# Paranoid Proof-Number Search[1]

Jahn-Takeshi Saito†‡ Mark H.M. Winands†

†*Games and AI Group, Department of Knowledge Engineering,*
‡ *Department of Bioinformatics BiGCaT,*
*Maastricht University, Maastricht, The Netherlands*

In the past 20 years a number of non-trivial two-player games have been solved [6], meaning that the perfect playing strategy for both players and the resulting game-theoretical value have been determined (from the starting state only for weakly solved games; from all positions for strongly solved games; for ultra-weakly solved games only the game-theoretical value is determined, not the playing strategy).

This abstract extends the scope to multi-player games (i.e., games with more than two players). So far, the notion of solving in multi-player games has attracted little if any attention by the games-and-search community. One of the reasons is that while two-player games have a unique game-theoretical value such a unique value is commonly not found in multi-player games because many equilibrium points can exist [5]. The standard search algorithms for multi-player games are max$^n$ [4] and paranoid search [5]. These search algorithms were designed to provide playing strength for programs but not to solve games.

Proof-Number Search (PNS, [2]), however, was designed for solving, but only for two-player games. The central idea of PNS is a heuristic that prefers expanding narrow subtrees over wide subtrees. Variants of PNS have been applied successfully to the endgame of Awari, Shogi, Othello, LOA, and Go; and contributed to solving Checkers.

In this abstract we propose solving multi-player games under the paranoid condition. This is equivalent to finding the optimal score a player can achieve independent of the other players' strategies. At the same time the paranoid condition is equivalent to reducing a multi-player game into a two-player game by assuming that one player (the paranoid player) plays against a coalition of all other players (the coalition players). We furthermore introduce and examine a PNS variant that we call Paranoid Proof-Number Search (PPNS) for weakly solving multi-player games under the paranoid condition. To reduce the memory requirements, we implemented PPNS in the PN$^2$ framework [1, 3], a two-level proof-number search variant.

Experimental results were obtained by performing PPNS to the game of Rolit, a multi-player variant of the game known as Reversi or Othello. The object of the game is to block your opponents' playing pieces and capture them by rolling their colors to your own. The player with the most playing pieces showing their color at the end of the game wins. In the experiments, we applied PPNS to solve $4 \times 4$, $6 \times 6$, and $8 \times 8$ Rolit, each for two, three, and four players, resulting in a total of 9 variants. A heuristic evaluation function was implemented to initialize the unknown leaf nodes of PPNS. The evaluation function simply takes the number of occupied corners for paranoid and coalition players into account. All experiments were carried out on a mixed Linux cluster with two kinds of nodes: 10 nodes with four 2.33 Opteron processors with 4 GB, and 2 nodes with eight 2.66 GHz Xeon processors and 8 GB of RAM.

The optimal scores found by PPNS are given in Table 1. The main results are that $4 \times 4$ Rolit has solutions that are non-trivial, i.e., scores larger than the minimum theoretical scores of 0 or 1 point exist for 3- and 4-player configurations. This is different in the case of $6 \times 6$ Rolit. Here, the main finding is that $6 \times 6$ Rolit is an unfair game: the paranoid player can always be forced to the minimum analytic score of 0 or 1 point. For $8 \times 8$ Rolit we did not attempt to solve the 2- and 3-player configurations because of the too large search space.

To assess the performance of PPNS we compared the empirically found search trees of PPNS with the analytical best-cases of the search trees that standard paranoid search would create when proving the optimal score for the Red player. We assume that standard paranoid search would not take advantage of the possible non-uniform nature of the game tree as PPNS does. Thus, we arrive at the best-case game-tree

[1]The full version of this paper is published in: *IEEE Conference on Computational Intelligence and Games (CIG-2010)*, pages 203–210, ISBN 978-1-4244-6297-1.

Table 1: Proven optimal scores for all players on $4 \times 4$, $6 \times 6$, and $8 \times 8$ Rolit for 2, 3 and 4 players under paranoid condition.

| | $4 \times 4$ | | | $6 \times 6$ | | | $8 \times 8$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | **2** | **3** | **4** | **2** | **3** | **4** | **2** | **3** | **4** |
| **Red** | 7 | 3 | 2 | 16 | 0 | 0 | ? | ? | 0 |
| **Green** | 9 | 4 | 1 | 20 | 0 | 0 | ? | ? | 0 |
| **Yellow** | - | 3 | 2 | - | 1 | 0 | - | ? | 0 |
| **Blue** | - | - | 1 | - | - | 1 | - | - | 1 |

Table 2: Comparison of search trees when proving the optimal score for Red. Estimated best cases in number of nodes for paranoid search compared to PPNS.

| | $4 \times 4$ | | $6 \times 6$ | |
|---|---|---|---|---|
| **Players** | **Paranoid** | **PPNS** | **Paranoid** | **PPNS** |
| **2** | 471 | $4.1 \times 10^4$ | $3.5 \times 10^{11}$ | $9.5 \times 10^{13}$ |
| **3** | $5.3 \times 10^3$ | $5.6 \times 10^4$ | $1.3 \times 10^{15}$ | $4.2 \times 10^{10}$ |
| **4** | $3.0 \times 10^4$ | $9.1 \times 10^4$ | $1.3 \times 10^{17}$ | $1.5 \times 10^6$ |

size of $O(b^{d(n-1)/n})$ for paranoid search. For *d* we use the maximum game-length of Rolit and for *b* the branching factor empirically approximated by one million Monte-Carlo simulations per Rolit configuration. The number of players is indicated by *n*.

Table 2 shows that for $4 \times 4$ Rolit PPNS creates larger search trees than paranoid search would do in the best-case. For $6 \times 6$ Rolit, PPNS generates larger search trees than the best-case for the two-player configuration. For the multi-player cases the outcome is different. The PPNS trees are smaller than the best-case trees of paranoid search. In the 3-player case, PPNS is smaller in the order of $10^5$, and in the 4-player case the factor is ca. $10^{11}$.

Based on the results in $6 \times 6$ Rolit we may conclude that PPNS is able to successfully exploit the non-uniformity of the game tree in multi-player games.

**Acknowledgments**

This research is financed by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project GO FOR GO, grant number 612.066.409.

# References

[1] L. V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, University of Limburg, The Netherlands, 1994.

[2] L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-Number Search. *Artificial Intelligence*, 66(1):91–124, 1994.

[3] D. M. Breuker, J. W. H. M. Uiterwijk, and H. J. van den Herik. The PN$^2$-search algorithm. In H. J. van den Herik and B. Monien, editors, *Advances in Computer Games 9*, pages 115–132. Universiteit Maastricht, Maastricht, The Netherlands, 2001.

[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'86)*, pages 158–162, 1986.

[5] N. R. Sturtevant and R. E. Korf. On pruning techniques for multi-player games. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'00)*, pages 201–207, 2000.

[6] H. J. van den Herik, J. W. H. M. Uiterwijk, and J. van Rijswijck. Games solved: now and in the future. *Artificial Intelligence*, 134(1-2):277–311, 2002.