NOTE

6×6 LOA IS SOLVED

Mark H.M. Winands[1]

Maastricht, The Netherlands

ABSTRACT

Lines of Action (LOA) is a two-person zero-sum game with perfect information; it is a chess-like game with a connection-based goal. In this note the 6×6 variant of LOA is weakly solved. A Proof-Number search method, PN^2, is used to compute that the game is a win for the first player (Black). Based on the results of solving the boards up to 6×6, an optimistic time frame is given for solving 7×7 and 8×8.

1. INTRODUCTION

In the last 20 years a number of non-trivial games have been solved (van den Herik, Uiterwijk, and van Rijswijck, 2002), meaning that the perfect playing strategy for all players as well as the resulting game-theoretic value are 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-theoretic value is determined, not the playing strategy). This includes games such as Connect-4 (Allis, 1988), Go-Moku (Allis, van den Herik, and Huntjens, 1996), and Nine Men's Morris (Gasser, 1995). More recently, the following games have been solved: Kalah (Irving, Donkers, and Uiterwijk, 2000), Renju (Wágner and Virág, 2001), Awari (Romein and Bal, 2003), Checkers (Schaeffer *et al.*, 2007), and Fanorona (Schadd *et al.*, 2008).

This note will focus on the game Lines of Action (LOA) (Sackson, 1969). It is a connection game, played on an 8×8 board. The game-tree complexity is estimated to be $O(10^{64})$ and the state-space complexity $O(10^{23})$ (Winands, 2004), making the game comparable in complexity to Othello (Allis, van den Herik, and Herschberg, 1991). Considering the current state-of-the-art computer techniques, LOA is not solvable by brute-force methods (Winands, 2004). Before 2001, Darse Billings (2002) was able to compute the game-theoretic values for the smaller LOA boards up to 5×5. The 6×6 board was regarded as (too) difficult at that time (Billings, 2002). The goal of this note is twofold, (1) weakly solving 6×6 LOA and (2) obtaining a better insight into the difficulties of solving standard (8×8) LOA.

The note is organised as follows. Section 2 provides background information on the game of LOA and the rules to play this game legitimately. Proof-Number search, which is used to solve 6×6 LOA, is explained in Section 3. Section 4 presents the results. Finally, a time frame is given for solving larger LOA boards in Section 5.

2. LINES OF ACTION

Lines of Action (LOA) is a two-person zero-sum game with perfect information; it is a chess-like game with a connection-based goal, normally played on an 8×8 board. LOA was invented by Claude Soucie around 1960. Sid Sackson (1969) described the game in his first edition of *A Gamut of Games*. The objective of a connection game is to group the pieces in such a way that they (1) connect two opposite edges (a static goal) *or* (2) form a fully-connected group (a dynamic goal). The precise definition of what constitutes a connection depends on the game in question. Whatever the case, the notion of connection became one of the great themes in the world

[1]Games and AI Group, Department of Knowledge Engineering, Faculty of Humanities and Sciences, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands. Email: m.winands@maastrichtuniversity.nl

of abstract gaming. Many prominent game inventors made a contribution to this theme. Other examples of connection games are TwixT (Bush, 2000) and Hex (Anshelevich, 2002). In contrast to the typical connection games, LOA is more chess-like because (1) pieces are moved over the board instead of placed on the board, and (2) pieces can be captured. The remainder of this note will focus on the 6×6 variant of the game.

2.1 The Rules

6×6 LOA is played by two sides, Black and White. Each side has eight (checker) pieces at its disposal. In this note the rules are applied, which are used for the standard board at the Computer Olympiads and at the MSO World Championships. Below they are formulated in 9 rules.

1. The black pieces are placed in two rows along the top and bottom of the board, while the white pieces are placed in two files at the left and right edge of the board (see Figure 1a).
2. The players alternately move a piece, starting with Black.
3. A move takes place in a straight line, exactly as many squares as there are pieces of either colour anywhere along the line of movement (see Figure 1b).
4. A player may jump over its own pieces.
5. A player may not jump over the opponent's pieces, but can capture them by landing on them.
6. The goal of a player is to be the first to create a configuration on the board in which all own pieces are connected in one unit. Connected pieces are on squares that are adjacent, either orthogonally or diagonally (e.g., see Figure 1c). A single piece is a connected unit.
7. In the case of simultaneous connection, the game is drawn.
8. If a player cannot move, this player has to pass.
9. If a position with the same player to move occurs for the third time, the game is drawn.

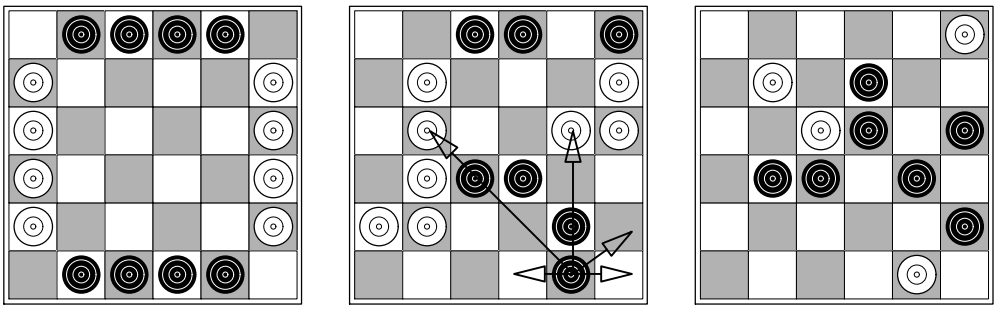

Figure 1: (a) The initial position. (b) An example of legal moves. (c) Terminal position: Black wins.

In the note, the standard chess notation is used for LOA. The possible moves of the black piece on **e1** in Figure 1b are indicated by arrows.

2.2 Characteristics of 6×6 LOA

Analysis of self-play matches showed that 6×6 LOA has an average branching factor of 18 and an average game length of 26 ply. The game-tree complexity is estimated to be $O(10^{32})$ and the state-space complexity $O(10^{13})$. A characteristic property of LOA is that it is a converging game (Allis, 1994); during the game the number of pieces (usually) decreases. However, since most 6×6 terminal board positions have on average 11 pieces remaining on the board, (small) endgame databases are not effectively applicable in LOA. Therefore, an endgame solver is applied, namely Proof-Number search to solve 6×6 LOA. The algorithm is discussed in the next section.

3. PROOF-NUMBER SEARCH

Proof-Number (PN) search is a best-first search method especially suited for finding the game-theoretic value in game trees (Allis, van der Meulen, and van den Herik, 1994). PN search was used successfully to solve LOA endgame positions by various researchers in the past (Winands, Uiterwijk, and van den Herik, 2003b; Sakuta *et al.*, 2003; Pawlewicz and Lew, 2007).

The aim of PN search 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 can be defined as the *most-proving* node (cf. Allis, 1994). 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 the naive implementation, proof and disproof numbers are each initialised to unity in the unknown leaves. In other implementations, the proof number and disproof number are set to 1 and *n*, respectively, for an OR node (and the reverse for an AND node), where *n* is the number of legal moves. In LOA it would mean that the *mobility* of the moving player in the position is taken into account, which is an important feature in the evaluation function as well (cf. Winands, Van den Herik, and Uiterwijk, 2003a) . In general, mobility speeds up PN with a factor 5 to 6 in time (Winands, 2004).

A disadvantage of PN search is that the whole search tree has to be stored in memory. Therefore, PN^2 can be used as an algorithm to reduce memory requirements in PN search (Allis, 1994; Breuker, Uiterwijk, and van den Herik, 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 (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 pn$_2$-search tree are preserved, but subtrees are removed from memory.

4. SOLVING LOA

PN^2 was used to compute the game-theoretic value of 6×6 LOA. After 100 hours of computation using an AMD Opteron 2.2 GHz., PN^2 was able to prove that the game is a win for Black. This is achieved by playing **b6-b4** in the initial position. For proving this, a total of 220,375,986,787 nodes were investigated. The maximum size of the first level tree during the search was 2,099,005 nodes. The development of the proof number of the root node is given in Figure 2. We see that the proof number (1) increases till 60,000 iterations, (2) appears to stabilize for some time, and (3) grows again after 100,000 iterations. When the top is reached after 160,000 iterations, the proof number drops quite fast to zero. Apparently, the number of lines in which Black faces considerable opposition is limited.

Taking symmetry into account, PN^2 was also applied to compute the game-theoretic values for the five remaining opening moves. The search was able to prove that **c6-e4** leads to a win for Black as well. In order to do this, a total of 753,110,070,891 nodes had to be investigated. Proving **c6-e4** took more than three times the effort as proving **b6-b4**. The other four opening moves, **b6-f6**, **b6-d4**, **c6:a4**, and **c6-c4** were all disproved by PN^2 (i.e., Black does not win).

An overview of the game-theoretic results of all the boards up 6×6 is given in Table 1. Except for 3×3, Black

Board Size	Outcome	Explored nodes
3×3	White wins	6
4×4	Black wins (b4-b2)	43
5×5	Black wins (c5-c3)	349,076
6×6	Black wins (b6-b4)	220,375,986,787

Table 1: Game-theoretic results up to 6×6.

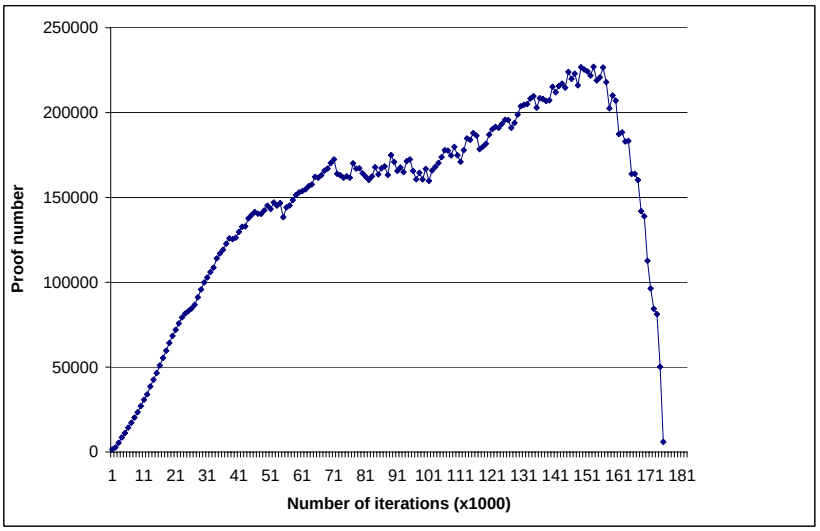

Figure 2: Development of the proof number.

always wins. Moreover, Table 1 shows that solving 5×5 took 10^4 times more effort than 4×4, and 6×6 took 10^6 times more effort than 5×5. Based on these results, one may estimate that solving 7×7 LOA would take at least 10^6 times more nodes to be searched than 6×6. The author expects that a similar complexity jump would hold for solving 8×8 as well.

5. FUTURE

In this note the game-theoretic value of 6×6 LOA was established to be a win for Black. The question now arises when 7×7 and 8×8 can be solved. As discussed in Section 4, 6×6 took 10^6 more time than 5×5. The author believes that at least a similar complexity jump of a factor 10^6 would occur for solving 7×7 LOA. There are four factors that would make solving 7×7 LOA feasible in 10 years: (1) processor speed, (2) massive parallelization, (3) running time, and (4) search improvements. Regarding the first point the author assumes that in ten years there will be at least an increase of processor speed by a factor 10.[2] The second point of massive parallelization will have a bigger impact. Although it seems that parallelizing PN search is quite difficult, a speedup of a factor 100 should be possible when using 100 processors with 16 cores. Another factor of 100 can be achieved by the third factor. Running the program for a year instead of 4 days will give us that number. A potential problem when running PN for a long time is the amount of main memory needed. Depth-first PN variants such as df-pn and PDS are good alternatives for solving LOA positions (Pawlewicz and Lew, 2007). Regarding the fourth factor, improving the search by a better leaf initialization, a specialized data structure for 7×7, and potential new PN-search enhancements could achieve a speedup factor of 10. When these factors are combined together ($10 \times 100 \times 100 \times 10 = 10^6$), it could be sufficient to solve the 7×7 board assuming it is only 10^6 more difficult than the 6×6 board.

When the 8×8 board will be solved is more speculative. Forward search methods such as PN search are probably not sufficient to solve the game in the next 20 years (i.e., before 2020). However, it is not unlikely that databases of at least 16 pieces, which are large enough to help the forward search, are available by then. This would have quite some impact on the size of the search tree. So, if all assumptions would hold, it means that in the best case 8×8 LOA could be solved around 2030.

6. REFERENCES

Allis, L. V. (1988). A Knowledge-Based Approach of Connect Four: The Game is Over, White to Move Wins. M.Sc. thesis, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands. Report No. IR-163.

[2]The author assumes that processor speed will approximately double every three years, leading to a tenfold increase in the next decade (cf. ITRS, 2007).

Allis, L. V. (1994). *Searching for Solutions in Games and Artificial Intelligence.* Ph.D. thesis, Rijksuniversiteit Limburg, Maastricht, The Netherlands.

Allis, L. V., Herik, H. J. van den, and Herschberg, I. S. (1991). Which Games Will Survive? *Heuristic Programming in Artificial Intelligence 2: the Second Computer Olympiad* (eds. D. N. L. Levy and D. F. Beal), pp. 232–243. Ellis Horwood, Chichester, England.

Allis, L. V., Herik, H. J. van den, and Huntjens, M. P. H. (1996). Go-Moku Solved by New Search Techniques. *Computational Intelligence*, Vol. 12, No. 1, pp. 7–24.

Allis, L. V., Meulen, M. van der, and Herik, H. J. van den (1994). Proof-Number Search. *Artificial Intelligence*, Vol. 66, No. 1, pp. 91–123.

Anshelevich, V. V. (2002). A Hierarchical Approach to Computer Hex. *Artificial Intelligence*, Vol. 134, Nos. 1–2, pp. 101–120.

Billings, D. (2002). Fred Kok (Netherlands) vs Mona. www.cs.ualberta.ca/∼games/LOA/fredkok-mona.html.

Breuker, D. M., Uiterwijk, J. W. H. M., and Herik, H. J. van den (2001). The PN^2-Search Algorithm. *Advances in Computer Games 9* (eds. H. J. van den Herik and B. Monien), pp. 115–132, Universiteit Maastricht, Maastricht, The Netherlands.

Bush, D. (2000). An Introduction to TwixT. *Abstract Games*, Vol. 1, No. 2, pp. 9–12.

Gasser, R. U. (1995). *Harnessing Computational Resources for Efficient Exhaustive Search.* Ph.D. thesis, Swiss Federal Institute of Technology, Zürich, Switzerland.

Herik, H. J. van den, Uiterwijk, J. W. H. M., and Rijswijck, J. van (2002). Games Solved, Now and in the Future. *Artificial Intelligence*, Vol. 134, Nos. 1–2, pp. 277–311.

Irving, G., Donkers, H. H. L. M., and Uiterwijk, J. W. H. M. (2000). Solving Kalah. *ICGA Journal*, Vol. 23, No. 3, pp. 139–148.

ITRS (2007). International Technology Roadmap for Semiconductors, 2007 Edition, Executive Summary. www.itrs.net/Links/2007ITRS/ExecSum2007.pdf.

Pawlewicz, J. and Lew, Ł. (2007). Improving Depth-First PN-Search: $1 + \epsilon$ Trick. *Computers and Games (CG 2006)* (eds. H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers), Vol. 4630 of *LNCS*, pp. 160–171, Springer-Verlag, Heidelberg, Germany.

Romein, J. W. and Bal, H. E. (2003). Solving the Game of Awari using Parallel Retrograde Analysis. *IEEE Computer*, Vol. 36, No. 10, pp. 26–33.

Sackson, S. (1969). *A Gamut of Games.* Random House, New York, NY, USA.

Sakuta, M., Hashimoto, T., Nagashima, J., Uiterwijk, J. W. H. M., and Iida, H. (2003). Application of the Killer-tree Heuristic and the Lamba-Search Method to Lines of Action. *Information Sciences*, Vol. 154, Nos. 3–4, pp. 141–155.

Schadd, M. P. D., Winands, M. H. M., Uiterwijk, J. W. H. M., Herik, H. J. van den, and Bergsma, M. H. J. (2008). Best Play in Fanorona Leads to Draw. *New Mathematics and Natural Computation*, Vol. 4, No. 3, pp. 369–387.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers is Solved. *Science*, Vol. 317, No. 5844, pp. 1518–1522.

Wágner, J. and Virág, I. (2001). Solving Renju. *ICGA Journal*, Vol. 24, No. 1, pp. 30–34.

Winands, M. H. M. (2004). *Informed Search in Complex Games.* Ph.D. thesis, Universiteit Maastricht, Maastricht, The Netherlands.

Winands, M. H. M., Herik, H. J. van den, and Uiterwijk, J. W. H. M. (2003a). An Evaluation Function for Lines of Action. *Advances in Computer Games 10: Many Games, Many Challenges* (eds. H. J. van den Herik, H. Iida, and E. A. Heinz), pp. 249–260. Kluwer Academic Publishers, Boston, MA, USA.

Winands, M. H. M., Uiterwijk, J. W. H. M., and Herik, H. J. van den (2003b). PDS-PN: A New Proof-Number Search Algorithm: Application to Lines of Action. *Computers and Games (CG 2002)* (eds. J. Schaeffer, M. Müller, and Y. Björnsson), Vol. 2883 of *LNCS*, pp. 170–185, Springer-Verlag, Berlin, Germany.