New Mathematics and Natural Computation

BEST PLAY IN FANORONA LEADS TO DRAW

MAARTEN P.D. SCHADD, MARK H.M. WINANDS, JOS W.H.M. UITERWIJK,
H. JAAP VAN DEN HERIK and MAURICE H.J. BERGSMA

MICC-IKAT Games and AI Group, Faculty of Humanities and Sciences
Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands
{*maarten.schadd,m.winands,uiterwijk,herik*}*@micc.unimaas.nl,*
m.bergsma@student.unimaas.nl

Received Day Month Year
Revised Day Month Year

Fanorona is the national board game of Madagascar. The game's complexity is approximately the same as that of checkers. In this article we present a search-based approach for weakly solving this game. It is a well-chosen combination of Proof-Number search and endgame databases. Retrograde analysis is used to generate the endgame databases in which every position with 7 or fewer pieces on the board has been solved. Then, a Proof-Number search variant, PN^2, exploits the databases to prove that the game-theoretical value of the initial position is a draw. Future research should develop techniques for strongly solving the game.

Keywords: Fanorona; proof-number search; endgame databases.

1. Introduction

Already for half a century, building strong game-playing programs is one of the main targets of Artificial-Intelligence researchers. The principal aim is to witness the "intelligence" of computers. A second aim is to establish the *game-theoretic value* of a game, i.e., the outcome of the game when all participants play optimally. The game-theoretic value indicates whether a game is won, lost, or drawn from the perspective of the player who has to move first.

Pursuing the second aim is an exciting task; there, game solvers are looking for new techniques and new achievements in research. For solving a game, three "layered" definitions exist.[1]

Ultra-weakly solved: For the initial position, the game-theoretic value has been determined.

Weakly solved: For the initial position, a strategy has been determined to obtain at least the game-theoretic value of the game.

Strongly solved: For all legal positions, a strategy has been determined to obtain the game-theoretic value of the position.

In the last 20 years quite a number of games have been solved.[2] Below we provide examples for each definition. (1) The game of Hex is an instance of an ultra-weakly solved game.[3] The proof was published in 1953 by Shannon.[4] (2) The following games were weakly solved: Qubic,[5,6] Go-Moku,[1,7] Nine Men's Morris,[8] Domineering,[9,10,11] and Renju.[12] Checkers is the latest addition to this list.[13,14] (3) Recently, we saw considerable development in "layer" three: the games Connect-Four,[15] Kalah,[16] and Awari[17] were strongly solved.

Here we remark that different methods are used for solving games. We divide these methods into three classes: knowledge methods, special search methods and endgame databases.

(1) Knowledge methods are used to prune the search tree. Uiterwijk *et al.*[18] introduced rules in Connect-Four which made it possible to determine the game-theoretic value for non-terminal positions. Using these rules the game of Connect-Four has been solved.

(2) Special search methods (Mate-Solvers) are used to search the space efficiently. For instance, Threat-Space Search[19] was introduced in Go-Moku. This technique searches as if the opponent is allowed to play all counter moves to a threat at once. Using this technique, a series of threat sequences to win the game can be established in such a way that it is indifferent what the opponent plays. The technique reduces the size of the search tree significantly. So, it was possible to solve the game.[7]

(3) Endgame databases are used to store the game-theoretical value of each possible endgame position. Retrograde analysis[20] is a method to solve endgame positions iteratively, starting with the terminal ones. Romein and Bal[17] were able to solve every possible Awari position using parallel retrograde analysis, thus solving the game strongly.

Below, we describe how we weakly solved the game of Fanorona. From the starting position, we have a computational proof that Fanorona is a draw, assuming optimal play by both sides. In the article we explain our domain-dependent search-based approach that establishes the game-theoretic value of Fanorona. We are also able to compute the results for all smaller boards. The search-based approach is a combination of a mate solver (Proof-Number (PN) search) and pre-constructed endgame databases.

The paper is organized as follows. Section 2 provides background information on the game of Fanorona and on the rules to play this game legitimately. Section 3 presents various characteristics of the game. Section 4 explains the outcome of a retrograde analysis and shows the results in Fanorona. The proof-number search variant, PN^2, is explained in Section 5. Section 6 presents the results of a nicely tuned combination of PN^2 search and the corresponding databases. Section 7 describes the verification of the results. Finally, Section 8 provides the conclusions and lists future research topics.

2. Fanorona

Fanorona is a board game with its roots in Madagascar. It is derived from the game "Alquerque" which might be over 3000 years old. Below we explain the rules of Fanorona. The explanation is based on the rules given in Bell,[21] and in Chauvicourt and Chauvicourt.[22] There exist some variations on the rules, but we will focus on the most common variant. The goal of the game is to capture all opponent pieces. The game is a draw if neither player succeeds. Fanorona has three standard versions: *Fanoron-Telo*, *Fanoron-Dimyand*, and *Fanoron-Tsivy*. The difference between these versions is the board size. Fanoron-Telo is played on a 3×3 board and the difficulty of this game can be compared to that of Tic-Tac-Toe. Fanoron-Dimyand is played on a 5×5 board and Fanoron-Tsivy is played on a 5×9 board. We will call Fanoron-Tsivy from now on Fanorona, because it is the best-known board size and the main subject of this article.

2.1. *Board*

The Fanorona board consists of lines and intersections. A line represents the path along which a piece can move during the game. There are weak and strong intersections. On a weak intersection it is only possible to move a piece horizontally and vertically, while on a strong intersection it is also possible to move a piece diagonally. Figure 1 shows a weak intersection at **e2** and a strong intersection at **e3**. A piece can only move from one intersection to an adjacent intersection.

In the initial position each player has 22 pieces. They are placed as shown in Figure 1. Players move alternately; White plays first.

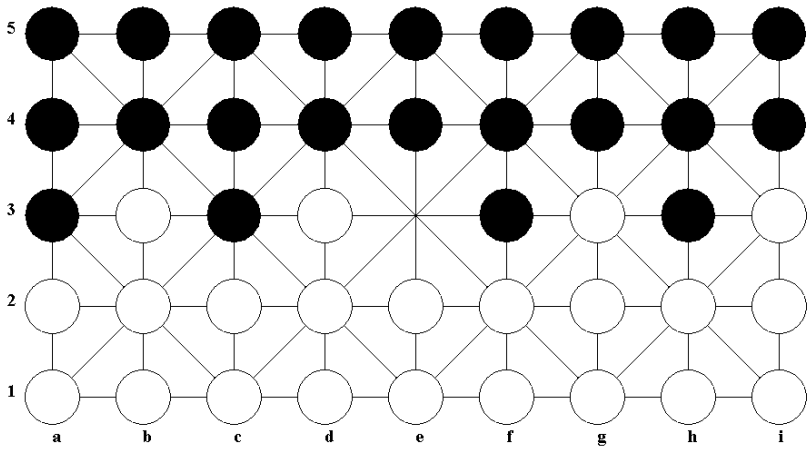
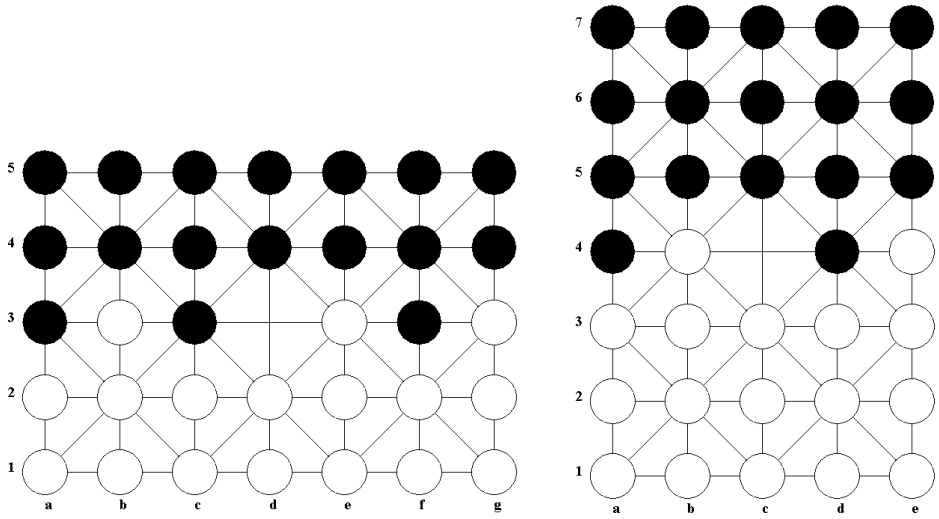


Fig. 1. The initial position of a Fanorona game.

Figure 2 shows two non-standard board sizes, the 5×7 and the 7×5 board. Please note that a game played on the 7×5 board is totally different from a game played on a 5×7 board. Rotating the 7×5 board to a 5×7 board creates a Fanorona game with a different 5×7 initial position (e.g., left Black, right White). It is standard in Fanorona that the white pieces are below the black pieces.



(a) 5x7: Initial position (b) 7x5: Initial position

Fig. 2. Initial positions on a smaller board.

2.2. *Movement*

We distinguish two kinds of moves, capturing and non-capturing moves. Capturing moves are *obliged* and have to be played, if possible, above non-capturing (paika) moves. We start explaining capturing moves because thereafter describing paika moves is straightforward.

Capturing implies removing one or more pieces of the opponent. It can be done in two different ways, either (1) by approach or (2) by withdrawal. An *approach* is the movement of the capturing piece to an intersection adjacent to an opponent piece provided that the opponent piece is situated on the continuation of the capturing piece's movement line. A *withdrawal* works analogously as an approach but the difference is that the movement is away from the opponent piece. When an opponent piece is captured, all opponent pieces in line with that piece and directly behind that piece (i.e., there is no interruption by an empty intersection or an own piece) are captured as well.

Figure 3 shows how the capturing mechanism works. We start with a straight-

forward capturing move. In the given position White can capture Black's piece on **d2** by moving his[a] white piece from **b2** to **c2**. By this move Black's piece on **e2** will be captured as well. **g2** will not be captured because there is no black piece on **f2**. This is called *capturing by approach* since White moves his piece towards Black's piece on **d2**.

White can also *capture by withdrawal* if he moves his white piece from **f4** to **e4** because now he is moving away from Black's piece on **g4**. The piece on **i4** will not be captured since there is a white piece on **h4** interrupting the line.

White cannot capture **c4** with **f4** because for a capturing move the own piece has to be next to the one captured after movement and **f4** is too far away to capture **c4**. In order to allow capturing the piece has to be moved to an intersection adjacent to the captured piece, if it is approached.

White also cannot capture **c4** with **b2** (moving diagonally) because **c4** is not on the extension of a movement line from **b2**. Only pieces can be captured which are located in the extension of the movement line of the capturing piece. Thus capturing "around a corner" is not allowed.

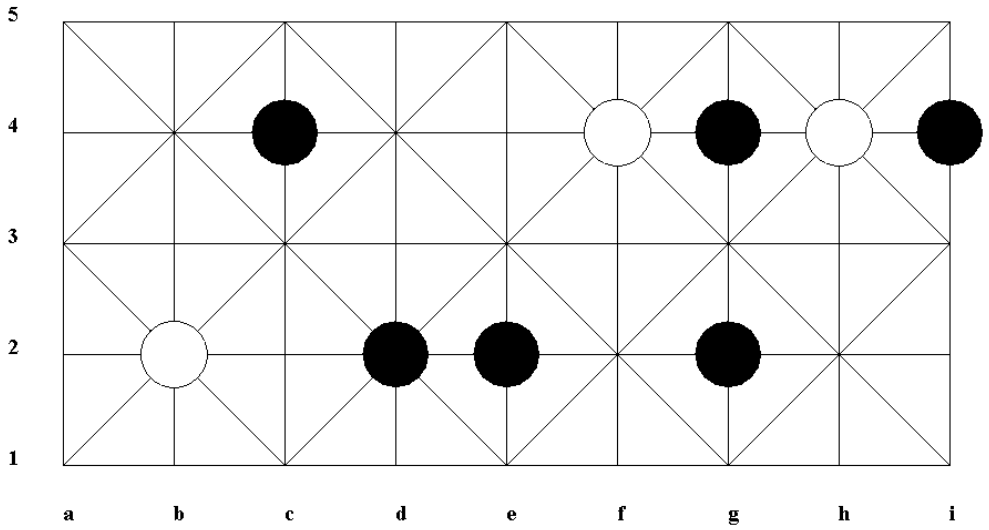


Fig. 3. An example position during a Fanorona game.

To describe a capturing by approach we define the following notation: **b2-c2A** meaning that the piece on **b2** moves to **c2** and approaches (A) the piece on **d2**. For a withdrawal the letter "W" is used. If a piece is moved without capturing any opponent piece then no letter "A" or "W" is used.

[a]For brevity, we use 'he' and 'his' when 'he or she' and 'his or her' are meant.

As in checkers it is allowed to continue capturing with the same piece as long as possible. We call this extended capturing. Figure 3 shows that White can capture **c4** with the move: **b2-c2A-c3A**. (Even if a move consists of multiple movements of one single piece it still counts as a single move.) Although a player is obliged to play a capturing move above a non-capturing move, the player may stop capturing after any number of opponent pieces are captured. This rule is different from the checkers rule where stopping a capturing sequence is not permitted. For instance, in Figure 3 White is allowed to stop early and plays only **b2-c2A**.

There are three more rules concerning capturing pieces. (1) It is not allowed to capture by approach and withdrawal at the same time. This is the case at the initial position shown in Figure 1 where White could play **d3-e3** as an approach or a withdrawal. In such a situation the player has to choose whether the current move is an approach or a withdrawal. (2) It is not allowed to make a capturing move in the same direction as the capturing move directly before. We illustrate this rule by referring to Figure 3. White is not allowed to play: **f4-e4W-d4A** because his piece would move twice in a row in the same direction. A player is allowed to play a capturing sequence two times into the same direction if a capturing movement in another direction is done in between. The last movement direction of the capturing move in the previous turn (i.e., before the last opponent move) does not influence possible capturing directions in the current turn. (3) The current capturing piece is not allowed to arrive on the same intersection twice during a capturing sequence. In Figure 3 White is not allowed to play **f4-e4W-e3A-f4W** because of this rule.

If no capturing move can be played, a non-capturing move is allowed. This is called a paika move and consists of moving one piece along a line to an adjacent intersection. White is not allowed to play the paika move **b2-b1** in the position depicted in Figure 3 because capturing is possible.

2.3. *End of the game*

The player who first captures all opponent pieces wins the game. The game is a draw if no player is able to win. In practice this is achieved by repetition of the same position with the same player to move.[23]

There does not exist any documentation stating what the outcome of the game would be if a player is not able to move. If this occurs during a game we define that a player who is not able to move forfeits the game. However, this situation rarely occurs and it is unlikely that the game-theoretical value would change if another outcome would be defined in this situation. It has still to be determined whether such situations can be reached legally.

3. Analyzing Fanorona

Two important indicators for establishing the questions (a) whether games can be solved and (b) which methods may be successful are (1) the game-tree complexity and (2) the state-space complexity.

An approximation for the game-tree complexity can be computed by performing a simulation. 10,000 games were played by two alpha-beta players, which performed a 4-ply deep search with a greedy evaluation function. The evaluation function consists of material difference and a random factor. We determined that the average game length of Fanorona is 43.81 moves (i.e., plies) and the average branching factor is 11.19 moves. This gives us a game-tree complexity of approximately 10^{46}. The state-space complexity has been computed as 10^{21} by using Eq. 4.1 (explained in the next section). These values are comparable to those of checkers, which has a game-tree complexity and state-space complexity of 10^{31} and 10^{20}, respectively. [1,13]

A typical game of Fanorona can be divided into two parts. In the first part of the game mostly capturing moves are played until most pieces are captured. In the second part, the endgame, mainly paika moves are played. Figure 4 shows the relation between capturing moves and paika moves as a function of the number of pieces on the board. This figure is based on 10,000 games played by alpha-beta players. In the initial position 44 pieces are on the board.

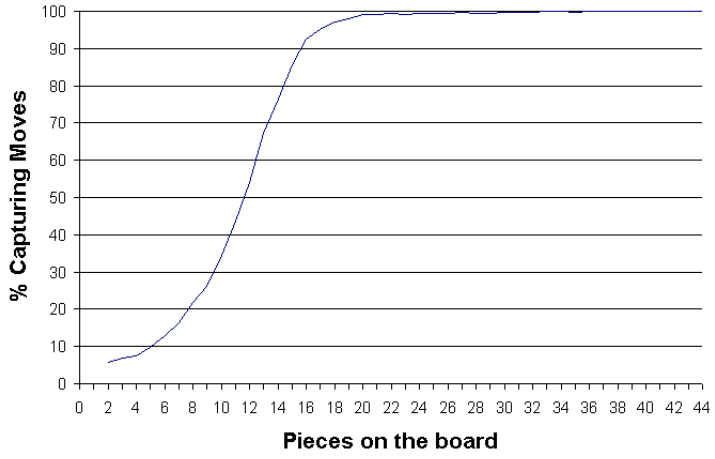


Fig. 4. Relation between capturing moves and paika moves.

Figure 4 shows that with 19 or more pieces on the board a capturing move is to be expected. We define the endgame as the part of the game where more paika than capturing moves are played. Figure 4 indicates that the endgame starts when there are fewer than 13 pieces on the board.

Figure 5 shows the average branching factor as a function of the number of pieces on the board, based on the same 10,000 games. We see in the figure that when the game enters the endgame the branching factor increases again. The reason for the increase of the average number of possible moves in the endgame is the occurrence of paika moves. The combination of a long endgame and a local maximum of the

branching factor results in a huge solution tree.

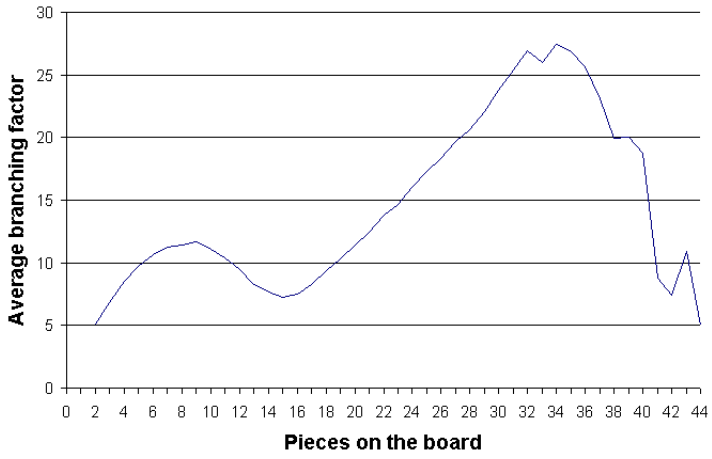


Fig. 5. The average branching factor as a function of the number of pieces on the board.

In order to cope well with the differences between both parts, different methods have been selected. For the endgame part, retrograde analysis has been selected to create endgame databases. There are three reasons, why endgame databases are a valuable tool for solving Fanorona. (1) Fanorona is a converging game.[2] (2) A large part of the game consists of positions with only a few pieces on the board. Fanorona has an average game length of 44 plies. But already after 21 plies there are on average fewer than 8 pieces on the board. (3) The endgame is not trivial. The branching factor has a local maximum in the endgame (see Figure 5) and there the game is converging more slowly.

Because of a large endgame where mostly paika moves are played, one would expect that an endgame database would decrease the size of the solution tree substantially. The expectation can be tested by doing a simulation. This was done in the following way.

A *virtual database* was used to finish a game early. For instance, if we assume that a 2-piece database is available, the game is stopped when a position with two pieces is reached. This is a terminal node because the outcome of the game can be retrieved from the virtual database. Using this approach we can estimate the change of both the average game length and the branching factor as a direct consequence of using larger databases. Table 1 shows that the virtual database was able to reduce the game-tree complexity substantially. In this experiment we see that the game-tree complexity decreases by a factor of more than 100 on average when the database size increases by one piece.

Because of the large amount of positions it is not possible to make an endgame

Table 1. Estimated complexities with increasing database size.

Database size	0	2	3	4	5
Average Game Length	43.81	42.60	40.23	36.23	31.62
Average Branching Factor	11.19	11.37	11.63	11.98	12.31
Log Game-Tree Complexity	45.96	44.97	42.88	39.07	34.48
Database size	**6**	**7**	**8**	**9**	**10**
Average Game Length	26.99	22.98	19.74	17.27	15.44
Average Branching Factor	12.59	12.82	13.02	13.26	13.53
Log Game-Tree Complexity	29.70	25.45	22.01	19.39	17.47

database up to the initial position. Therefore, for the first part of the game, a dedicated search method is needed. PN search has been chosen because the method is efficient for non-uniform trees. A non-uniform tree can be the result of many forced moves (e.g., capturing in Fanorona). Moreover, the use of an endgame database in the search tree makes the search tree non-uniform.

During the search the most-promising lines in the tree (i.e., lines where relatively the weakest resistance is expected) will be examined first because PN search uses a best-first tree traversal to find the solution tree. The combination of endgame databases and the fact that Fanorona converges fast may make PN search an efficient technique for this game. In the next two sections we will describe the two methods, retrograde analysis and PN search, in detail.

4. Retrograde Analysis

Retrograde analysis is a method to create omniscient endgame databases for board games.[24,20] Such endgame databases have proved to be vital to the strength of computer programs in quite a number of games.[25,26] For instance, in the game of checkers endgame databases have been built for up to 10 pieces remaining on the board.[27] The more board positions are stored in the endgame database, the earlier the search process can be terminated. Moreover, the method makes a deeper search possible in the middle game. Besides improving the playing strength of a program in the middle game, endgame databases can be used to solve a game as well. For instance, Romein and Bal[17] solved the game of Awari by storing the complete state space in a database.

A requirement for retrograde analysis is an index function. This function has to be a one-to-one mapping. Making such a function efficient can save a significant amount of space in the database.[28,29] The function we used consists of two parts: (1) a function to transform the pieces on the board to a number, independent of

the color of a piece and (2) a function to convert the order of black and white pieces to a number. The combined index function identifies uniquely all possible board positions disregarding symmetry. Such an index function is called gap-less. A gap-less function uses each index between the minimum and the maximum index. It is also invertible so that given an index, the corresponding board position can be computed.

Eq. 4.1 shows the index function used. We denote the total number of pieces on the board by M and the position of a piece i by S_i (which is a number between 0 and 44). We define W as the total number of white pieces and W_i is the place of white piece i in the sequence of white and black pieces on the board.

$$index = \sum_{i=1}^{M} \binom{S_i}{i} + \binom{45}{M} \times \sum_{i=1}^{W} \binom{W_i}{i} \tag{4.1}$$

Table 2 shows the size of each database if the above index function is used. Moreover, each position uses 2 bits of space on the hard disk, indicating that the position is a win, draw, or loss. So, for instance, we may state that the computation of the 9-piece database would be feasible on a regular desktop machine, where a total of 119.3 GB of hard disk space would be needed.

Table 2. Theoretical database size for up to 15 pieces.

# of Pieces	# of Positions
1	90
2	1,980
3	85,140
4	2,085,930
5	36,652,770
6	504,993,720
7	5,717,832,120
8	54,750,511,530
9	451,943,198,850
10	3,260,371,406,292
11	20,768,119,231,860
12	117,743,529,024,030
13	597,920,852,078,550
14	2,733,686,209,314,720
15	11,299,926,066,685,300

For speeding up the creation process paging was implemented. Paging[29] is a technique which stores frequently used parts of the database in the memory without

Table 3. Number of won, drawn, and lost positions in the databases for the 5×9 board. *a*-*b* indicates that the player with *a* stones is to move.

Db.	1-1	2-1	1-2	3-1	2-2	1-3
Win	158	10,366	717	149,458	127,756	4,188
Draw	334	398	3,231	91	79012	15,875
Loss	26	6	6,822	1	17,386	129,487
Db.	4-1	3-2	2-3	1-4		
Win	1,529,142	2,711,327	774,043	19,814		
Draw	12	327,836	1,252,162	88,187		
Loss	0	18,297	1,031,255	1,421,153		
Db.	5-1	4-2	3-3	2-4	1-5	
Win	12,223,788	30,095,407	24,137,779	4,180,200	81,728	
Draw	0	426,350	13,955,354	7,926,733	391,405	
Loss	0	32,491	2,644,731	18,447,315	11,750,655	
Db.	6-1	5-2	4-3	3-4	2-5	1-6
Win	79,431,164	237,393,018	344,370,238	145,408,435	18,659,090	302,021
Draw	0	774,868	46,020,564	170,633,688	41,896,491	1,509,775
Loss	0	108,614	6,724,158	81,072,837	177,720,919	77,619,368

writing them to the hard disk. When data from a page is needed the information can be retrieved from memory and no hard-disk access is needed.

During this research all databases with 7 or fewer pieces were computed for Fanorona. The computation was done by using a regular desktop pc with a Pentium IV 3.0 GHz processor and 256 MB RAM and took multiple months.

Table 3 shows the number of wins, draws, and losses in the databases for the 5×9 variant. Symmetric positions have been removed. *a*-*b* denotes that the player to move has *a* pieces and its opponent *b* pieces.

Table 3 indicates that the player to move has an advantage. One might suspect that a player who is to move and has more pieces than the opponent would win the game. The results are that (1) in the 4-1 database the player to move cannot lose any position, (2) in the 5-1 and 6-1 databases the player to move can win every position. So, an advantage of more than three pieces is needed for a forced win.

As an aside, we have created databases for the smaller Fanorona boards. We have computed (1) all endgame databases up to 7 pieces for the 5×7 board, (2) all endgame databases up to 9 pieces for the 5×5 board, and (3) all endgame databases up to 7 pieces for the 3×9 board.

5. Proof-Number Search

This section discusses the search procedure applied for weakly solving the game. The method used is Proof-Number (PN) search; it is briefly described in Subsection 5.1. A variant of PN search, called PN^2, is explained in Subsection 5.2.

5.1. *PN search*

PN search is a best-first search algorithm especially suited for finding the game-theoretical value of a game.[30] Its aim is to prove the true value of the root of a tree. A tree can have three values: *true*, *false*, or *unknown*. In the case of a forced win, the tree is *proved* and its value is *true*. In the case of a forced loss or draw, the tree is *disproved* and its value is *false*. Otherwise the value of the tree is *unknown*. In contrast to other best-first algorithms PN search does not need a domain-dependent heuristic evaluation function to determine the most-promising node to be expanded next. In PN search this node is usually called the *most-proving* node. PN search selects the most-proving node using two criteria: (1) the shape of the search tree (the branching factor of every internal node) and (2) the values of the leaves. These two criteria enable PN search to treat game trees with a non-uniform branching factor efficiently.[30]

5.2. *PN^2*

A disadvantage of PN search is that the whole search tree has to be stored in memory. Therefore, we use PN^2 as an algorithm to reduce memory requirements in PN search.[1,31] PN^2 consists of two levels of PN search. The first level consists of a PN search (pn_1), which calls a PN search at the second level (pn_2) for an evaluation of the most-proving node of the pn$_1$-search tree. This pn$_2$ search is bound by a maximum number of nodes N to be stored in memory. In our implementation (analogously to Allis[1]), N is equal to the size of the pn_1 tree. The pn$_2$ search is stopped when the number of nodes stored in memory exceeds N or the subtree is (dis)proved. After completion of the pn$_2$ search, the children of the root of the pn$_2$-search tree are preserved, but subtrees are removed from memory.[b]

The maximum size of the subtree for determining the PN numbers is an important factor for PN^2 search. With a larger tree the search would be more directed but would take more time to compute. Allis[1] proposed a variable size for the pn_2 tree which is dependent on the size of the pn_1 tree. Breuker *et al.*[31] used a fraction function to determine the size of the pn_2 tree. We did not use the latter approach because the fraction function is not efficient for large problems. We initialized the maximum size of the pn_2 tree equal to the size of the pn_1 tree.

[b]This only holds if the root is not proved or disproved.

Table 4. The game-theoretic values for different board sizes.

Board Size	*Winner*	*DB Size (Pieces)*	*Nodes*
3×3	White	0	122
3×5	White	0	2,490
5×3	White	0	1,491
3×7	White	0	87,210
7×3	White	0	172,101
5×5	Draw	9	108,593
3×9	White	5	209,409
9×3	White	5	262,217,017
5×7	Black	7	72,826,963
7×5	White	7	1,053,126
5×9	Draw	7	130,820,097,938

6. Results

This section presents the results obtained during our research. In Section 6.1 the game-theoretic values of Fanorona and its smaller board variants are given and the optimal solution for the 3×3 board is presented. Section 6.2 discusses an interesting observation regarding the behavior of the proof and disproof numbers.

6.1. *Solving Fanorona and its Board Variants*

PN^2 search is used in combination with endgame databases to compute the game-theoretic value of Fanorona and its smaller variants. After some tuning we arrived at the "ideal" combination for this configuration (i.e., 5×9 board on current hardware) viz. PN^2 with all databases including 7 or fewer pieces. By doing so, we were able to prove that Fanorona is a draw. For proving this, a total of 130,820,097,938 nodes were created during the search. Solving the initial position of Fanorona took more than a week when using our search-based approach on a computer with an AMD Opteron 2.6 GHz processor and 32 GB of RAM. Moreover, it has been proved that both the move **f2-e3A** and **d3-e3A** lead to a draw. Preliminary results suggest that by optimal play of both sides the move **e2-e3A** will lead to a win for Black. At this moment, we are unable to predict the game-theoretical outcomes of the other two opening moves (i.e., a win for Black or a draw).

Furthermore, we have solved the smaller variants of the game as well. An overview of the results of Fanorona and all smaller boards is given in Table 4. The column labeled *DB size* indicates which databases have been used. The column labeled *Nodes* indicates the total number of created nodes.

For smaller boards we may remark the following. If we have a look at Table

4, we see that all boards with a side equal to size 3 are a win for White. Thus, the starting player can exploit a narrow board and force a win. However, for most boards, with sides of at least size 5, White does not have this advantage anymore.

Table 4 shows differences between horizontal and vertical boards. The difference in number of nodes between the 5×7 and 7×5 boards (see Table 4 and Figure 2) can be explained by the fact that 5×7 is a win for Black (the second player), which is harder to prove. Furthermore, a substantial difference between the 3×9 and 9×3 boards can be observed. We conjecture that the average distance between the players' pieces is larger on vertical boards than on horizontal boards when entering the endgame. This would result in a slower convergence.

In order to provide insight into the strategies behind optimal play we show the solution for the 3×3 board. Figure 6 shows the game which will be played if both players play optimally.

Below, some interesting positions in this game are discussed. In Subfigure 6b Black has no choice of movement because there is only one possible capturing move. In Subfigure 6d Black has two choices. **b3-c3** would lead to a loss in 1 because White would answer with **b2-a1W-a2A**. Subfigure 6f is the most interesting one. Black decides to stop the capturing sequence after the capturing of only one of the two possible pieces. The reason for this is to postpone the eventual loss by one move.

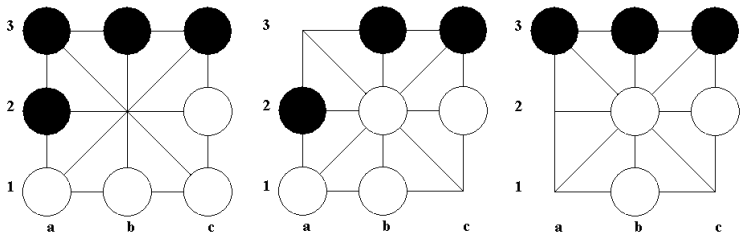
6.2. *Behavior of the PN search*

During this research we made an interesting observation. Figure 7 and Figure 8 show the development of proof and disproof number of the root node during the search. Figure 7 depicts a search with the goal that White can win the initial position. Figure 8 visualizes a search with the goal that White can at least draw the initial position. We found that the development of the proof and disproof number has a similar pattern for smaller boards and looks like a Chi-Square distribution. The number which goes to zero, independent of its nature (proof or disproof number), always reaches its peak in the beginning and has a long tail at the end. A similar pattern can be found when solving the small boards. For instance, this can be seen when solving the 7×3 board (See Figure 9).

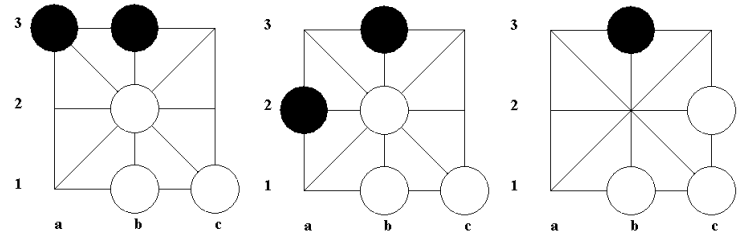
7. Correctness

It is important that the obtained results are free of errors. This section will describe two arguments why we are convinced that the resuls presented in this paper are correct.

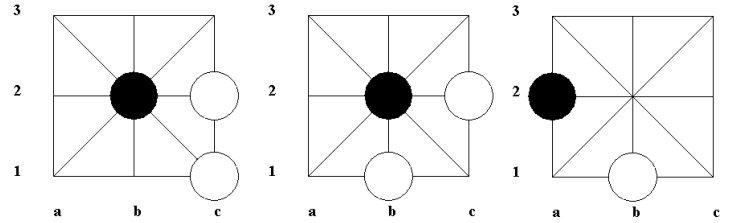
First, we verified the code for the retrograde analysis in three ways. (1) We solved endgame positions with the help of forward search and compared the obtained result with the entry in the database. (2) We did use the database code to construct Lines of Action (LOA) and Surakarta endgame databases which were verified independently by the programs MIA[32] and SIA,[33] respectively. (3) A consistency check was done in order to check for possible bit flips.[26] It turns out that



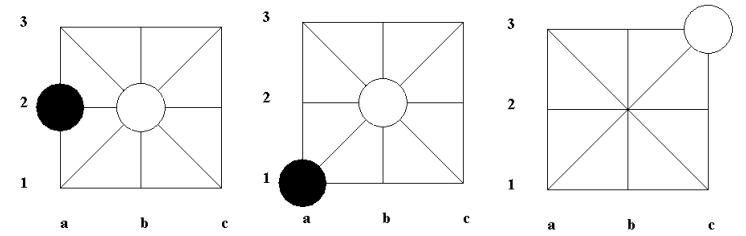
(a) Start (b) c1-b2A (c) a2-a3W



(d) c2-c1W (e) a3-a2 (f) b2-c2W



(g) b3-b2A (h) c1-b1 (i) b2-a2W



(j) b1-b2 (k) a2-a1 (l) b2-c3W

Fig. 6. 3×3 board: White can force a win.

no bit flips had occured.

Second, in order to verify PN search three actions have been taken. (1) The smaller boards were checked manually. This was possible for the 3×3, 5×3, and 3×5 boards. (2) PN search was compared to a standard alpha-beta search. Non-trivial 5×9 positions were solved by both solvers in order to detect differences.

(3) To handle the Graph-History-Interaction (GHI) problem[34,35] no transposition tables were used during the search. There exist methods which handle the GHI problem with PN search[36,37] but these were not implemented.

8. Conclusions and Future Research

This section starts with two conclusions drawn from our research. Thereafter, a discussion is presented. The section ends with future research.

8.1. *Conclusions*

Our first and main conclusion is that the game of Fanorona (played on the 5×9 board) has been weakly solved and is drawn when both players play optimally. This conclusion was arrived at by a well-chosen combination of proof-number search and endgame databases. Combining these two methods is a relatively new approach for solving a game. Simultaneously to our research, Schaeffer *et al.*[13,14] used a similar method for solving checkers. Endgame-database statistics show that (1) the player to move has an advantage and (2) that a draw can often be achieved in spite of having fewer pieces than the opponent. Second, we may conclude that White is

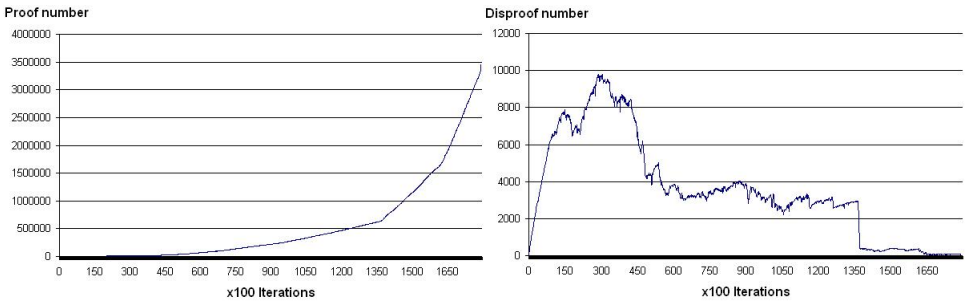


Fig. 7. White cannot win the initial Fanorona position.

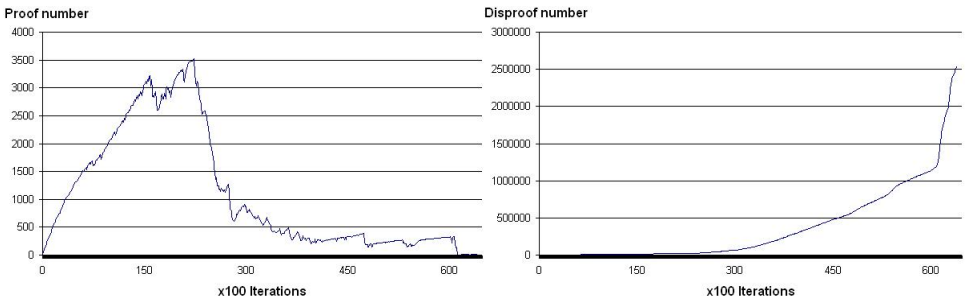


Fig. 8. White can at least draw the initial Fanorona position.

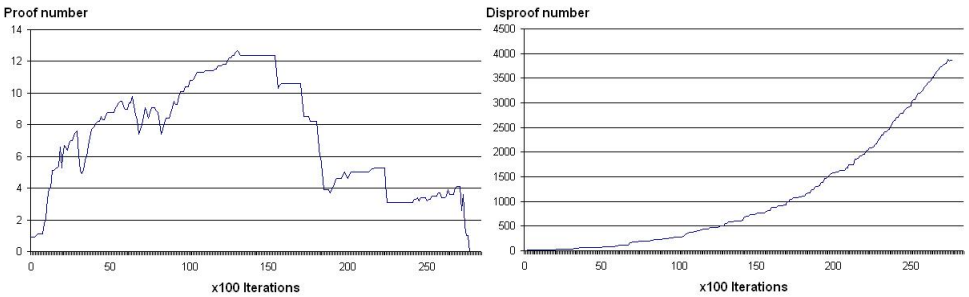


Fig. 9. White can win the initial position on the 7×3 board.

able to force a win on board sizes with one side equal to 3. We conjecture that for boards where both sides have at least size 5, White does not have this advantage for the majority of cases (so, we consider 7×5 as an exception because White still wins). The 9×5 board (please note the inversion) of the game has not been fully weakly solved up to now. Preliminary results suggest that this board is much harder to solve than the 5×9 board. We anticipate that the reason for this is the larger distance to the opponent when entering the endgame (see 6.1).

8.2. *Discussion*

The game-tree and state-space complexity of Fanorona are somewhat higher than those of checkers. Therefore, the following question may arise: why is this game easier to solve than checkers? The answer lies in the properties of the game (e.g., the decision complexity[38]). In Fanorona, capturing is almost always possible in the opening and middle game; often a large amount of pieces is then captured. Thus, the game converges fast to the endgame, where the endgame databases can take over the best-play procedure. The speed of the game convergence is not represented in the game-tree and state-space complexity. We are confident that if such a measure would be established, it would be higher for Fanorona than for checkers.

8.3. *Future Research*

In this contribution Fanorona has been *weakly* solved. We determined a strategy to achieve the game-theoretic value against any opposition starting from the initial position. Solving Fanorona strongly is a challenging subject for future research.

At this moment of time (2007) the 9×5 and all larger boards are unsolved. We believe that for solving these board sizes larger databases are required.

Finally, it would be interesting to investigate whether the proof and disproof number show a similar pattern as seen in Subsection 6.2 when solving other games.

Acknowledgments

This work is funded by the Dutch Organization for Scientific Research (NWO) for the project TACTICS, grant number 612.000.525.

References

1. L. V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, Rijksuniversiteit Limburg, Maastricht, The Netherlands, 1994.
2. H. J. van den Herik, J. W. H. M. Uiterwijk, and J. van Rijswijk. Games Solved: Now and in the Future. *Artificial Intelligence*, 134(1–2):277–311, 2002.
3. V. V. Anshelevich. The game of hex: An automatic theorem proving approach to game programming. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 189–194, AAAI Press, Menlo Park, CA, 2000.
4. C. E. Shannon. Computers and Automata. In *Proceedings of Institute of Radio Engineers*, volume 41, pages 1234–1241, 1953.
5. L. V. Allis and P. N. A. Schoo. Qubic solved again. In H. J. van den Herik and L. V. Allis, editors, *Heuristic Programming in Artificial Intelligence 3: The Third Computer Olympiad*, pages 192–204, Ellis Horwood Limited, Chichester, UK, 1992.
6. O. Patashnik. Qubic: 4x4x4 tic-tac-toe. *Mathematics Magazine*, 53(4):202–216, 1980.
7. L. V. Allis, H. J. van den Herik, and M. P. H. Huntjes. Go-moku solved by new search techniques. *Computational Intelligence*, 12(1):7–23, 1996.
8. R. U. Gasser. *Harnessing Computational Resources for Efficient Exhaustive Search*. PhD thesis, Swiss Federal Institute of Technology, Zürich, Switserland, 1995.
9. D. M. Breuker. *Memory versus Search in Games*. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands, 1998.
10. D. M. Breuker, J. W. H. M. Uiterwijk, and H. J. van den Herik. Solving 8×8 Domineering. *Theoretical Computer Science*, 230(1–2):195–206, 1999.
11. N. Bullock. Domineering: Solving large combinatorial search spaces. *ICGA Journal*, 25(2):67–84, 2002.
12. J. Wágner and I. Virág. Solving renju. *ICGA Journal*, 24(1):30–34, 2001.
13. J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007.
14. J. Schaeffer. Game over: Black to play and draw in checkers. *ICGA Journal*, 30(4):187–197, 2007.
15. J. T. Tromp. John's connect four playground. Retrieved 26.6.2007. http://homepages.cwi.nl/~tromp/c4/c4.html.
16. G. Irving, H. H. L. M. Donkers, and J. W. H. M. Uiterwijk. Solving Kalah. *ICGA Journal*, 23(3):139–147, 2000.
17. J. W. Romein and H. E. Bal. Solving awari with parallel retrograde analysis. *IEEE Computer*, 36(10):26–33, 2003.
18. J. W. H. M. Uiterwijk, L. V. Allis, and H. J. van den Herik. A knowledge-based approach to connect-four. In D. N. L. Levy and D. F. Beal, editors, *Heuristic Programming in Artificial Intelligence: the First Computer Olympiad*, pages 113–133. Ellis Horwood Limited, Chichester, UK, 1989.
19. L. V. Allis, H. J. van den Herik, and M. P. H. Huntjes. Go-moku and threat-space search. Technical Report CS 93-02, Department of Computer Science, Faculty of General Sciences, Rijksuniversiteit Limburg, Maastricht, The Netherlands, 1993.
20. T. Ströhlein. *Untersuchungen über kombinatorische Spiele*. PhD thesis, Fakultät für Allgemeine Wissenschaften der Technischen Hochschule München, München, Germany, 1970. In German.

21. R. C. Bell. *Board and Table Games from Many Civilizations*. Dover Publications, New York, USA, 1980.
22. J. Chauvicourt and S. Chauvicourt. *Le Fanorona – Jeu National Malgache*. Nouvelle Imprimerie des Arts Graphiques, Tananarive, Madagascar, 1980. In French.
23. M. P. D. Schadd. *Solving Fanorona*. Master's thesis, Universiteit Maastricht, Maastricht, The Netherlands, 2006.
24. H. J. van den Herik and I. S. Herschberg. The construction of an omniscient endgame database. *ICGA Journal*, 8(2):66–87, 1985.
25. E. A. Heinz. Endgame databases and efficient index schemes. *ICGA Journal*, 22(1):22–32, 1999.
26. J. Schaeffer. *One Jump Ahead : Challenging Human Supremacy in Checkers*. Springer-Verlag, New York, USA, 1997.
27. J. Schaeffer, Y. Björnsson, N. Burch, R. Lake, P. Lu, and S. Sutphen. Building the checkers 10-piece endgame databases. In H. J. van den Herik, H. Iida, and E. A. Heinz, editors, *Advances in Computer Games 10*, pages 193–210, Dordrecht, The Netherlands, 2003. Kluwer Academic Publishers.
28. S. T. Dekker, H. J. van den Herik, and I. S. Herschberg. Perfect knowledge revisited. *Artificial Intelligence*, 43(1):111–123, 1990.
29. R. Lake, J. Schaeffer, and P. Lu. Solving large retrograde-analysis problems using a network of workstations. In H. J. van den Herik, I. S. Herschberg, and J. W. H. M. Uiterwijk, editors, *Advances in Computer Chess 7*, pages 135–162, Rijksuniversiteit Limburg, Maastricht, The Netherlands, 1994.
30. L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-number search. *Artificial Intelligence*, 66(1):91–124, 1994.
31. 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.
32. M. H. M. Winands. *Informed Search in Complex Games*. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands, 2004.
33. M. H. M. Winands. SIA wins Surakarta Tournament. *ICGA Journal*, 30(3):162, 2007.
34. M. Campbell. The graph-history interaction: On ignoring position history. In *1985 Association for Computing Machinery Annual Conference*, pages 278–280, 1985.
35. A. J. Palay. *Searching with Probabilities*. PhD thesis, Boston University, Boston MA, USA, 1985.
36. D. M. Breuker, H. J. van den Herik, J. W. H. M. Uiterwijk, and L. V. Allis. A solution to the GHI problem for best-first search. *Theoretical Computer Science*, 252(1–2):121–149, 2001.
37. A. Kishimoto and M. Müller. A general solution to the graph history interaction problem. In D. L. McGuinness and G. Ferguson, editors, *AAAI*, pages 644–649, San Jose, California, USA, 2004. AAAI Press / The MIT Press.
38. L. V. Allis, H. J. van den Herik, and I. S. Herschberg. Which games will survive? In D. N. L. Levy and D. F. Beal, editors, *Heuristic Programming in Artificial Intelligence 2: the Second Computer Olympiad*, pages 232–243. Ellis Horwood Limited, Chichester, UK, 1991.