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. Email: {maarten.schadd,m.winands,uiterwijk,herik}@micc.unimaas.nl, m.bergsma@student.unimaas.nlIntroduction

1. Introdcution

Solving a game is an exciting task; there, game solvers are looking for new achievements in research. In the last 25 years quite some games have been solved [van den Herik et al. (2002)]. Here solving means that the program is always able to achieve the best (i.e., game-theoretic) value independent of the opponent. This is called weakly solving a game [Allis (1994)]. We mention Connect-Four [Allis (1988)], Qubic [Patashnik (1980); Allis and Schoo (1992)], Go-Moku [Allis (1994)], Nine Men's Morris [Gasser (1995)], and Domineering [Breuker (1998)] as weakly solved games. More recently we saw the solutions of Kalah [Irving et al. (2000)], Renju [W´agner and Vir´ag (2001)], and Awari [Romein and Bal (2003)].

To the above list we may now add 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 this paper we present our search-based approach to establish the game-theoretic value of Fanorona. Moreover, we provide some results of its smaller variants. Our search-based approach is a combination of mate solver (Proof-Number (PN) search) and endgame databases.

2. Fanorona

Below we explain the rules of Fanorona. The explanation is based on the rules given in Bell (1980), and in Chauvicourt and Chauvicourt (1980). Fanorona is a board game with its roots in Madagascar. It is a descendent derived from the game "Alquerque" which might be over 3000 years old. The goal of the game is to capture all opponent stones. The game is a draw if neither player succeeds.

Fanorona is played on a 5×9 board, consisting of lines and intersections. A line represents the path along which a stone can move during the game. There

are weak and strong intersections. At a weak intersection it is only possible to move a stone horizontally and vertically, while on a strong intersection it is also possible to move a stone diagonally. A stone can only move from one intersection to an adjacent intersection.

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

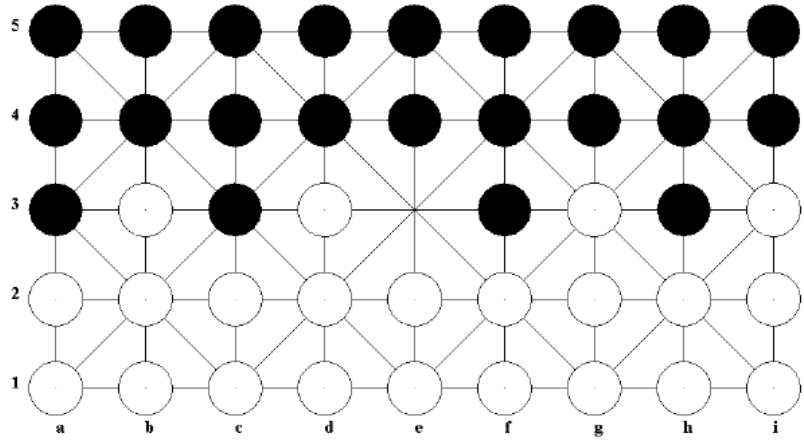

Fig. 1 The initial position of a Fanorona game.

We distinguish two kinds of moves, non-capturing and capturing moves. A non-capturing move is called a *paika* move and consists of moving one stone along a line to an adjacent intersection. Capturing moves are *obliged* and have to be played above paika moves.

Capturing implies removing one or more stones 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 stone to a point adjacent to an opponent stone provided that the stone is situated on the precise extension of the movement line of the capturing stone. A *withdrawal* works analogously to an approach but the difference is that the movement is away from the opponent stone. When an opponent stone is captured, all opponent stones in line behind that stone (as long as there is no interruption by an empty point or an own stone) are captured as well.

As in Checkers it is allowed to continue capturing with the same stone as long as possible. A player is obliged to play a capturing move above a non-capturing move but a player is allowed to stop with capturing after any number of opponent stones are captured. This rule is different from the Checkers rule where stopping a capturing sequence is not permitted. More details on the rules can be found in Schadd (2006).

3. Solving Fanorona

Two important indicators whether games can be solved and which methods may be successful are the game-tree and state-space complexity. Using random players, an upper bound for these values can be computed. We determined that Fanorona has a game-tree complexity of at most 10^{47} and a state-space complexity of at most 10^{21} [Schadd (2006)]. 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.

A typical Fanorona game can be divided into two parts. In the first part mostly capturing moves are played until the majority of stones are captured. In the second part, the endgame, mostly paika moves are played. Analysis based on random players showed that with 11 stones or fewer more paika than capturing moves will be played.

In order to cope well with the differences between the two parts different methods have to be selected. For the second part, retrograde analysis has been selected to create endgame databases. For three reasons, endgame databases are a valuable tool for investigating Fanorona. (1) Fanorona is a converging game. (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 45 ply. But already after 21 ply there are on average fewer than 8 pieces. (3) The endgame is not trivial. The branching factor has a local maximum in the endgame (because mainly paika moves are played).

Because of the large amount of positions it is not possible to make an endgame database up to the initial position. Therefore, for the first part, a 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.

4. Retrograde Analysis

Retrograde analysis is a method to create endgame databases for board games [Strohlein (1970); van den Herik and Herschberg (1985)]. Endgame databases have proven to be vital the strength of computer programs in quite some games [Schaeffer (1997); Heinz (1999)]. The more board positions are stored in the

endgame database, the earlier the search process can be supported by interruption. 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 (2003) 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 [Dekker et al. (1990); Lake et al. (1994)]. The function we used consists of two parts. (1) A function to transform the stones on the board to a number, independent of the color of a stone and (2) a function to convert the order of black and white stones to a number. This gives 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 can be computed.

Each position uses 2 bits of space on the hard disk, indicating that the position is a win, draw, or loss. The computation of the 9-piece database is feasible on a normal desktop machine, where a total of 119.3 GB of hard disk space would be needed.

For speeding up the creation process paging was implemented. Paging [Lake et al. (1994)] is a technique which stores frequently used parts of the database in the memory without writing them to the hard disk. If data on a page is needed, then the information can be retrieved from memory and no hard-disk access is needed. Furthermore, a consistency check was run to detect possible bit flips [Schaeffer (1997)].

During this research all databases with 7 or fewer pieces were computed for Fanorona. The computation was done on a normal desktop pc with a Pentium IV 3.0 Ghz processor and 256 Mb RAM.

Table 1 shows the amount of wins, draws and losses in the database for the 5×9 variant. Symmetric positions have been removed. *a-b* denotes that the player to move has *a* stones and its opponent *b* stones. Table 1 indicates that the player to move has an advantage. One might suspect that a player who is to move and has more stones 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 against optimal play, (2) in the 5-1 and 6-1 databases the player to move wins every position against optimal play. So, an advantage of more than three stones is needed for a secured win.

As an aside, we have created databases for the smaller Fanorona variants. We have computed (1) all endgame databases up to 7 pieces for the 5×7 variant,

(2) all endgame databases up to 9 pieces for the 5×5 variant, and (3) all endgame databases up to 7 pieces for the 3×9 variant.

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

Table 1 Number of win, draw and loss positions in the databases for the 5×9 variant.

5. Proof-Number Search

PN search is a best-first search algorithm especially suited for finding the game-theoretical value in game trees [Allis et al. (1994)]. 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.

In PN search each node has a proof and a disproof number. In the naïve implementation, proof and disproof numbers are each initialised to unity in the unknown leaves. In our implementation we used a greedy approach to initialize proof and disproof numbers. The proof number is initialized as the ratio of black and white stones. The disproof number is calculated as the inverse of the proof number. With this approach branches where a player has more pieces than the opponent are searched first. This method resembles *stones pn* described by Allis (1994) with the difference that, e.g., 4-2 positions will be preferred above 16-14 positions. This heuristic was tested on the 5×5 variant of the game and reduces the size of the solution tree significantly [Schadd (2006)].

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 [Allis (1994); Breuker et al. (2001)]. 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, N is equal to the size of the pn_1 tree [cf. Allis (1994)]. 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 pn2-search tree are preserved, but subtrees are removed from memory.

6. Results

PN2 search in combination with endgame databases were used to compute the game-theoretic values of Fanorona and its smaller variants. The results are given in Table 2. The column labeled DB *size* indicates which databases have been used. The column labeled *Nodes* indicates the total number of created nodes. We were able to prove that Fanorona is a draw as shown in Table 2. Solving the initial position of Fanorona took more than a week when using our search-based approach. Moreover, it has been proven that both the move **f2-e3A** and **d3-e3A** lead to a draw.

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

Table 2 The game-theoretic values for different board sizes.

For smaller variants we may remark the following. If we have a look at Table 2, we see that all variants 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 variants, with sides of at least size 5, White does not have this advantage anymore.

7. Conclusion and Future Research

Our main conclusion is that the game of Fanorona (played on the 5×9 board) is drawn when both players play optimally. This result was achieved by a combination of proof-number search and endgame databases. Endgame-database statistics show that the player to move has an advantage and that a draw can often be achieved in spite of having fewer pieces than the opponent. Furthermore, we may conclude that White is 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 (i.e., we consider 7×5 as an exception).

In this contribution Fanorona has been *weakly* solved. We determined a strategy to achieve the game-theoretic value against any opposition for the starting position. When we would have determined such a strategy for all legal positions, the game would have been strongly solved [Allis (1994)]. This is subject for future research.

Acknowledgement

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

Reference

1. Allis, L. V. (1988). A Knowledge-Based Approach of Connect-Four. The game is solved: White wins, Master's thesis, Faculty of Mathematics and Computer Science, Free University, Amsterdam, The Netherlands.
2. Allis, L. V. (1994). Searching for Solutions in Games and Artificial Intelligence, Ph.D. thesis, Rijksuniversiteit Limburg, Maastricht, The Netherlands.
3. Allis, L. V. and Schoo, P. N. A. (1992). Qubic solved again, in H. J. van den Herik and L. V. Allis (eds.), Heuristic Programming in Artificial Intelligence 3: The Third Computer Olympiad (Ellis Horwood Limited, Chichester, UK), pp. 192–204.
4. Allis, L. V., Meulen, M. van der and Herik, H. J. van den (1994). Proof-number search, Artificial Intelligence 66, 1, pp. 91–124.
5. Bell, R. (1980). Board and Table Games from Many Civilizations (Dover Publications).
6. Breuker, D. M. (1998). Memory versus Search in Games, Ph.D. thesis, Universiteit Maastricht, Maastricht, The Netherlands.
7. Breuker, D. M., Uiterwijk, J. W. H. M. and Herik, H. J. van den (2001). The
8. PN2-search algorithm, in H. J. van den Herik and B. Monien (eds.), Advances in Computer Games 9 (Universiteit Maastricht, Maastricht, The
9. Netherlands), pp. 115–132.

10. Chauvicourt, J. and Chauvicourt, S. (1980). Le Fanorona − Jeu National Malgache (Nouvelle Imprimerie de Arts Graphiques).
11. Dekker, S. T., Herik, H. J. van den and Herschberg, I. S. (1990). Perfect knowledge revisited, Artificial Intelligence 43, 1, pp. 111–123.
12. Gasser, R. U. (1995). Harnessing Computational Resources for Efficient Exhaustive Search, Ph.D. thesis, Swiss Federal Institute of Technology, Z¨urich, Switserland.
13. Heinz, E. A. (1999). Endgame databases and efficient index schemes, ICGA Journal 22, 1, pp. 22–32.
14. Herik, H. J. van den and Herschberg, I. S. (1985). The construction of an omniscient endgame database, ICGA Journal 8, 2, pp. 66–87.
15. Herik, H. J. van den, Uiterwijk, J. W. H. M. and Rijswijk, J. van (2002). Games solved: Now and in the future, Artificial Intelligence 134, 1–2, pp. 277–311.
16. Irving, G., Donkers, H. H. L. M. and Uiterwijk, J. W. H. M. (2000). Solving
17. kalah, ICGA Journal 23, 3, pp. 139–1–48.
18. Lake, R., Schaeffer, J. and Lu, P. (1994). 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 (eds.), Advances in Computer Chess 7 (Universiteit Maastricht, Maastricht, The Netherlands), pp. 165–162.
19. Patashnik, O. (1980). Qubic: 4x4x4 tic-tac-toe, Mathematics Magazine 53, pp. 202–216.
20. Romein, J. W. and Bal, H. E. (2003). Solving awari with parallel retrograde
21. analysis, IEEE Computer 36, 10, pp. 26–33.
22. Schadd, M. P. D. (2006). Solving Fanorona, M. Sc. thesis, Universiteit Maastricht, Maastricht, The Netherlands.
23. Schaeffer, J. (1997). One Jump Ahead : Challenging Human Supremacy in Checkers (Springer-Verlag, New York, USA).
24. Strohlein, T. (1970). Untersuchungen uber kombinatorische Spiele, Ph.D. thesis, Fakult¨at f¨ur Allgemeine Wissenschaften der Technischen Hochschule M¨unchen, Munchen, Germany.
25. W´agner, J. and Vir´ag, I. (2001). Solving renju, ICGA Journal 24, 1, pp. 30–34.