THE QUAD HEURISTIC IN LINES OF ACTION

Mark H.M. Winands, Jos W.H.M. Uiterwijk and H. Jaap van den Herik[1]

Maastricht, The Netherlands

ABSTRACT

Lines of Action (LOA) is a two-person zero-sum chess-like connection game with perfect information. The goal of each side is to connect its own pieces. The state-space complexity and game-tree complexity are comparable with those of Othello. A LOA evaluation function usually consists of a combination of the following components: threats, solid formations, (partial) blocking, centralisation, material advantage, and mobility.

In this article we introduce the quad heuristic and disregard mobility. The quad heuristic is used in three qualities, viz. (1) to assess whether a position is non-terminal, (2) to contribute to the evaluation function, and (3) to support the quiescence search. The heuristic is based on quad counts and Euler numbers, and derived from the field of optical character recognition.

We show that an evaluation function using the quad heuristic outperforms evaluation functions based on the centre-of-mass approach and on (partial) blocking, i.e., evaluation functions without the quad heuristic. Moreover, it turns out that a quiescence search, which only looks at capturing moves that destroy or create connections, is quite effective when using the quad heuristic. Finally, we provide three conclusions and suggest three ways of improving the use of the quad heuristic in LOA.

1. INTRODUCTION

Sixty years ago the first true connection game, called Hex, made its debut. The objective of a connection game is to group the pieces in such a way that they connect two opposite edges (a static goal) or 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 of abstract gaming. Many prominent game inventors made a contribution to this theme. Besides Hex, examples of connection games are TwixT and Lines of Action (LOA); the latter is discussed in this article.

LOA is a two-person zero-sum game with perfect information; it is a chess-like game with a connection-based goal played on an 8×8 board. LOA is invented by Claude Soucie around 1960. Sid Sackson (1969) described it in his first edition of *A Gamut of Games.* After this publication, LOA received some attention of AI researchers. For instance, the first LOA program was written at the Stanford AI laboratory around 1975 by an unknown author. In the 1980s and 1990s "hobby" programmers wrote several LOA programs. However, all were beatable by human beings (Dyer, 2000). At the end of the nineties LOA became a target of AI researchers. Some of them used LOA only as a test domain for their algorithms (Kocsis, Uiterwijk, and Van den Herik, 2000), others tried to build strong LOA programs by using new ideas. The programs YL, MONA, and MIA – winners of the gold, silver and bronze medal at the fifth Computer Olympiad, respectively (Björnsson, 2000) – belong to the latter category. The program MIA (Maastricht In Action) is discussed in this paper. The main issue of this article is the quad heuristic. It is used (1) to assess whether a position is non-terminal, (2) to contribute to a static board evaluator, and (3) to support the quiescence search. YL and MONA do not use the quad heuristic.

The article is organised as follows. Section 2 explains the rules of LOA, and Section 3 describes the complexity of the game. In Section 4 the quad heuristic is explored in detail. Section 5 gives the background and implementation of three evaluation functions among which one that relies on the quad heuristic; three matches of 200 games each show the superiority of the quad heuristic. Section 6 describes the quiescence search and

[1] Universiteit Maastricht, Department of Computer Science, P.O. Box 616, 6200 MD Maastricht, The Netherlands. Email: {m.winands, uiterwijk, herik}@cs.unimaas.nl.

how the quad heuristic is used there; it turns out that the normal evaluator and the quad evaluator perform equally well. Section 7 contains our conclusions and some suggestions for future research.

2. RULES OF LOA

LOA is played on an 8×8 board by two sides, Black and White. Each side has twelve pieces at its disposal. Sackson (1969) describes the original LOA rules of the inventor Claude Soucie in the first edition of *A Gamut of Games*. Below they are formulated in eight points.

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 side of the board. The initial position of the game is shown in Diagram 1.
2. The players alternately move, starting with Black.
3. A player to move must move one of its pieces. 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. (These are the *Lines of Action*.)
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 turn all own pieces on the board into one connected unit. The first player to do so is the winner. The connections within the group may be either orthogonal or diagonal. For example, in Diagram 2 Black has won because the black pieces form one connected unit.
7. If one player's pieces are reduced by captures to a single piece, the game is a win for this player.
8. If a move simultaneously creates a single connected unit for both players, the game is drawn[2].

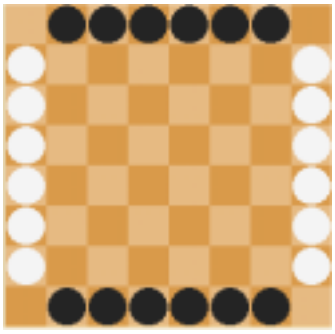


Diagram 1: The initial position.

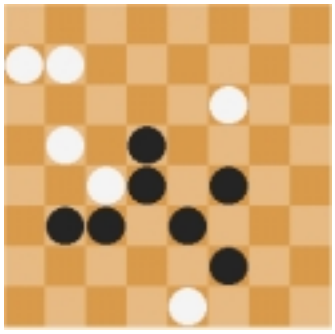


Diagram 2: A terminal position.

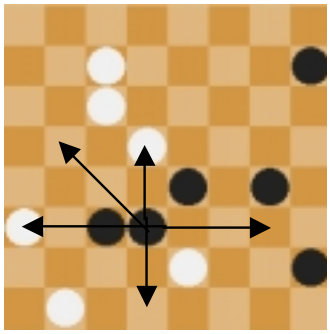


Diagram 3: Movement of pieces.

Since some situations are not covered by the original rules, ad hoc rules are introduced. For instance: (1) repetition of positions is considered as a draw, and (2) if a player cannot move, this player loses.[3]

In the article we use the standard chess notation. The possible moves of the black piece on **d3** in Diagram 3 are indicated by arrows. The piece cannot move to **f1** because its path is blocked by an opposing piece. The move to **h7** is not allowed because the square is occupied by a black piece.

3. COMPLEXITY OF LOA

Below we distinguish two types of complexity, viz. the state-space complexity and the game-tree complexity. Of both complexities we give an estimation of their size. Moreover, we argue that LOA is not crackable with the current state-of-the-art computer techniques. For this goal, new techniques should be developed.

---

[2] In the second edition of *A Gamut of Games* (1982) simultaneous connection is described as a win for the moving player, but the draw variant is still in force in the main tournaments, such as the Mind Sports Olympiad 2000, and on Richard's PBeM server (e-mail championship).

[3] The use of this rule is disputable, some LOA players allow the opponent to pass in such positions.

3.1 State-Space Complexity

An upper bound for the state-space complexity of LOA can easily be derived (cf. Schaeffer and Lake, 1996). Let B be the number of black pieces and W the number of white pieces. There are at most twelve black and twelve white pieces. Each of the 64 squares on the board can be occupied by a piece. The number of positions with 24 pieces or fewer is derived by the following formula.

$$\sum_{B=1}^{12}\sum_{W=1}^{12} Num(B,W) - Num(1,1) \qquad \text{where} \quad Num(B,W) = \binom{64}{B} \cdot \binom{64-B}{W}$$

In the formula we have corrected the counting formula for all positions with one piece at each side, because they are impossible to reach. One of the players would have won earlier. For the same reason we do not count all positions with pieces of one colour only.

In Table 1, we present the number of positions distinguished by the number of pieces on the board. For instance, the number of positions of three pieces equals $Num(2,1) + Num(1,2)$, resulting in 249,984. Totalling the numbers for 3 pieces up to 24 pieces results in 1.3×10^{24} different positions. Of course, we then have included some positions, which are not reachable from the start (Dyer, 2000), but we believe that this fact only lowers the actual state-space complexity marginally. If we take the board symmetries into account, the state-space complexity is estimated to be $O(10^{23})$.

# of pieces	# of positions
3	249,984
4	8,895,264
5	228,735,360
6	4,648,410,816
7	78,273,240,192
8	1,124,246,003,472
9	14,045,698,101,120
10	154,805,625,541,952
11	1,521,396,969,611,904
12	13,445,574,994,311,264
13	107,590,873,672,114,560
14	782,632,117,126,476,864
15	5,188,514,989,534,830,720
16	31,335,094,798,158,420,360
17	171,930,216,128,039,066,880
18	853,283,294,857,675,368,960
19	3,807,935,897,946,939,333,120
20	15,158,514,055,288,777,729,920
21	53,164,643,498,259,191,458,560
22	160,592,373,542,262,268,414,080
23	396,757,628,751,471,486,670,080
24	677,794,282,450,430,456,394,720
Total:	1,308,338,020,676,454,019,380,152

Table 1: State-space complexity.

3.2 Game-Tree Complexity

The game-tree complexity can be approximated by the size of a uniform game tree in which the depth is the average game length and the branching factor the average number of legal moves (Allis, 1994). Since we do not have a large database of LOA games we have approximated the numbers (i.e., game length and branching factor) by experiments in which our program played against itself. The results were: an average game length of 38 ply and an average branching factor of 30 (Winands, 2000). The game-tree complexity thus is about $O(10^{56})$.

3.3 Solving LOA

The state-space complexity and game-tree complexity are comparable with those of Othello (Allis, Van den Herik, and Herschberg, 1991). Considering the current state-of-the-art computer techniques, it seems that LOA is not crackable or solvable. We believe that for solving the game new techniques have to be developed. A characteristic property of LOA is that it is a converging game (Allis, 1994), since the initial position consists of 24 pieces, and during the game the number of pieces (usually) decreases. However, since most terminal positions have still more than 10 pieces remaining on the board (Winands, 2000), endgame databases are (probably) not effectively applicable in LOA. As a case in point, we remark that an endgame database of ten pieces would require approximately 10 terabytes.

4. THE QUAD HEURISTIC

According to the rules given in Section 2, LOA is a win for the player who first succeeds in creating a position in which all remaining pieces are connected. This is called a terminal position, comparable to a mate position in chess. However, it is not easy to detect whether a LOA position is terminal, contrary to the analogous procedure for mate positions in chess. An obvious solution is to update incrementally the number of pieces after a move is made. The number of pieces of an arbitrary group is then determined, using a brute-force search. If this number equals the total number of pieces of that colour, the game is over. However, the brute-force search necessary to retrieve the number of pieces in a group can be a time-consuming process.

The use of the quad heuristic for LOA was first proposed and implemented by Dave Dyer in 1996, in his program LOAJAVA (Dyer, 2001). We have incorporated it into a heuristic that easily detects non-terminal positions. The heuristic is based on the use of *quads*, an OCR (Optical Character Recognition) method. A *quad* is defined as a 2×2 array of squares (Gray, 1971). In LOA there are 81 possible quads for each side, including also quads covering only a part of the board along the edges. Considering rotational equivalence, there are six different quad types, depicted in Figure 1.

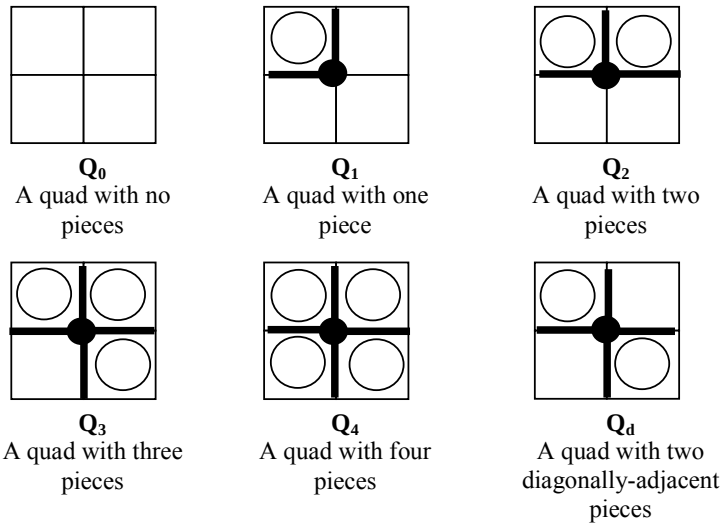


Figure 1: Six different quad types.

If a quad only partially covers the board, the cells not covering the board are considered empty. We note that there are two quad types with two pieces: one in which the pieces are orthogonally adjacent, and another one in which they are diagonally adjacent. In Figure 1, the black points, thick lines and occupied squares denote the *vertices*, *line segments* and *filled regions* (Gray, 1971), respectively. The Euler number of a grid (e.g., the 8×8 board) represents the number of connected groups minus the number of holes (Minsky and Papert, 1988). According to Gray (1971), the quads can be used to derive the Euler number in the following way. Let n_0 be the total number of vertices, n_1 be the total number of line segments, and n_2 be the total number of filled regions. Then the Euler number E is given by:

$$E = n_0 - n_1 + n_2$$

Moreover, we remark that a vertex occurs in one quad only, a line segment in two different quads, and a filled region in four different quads. Hence, we use the following weight function: a vertex is given weight 1, a line segment weight 1/2, and a filled region weight 1/4. Clearly, the contribution of each quad to n_0 , n_1 and n_2, (denoted by Δn_0, Δn_1, and Δn_2) depends only on its type. The same holds for each quad contribution to E, $\Delta E = \Delta n_0 - \Delta n_1 + \Delta n_2$. Looking at Figure 1 and taking into account the weight function, the distinct quad-type contributions are given in Table 2.

Quad Type	Δn_0	Δn_1	Δn_2	ΔE
Q_0	0	0	0	0
Q_1	1	1	1/4	1/4
Q_2	1	3/2	1/2	0
Q_3	1	2	3/4	-1/4
Q_4	1	2	1	0
Q_d	1	2	1/2	-1/2

Table 2: Contributions to the Euler number.

We note that the quads of type Q_0, Q_2 and Q_4 have a contribution $\Delta E = 0$. Thus, when summing ΔE over all quads we arrive at the following formula of the Euler number of a player.

$$E = \left(\sum Q_1 - \sum Q_3 - 2\sum Q_d\right)/4$$

For the purpose of determining the Euler number of a player, the opponent's pieces are ignored. In Diagram 4 a special example is given. The Euler number for Black is 3; the interpretation is: Black has three connected units. However, the Euler number for White is 1, although White has two connected units. Here it is essential that the white pieces are surrounding the square **e5**, called a *hole*. Therefore, the Euler number for White is equal to two connected units minus one hole. The example shows that an Euler number of 1 does not refer with certainty to a terminal position. If both Euler numbers are larger than one, the game is definitely not over (cf. Minsky and Papert, 1988). When one Euler number of the two sides is 1 or less, we have to use another method. In an experiment, in which twenty games were played, we observed that the quad heuristic was able to determine correctly that in ninety-eight percent of the positions in the search tree the game was not over. In the remaining two percent we used another method (see at the beginning of this section).

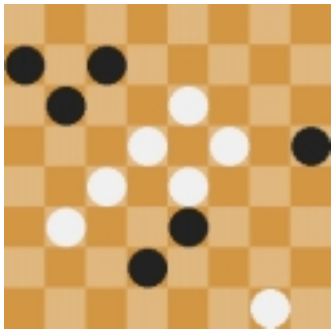


Diagram 4: A position with a hole on e5.

The advantage of the quad method is that we can update incrementally the bit quads and their counts. The number of quads that change as a result of a non-capture move is eight, whereas for a capture move this number is twelve. Although the bookkeeping is not trivial, it is still sufficiently cheap to outperform the method that straightforwardly calculates the number of connected units by recursive expansion of pieces to their neighbours.

The LOA-playing computer program MIA (Winands, 2000) uses the quad heuristic. In Table 3 we compare the quad heuristic with an ordinary depth-first α-β search by presenting the number of nodes visited for different search depths, with and without using the quad heuristic, for 15 positions (see Appendix). The criterion for selecting the positions was that (1) they come from regular tournament games, and (2) they still have twelve or more pieces on their *initial* square. The latter criterion is rather arbitrary, but in discussions with other LOA

programmers this was seen as a reasonable measure for "opening" positions. For this particular set of positions, using the quad heuristic resulted in a speed up of 10 to 15 percent. These results can be seen as an indication not as a proof. A possible drawback of the quad heuristic is its low performance when the total number of pieces is low. Since LOA usually ends with ten or more pieces on the board (cf. Subsection 3.3), this drawback rarely occurs. An additional advantage is that we can use the quad heuristic for two other purposes: (1) to contribute to the evaluation function, and (2) to support the quiescence search (see Sections 5 and 6).

Depth	Time (s) with quads	Time (s) without quads	# of nodes
1	0.1	0.1	511
2	0.3	0.3	2,709
3	1.2	1.2	23,538
4	3.7	4.3	95,339
5	18.1	21.2	692,663
6	79.1	81.8	2,758,940
7	407.8	467.9	17,123,728

Table 3: Experiments with the quad heuristic.

5. EVALUATION FUNCTIONS

Although not much strategic knowledge about LOA is available, there are a few basic principles. Carl von Blixen (2000) provides some guidelines on his homepages, which we have taken as a start for developing an evaluation function. We identified six strategic principles, i.e., threats, solid formations, (partial) blocking, centralisation, material advantage, and mobility. In contrast to chess and checkers, where material is the dominant factor, in LOA none of them is dominating. The principles are transformed into the components of a LOA evaluation function. Assuming that the quad heuristic is in some sense correlated to mobility, we concentrate on five components, disregarding mobility. Moreover, we refrain from an analysis of interdependent relations. The five components are used in MIA; they are explained in Subsection 5.1. Based on these components, three different evaluation functions have been constructed, called *normal*, *quad*, and *blocking*. Subsection 5.2 describes the composition of the evaluation functions. Subsection 5.3 provides results. Although we do not use mobility as a component in the evaluation function, two other programs do, viz. YL and MONA (Billings and Björnsson, 2000).

5.1 General Principles of LOA

As stated before LOA is a chess-like connection game based on a few general strategic principles. Above we emphasised five of them as valuable components for MIA's evaluation function. Below we discuss the components.

Threats
If a player can connect in one move all its pieces in a single connected unit, the position contains a *threat*. The first player to create a threat has a big advantage, because the opponent possibly has to break up its own formation to stop this threat. For instance, in Diagram 5 we see that White threatens to win by the move **e8-e6**. Black can only prevent this by **g4×e4**, weakening its own formation. Although threats are important, they are not essential. YL has no detection of threats for either side. In contrast, MONA does extensive threats analysis. According to Billings and Björnsson (2000) the extensive threat analysis has significantly improved MONA's performance. MIA has a superficial threat analysis (implicit in the centre-of-mass approach, see Subsection 5.2).

Solid formations
If pieces are connected in more than one direction it is harder for the opponent to disconnect them. In Diagram 6 Black has created a solid formation in the centre of the board, which is hard to destroy. However, creating such a formation usually takes many moves, during which the opponent may create a threat. It is good practice to investigate first whether it is possible to connect a piece to a group of connected pieces, and only thereafter to see whether a solid connection can be made. In MIA, solid formations are implicit in the centre-of-mass approach and explicit in the quad evaluator (see Subsection 5.2).

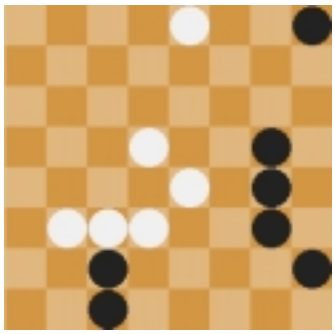


Diagram 5: A position with a threat.

Diagram 6: A solid formation.

Blocking

Because a piece is not allowed to jump over the opponent's pieces, it can happen that the piece is blocked, i.e., cannot move. Blocking a piece far away from the other pieces is an effective way of preventing the opponent to win. For instance, in Diagram 7 White must first spend many moves to free the blocked piece, i.e., capturing pieces around the blocked piece on **a8**, before it may attempt to reach the connection goal. During that time, Black may set up a winning strategy. Even partial blocking can be quite effective, especially if it forces a player to find a way around the opponent's pieces. Handscomb (2000a) describes that it is a common tactic to create a wall on the b- or g-file, or the 2nd or 7th rank in the opening phase of the game, which partially blocks pieces of the opposing side. In MIA, blocking is a separate component.

Diagram 7: Blocking a piece.

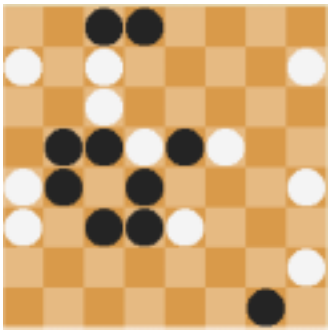


Diagram 8: LOA email tournament, 1999, Roessner vs. Handscomb, after 10. e1-c3.

Centralisation

Pieces in the centre or controlling the centre are assumed to be more important than pieces without any centre relation. However, Handscomb (2000b) argues that the benefits of actual centralisation in LOA are sometimes exaggerated. Because of the nature of the game, pieces huddled together in the middle of the board are incapable of capturing each other. Even with only two pieces in a given line of action a distance of two squares is needed for a capture, and if there are three pieces a considerable distance between the pieces is required. In Diagram 8 the white pieces are scattered around the edges of the board. Nevertheless, the white piece on **h4** can destroy Black's formation in the middle by capturing the black piece on **d4**. So, the white pieces on the edges of the board are acting as cruise missiles. Of course, this is not possible when all the pieces are in the centre. This example shows that controlling the centre might be better than occupying it. Hence, actual centralisation has a limited importance, but successfully controlling the centre is an important component in the evaluation function. Through an effective control of the centre, the two starting groups of the opponent remain separated, whereas the own groups can be brought together. The notion of centralisation seems to be more important in LOA than in chess. As a direct consequence we have found that the dynamic principle of favouring pieces to move towards their centre of mass is more profitable than the static principle of favouring pieces to move towards the centre of the board.

Material advantage
The last feature to be discussed is material advantage. At first sight capturing the opponent's pieces is not a good idea, because the opponent then needs to connect fewer pieces. However, it is not hard to see that capture moves are beneficial in some positions. First, capturing a piece such that the opponent's formation is destroyed is mostly a good move. Second, it also happens that a player is able to connect its own formation only by a capture move. Third, a capture move may be the only way to free a blocked piece. Some authors argue that capturing pieces for the sake of a material advantage is sometimes appropriate (Handscomb, 2000c), because a player with a material advantage has more opportunities to prevent the opponent's threats and create its own threats. Whether material really is an advantage depends on the position. Moreover, material can be interrelated to other components of the evaluation function used. For instance, gaining material contributes heavily to mobility. Therefore, MONA gives a zero weight to the material component and YL even gives a negative weight. Since, a straight material component in MIA's evaluation function would be dangerous – the program might wildly capture pieces – we have assigned the material component a zero weight too. From these considerations we may conclude that material is probably the least important component of alls six LOA evaluation-function components.

5.2 LOA Evaluators

Based on the general principles described in Subsection 5.1, we have implemented three evaluators: *normal*, *quad* and *blocking*. We want to stress that the *blocking* and the *quad* evaluator are both extensions of the *normal* evaluator.

Normal evaluator
The core of the normal evaluator is the centre-of-mass approach, consisting of four steps. First, the centre of mass of the pieces on the board is computed for each side. Second, we compute for each piece its distance to the centre of mass. The distance is measured as the minimal number of squares the piece is removed from the centre of mass. These distances are summed together, called the *sum-of-distances*. Third, the *sum-of-minimal-distances* is calculated. It is defined as the sum of the minimal distances of the pieces from the centre of mass. This computation is necessary since otherwise boards with a few pieces would be preferred. For instance, if we have ten pieces, there will be always at least eight pieces at a distance of 1 from the centre of mass, and one piece at a distance of 2. In this case the total sum of distances is minimal 10. Thus, the sum-of-minimal-distances is subtracted from the sum-of-distances. Fourth, the average distance towards the centre of mass is calculated and the inverse of the average distance is defined as the *concentration*.

The normal evaluator does not look at connections and has no knowledge of *solid formations*. However, since we reward positions where pieces are in the neighbourhood of each other, eventually they will be connected in *solid formations* and they will create *threats*. Capture moves are sometimes favoured. For example, if **1. d1-d3** is played in the initial position, the program will respond with **a3×d3**. This move will not only improve its own concentration, but will also undermine the opponent's concentration, since its pieces are a bit more scattered over the board.

In addition to the centre-of-mass approach, positions with a somewhat more *centralised* centre of mass are slightly preferred, preventing formations built on the edges of the board, which are easy to destroy or to block. Finally, penalties are given for each piece on the edge. This is done for two reasons: (1) on average they can move in fewer directions than other pieces (restricted mobility), and (2) they are usually not in the neighbourhood of the centre of mass.

Quad evaluator
The quad evaluator is named after its use of the quad count, which is added as a fifth step to the *normal* evaluator. Since it is impossible to destroy formations with quads of three or four pieces by a single capture, it seems reasonable to favour boards with many Q_3's and Q_4's. However, the danger exists that many of those quads are created outside the neighbourhood of the centre of mass. So, we have rewarded only Q_3's and Q_4's in our quad evaluation, which are at a distance of at most two of the centre of mass. In passing we note that this implementation implicitly favours a *material advantage*. The effect of implicitly favouring a component due to the introduction of another is first described by Schaeffer (1984) for chess. Obviously, it is a challenge to analyse the interrelationship in LOA too, since it turns out to be an issue for almost all components.

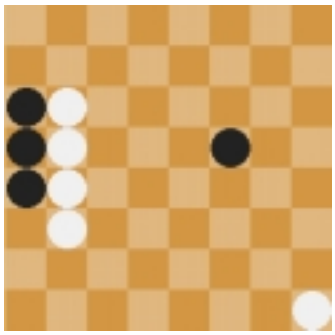


Diagram 9: A wall position.

Diagram 10: An example of pseudo blocking.

Blocking evaluator

The problem with the previous two evaluators is that they are not able to detect *blocking* positions. See, for instance, Diagram 9 in which Black is to move. Playing **f5-c5,** will result in a position with a higher evaluation value than after playing **f5-e6**, but the piece on **c5** cannot move any further to the centre of mass because of the white wall. However, the piece on **e6** has a threat to win (**e6×b6**). Therefore, we have to take into account whether pieces are partially blocked. If a piece outside the neighbourhood of the centre of mass has two moves or less, a penalty is given inversely proportional to the number of possible moves. For pieces outside the region of the centre of mass it is determined whether they can move directly to this region. If so, a fixed bonus is given. These features are implemented in the blocking evaluator *in addition* to the *normal* evaluator. For the blocking evaluator we need the move generator. Consequently, this evaluator is much slower than the others. Whether the benefit of better evaluations outperforms the disadvantage of searching to a lesser depth is an open question. Finally, we notice that a position with a blocked piece is not necessarily a bad position. Sometimes a piece can be freed by a capture move and be connected. For instance, in Diagram 10 the black piece on **f8** is completely blocked. However, this board position is too negatively evaluated by our current static evaluation function, because Black can free this piece and win the game by **c4×f7**. These positions can easily be detected by the quiescence search described in the Section 6.

5.3 The Tree Evaluators Compared

We have played three matches, one for each combination of evaluators. All evaluators played 200 games against each other, switching sides halfway. To measure the trade-off between added information and search depth, the time was limited to 5 seconds per move. The match results are given in Table 4. For the normal and quad evaluator, the speed of the search performed was between 35,000 and 40,000 nodes per second (nps) on a Pentium III 550 MHz machine; for the blocking evaluator a speed of only 10,000 nps was achieved. On the average, the blocking evaluator searches one ply less than one of the other two evaluators, i.e., 6 ply vs. 5 ply.

Quad vs. Normal	Normal vs. Blocking	Quad vs. Blocking
127-71-2	117-81-2	119-76-5

Table 4: Comparing the evaluators with each other (wins-losses-draws).

From Table 4 we see that the quad evaluator outperforms normal and blocking by large margins. We conclude that the quad evaluator is better than the normal evaluator and that the added information in the blocking evaluator does not compensate for the cost of computation. Interestingly, the quad evaluator defeats the blocking evaluator by a margin not significantly larger than the normal evaluator does. As an explanation we conjecture that both evaluators take advantage of the same weaknesses of the blocking evaluator (i.e., a lesser search depth). It is a subject of future research whether these results still hold when searching more deeply, e.g., 10 ply vs. 9 ply.

We tested the quad evaluator in the fifth Computer Olympiad. Diagram 11 shows a position, which occurred in the tournament. Owing to the quad evaluator MIA has built a seemingly strong black formation, which partially blockades the h-file (Björnsson, 2000). Now White did not play **7. h3×f1** and MIA did not play **f1×h3** on move 8 or later in the game – it would have prevented the white pieces on the h-file to escape in the southern direction. Clearly, MIA's evaluator found better things to do. Later in the game (Diagram 12), the black pieces

on the left and right became physically separated by a wall of white pieces. The wall demonstrated the drawback of ignoring the position of the opponent's pieces (Billings, 2001).

Diagram 11: The fifth Computer Olympiad, 2000 MIA vs. MONA, fifth game, after 6. g8-g4.

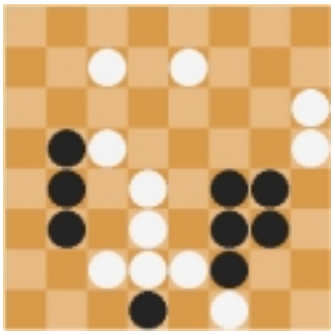


Diagram 12: The fifth Computer Olympiad, 2000 MIA vs. MONA, fifth game, after 14. … e5-f4.

6. QUIESCENCE SEARCH

When the search reaches the depth limit, a static evaluation function should be applied in the leaf node reached. This approach can have disastrous consequences because of the approximate nature of the evaluation function (Kaindl, 1982). Obviously, a more sophisticated depth limit is needed. The evaluation function should only be applied to positions that are *quiescent* (cf. Shannon, 1950).

6.1 Quiescence Search in LOA

Tactical capture moves are mostly responsible for swings in the evaluation of a LOA position. However, it is hard to define what a tactical capture move is, because this heavily depends on the board position. In general capture moves, which destroy connections or form connections, are considered tactical. Based on this definition of tactical, we use a quiescence search that involves only this kind of capture moves. For instance, if Black plays **d7×d4** in Diagram 13, Black will both connect its pieces in the centre and destroy White's connection. Using the quad heuristic, capture moves destroying or forming connections can be detected easily. In Section 4 we have seen that the use of the Euler number to detect the number of isolated groups is not always correct. Therefore, we have to adjust the result in some particular cases ($E \leq 1$). Yet, in general using the quad heuristic leads to a fast and efficient quiescence search.

We have implemented in our program an extended version of quiescence search (cf. Schrüfer, 1989). This kind of quiescence search limits the set of moves considered and uses the evaluations of interior nodes as lower / upper bounds of the resulting search value.

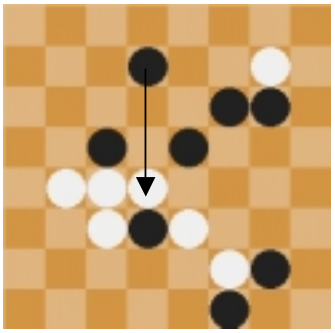


Diagram 13: A tactical capture move.

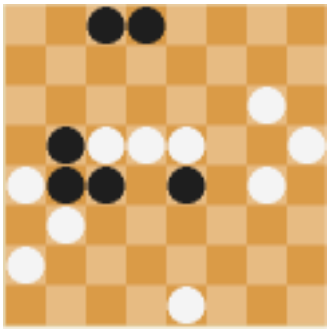


Diagram 14: The fifth Computer Olympiad, 2000, MIA vs. MONA, seventh game, after 12. … c7×e5.

6.2 Testing the Quiescence Search

To investigate whether a quiescence search is beneficial we did some experiments. Two players using the same evaluator played 200 times against each other. The time was limited to 5 seconds per move. One player used quiescence search, the other did not. After 100 games, the players switched colour. Table 5 shows that using quiescence search is an advantage. Nevertheless, the utility of the quiescence search is dependent on the evaluator used. If the blocking evaluator is combined with the quiescence search, no large gain is obtained. It might be due to the restricted depth or to the interrelationship of blocking and quiescence search. We leave it to future research. Obviously, the combination of the quiescence search with the quad or the normal evaluator is powerful. Table 5 shows that there is no significant difference between the quad and normal evaluator. One would expect a higher score for the combination with the quad evaluator than the one with the normal evaluator, because the former looks explicitly at multiple connections and the latter does not. The reason that the combination with the normal evaluator works equally well is that in this case the quiescence search handles tactical positions faster, and presumably multiple connections are not important in this stage. Destroying or forming connections by capture moves is common play in this kind of positions. For example, because of quiescence search MIA played **b4-b7** instead of **d8-b8** in Diagram 14. The former move is a win in 15 (Billings and Björnsson, 2000) whereas the latter is not. Thanks to the use of quiescence search this game was won.

Quad Evaluator	**Normal Evaluator**	**Blocking Evaluator**
Quiescence vs. No Quiescence	Quiescence vs. No Quiescence	Quiescence vs. No Quiescence
125-71-4	134-65-1	106-93-1

Table 5: Comparing quiescence search with no quiescence search (wins-losses-draws).

Finally, we note that the profit of the quiescence search is strongly correlated to the type of moves investigated (e.g., looking at all capture moves is not beneficial in our program). Of course, the profit is dependent of other components in the evaluation function too. Therefore, quiescence search is not an essential part of the search process. It may be compensated by other components such as blocking and mobility. For instance, YL and MONA do not use quiescence search.

7. CONCLUSIONS AND FUTURE RESEARCH

In this article we have introduced the use of the quad heuristic in the board game LOA. We showed that the heuristic may be beneficial in three ways. First, the heuristic enables us easily to assess non-terminal positions. Second, the heuristic is successfully incorporated in the static board evaluator. Third, the heuristic is quite powerful in combination with a quiescence search that looks only at capture moves destroying or creating connections. The quiescence search is similarly helpful for evaluation functions not looking explicitly at multiple connections, such as the normal evaluation function. The reason is that quiescence search is applied on special tactical positions only. Future research on this point may lead to a more detailed answer. Another advantage of the quad heuristic in the quiescence search is that the cost to check whether connections are destroyed or formed is low. The effectiveness of the quad heuristic as applied in our program MIA leads us to the conclusion that the heuristic is quite useful in LOA programs.

Below we suggest three ways of improving the use of the quad heuristic in LOA. First, when the Euler number is 1 or less, we have to use another method to detect whether a position is terminal (because of the possibility of holes in LOA). We will investigate whether it is possible to adjust the quad heuristic in such a way that it is able to cope with holes. Second, the quad heuristic could have been used in alternative ways in the evaluation function, for example, by using the Euler number of each side. If we assume that holes do not occur, we then have also computed the number of connected groups. Because the goal of the game is to create one connected group, it is tempting to consider minimising the number of connected groups as a subgoal of the game. However, at the start of the game this number is just two and it is common knowledge that we have to break up the formation in the opening phase of the game to reach the connection goal. Keeping the Euler number at two would be disastrous in this phase. Disregarding the temporal increase of the Euler number, we believe that the quad heuristic can be incorporated in the evaluation function in a more effective way. Further, the current quad evaluation function can be improved by taking into account specific piece configurations. Moreover, the interdependent relations of evaluation-function components have to be investigated. Using the results obtained we have to perform experiments to test whether the quad evaluator then still is the best when searching more deeply. Third, there is room to improve the quiescence search, e.g., in the selection of moves to investigate. For instance, a closer examination of the relation between the quad heuristic and material advantage may influence

this selection. It has to be investigated whether quiescence search in MIA will still be effective because it might be compensating for inadequacies in other components of the program and its effectiveness might be less with an enhanced evaluation function.

ACKNOWLEDGEMENTS

The authors thank the anonymous referees for their constructive comments and suggestions for improvements. We also would like to thank Dave Dyer for his additional information on the quad heuristic. Moreover, we thank Kerry Handscomb for making articles of *Abstract Games* available. Next, we gratefully acknowledge the possibility to use additional LOA knowledge, made available to us by the former LOA World Champion Fred Kok. Finally, we recognise the support of members of the Alberta GAMES Group, Darse Billings, Yngvi Björnsson and Jonathan Schaeffer.

8. REFERENCES

Akl, S.G. and Newborn, M.M. (1977). The Principal Continuation and the Killer Heuristic. *1977 ACM Annual Conference Proceedings*, pp. 466-473. ACM, Seattle.

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. ISBN 0-13-382615-5.

Allis, L.V. (1994). *Searching for Solutions in Games and Artificial Intelligence*. Ph.D. thesis Rijksuniversiteit Limburg, Maastricht, The Netherlands. ISBN 90-9007488-0.

Billings, D. and Björnsson, Y. (2000). Mona and YL's Lines of Action Page. http://www.cs.ualberta.ca/~darse/LOA.

Billings, D. (2001). *Personal communication.*

Björnsson, Y. (2000). YL wins Lines of Action Tournament. *ICGA Journal*, Vol. 23, No. 3, pp. 178-179.

Blixen, C. von (2000). Lines-of-Action. http://www.student.nada.kth.se/~f89-cvb/loa.html.

Breuker, D.M., Uiterwijk, J.W.H.M. and Herik, H.J. van den (1996). Replacement Schemes and Two-Level Tables. *ICCA Journal*, Vol. 19, No. 3, pp. 175-180.

Dyer, D. (2000). Lines of Action Homepage. http://www.andromeda.com/people/ddyer/loa/loa.html.

Dyer, D. (2001). *Personal communication*.

Gray, S.B. (1971). Local Properties of Binary Images in Two Dimensions. *IEEE Transactions on Computers*, Vol. C-20, No. 5, pp. 551-561.

Handscomb, K. (2000a). Lines of Action Strategic Ideas – Part 1. *Abstract Games*, Vol. 1, No. 1, pp. 9-11.

Handscomb, K. (2000b). Lines of Action Strategic Ideas – Part 2. *Abstract Games,* Vol. 1, No. 2, pp. 18-19.

Handscomb, K. (2000c). Lines of Action Strategic Ideas – Part 3. *Abstract Games,* Vol. 1, No. 3, pp. 18-19.

Kaindl, H. (1982). Quiescence Search in Computer Chess. *SIGART Newsletter*, 80, pp. 124-131.

Kocsis, L., Uiterwijk, J.W.H.M., and Herik, H.J. van den (2000). Learning Time Allocation using Neural Networks. *Working Notes of the Second International Conference on Computers and Games CG'2000*, pp. 297-314.

Minsky, M. L. and Papert, S.A. (1988). *Perceptrons, An Introduction to Computational Geometry*. Expanded edition. MIT Press, Cambridge, MA, USA. ISBN 0-262-63111-3. (First edition, Massachusetts Institute of Technology, 1969).

Sackson, S. (1969). *A Gamut of Games*. Random House, New York, NY, USA. A second edition (1982) has been republished in 1992 by Dover Publications, New York, NY, USA. ISBN 0-486-27347-4.

Schaeffer, J. (1983). The History Heuristic. *ICCA Journal*, Vol. 6, No. 3, pp. 16-19.

Schaeffer, J. (1984). The Relative Importance of Knowledge. *ICCA Journal*, Vol. 7, No. 3, pp. 138-145.

Schaeffer, J. and Lake, R. (1996). Solving the Game of Checkers. *Games of No Chance* (ed. J. Nowakowski). *MSRI Publications*, Vol. 29, pp. 119-133. Cambridge University Press, Cambridge, UK. ISBN 0-521-57411-0.

Schrüfer, G. (1989). A Strategic Quiescence Search. *ICCA Journal*, Vol. 12, No. 1, pp. 3-9.

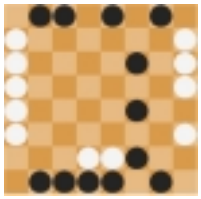
Shannon, C.E. (1950). Programming a Computer for Playing Chess. *Philosophical Magazine*, Vol. 41, pp. 256-275.

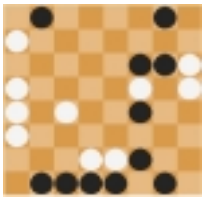
Winands, M.H.M. (2000). *Analysis and Implementation of Lines of Action*. M.Sc. thesis. Universiteit Maastricht, Maastricht, The Netherlands.

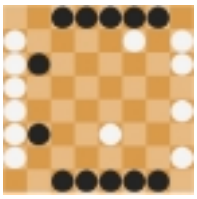
9. APPENDIX: EXPERIMENTAL SET-UP

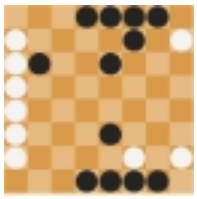
All experiments have been performed in the framework of MIA. MIA has been written in Java and runs on every well-known operating system. It can be played at the website: http://www.cs.unimaas.nl/m.winands/loa/. MIA performs an alpha-beta depth-first iterative deepening search. The program uses a 2^{19}-entry, two-deep transposition table (Breuker, Uiterwijk, and van den Herik, 1996), the history heuristic (Schaeffer, 1983) and killer moves (Akl and Newborn, 1977). For all experiments described in this article, the null move is not used because of possible negative effects of zugzwang in LOA (Winands, 2000). Below the positions are given, which are used in the experiment of Section 4. These positions are taken from matches MIA played at the fifth Computer Olympiad.

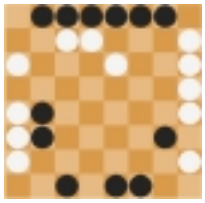

TP 1: Black to move

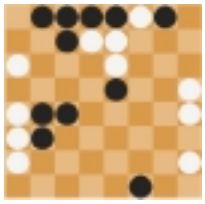

TP 2: White to move

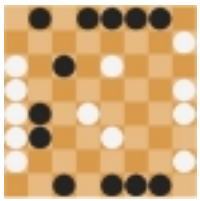

TP 3: Black to move

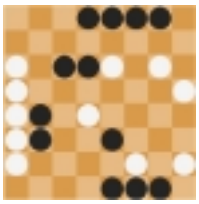

TP 4: Black to move

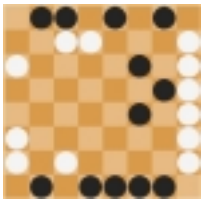

TP 5: White to move

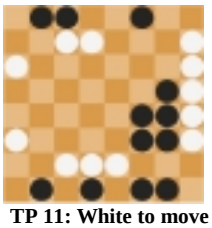

TP 6 Black to move

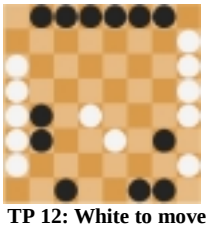

TP 7: White to move

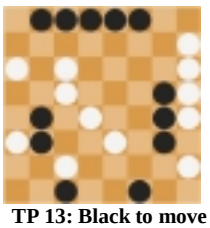

TP 8: Black to move

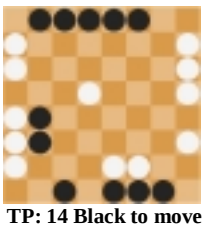

TP 9: Black to move

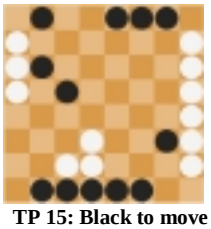

TP 10: Black to move


TP 11: White to move


TP 12: White to move


TP 13: Black to move


TP: 14 Black to move


TP 15: Black to move