. Björnsson. $\alpha\beta$-based Play-outs in Monte-Carlo Tree Search. In *2011 IEEE Conference on Computational Intelligence and Games (CIG 2011)*, pages 110–117. IEEE Press, 2011.