A Knowledge-based Approach of Connect-Four

The Game is Solved: White Wins

Victor Allis

Department of Mathematics and Computer Science

Vrije Universiteit

Amsterdam, The Netherlands

Masters Thesis, October 1988 †

ABSTRACT

A Shannon C-type strategy program, VICTOR, is written for Connect-Four, based on nine strategic rules. Each of these rules is proven to be correct, implying that conclusions made by VICTOR are correct.

Using VICTOR, strategic rules where found which can be used by Black to at least draw the game, on each $7 \times (2n)$ board, provided that White does not start at the middle column, as well as on any $6 \times (2n)$ board.

In combination with conspiracy-number search, search tables and depth-first search, VICTOR was able to show that White can win on the standard 7×6 board. Using a database of approximately half a million positions, VICTOR can play real time against opponents on the 7×6 board, always winning with White.

† Published in 1988 as Report IR-163 by the Faculty of Mathematics and Computer Science at the Vrije Universiteit Amsterdam, The Netherlands. Also published in 1992 as Report CS 92-04 by the Faculty of General Sciences at the University of Limburg, Maastricht, The Netherlands.

Contents

Introduction

One of the first domains in Artificial Intelligence (AI) research, has been computer chess. This is not surprising, if we consider the possibilities people thought computers would have in the near future. Using these possibilities, people thought it would be possible to let the computer perform tasks for which humans need intelligence, whatever that may be.

Chess has several features, which makes it a interesting subject for computer programming, in the context of letting the computer perform tasks for which intelligence is needed.

- chess is defined by a set of strict rules.
- matches can be organised between computer programs and humans to measure performance.
- many people (especially among computer scientists and the like) play chess.
- the complexity of the game is large enough to be unable to make an exhaustive analysis of the whole game, using brute force only.

During the years, a large amount of work has been done in the area of computer chess. Some of the results are specific for chess, others have a wider impact. Especially the different search techniques in game trees are used in many other areas of Computer Science in general and Artificial Intelligence in particular.

Another result of the last decades of research in the area of computer chess has been the observation that it is very hard to develop programs which play chess in a way which is more or less similar to the way humans play chess. For an excellent description of the state of the art of computer chess, we refer to [1] and [2].

Programs which play chess in a way similar to chess masters, are called Shannon type C programs [3]. At this time, no strong Shannon type-C chess programs exist. Earlier attempts have failed for reasons which are mostly due to the difficulty to describe the way human chess masters play chess. The only promising results have been in subfields of chess, such as how to solve a mate-in-n chess problem (even with retrograde analysis) and how to find an adequate plan in complex tactical mid-game positions. An example of this is the program Paradise [4].

The fact that the Shannon-C strategy has not lead to good results in the computer chess area made many people (especially chess masters) believe that it will never be possible to develop a program that plays chess at the level of the World Champion. The main argument used by these people is the impossibility to implement intuition, which is considered to be necessary to play chess at grand master level.

We believe that, although it seems to be very hard to develop programs that play chess using the Shannon-C strategy, it is not at all impossible. We therefore want to show that it is possible to develop a program using this strategy. Furthermore, we would like to show that such a program can play at World Champions level, or even better. That would show that, how difficult it may be, good results can be achieved. It would, of course, be an impossible job to try to do this for chess now, where so many attempts have failed in the past. The best way to choose the subject is to take a game which is less complex than chess, but more complex than can be dealt with using brute force only.

Connect-Four has been chosen as a game which meets these restrictions.

A disadvantage of chosing Connect-Four as subject is the fact that, although many people know the rules of the game, most of them know little about the way the game should be played.
This gap of knowledge is tried to be filled in. Another aspect of the choice of Connect-Four, is the fact that we do not know of any research done on the subject. Although some programs exist which

play by using simple alfa-beta algorithms, we do not know of research where knowledge about the game is used. Therefore nowhere in this text is referred to other articles about Connect-Four.

Chapters 1, 2, 3 and 4 are written in a way, which expects no knowledge about the game. Chapter 1 explains the rules of the game and nomenclature used throughout this text. Chapter 2 compares the different possible approaches to find the game-theoretical value of Connect-Four. Chapters 3 and 4 describe essential observations needed to play Connect-Four at an acceptable level.

Chapter 5 and 6 describe the set of strategic rules which are implemented in the program. This set of rules is the knowledge which is given to the program. In chapter 5 it is also described how to use this knowledge to find the move which should be played in a given position. Chapter 7 describes the way the strategic rules interact. Chapter 8 discusses the situations in which the strategic rules can be used to find the game-theoretical value of a position. Chapter 9 describes the program developed using the knowledge of chapters 6, 7 and 8.

Chapter 10 discusses the results found using the program of chapter 9, for several board sizes. It will be shown that the described knowledge is strong enough to find the game-theoretical value for all initial situations on boards smaller than 7 x 6. It will also be made clear, that for the 7 x 6 board the knowledge is not strong enough. Therefore search techniques are used to replace the gap of knowledge. Chapter 11 describes the used search techniques, while chapter 12 describes the implementation of these. Chapter 13 describes the result of the search, as well as some of the variants which were hard to solve. Chapter 14 evaluates the whole project. Chapter 15 describes conclusions and topics for further research.

Acknowledgements

I would like to thank several people without whom this project would not have succeeded:

Prof.Dr. H.J. van den Herik, my thesis supervisor, a specialist in the research field of computer chess, for the stimulating way he critically followed and corrected the project.

I gratefully recognise that Prof.Dr. van de Riet and Prof.Dr. Siklóssy jointly brought me into contact with Prof.Dr van den Herik.

Ir. Matty Huntjens for operating the 2 Sun-4's at times I was unable to operate them myself, as well as for being optimistic about the eventual result, which helped me through times where difficult variants seemed to never end.

Maarten Litmaath for writing a part of the interface which made it possible to collect the results of the games played against the program.

Drs. Aart Middeldorp for reading several preliminary versions of this text.

Drs. Petra Allis-Meinsma for her support and encouragement during the long time I have spent on the project.

Dr. J.W.H.M. Uiterwijk for carefully reading and correcting a preliminary version of the text.

Ruud Wiggers for the way he solved all hardware problems I caused. Ruud is a system administrator who even takes the time to help you after midnight!

1. Introduction into Connect-Four

In this introduction the rules of the game Connect-Four are described, as well as some nomenclature used throughout this text.

§ 1.1. The Rules of the Game

Connect-Four is a game for two persons. Both players have 21 identical men. In the standard form of the game, one set of men is yellow and the other set is red. The game is played on a vertical, rectangular board consisting of 7 vertical columns of 6 squares each. If a man is put in one of the columns, it will fall down to the lowest unoccupied square in the column. As soon as a column contains 6 men, no other man can be put in the column. Putting a man in one of the columns is called: a move.

The players make their moves in turn. There are no rules stating that the player with, for instance, the yellow men should start. Since it is confusing to have to identify for each new game the colour that started the game, we will assume that the sets of men are coloured white and black instead of yellow and red. Like chess and checkers (and unlike go) it is assumed that the player playing the white men will make the first move.

Both players will try to get four connected men, either horizontally, vertically or diagonally. The first player who achieves one such group of four connected men, wins the game. If all 42 men are played and no player has achieved this goal, the game is drawn.

Diagrams 1.1, 1.2 and 1.3 show positions in which White has won the game:

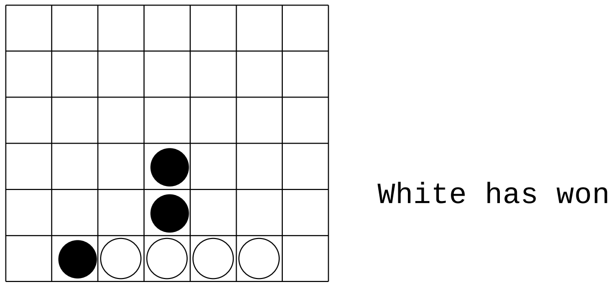


Diagram 1.1.

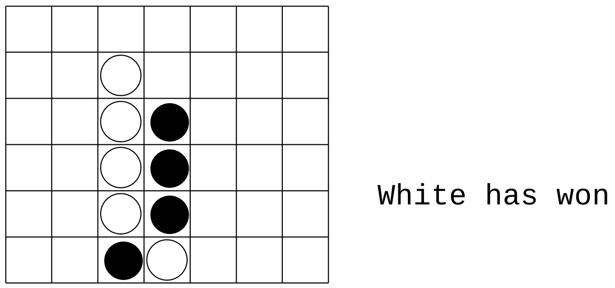


Diagram 1.2.

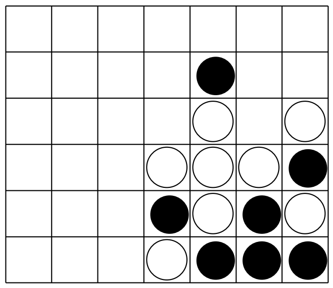


Diagram 1.3.

In the position of diagram 1.1, White has made a horizontal winning group, while his winning groups were resp. vertical and diagonal in the other two diagrams.

A possible drawn position is shown in the diagram 1.4:

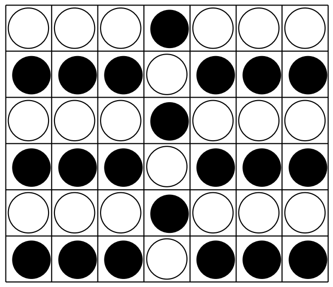


Diagram 1.4.

§ 1.2. Nomenclature

In order to be able to talk about a position, it is useful to give each square a name. We have chosen to use a nomenclature as used by chess players. The 7 columns are labeled ’a’ through ’g’, while the rows are numbered 1 through 6. In this way the lowest square in the middle column is called d1. Unlike the chess nomenclatures, we talk about the a-column, b-column etc, instead of the a-file, b-file etc. Using the word file in a computer science environment would be confusing. Therefore we prefer the word column.

It is now possible to make a list of the moves made during a game. For the game of diagram 1.1 this could have been:

1. d1, d2
2. c1, d3
3. e1, b1
4. f1, White wins.

It is also easy to use the names of the squares to show where the winning group was created. In diagram 1.1 the winning group was on squares c1, d1, e1 and f1. Since the squares must lie on a straight line, it is enough to specify the two endpoints of the group. In this case the group can be identified with c1-f1. In general the notation <`square1`>-<`square2`> will be used to identify all squares on the line with square1 and square2 as endpoints.

2. Different Approaches

In this chapter we show why a brute-force approach will not be successful at this time. Moreover, we show that in certain positions a few strategic rules are strong enough to show that a player can at least draw the game.

§ 2.1. Complexity of the Game

In order to get an idea about the complexity of the game an estimate is presented of the number of different positions which can be achieved, if the game is played according to the rules. A position which can occur during a game is called a legal position, while a position which cannot be achieved is called illegal.

Each square can be in one of three states: empty, white or black. Therefore it is easy to see that the number of possible positions is at most 3^{42} ($\geq 10^{20}$). This upper bound is a very crude one, and can be brought into better proportions.

If the total number of occupied squares in a given position is odd, the number of white men is one more than the number of black men. If the total of occupied squares is even, these numbers are equal. Furthermore, if a column contains an empty square, all squares higher than this square are also empty. If a position contains four connected men, the position concludes a game. Since the last move ended the game, at least one of the four squares in the connected group must be the highest filled square in its column. If this is not the case, or both players have connected four men, the position is illegal. If one player has more than one connected group this position can only be legal if these groups share a square which contains the last man played. In the calculations we are going to make, we do not rule out positions in which are illegal for the reason mentioned above. We also do not rule out positions which are not legal, because they cannot be achieved, during normal play. Diagram 2.1 shows such a position.

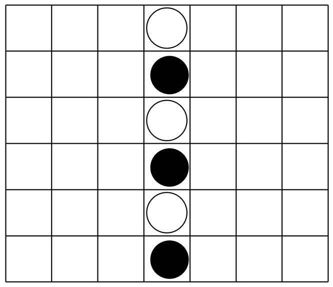


Diagram 2.1.

Although the position looks perfectly normal, it is clear that Black has made the first move. Therefore it is not legal.

We have calculated the number of different positions, including all illegal positions which contain too many connected groups of four men, and illegal positions as shown in diagram 2.1. For this purpose a program was written in the C programming language, which can be found in appendix A. For the standard, 7 x 6 board, the program found an upper bound of $7.1*10^{13}$.

To determine the amount of memory needed to construct a database for Connect-Four this upper bound is useful. In order to show that such a construction takes too much memory, we need a lower bound instead of an upper bound. If we want to find a (good) lower bound of the number of possible positions, we have to make sure that each position we count is legal. Therefore all positions which cannot be achieved during normal play, e.g. diagram 2.1, should be ruled out. We will show that this is not a trivial task. For this reason we have not determined a lower bound. Diagram 2.2 illustrates the difficulties we are faced with in determining if a position is legal.

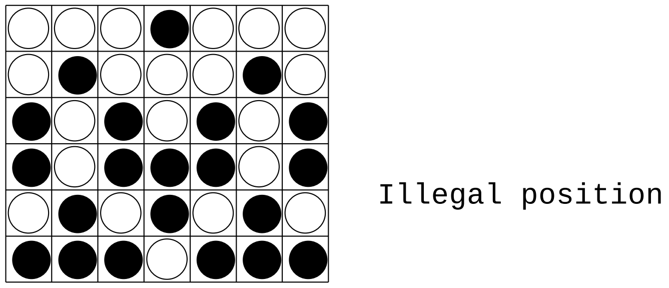


Diagram 2.2.

The position of diagram 2.2 is a draw. Although at first sight it might look like a normal position, it cannot be achieved during normal play. This can be seen as follows: the first move White made must have been d1. If Black played as his first move one of b1, d2 and f1, there is no possible second move for White. Therefore Blacks second move was one of a1, c1, e1 and g1. Suppose Black played a1, White then must have played a2 as second move, giving the position of diagram 2.3:

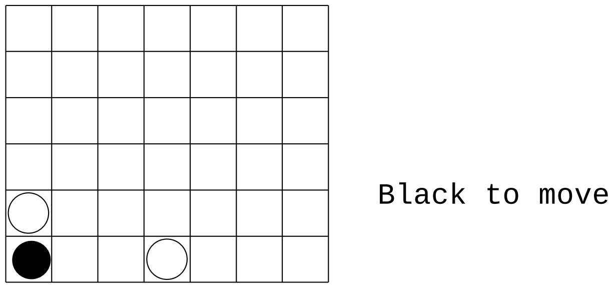


Diagram 2.3.

Now Black still cannot have played b1, d2 or f1, for the same reason as before. The move on a3 is not possible either. Therefore Black must have played one of the remaining c1, e1 or g1. After one of these, and after White's answer to it, the position did not get any better. The farthest we can get with this game is shown in diagram 2.4.

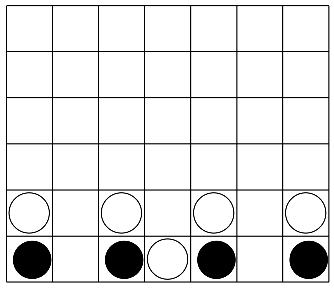


Diagram 2.4.

In this position Black has to move. For all seven columns, the lower two squares should be filled by black men. Therefore after the next move of Black there is no move White can make which will eventually result in the position shown in diagram 2.2. Therefore that position is illegal.

This diagram shows that it can be rather difficult to detect if a position is illegal. It is equally difficult to show which of the positions counted by the program of Appendix A, are not legal because more than one group of four connected men is present. We therefore assume that a database should contain a large number of illegal positions. We believe that in that case the order of magnitude of the upper bound presented before, is a good estimate for the magnitude of the database. This number is by far too big, to think seriously about making a database for Connect-Four. To see this, we have to consider the number of positions which must be stored at the same time, when we build the database. When a retrograde analysis is applied, as has been done for many endgames in chess [5], we need not necessarily store the positions consisting of, say, 20 men, as long as we have not yet determined the value of all positions of 21 men. When we have determined the value of these positions, we no longer need the positions consisting of 22 men or more. Therefore we only need to be able to store all positions of n and n+1 men at the same time. For the 7 x 6 board, this means that we must be able to hold all positions of 36 and 37 men at the same time, a total of over $1{,}6 \cdot 10^{13}$ positions. Appendix C contains a table for the number of different positions for each number of men. We can store the value of a position in 2 bits, since we have 4 possible states: win for White, win for Black, draw or not checked (we can use the address of the 2 bits as identification for the position). This way we need at least 4 Terabyte. Therefore making a database does not seem realistic yet.

§ 2.2. Knowledge-Based Approach

Although the results of the last section show that an exhaustive brute-force approach of the problem is too complex, this does not imply anything for a knowledge-based approach. It is possible that some strategic rules can be found which ensure a win for one of the players or a draw for both. If the correctness of these rules can be proven, it is not necessary to evaluate a large number of positions to obtain the result of the game if both players play correctly.

As an example of such a set of strategic rules we look at Connect-Four on other board sizes. Suppose we play on a board of *n* columns, which are each only 2 squares high. It is easy to see that the only way one can connect four men, is horizontally.

The following strategy will ensure Black at least a draw:

Suppose *n* is even. Group the columns in pairs, giving pairs of columns 1 & 2, 3 & 4, ..., (*n*-1) & *n*. Each time White plays in one column of a pair of columns, Black plays in the other column of the pair. For *n* is odd, groups are made in the same way, leaving the *n*-th column as a single column. The same rules apply to this position. Only if White plays in the *n*-th column, Black plays in that column, too.

Let us try to see why this simple strategy works. Suppose Black has used this strategy and White has won the game. In that case White has four men, horizontally connected. We suppose this has been done on the first row. A group in the second row is dealt with in the same way. It is easy to see that at least two of the four men in the winning group must lie within the same pair of columns. Since the rules show that as soon as White plays in one column of the pair, Black will take the other square, it is clear that it is impossible for White to get both lower squares in a pair of columns. Therefore the assumption that White has won somewhere in the first row is wrong. The same reasoning applies to the second row and therefore White cannot have won. This shows that this simple strategy ensures Black that he will draw the game.

The previous set of strategic rules, shows that it is sometimes possible to find some easy rules which can be used to play a game in a way which ensures a good result, although there are a large number of possible different positions.

Surprisingly enough, it is also possible to give such a simple set of rules for more interesting board formats. For all boards which consist of at most 6 columns and 2*n* rows (an even number of rows), the following strategy will ensure Black at least a draw. For simplicities sake, the rules are given for a board of exactly 6 columns. However, it is easy to see that it works for less columns, too.

For columns 1, 2, 5 and 6 the following rule is used:

(1) Black answers in this column if and only if White has just played in this column.

For columns 3 and 4 the following rule is used:

(2) If White plays for the first time in one of these columns, Black answers in the other column. Otherwise if Black still can play in the column in which White just played, Black does so, otherwise Black moves in the other column.

First we have to check that Black always can play according to these rules. The rules state that every time White plays in one of the columns 1, 2, 5 or 6, Black plays in the same column. This will give Black the squares in the even rows of these columns, and White the squares in the odd rows. Since the number of rows is even, Black always can answer a White move in one of these columns.

For columns 3 and 4, after the first move of both White and Black, the lowest, odd squares are occupied, one by Black and one by White. When White moves in one of these columns, he gets an even square. Black will take the odd square above it if such a square exists. If not, he takes the first empty square in the other column, which is even, and must exist, since a column cannot be filled up, if the last square played is odd.

What will this strategy give to Black? He gets all even squares in columns 1, 2, 5 and 6. Furthermore he gets one of the lower two squares in the middle columns. And most importantly, he gets at least one of the middle squares in every odd row. This follows directly from rule (2).

In this way White clearly cannot get four men connected vertically. If he wants to have four men connected horizontally, he will not get them in an odd row, since Black occupies at least one of the middle squares. Each horizontal group of four connected men contains both middle squares. (This observation is not true for a board of more than 6 columns wide, which limits the power of the proof to these small boards.) It is obvious that four horizontally connected men in an even row is not possible, either. For a possible diagonal group of four, we first observe that the four men in such a group lie alternately in odd and even squares. This shows that White cannot get a diagonal group which lies partly in column 1 and column 2, since in that case both squares in the diagonal group would be odd. For the same reason it is impossible to have a diagonal group partly in column 5 and column 6. The only remaining possibility is a group spread over columns 2, 3, 4 and 5. However, White only will get odd squares in columns 2 and 5, while he also needs an even square in one of those columns to have a diagonal group. This completes the proof that White cannot win, giving Black a draw.

We have shown in this section that it is sometimes possible to give a set of strategic rules with which one of the players will never lose. This shows that a knowledge-based approach in at least some cases can give results which cannot be obtained using brute force only. Therefore it is useful to investigate the possibilities of this approach for the standard board of 7 columns and 6 rows.

§ 2.3. Correctness of Strategic Rules

Let us consider a set of strategic rules for the well-known game tic-tac-toe, which some players might like to use. The rules have priority in the order given:

1. If there is a winning move, make it.
2. If the opponent can win at a square by his next move, play that move.
3. Taking the central square is more important than taking other squares.
4. Taking corner squares is more important than taking squares on the edges.

These four rules seem all to be very reasonable. Now let us see what happens if we use these rules to play against a human player, who does not use the rules. The human player starts playing. The whole game is shown in diagram 2.5:

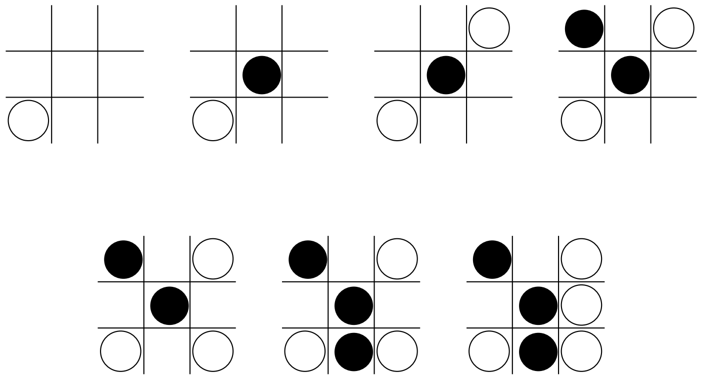


Diagram 2.5.

This game shows that the four strategic rules do not always assure that the best result possible (draw) can be reached, although the rules look perfectly reasonable. This example is due to [6].

For this reason we must make clear what the differences are between the four strategic rules for tic-tac-toe and the rules used in this article concerning Connect-Four.

After the strategic rules in section 2.2 were described, we also showed that there was no way the opponent could win. That proved that the rules would guarantee the player who uses them a draw. The four rules for tic-tac-toe seem as reasonable as the rules for Connect-Four. Only this time nothing is proved about the result of the game, when the rules are applied. Therefore it should not be surprising that there is a way to win against someone using these rules.

This example shows that the formulation of strategic rules is not enough. Each set of rules should be classified as guaranteeing a certain result (in which case a proof should be supplied) or as being merely a heuristic rule. A difficulty is the fact that we do not have a formal framework (model) which can be used to describe a proof in. For that reason, proofs will be as given in section 2.2. Still we claim that these informal proofs are correct, meaning that the results which are promised by the rules, can always be achieved.

3. Some Strategic Rules for Connect-Four

Most articles on computer chess assume that readers have some basic knowledge about chess. If an endgame of King and Queen versus King is considered it is tacitly assumed that the readers know that it is very easy for the King and Queen to mate the other King. Although many people have seen a ’Connect-Four’ set before, and a lot of them have played the game sometimes, very few seem to have encountered at least some of the basic strategics of the game. For this reason in this chapter we try to fill in this gap of knowledge.

§ 3.1. Useless Threats

Let us consider the position of diagram 3.1:

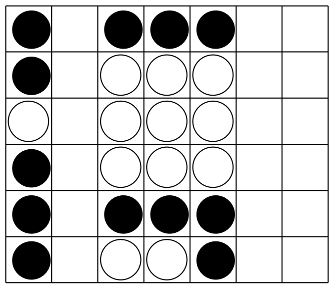


Diagram 3.1.

This position arose after the following moves:

1. d1, d2
2. d3, e1
3. d4, e2
4. d5, d6
5. c1, c2
6. c3, a1
7. c4, a2
8. c5, c6
9. e3, a3
10. a4, a5
11. e4, a6
12. e5, e6

Which player will win the game? Let us consider the different possibilities to get a group of four connected men for both players.

White can get groups at a4-d4, b2-e5, b3-e3, b4-e4, b5-e5, b6-e3, c5-f2, c3-f3, c4-f4, c5-f5 and c3-f6, while for all these groups he needs just one more man. Black can get groups at a2-d2, b2-e2, c2-f2, a6-d6 b6-e6 and c6-f6 for which he also needs just one more man.

First we should note that the possibilities a2-d2 and b2-e2 are not really different possibilities.

For both only the square b2 is needed. This means that both are completed or none is. So after White's 7th move, getting the possibility a2-d2 could not be the reason to play a2, since it would add nothing to Blacks chances to win (there could be other reasons, of course).

It is therefore better not to talk about which groups can be completed, but instead talk about the squares needed to complete one of these groups. In our example White needs one of the squares b2, b3, b4, b5, b6, f2, f3, f4, f5 or f6. Black needs one of the squares b2, b6, f2 of f6.

Clearly White has more possibilities to complete groups. In this case it does not help him much, because it is not important at all that White can win at b3, b4, b5, b6, f3, f4, f5 of f6. To see this let us suppose that play has continued:

13. g1, g2
14. g3, g4
15. g5, g6

This results in the position of diagram 3.2.

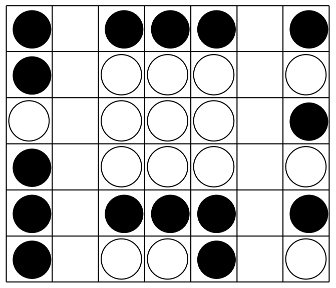


Diagram 3.2.

Now the time has come that White has to move in the b-column or the f-column. As soon as this has been done, Black will take the square in the second row and win the game. Therefore it is clear that in this game no square of b3-b6 and f3-f6 will be used. Therefore most of White's unfinished groups are useless, as well as some of Blacks groups.

Now that we know this, we can go back to diagram 3.1 and list once again the squares for both players which can complete a group of four connected men. For both players the lists consist of only the squares b2 and f2.

A threat for player A, is a square which, if taken by player B, connects four of B's men. A useless threat then is a threat which can never be taken by the opponent for some reason. In the position of Diagram 3.1 we have found two serious threats: b2 and f2, while all other threats were useless.

The main question is: Which player will be so unlucky that he has to play b1 or f1, because no square in the other columns is left empty. As we have seen in diagram 3.2, it is White's turn. Therefore Black will win. Most unexperienced players feel White was really unlucky. This is however not the case. In the position of diagram 3.1, it was already easy to count the empty squares to see that eventually White would have to play b1 or f1. A better approach is to look more generally to this principle of having to play a move which one would rather not: Zugzwang.

Suppose we have, like in diagram 3.2, five columns filled up completely and the two other columns empty. Without knowing anything about the way the columns are filled, it is easy to see that it is White’s turn. This does not depend on the way the players filled the board, but depends only on the number of moves made. If an even number of men are played, it is always White’s turn, while Black has to move after an odd number of men played.

Since five full columns consist of thirty men, it must be White’s turn. The fact that five full columns consist of an even number of men, is due only to the fact that a column consist of an even number of men. Therefore in any position where some columns are completely filled and the others are still empty, the number of men played is even and therefore it is White’s turn. So filling up the g-column, (between diagram 3.1 and 3.2), does not help White at all, since he has to start playing again afterwards. To put it another way: filling up another column does not change the Zugzwang.

Now let us look once again at a position of the last game, shown in diagram 3.3:

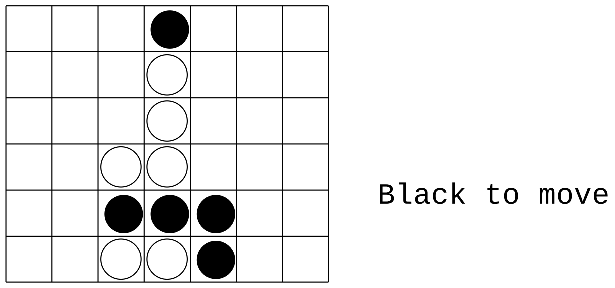


Diagram 3.3.

White has just played his 6th move and it is Blacks turn. Black already has his threats at b2 and f2, and, having read the former sections, Black knows that eventually White has to play b1 or f1 (if Black can prevent White from winning before that time). So Black tries to find possible other threats White can compose. There are of course the vertical possibilities, but these are always easy to refute. Diagonally White can achieve nothing, without a square in the range b2-b6 or f2-f6. So Black does not have to worry about that either. Horizontally the same is true, except for the group a1-d1. This group can be completed without using a square in b2-b6 or f2-f6. If White takes a1, Black has to play b1 and then White can get b2. So to destroy White’s last hope, Black plays a1. We now know that White cannot complete groups either horizontally, vertically or diagonally. Moreover we know that White will be forced to play b1 or f1 eventually. Therefore we know that Black will win.

§ 3.2. Another Useless Threat

Let us look at diagram 3.4:

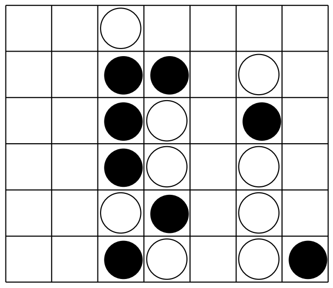


Diagram 3.4.

In this position Black can win if he gets square e3, while White can win if he gets e4. Now suppose the remainder of the board is filled in without one of the players winning, before one of the players starts playing in the e-column. As we know, White will have to play e1. Black then has to play e2. White then takes e3, preventing Blacks possible win there, after which Black does the same on e4, preventing White from winning. This way the game will end in a draw.

Although this time the threat on e3 did not help Black to win the game, it was easy to see that e4 was without a chance from the beginning. If White was to get e4, this would have meant that Black was forced to play e3. Black probably would not mind, since this would immediately let him win the game.

In general: a threat for one player one square above a threat for the opponent is useless, since either the opponent will win before the square is played, or both threats are refuted.

§ 3.3. Odd and Even Threats

As we have seen before, a threat can only be used to win, if at a certain time during the game the opponent has, for some reason, to play the square below the threat. Most of the time, this happens if all other columns are filled up completely. If that is the case, we have shown that White will get the odd squares and Black will get the even squares. If in the last column White has only even threats and Black has only odd threats, all threats will be refuted and the game will be drawn. This shows that for Black the best threats are even threats, while White likes to get odd threats.

Now suppose both players have been able to create a good threat. White has an odd one, while Black has an even one. If these threats are in the same column, the lower threat will win. Let us see what happens if the threats are in different columns, as is the case in the position of diagram 3.5.

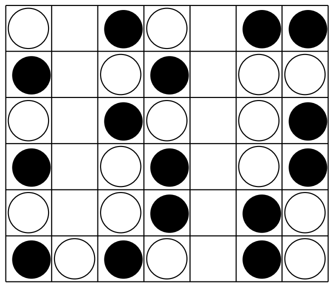


Diagram 3.5.

In diagram 3.5 White has an odd threat at b3 to solve a4-d1, while Black has an even threat at e2, to solve c4-f1. The remainder of the board has been filled up and now one of the squares b2 or e1 has to be played. Since the number of played men (as well as the number of empty squares) is odd, Black has to move. If he plays b2 he will lose immediately. If he plays e1, White can refute Blacks threat. He chooses the lesser of two evils and plays e1. Now White and Black will alternately play in the e-column giving the position of diagram 3.6.

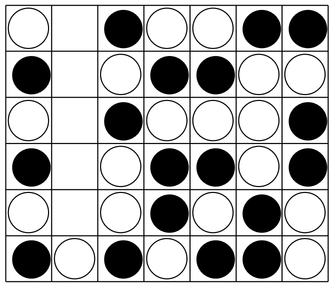


Diagram 3.6.

Once again the number of empty squares is odd and it is Blacks turn. This time there is no way to escape. He plays b2, White answers b3 and wins.

What can be concluded from the last example? If both players have a good threat in different columns (an odd threat for White and an even threat for Black), the white threat is the stronger one. This is caused by the fact that the column in which the white threat lies (the b-column in the last example) will remain empty for an odd number of squares (b2-b6 in the last example). Therefore if Zugzwang happens on the remainder of the board, the squares which will be taken by the players are of the kind they normally do not get: White gets even squares, while Black gets odd squares. In our last example this allowed White to win.

There are other possible combinations of threats for the two players. Let us suppose the combinations of one threat per player, assuming that the threats are in different columns. We can now easily deduce whose threat will be the strongest (we assume that no other threats will be created).

1) White has an odd threat, Black has an even threat.
As we have seen, White will win the game in this case.

2) White has an even threat, Black has an even threat.
Since there is no column in which only an odd number of men can be played, both players will get their normal squares. Therefore Black can refute White’s threat, and win afterwards on his own threat.

3) White has an even threat, Black has an odd threat.
Both players have a threat they cannot really use. Black will get even squares and White gets odd squares. Therefore the game will be drawn. Notice that Black has to give up his odd threat, before he will be able to play in the column where White has his even threat. If he does not want to do this, an odd number of squares remain empty in the column where Black has his threat. This will cause odd and even to change sides in the other column, giving White even squares and let him win the game.

4) White has an odd threat and Black has an odd threat.
For this position we look at diagram 3.7.

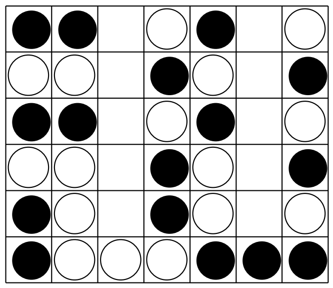


Diagram 3.7.

White has an odd threat at f3 (d1-g4), while Black has an odd threat at c3 (b4-e1). Both columns are filled as far as possible and now one of the players has to move. Since both columns contain an odd number of empty squares, and odd plus odd makes even, it is White's turn. Therefore White has to give up his own odd threat. This will result in filling up the f-column. If the f-column is filled up completely, Black has to move, which will cause him to get c2 instead of c3. In general both players will therefore lose their odd threats. In this particular example Black has another trick left for White. While filling up the f-column, Black gets f5. Now he can win by playing c2 (c2-f5) or c3 (b4-e1). Since White cannot stop both, Black wins.

The last example shows that although the given rules apply to all positions where the two mentioned threats are the only possible ways to win for one of the players, one has to be careful if other threats can still be created.

The main conclusion is, however, that there is no luck involved in winning by an early created threat. By simply looking at the difference between odd and even threats one can determine whose threat will be the most important.

§ 3.4. Tactics

Were the former sections devoted to strategic rules, rules which will affect play for more than only a few moves, tactics is also very important in order to play Connect-Four at an high level. A simple trick each player who has played the game at least a few times will have seen before, is shown first.

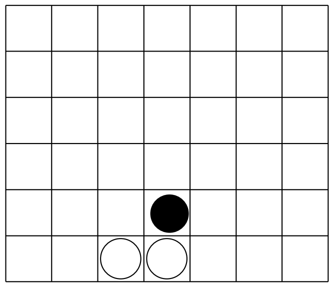


Diagram 3.8.

The position of diagram 3.8 arose after

1. d1, d2
2. c1, ..

Black can now very easily lose if he does not play b1 or e1. Suppose Black answers d3. White then plays e1 and the position of diagram 3.9 arises:

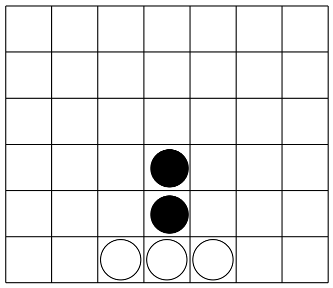


Diagram 3.9.

Now Black has to defend two threats, one on b1 and one on f1. This is not possible and therefore White will win the game. On all second moves of Black except for b1 and e1, White can play this trick.

This diagram shows that one should not only try to create threats which can work in the long run. Care should also be taken to avoid a sudden loss. Another position which is well-known is shown in diagram 3.10.

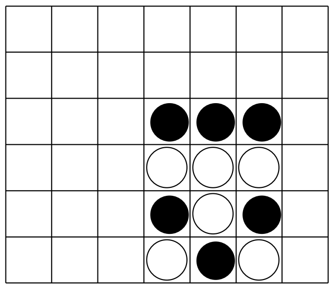


Diagram 3.10.

White now can win early by starting to fill the g-column. After White has played g2, Black must defend g3 and White will win on g4 (d1-g4). This shows that a player which has two threats in one column directly above each other, can simply start to fill the column, until he wins. This cannot be done, if the other player has a threat in the same column, which is below the highest square of the double threat. In diagram 3.1 White not only had two threats above each other, but no less than five, since he could win at any square between b2 and b6. As we have seen, it did not help him at all, since Black could easily win the game, because he had a threat on square b2.

A deeper tactical trick is shown in diagram 3.11:

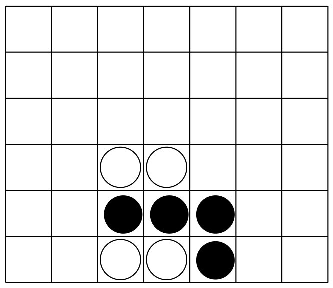


Diagram 3.11.

This position arose after:

1. d1, d2
2. c1, e1
3. d3, c2
4. c3, e2

White is now to move. We have seen before that if Black can play a1, he will win on b2 or f2. In this position White can get a winning position. White plays first a1. Black has to answer b1. This position is shown in diagram 3.12:

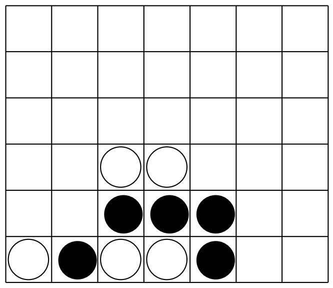


Diagram 3.12.

Now White plays b2 (forced) and Black again has no choice playing d4, giving the position of diagram 3.13.

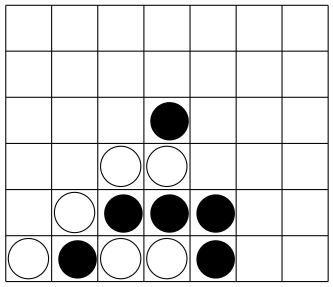


Diagram 3.13.

Now White plays b3, creating an odd and therefore nice threat on a3. Black has to answer e3. White then has to play e4. Therefore the position of diagram 3.14 can be reached from diagram 3.11, without Black having had any choice.

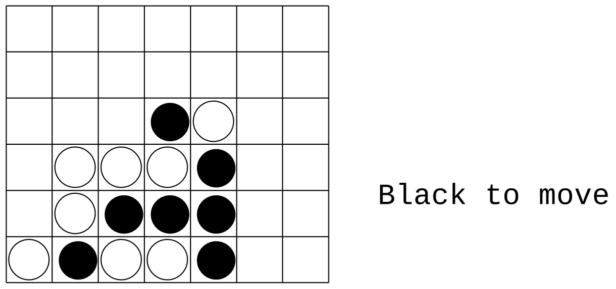


Diagram 3.14.

Using VICTOR, which is described in chapter 9, it can be shown that the odd threat White has on a3 is strong enough to win the game. In this example a good looking position for Black, turned out to be lost after Black had to play 3 forced moves! A nice example of the tactics sometimes involved. Each player should therefore try to achieve some good threats, but he should always be aware of tactical tricks which might ruin a position for him.

4. Control of Zugzwang

As has been shown in chapter 3, Zugzwang is a very important concept in playing Connect-Four. In finding the result of a game starting at a given position after correct play of both players, it is very important to determine which player has control of the Zugzwang.

First we try to define Zugzwang: Zugzwang forces a player to make a move which he would rather not make. The force simply consists of the fact that each turn exactly one move must be made.

§ 4.1. Definition of Control of Zugzwang

To show what is meant by this, we look again at a position we have encountered before (diagram 4.1):

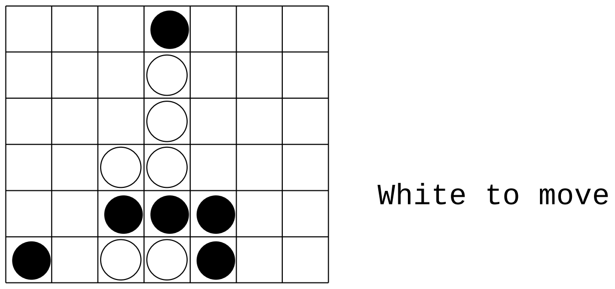


Diagram 4.1.

We have already seen that this position can be won by Black, since he will win at b2 or f2, depending on which column White chooses to lose in. Now suppose the following moves are made:

7. c4, a2.

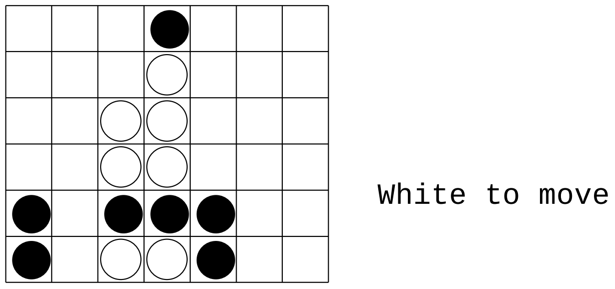


Diagram 4.2.

In the position of diagram 4.2 all columns contain an even number of men, consequently leaving an even number of squares empty. This means that it is White’s turn. Now White has to play in one of the columns, which will result in an odd number of empty squares left in that column. Since this means that White cannot fill a column completely with his next move, Black can then answer in the same column as White just played. That would result in a position in which all columns contain again an even number of empty squares.

Now let us suppose that Black is going to use this simple strategy for the rest of the game: each move he plays in the same column as White just played in (we call this strategy ’to play follow-up’.) Which squares will White get this way and which squares will Black get? The answer to this question is simple: White will only get odd squares, while Black will only get even squares. White can choose the order in which he fills the columns, but there is no way he can force Black to take an odd square. If Black wants to play follow-up, he can do so, and White can do nothing about it.

In our example, White cannot play b1 or f1 without losing immediately. Let us therefore assume that he starts filling up the other columns, while Black is playing follow-up. This will eventually result in the position of diagram 4.3:

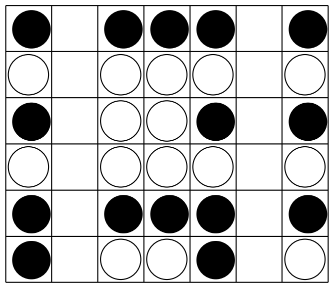


Diagram 4.3.

The order in which White has filled up the columns is of no importance. The result will always be the same. It is now clear that White will lose after his next move.

In this diagram, to play follow-up was a winning strategy for Black. One nice feature of the strategy was, that White had no chance to interfere. Black wanted to get all even squares and White could not do anything about it. If, however, Black wanted to get one or more odd squares, he could have taken them. The only price he would have to pay, is that it would give White the opportunity to get an even square.

Concluding this section, we see that Black is able to take all even squares, without White being able to do anything about it, while Black can take some odd squares if he wants to, the price of which is the loss of some even squares. This enables us to define the concept of Control of Zugzwang: A player is in control of the Zugzwang if he is able to guide the way odd and even squares are divided among both players.

Being in control of the Zugzwang is important, since it can be used in positions like diagram 4.2, to win without any possible tactical problems.

§ 4.2. Control in Initial Position

Let us now try to see, what can be said on which of the players is in control of the Zugzwang as the game starts. Since all columns are empty, the number of empty squares in each column is even. Therefore Black is in a position to play follow-up, which means that Black is the person who controls the Zugzwang. Now suppose Black tries to use this control to play follow-up from the beginning. If he could do that for all of his 21 moves, the position of diagram 4.4 would result:

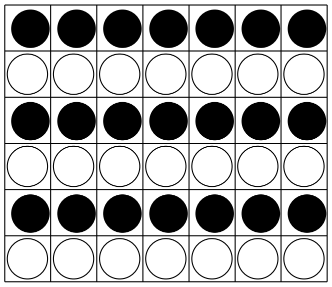


Diagram 4.4.

It is clear that this position is illegal. Both players have connected four men at several positions. Still it is interesting to try to find out which player must have won first, after which the players just played on. Since White controls the whole first row, it is clear that before Black could have finished one of his groups of four, White must have connected four men in the first row. Therefore it is clear that White must have won the game.

This shows that playing follow-up is not a good strategy to start the game, for Black. He will have to get some squares at the bottom row and give White the opportunity to take some even squares in return. If White does not take the right squares, like he did in the game which brought him to diagram 4.1, Black can always switch to playing follow-up and win.

§ 4.3. Control for White

Now let us see what White can do. Is there, for instance, a possibility that he can take over control of the Zugzwang? To see that this is indeed possible, we look at the position of diagram 4.5:

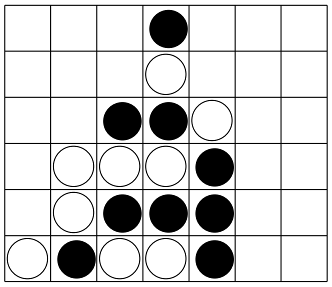


Diagram 4.5.

White is to move. Two columns contain an odd number of empty squares, and five columns contain an even (possibly zero) number of empty squares. If we look at the a-column, it is clear why Black has not yet played there. As soon as Black plays a2, he will lose the game, since White has a threat at a3. This odd threat is a good one for White, as we have seen before. Therefore, from Blacks point of view, playing in the a-column is out of the question. Since the a-column contains an odd number of empty squares, this will however change the way odd and even squares are taken normally by the players, on the remainder of the board. White, of course, knows this and plays b4. Now all six columns of the remainder of the board contain an even number of empty squares. At this time Black has to move. Since he cannot play in the a-column, he has to concentrate on the remainder of the board and play an odd square. White can, if he wants, play in the same column as Black, creating a new position in which Black has to play an odd square. In other words, White can play follow-up on the remainder of the board. Therefore, in this position, White controls the Zugzwang. Let us see what happens if Black and White fill up the board, except for the a-column, White playing follow-up all the time (diagram 4.6):

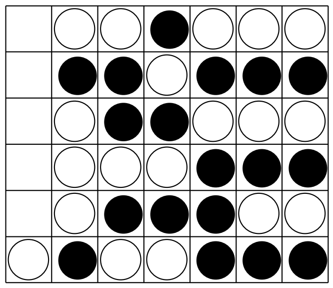


Diagram 4.6.

Neither of the players has connected four men. Therefore the game will be still in progress, when this position occurs. Black has to move, and loses as White plays a3 (Black, knowing all this, could have forfeited the game before, naturally).

This diagram made clear that an odd threat for White gives him control of the Zugzwang as soon as all other columns contain an even number of empty squares. White will win the game, if Black cannot connect four men, when White plays follow-up, since Black will be forced to play eventually the square below the odd threat. If, however, Black will be able to connect four men - if he is forced to play against the follow-up - White will have to take some odd squares, giving Black some even squares. Still, White is the one who can decide when to stop playing follow-up and is therefore the one in control of the Zugzwang.

§ 4.4. Some Special Control Positions

In the last sections, we stated that playing follow-up is possible if all columns (except for columns as the a-column in the last diagram) contain an even number of empty squares. The position of diagram 4.7 shows that it can be played in other positions as well:

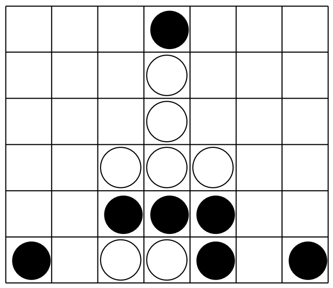
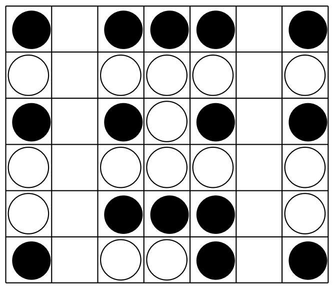


Diagram 4.7.

This position arose from diagram 4.1, after playing
7. e3, g1.

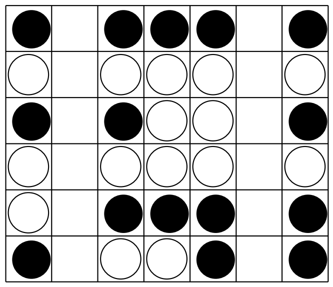
Here, Black is just waiting for White to play b1 or f1. Therefore Black would like to play follow-up. If we count the number of empty squares in the different columns, we see 4 columns with an odd number of empty squares. This gives White a choice between four different even squares, something which the player of follow-up, does not want to happen. Still Black can play a somewhat modified follow-up, by observing that no real harm is done if White gets one or more of these even squares. Therefore Black will try the following: as soon as White plays a2, Black answers at b4. If White plays b4, Black answers at a2. As soon as White plays e4, Black answers at g2. If White plays g2, Black answers e4.

This will give Black exactly one of the pair of squares a2, b4 and one of the pair e4, g2. After the squares of such a pair are used, the number of empty squares in the column is even and follow-up can be played. From diagram 4.7 as starting position, this will result in one of the following four positions of diagrams 4.8 to 4.11. (if White does not give up before by playing b1 or f1).



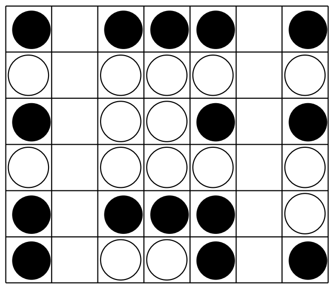
White to move

Diagram 4.8



White to move

Diagram 4.9



White to move

Diagram 4.10

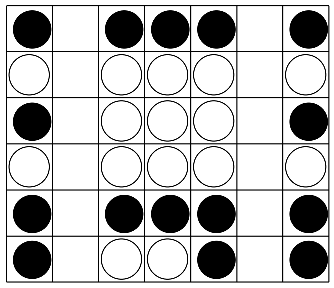

White to move

Diagram 4.11

As can easily be detected, in each of the four given positions, White has not been able to connect four men and will lose after his next move.

This shows that follow-up can also be played if some columns contain an odd number of squares. The number of these columns must be even, since the total number of empty squares is even, if the opponent of the follow-up player has to move. The follow-up player simply groups the playable even squares (in our example a2, b4, e4 and g2) in pairs of two (like we paired a2, b4 and e4, g2). Of each pair exactly one square will be taken by White and one square will be taken by Black. Which square is taken by the follow-up player, can be chosen by his opponent, while the pairing can be done by the follow-up player himself. For all other squares, the normal follow-up rules apply: even squares are taken by the follow-up player, while the odd squares are for his opponent.

§ 4.5. Benefits of Controlling the Zugzwang

As we have seen in the last sections, Black controls the Zugzwang in the initial position, but cannot play follow-up, since that way he would lose the game. Because Black cannot play follow-up, there is a chance that White will create an odd threat, which gives him the control of the Zugzwang, in all columns, except the column where he has the odd threat. If White cannot win from such a position by playing follow-up, he also has a chance of losing the control again. Although we can talk about one player controlling the Zugzwang, it cannot be used (yet) to win the game, or at least draw the game.

If White controls the Zugzwang, like in diagram 4.5, where follow-up would give Black no chance to connect four men on the remainder of the board, White will win the game, using his odd threat. In this case the control is winning for White. If Black controls the Zugzwang, like he did in diagrams 4.2 and 4.7, where follow-up would give White no chance to connect four men, and Black can complete a group of his own (b2-e2 of c2-f2), Black will win the game. It is, however, not necessary for Black to have an even threat to be able to use follow-up. Diagram 4.12 will show such a position:

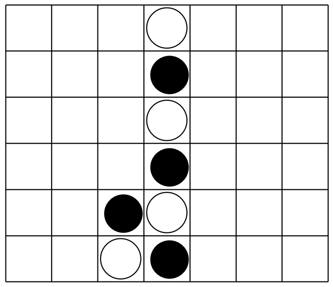


Diagram 4.12.

This position arose after

1. c1, d1
2. d2, d3
3. d4, d5
4. d6, c2.

Since all columns contain an even number of empty squares, Black can apply follow-up in its purest form. This will result eventually in the position of diagram 4.13:

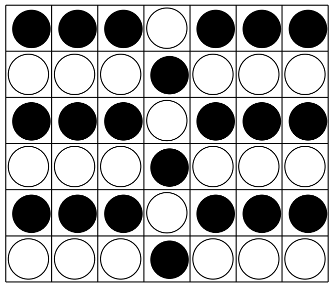


Diagram 4.13.

The position of diagram 4.13 is a draw. Since this position was obtained by playing follow-up by Black, starting in the position of diagram 4.12, we see that the follow-up strategy can sometimes be used to get a draw. Although we now know that the position of diagram 4.12 is at least a draw for Black, we do not know yet if White can also be sure to get a draw, or Black can actually win. Still the result that we can show that a position can be drawn by Black, can be used while we try to show that a position can be won by White. As soon as we encounter a variant in which Black can draw the game, we know that White has made wrong move before, if White is trying to win. This observation will be used in chapters 10 and further.

5. More Strategic Rules

In chapter 4 we have shown that a simple strategic rule as playing follow-up might reveal that a given position can be won by one of the players, although the actual win may take another 20 moves. We have also shown that in some positions where not all columns are filled with an even number of men, it was possible to play a special kind of follow-up, after pairing some of the even squares which were directly playable. In this chapter we will look at some other strategic rules. All these rules can be used to show that a given possible potential threat for the opponent can be refuted. A potential threat for player A is a group of four men which can be connected by player B. The only restriction for the squares of the potential threat is the fact that player A may not have occupied any of the squares. A potential threat becomes a threat, as soon as exactly three squares are occupied by player B. The rules in this chapter are described in an informal way. For exact definitions of the rules, with proofs of their validity, we refer to chapter 6. The given rules are valid for the player who controls the Zugzwang. In this chapter we always assume Black controls the Zugzwang.

§ 5.1. Baseinverse

Let us look at the position of diagram 5.1:

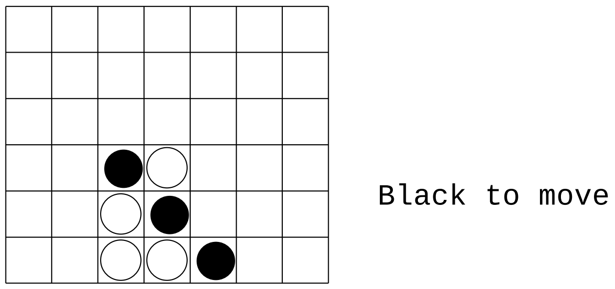


Diagram 5.1.

Black is to move. Black controls the Zugzwang, since White has not yet created an odd threat. Black cannot, however, play follow-up, since that way he would lose on a1-d1. Does this mean that he has to stop White immediately by playing a1 or b1? If White plays a1, he has a group of three men, needing only b1, an odd square. Still it is clear that a1-d1 is not a serious threat since as soon as White plays a1, Black can answer b1 and as soon as White plays b1, Black can answer a1.

This gives rise to a more general rule: if two squares are directly playable, and both squares are part of the same group, player A can always prevent player B from completing the group, by playing one square as soon as player B plays the other.

In diagram 5.1 this will assure Black, that he gets one of the squares a1 and b1. Which square he gets depends on White, unless Black will play a1 or b1 immediately, since it is his turn. Now suppose Black wants to get b1, and in the same way he not only wants to have one of c4 and d4, but wants to have d4 for sure. He can have one of each group, and for one group can choose which one he wants, since he can make one move immediately. In this way it is important to know if he has used up his immediate move at b1 or not. It would be much easier to look at the position where White is to move. In that case Black cannot claim to get b1, since White could play it immediately, so Black has to settle for one out of a1 and b1 and one out of c4 and d4, since it is not his move. Therefore let us look at a position where Black made his move (diagram 5.2):

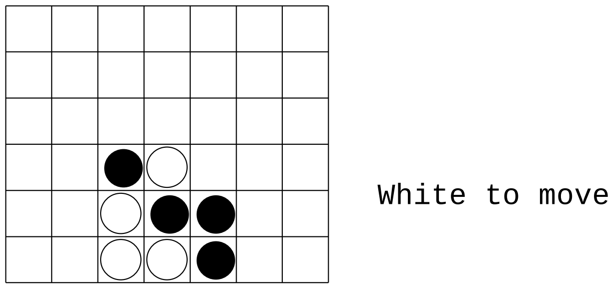


Diagram 5.2.

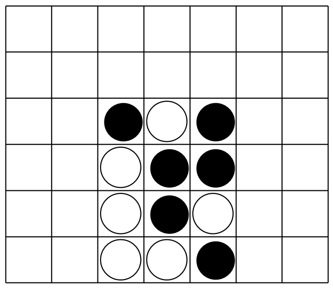
We have seen that Black can get one of a1 and b1, but White can decide which it is going to be. We call this position a Baseinverse. Each rule which gives one of the players a square which he would not get, if he played follow-up, is called an inverse. In this case Black gets a square, which can be taken immediately by the opponent. The reason why Black can claim one of the squares, is that they are both located at the bottom (base) of a column. This rule applied to a1 and b1 ensures Black that White will not win the game at a1-d1. A complete description of the Baseinverse can be found in section 6.2.

§ 5.2. Claimeven

Now let us take a look at the group b1-e4 of diagram 5.2. White already occupies c2 and d3. b1 can be taken by White if he wants to, so it is important for Black to be able to get e4. Fortunately, this is an even square and Black will normally get this square. Black can therefore get e4 if he will be able to hold control over the Zugzwang long enough to force White to play e3. This will be the case if Black can find answers to all White’s threats. How Black can try to refute all White’s threats, will be shown in the next chapter. For now it suffices to note that if Black controls the Zugzwang, he can claim an even square, if the odd square below the even square is still empty. In that case he will take the even square as soon as White has played the odd square below.

§ 5.3. Combination of Rules

Let us take a look at diagram 5.3:



White to move

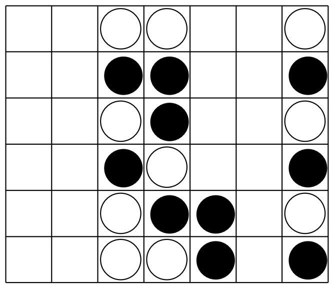
Diagram 5.3.

Suppose Black wants to be sure that he will not lose at a1-d1. Therefore he applies the Baseinverse rule, which ensures him one of the squares a1 and b1. On the other hand he wants to be sure he will not lose at a1-d4. Therefore he applies the Claimeven rule for b2. Now it must be clear that something is wrong. The Baseinverse expects Black to play a1 as soon as White plays b1, while the Claimeven expects Black to play b2 as soon as White plays b1. Therefore these rules cannot be applied both in the given position. It is interesting, to find out in what cases these rules can be applied simultaneously. For the two given rules, this is easy to detect: a Baseinverse is concerned with two squares. A Claimeven is also concerned with two squares. Both rules state that if one of the squares is played by White, the other should be played by Black immediately. If therefore the groups of squares are not disjoint, we cannot be sure that these conditions can both be met, and therefore these rules can only be applied simultaneously, if the sets of squares concerned are disjoint.

§ 5.4. General Usage of Rules

We have seen that some simple rules can be used to claim one of a set of squares by the controller of the Zugzwang. For a Claimeven this only works, if the Zugzwang can be controlled until the end of the game. On the other hand, Black can only be sure that the Zugzwang will be controlled to the end, if White cannot make a serious threat. So what we have is a set of rules which can be used to show that certain potential threats of White can be refuted. Some of these rules only work if Black can be sure that White cannot make a serious threat. Somehow this sounds like a circular reasoning. It is not, however.

To clarify this, we look at the position of diagram 5.4.



White to move

Diagram 5.4.

Let us look at all positions where White might be able to create a position of four connected men:

(1) a1-d1,
(2) a6-d6, b6-e6,
(3) c6-f6, d6-g6,
(4) d3-g6, c2-f5, b1-e4, e3-e6,
(5) a1-a4, a2-a5, a3-a6, a4-d1,
(6) b1-b4, b2-b5, b3-b6,
(7) f1-f4, f2-f5, f3-f6,
(8) a6-d3.

Now let us see how we can refute all these potential threats.

a1-d1 can be refuted by a Baseinverse on a1 and b1.
All groups in (2) can be refuted by claiming the square b6, using Claimeven.
All groups in (3) can be refuted by claiming the square e6, using Claimeven.
All groups in (4) can be refuted by claiming the square e4, using Claimeven.
All groups in (5) can be refuted by claiming the square a4, using Claimeven.
All groups in (6) can be refuted by claiming the square b4, using Claimeven.
All groups in (7) can be refuted by claiming the square f4, using Claimeven.
a6-d3 can be refuted by claiming the square a6, using Claimeven.

This list shows that for each potential threat we can find a way to solve the threat. This means that if we hold on to a strategy which gives us all those squares needed, we will not lose control over the Zugzwang, which will guarantee, that all squares can be taken. So in order to be able to claim a square at a spot, which refutes some potential threats, we must be sure that there is a set of rules which will ensure that all threats can be refuted. If there is at least one potentially dangerous group, for which we cannot find a refutation, we cannot be sure of the other claims we would like to make either.

This situation can be compared with the building of a house of cards. When we build the house, all components must be placed in the right way. If all components are placed correctly, the house will be finished (until a nice passer-by blows it away). If one component is placed incorrectly, the whole thing will come down.

It is important to see that there is no circular reasoning involved here. Merely we have that, when this approach succeeds in finding refutations for all positions, Black can at least get a draw. If there is only one spot left where we cannot find a rule which shows how to refute the potential threat, the whole set of rules comes down like a house of cards. We cannot claim anything any more.

Concluding, we have found a method where we can sometimes find a set of rules which ensures the controller of the Zugzwang that he will not lose, since he can refute all the opponents threats. If the method does not work, we can say nothing about the given position. Therefore it is useful to find as many rules as possible, which can be applied, in order to be able to predict the result of a game in as many positions as possible. In the next chapter the set of rules found so far is described.

6. Formal Definitions of Rules

In this chapter a formal definition is given of the nine rules which are used to refute potential threats of the opponent. In section 6.10 it is described how these rules where found. As has been stated before, the rules can only be applied by the person who controls the Zugzwang. Rules are always applied in positions where the opponent is to move. Since Black controls the Zugzwang if White has not created an odd threat, the rules can be applied on the whole board. If White controls the Zugzwang, the rules can only be applied on the board without the column in which White has his odd threat. Even if White has created an odd threat, Black can still try to use the rules to show that he can achieve a draw. Normally this will not work, since there are no possibilities to refute the odd threat. There are however positions where an odd threat can be refuted. In that case Black clearly can get back control of Zugzwang. Therefore in a position in which White is to move, Black can always try to find a good set of rules, while White can do so, if he has created at least one odd threat (later we will encounter another position in which White can apply the rules). For simplicities sake, the rules are described without referring to the player who controls the Zugzwang. The reader should keep in mind that if White is in charge, he can only apply the rules on squares which are not in the column of his odd threat.

The names of the rules are chosen in a way to make clear on what observation the rule is based. It is described which potentially threats are solved by applying a rule in a given position. Instead of the term ’potential threats’ the term ’groups’ is used to indicated a position of 4 connected squares, which could be used by the opponent to connect four of his men. The term ’opponent’ is used to refer to the opponent of the controller of the Zugzwang.

§ 6.1. Claimeven

A Claimeven makes direct use of the fact that the controller of the Zugzwang can get all empty even squares (which are not directly playable), by letting the opponent play all empty odd squares. So in individual cases, the controller gets an even square, for the price of the odd square below the even square. A Claimeven is therefore concerned with two squares, an odd and an even square, both empty, lying directly above each other. All groups which contain the even square are solved by the Claimeven.

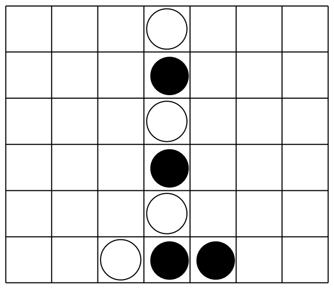


Diagram 6.1.

In the position of diagram 6.1, the following Claimeven instances can be applied:

a1-a2, a3-a4, a5-a6,
b1-b2, b3-b4, b5-b6,
c3-c4, c5-c6,
e3-e4, e5-e6,
f1-f2, f3-f4, f5-f6,
g1-g2, g3-g4, g5-g6.

Since, as will be shown in chapter 7, all these instances of the Claimeven can be applied simultaneously and each potential threat of White contains at least one of the even squares of the listed Claimeven instances, this set of rules can be used to show that the position of diagram 6.1 is at least a draw for Black.

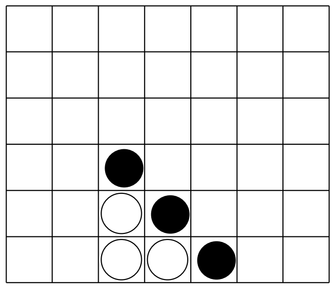
Claimeven formally:

Required: Two squares, directly above each other. Both squares should be empty. The upper square must be even.

Solutions: All groups which contain the upper square.

§ 6.2. Baseinverse

A Baseinverse is based on the fact that a player cannot play two directly playable moves in one turn. Therefore, after the opponent has made his move the controller of the Zugzwang can still play at least one of the two squares. Since the opponent can move first, he is the one who can decide which square he gets, and which square will be taken by the controller of the Zugzwang. Therefore the controller of the Zugzwang can only be sure that he gets at least one of the two squares, in that way solving all groups which contain both squares.



White to move

Diagram 6.2.

In the position of diagram 6.2, the following Baseinverses can be applied by Black:

a1-b1, c4-d3, c4-e2, c4-f1, d3-e2, d3-f1, e2-f1.

A Baseinverse like a1-c4 is also possible, but useless, since there are no groups which contain both a1 and c4. Therefore such a Baseinverse would solve nothing. Another possible Baseinverse in diagram 6.2 is f1-g1. Although d1-g1 contains both squares, this Baseinverse is of no use either, since d1-g1 is no potential threat as Black occupies e1.

Which groups are solved by some of the listed Baseinverse instances?

a1-b1 solves a1-d1,
c4-d3 solves a6-d3, b5-e2 and c4-f1,
d3-f1 solves c4-f1.

These three instances show that it is very well possible that one kind of rule (Baseinverse) can have different instances (c4-d3 and d3-f1) to solve the same group (c4-f1).

Baseinverse formally:

Required: Two directly playable squares.

Solutions: All groups which contain both squares.

§ 6.3. Vertical

A Vertical is based on the fact that a player cannot play two men in the same column in one turn, while by playing one men in the column, the square above that men becomes immediately playable. This means that a player can always claim one of two consecutive squares in a column: if the opponent plays the lower square, the claiming player can take the upper square immediately. If the opponent never plays the lower square, the player can take that square himself sometime (he might even be forced to do so). In both ways, at least one of the two squares will be occupied by the player in control of the Zugzwang. If the upper square is even, a stronger claim can be made which has been done by Claimeven. Therefore the Vertical rule is only used if the upper square is odd. Since the opponent can determine which square will be taken by which player, we can only be sure that groups which contain both squares are solved by this rule. The name of the rule is based on the observation that all groups which contain both squares are Vertical.

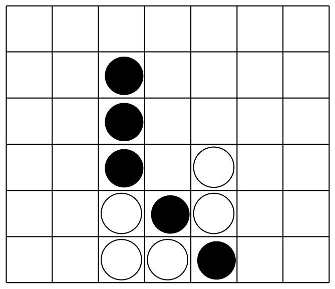


Diagram 6.3.

In the position of diagram 6.3 a great number of Verticals is possible. The most interesting is e4-e5. At this point, where we cannot use a Claimeven for e4 and cannot use a Baseinverse to solve the potential threat e2-e5, we should apply the Vertical e4-e5 to make sure that White cannot win at e2-e5. Note that application of this rule, also solves e3-e6.

Vertical formally:

Required: Two squares directly above each other. Both squares should be empty. The upper square must be odd.

Solutions: All groups which contain both squares.

§ 6.4. Aftereven

Aftereven uses a special side-effect of the usage of one or more Claimevens. If a group can be completed by the controller of the Zugzwang, by filling squares which all can be claimed using Claimevens, then this player can be sure that he will eventually complete the group. In the column where he plays the last man which completes the group, no other man will be thrown. Therefore the opponent cannot get any of the squares above the last square of the Aftereven group. Which column will be the last, if the Aftereven group needs at this time more than one man, cannot be determined beforehand. The order in which those columns are filled can most of the time be determined by the opponent. For this reason the only thing we can conclude is, that in at least one of the columns which contain empty squares of the Aftereven group, no men will be put by the opponent, above the square of the Aftereven group. Therefore groups of the opponent, which have squares in all columns where a man of the Aftereven group is still missing, all higher than the Aftereven group, cannot be completed. These groups all can only be completed after the group which needs only even squares, clarifying the chosen name for the rule.

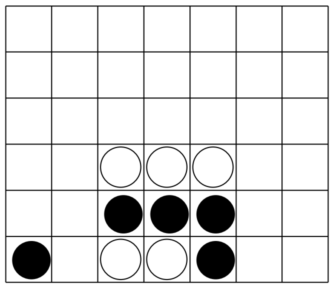


Diagram 6.4.

In the position of diagram 6.4, a Claimeven can be used on the squares f1-f2, while f2 is the only square missing to complete c2-f2. For this reason we can see c2-f2 as an Aftereven group, with only one empty square. Therefore no squares will be played after f2, which solves all groups which need a square in the range f3-f6. As a side effect of the Aftereven, all groups which need the square f2 itself, are also solved. In the same way b2-e2 is an Aftereven group, which needs only b2. This solves all problems that need one or more squares in the range b2-b6. These two Afterevens and some Claimevens in the other columns, solve all groups. This assures Black that he will win the game, at b2 or f2.

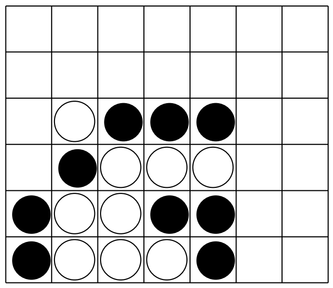


Diagram 6.5.

In the position of diagram 6.5 Claimevens can be used for f2 and g2. Together they form an Aftereven solution, with d2-g2 as Aftereven group. This time more than one square is needed and therefore all squares in f3-f6 will not be played, or all squares in g3-g6 will not be played. The choice between these columns can be made by White. Therefore only groups which contain squares in both the f and g-column are solved by this Aftereven. c3-f3, for instance, is not solved by this Aftereven. This group will become a serious threat for Black, since there is no rule which can be used to solve this problem.

Aftereven formally:

Required:
A group which can be completed by the controller of the Zugzwang, using only the even squares of a set of Claimevens. This group is called the Aftereven group. The columns in which the empty squares lie are called the Aftereven columns.

Solutions:
All groups which have at least one square in all Aftereven columns, above the empty square of the Aftereven group in that column.
All groups which are solved by the Claimevens, which are part of the Aftereven.

§ 6.5. Lowinverse

A Lowinverse is based on the fact that the sum of two odd numbers is even. Normally the controller of the Zugzwang will be forced to play the lowest, even square of a column which contains an odd number of empty squares. If, however, two columns with an odd number of empty squares exist, the total number of empty squares in these two columns is even. This will force the opponent to play first in one of these column. In that way the controller of the Zugzwang gets a chance to take the odd square above the square which will be played by the opponent. The opponent decides in which column he will play first, implying that we can only conclude that we get one of the two odd squares.

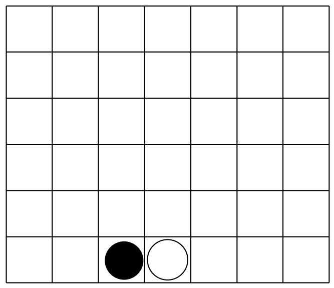


Diagram 6.6.

In the position of diagram 6.6, a Lowinverse can be used for the squares c2-c3-d2-d3. The result of this is, that Black gets at least one of the squares c3 and d3. This effect is the goal of the Lowinverse. This rule has a side-effect, too. The rule can be seen as consisting of two Verticals. Black will play d3 as soon as White plays d2, or c3 as soon as White plays c2. After one of these variants occurred, two squares remain, which Black can start treating as a normal Vertical. In general the controller of the Zugzwang will get at least one square out of each of three pairs of squares: the squares in the first column (c2-c3), the squares in the other column (d2-d3) and the two upper squares (c3-d3). Note that a Lowinverse on c4-c5-d4-d5 is also possible in the position of diagram 6.6. It is not necessary for the lower squares of the Lowinverse to be playable.

It is necessary to see that there is no restriction which states that the upper squares must lie in the same row. A Lowinverse on, for instance, a2-a3-c4-c5 is very well possible. The only observation needed is the fact that the upper fields (a3 and c5 in this example) are odd

Lowinverse formally:

Required:
Two different columns, called the Lowinverse columns. In each Lowinverse column two squares, lying directly above each other.
All four squares must be empty.
In both columns the upper of the two squares is odd.

Solutions:
All groups which contain both upper squares.
All groups which are solved by the Verticals, which are part of the Lowinverse.

§ 6.6. Highinverse

A Highinverse is based on the same principle as the Lowinverse. This time the two even squares above the upper odd squares are also taken into account. What happens at the time the opponent is forced to play in one of the two columns? To see this we look again at the position of diagram 6.6. Suppose White plays c2. Black then can play c3 and four squares remain empty: c4, d2, d3 and d4. Since Black has already achieved his main goal of the inverse: getting one of the squares c3 and d3, he now can try to get some more. He could for instance change d3 and d4 into a Claimeven. In that case he gets at least c3 and d4. If we suppose that White plays d2 first, we can show in the same way that Black gets at least c4 and d3. The choice between these two alternatives can be made by White. Therefore, all we can conclude from this is that Black gets from each of the following pairs at least one square: c3 and c4, d3 and d4, c3 and d3, c4 and d4. The last two of these pairs should be seen as the main goals of the Highinverse, while the other two are Vertical side-effects.

Note that it is easy to achieve only one of the main goals by using Claimevens (c4 and d4) or a Lowinverse (c3 and d3), but for the combination of these goals a Highinverse is needed.

In the position of diagram 6.6 Black can claim something more. After White played c2 and Black answered c3, both c4 and d2 can be played immediately. Therefore Black can not only use a Claimeven for d3-d4 but can also use a Baseinverse for c4 and d2. Although that is no use in this position, it would help if the columns where one column apart. In the other variant Black gets c4 for sure, which means that anyhow, Black gets one of c4 and d2, because d2 is directly playable. In the same way Black gets either d4 or c2, since c2 is directly playable.

As has been stated for the Lowinverse, there is no restriction which says that the upper squares of the Highinverse must be in the same row. A Highinverse on, for instance, a2-a3-a4-c4-c5-c6 is therefore possible.

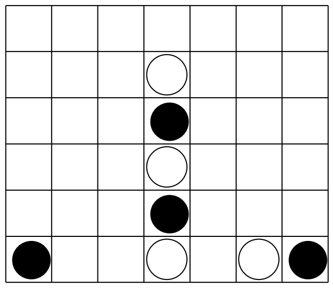
Highinverse formally:

<u>Required:</u>
Two different columns, called the Highinverse columns. In each Highinverse column three squares, lying directly above each other.
All six squares are empty.
In both columns the upper square is even.

<u>Solutions:</u>
All groups which contain the two upper squares.
All groups which contain the two middle squares.
All (vertical) groups which contain the two highest squares of one of the Highinverse columns.
If the lower square of the first column is directly playable:
All groups which contain both the lower square of the first column and the upper square of the second column.
If the lower square of the second column is directly playable:
All groups which contain both the lower square of the second column and the upper square of the first column.

§ 6.7. Baseclaim

A Baseclaim is the combination of two Baseinverses and a Claimeven. Diagram 6.7 will clarify this rule:



White to move

Diagram 6.7.

Suppose both the groups b1-e4 and c1-f1 have to be solved and no Claimeven can be used for e4. In that case a Claimeven can be tried at c1-c2, to solve b1-e4. A Baseinverse at c1-e1 is not possible, since these rules meet at square c1. This would make it impossible to solve c1-f1. The playable square b1 of b1-e4 gives another possibility. To see how this can help we look at all possible first moves of White in one of the squares b1, c1 or e1. White plays b1. Black plays e1 and a Claimeven at c1-c2 can be used to solve b1-e4. White plays c1. Black plays e1 and a Baseinverse at b1-c2 can be used to solve b1-e4. White plays e1. Black plays c1 and a Baseinverse at b1-c2 can be used to solve b1-e4.

In all variants both groups are solved. Since b1-e4 is solved sometimes by a Claimeven and sometimes by a Baseinverse, the name Baseclaim is chosen.

In general we need three directly playable squares and an even square directly above one of these squares. To simplify things, we number the columns in such a way that the even square lies above the directly playable square in the second column. The second and third playable square form the Baseinverse and the first square and the non-playable square form the real Baseclaim part. In this case the groups which contain both the second and third directly playable squares and the groups which contain both the first playable square and the non-playable square, are solved.

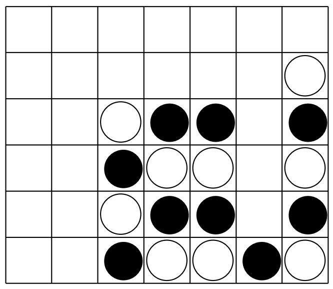
Baseclaim formally:

Required:
Three directly playable squares and the square above the second playable square.
The non-playable square must be even.

Solutions:
All groups which contain the first playable square and the square above the second playable square.
All groups which contain the second and third playable square.

§ 6.8. Before

A Before is based on a combination of Claimevens and Verticals. The following diagram shows the goal of the Before rule:



White to move

Diagram 6.8.

In the position of diagram 6.8 no solution can be found for d3-g3 using the previous rules. Still it is clear that d3-g3 is no real threat. If White plays f2, Black can play f3, and if White does not play f2, Black will, sometime, and win before White can win. In both cases White cannot win at f3. In general we need a group for the controller of the Zugzwang, which is not yet completed. If this group will be completed, not all squares which directly succeed the at this moment empty squares of the Before group, can be played, since the game will be over, as soon as the last man has completed the group. Suppose the opponent plays one of the squares needed to complete the Before group (like he would be doing by playing f2 in diagram 6.8). In that case the controller of the Zugzwang can immediately play the successor of the squares played by the opponent. This would mean that again not all squares which succeed the empty squares of the Before group can be taken by the opponent. A group which contains all those successor squares, can therefore not be solved.

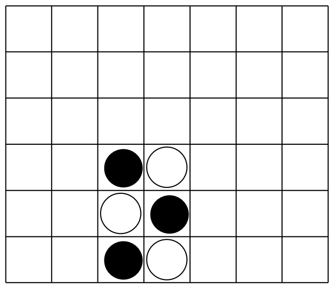


Diagram 6.9.

In the position of diagram 6.9 we take b4-e1 as Before group. b4 and e1 are still empty, which means that the Before works in this position for all groups which need both b5 and e2. The only group which needs both is b5-e2. The Before rule uses this time the squares b4-b5 and e1-e2. As soon as the opponent plays b4, b5 is played and as soon as e1 is played, e2 is answered. This will ensure Black completing b4-e1, or stopping White from completing b5-e2, in both cases b5-e2 is not dangerous. Side effect is that of both pairs b4-b5 and e1-e2 at least one square can be taken, therefore solving the vertical groups which contain both.

Now let us suppose we use a Claimeven at b3-b4 instead of the Vertical b4-b5. Is the conclusion still valid that White cannot complete b5-e2? It is, as can easily be seen by looking at the goal Black has at each empty square of his Before group: taking it, or giving it away and taken the successor square instead. If a Claimeven at b3-b4 is used, Black chooses the first variant. In general, a Before consists of a number of Claimevens and Verticals. Some of the Verticals may have an even upper square, as e1-e2 does. This does not matter. If the Before would consist of Claimevens only, it would be equal to an Aftereven. In that case we can draw stronger conclusions from the Aftereven, and therefore we only use Befores, if an Aftereven cannot be used.

Before formally:

Required:

A group without men of the opponent, which is called the Before group.

All empty squares of the Before group should not lie in the upper row of the board.

Solutions:

All groups which contain all squares which are successors of empty squares in the Before group.

All groups which are solved by the Verticals which are part of the Before.
All groups which are solved by the Claimevens which are part of the Before.

§ 6.9. Specialbefore

The Specialbefore is (what’s in a name) a special version of the Before.

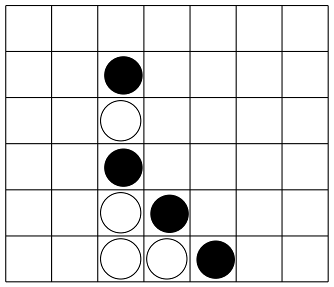


Diagram 6.10.

In the position of diagram 6.10, we can use d2-g2 as Before group. This Before can consist of Claimevens at f1-f2 and g1-g2 and a Vertical at e2-e3. This Before would solve d3-g3, while there are some side-effects caused by the Claimevens and Vertical. We have to use a Baseinverse to solve a1-d1. This way we could lose square b1, which gives us only one way to solve b1-e4: a Claimeven at e3-e4. This Claimeven would, however, be in conflict with the Vertical at e2-e3. Therefore, we have a problem.

Still something can be done. The only reason why we play e3 after e2, is to stop White from getting d3-g3. Black could also play d3 instead of e3 with the same effect. This time White is stopped and Black has the possibility to use a Claimeven for e4. What would happen if White plays d3, before he wants to play e2? In that case Black should immediately play e2 himself, in that way still doing as is dictated by the Before. To be able to play d3 or e2 when necessary, it is important that d3 and e2 are directly playable.

In general we need a normal Before group, of which one of the empty squares is directly playable. Furthermore we need another playable square. All groups are solved which contain the successors of the empty squares of the Before group and the extra playable square. Furthermore all groups which contain both directly playable squares, are solved, since at least one of these is taken (some kind of a Baseinverse). For the other parts of the Specialbefore (Claimevens or Verticals) the normal rules apply.

Specialbefore formally:

Required:
A group without men of the opponent, which is called the Specialbefore group.
A directly playable square in another column.
All empty squares of the Specialbefore group should not lie in the upper row of the board.
One empty square of the Before group must be playable.

Solutions:

All groups which contain all successors of empty squares of the Specialbefore group and the extra playable square.
All groups which contain the two playable squares.
All groups which are solved by one of the Claimevens.
All groups which are solved by one of the Verticals.

§ 6.10. Creation of Strategic Rules.

The most elementary rules are the Claimeven, Baseinverse, Lowinverse and Vertical. These four rules are based on simple observations which can be made by any experienced player of the game. These observations were made by the author of this article, and implemented in VICTOR, a program described in chapter 9.

Testing of several positions, using VICTOR revealed some problems: Positions as diagram 3.1 where not yet shown to be at least a draw for black. This was due to the fact that the white threats at b2-b6 were considered to be a problem. The evaluation function lacked the knowledge to see that after black would play b2, the game would be ended. This resulted in the invention of the Aftereven by the author. The addition of the Aftereven to the evaluation function resulted in a significant improvement in performance.

New tests showed that positions as shown in diagram 6.8 where a problem. This resulted in the invention of the Before. Although the costs of the Before where very high in implementation (the number of different instances of the Before in most positions is about one-third of the total number of instances of solutions of all different types), this resulted in a major improvement of strength of the evaluation function. The analysis given in appendix B of three start moves shows that the Before is very important. It was after the Before was implemented that the general rules for all boards with at most 6 columns where encountered.

Special situations where the usage of the Lowinverse was necessary but resulted in problems just above the fields of the Lowinverse, resulted in the addition of the Highinverse.

The Baseclaim and Specialbefore were added after special positions were encountered where the evaluation function did not find a solution, while the author did see no reason why a solution was impossible. Detailed analysis then showed the general conditions for these rules. The impact of these two rules is probably not significant. The costs are low also, and therefore this knowledge has been added to the evaluation function.

It is possible to find more rules, when more positions are analysed where VICTOR cannot find a solution yet, while it is known what the result of the game will be after correct play. The last two rules show that this will most of the times result in special rules, which can be applied only in special situations. For this reason no search for extra rules was made.

7. Interaction of Strategic Rules

In Chapter 6 we have described nine strategic rules. Each rule can be used to show that one or more groups cannot be completed by the opponent. In chapter 5 we made clear, that these rules are only useful, if we show that all groups are harmless. Therefore we need to combine rules, which together will ensure that the opponent cannot complete any group. We also have seen before that some rules cannot be combined, if they partly use the same set of squares. Therefore it is important to show under which constraints rules can be applied simultaneously. These conditions are deduced in this chapter.

§ 7.1. Dependence of Zugzwang

Some of the strategic rules depend on Zugzwang. The Claimeven, for instance, forbids the controller of the Zugzwang to play in the column of the Claimeven, if the directly playable square in the column is the lower square of the Claimeven. The number of squares which are forbidden is in this case even, since this lower square is odd-numbered. Therefore this does not influence the situation on the remainder of the board. The Lowinverse forbids the controller of the Zugzwang to play any of the lower squares of the Lowinverse, until the opponent has played one of them. Since in both columns an odd number of squares is forbidden, the total number of forbidden squares due to a Lowinverse is even. Therefore this has no influence on the remainder of the board. This means that, for instance, a Lowinverse on columns a and b, and a Claimeven in column e, do not interfere.

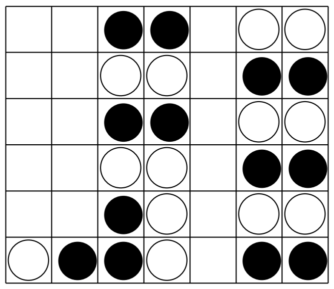


Diagram 7.1.

In diagram 7.1 we have a Lowinverse at a2-a3-b2-b3 and a Claimeven at e1-e2. Since both rules forbid an even number of squares, this has no influence on the Zugzwang which forces White to move. Suppose White plays e1, Black then answers e2, and the whole e-column has come available. This means a total of six squares, which is an even number, and therefore White will eventually be forced to play a2 or b2, showing that the Lowinverse can be used to refute a3-d3. If White first played in the Lowinverse, say a2, Black would have answered a3. Now b2-b3 has turned into a Vertical, which does not depend on Zugzwang. Therefore all squares in both columns have become available, for a total of 10 squares. Since this number is even again, we can still be sure that Black will get e2.

In general it is necessary that as soon the squares of one of the rules are filled in, the number of extra squares available will be even. If this is not the case, Zugzwang changes and a problem occurs.

Now suppose that Black also wants to have square e6 in diagram 7.1 and used a Claimeven on e5-e6 to get it. What would happen as soon as White plays e1? Black answer e2 and the e-column can now be filled up to e4. The total number of squares coming available in the e-column is 4, an even number. Therefore these two Claimevens can be combined, and they both can be combined with the Lowinverse at the a and b column.

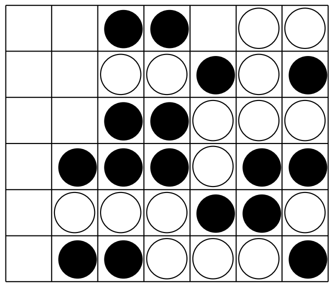


Diagram 7.2.

In the position of diagram 7.2, White is to move. Dangerous groups for Black are a2-d2, for which White only needs a2, and a5-d5, for which White needs both a5 and b5. Now Black wants to solve these problems by using a Claimeven at a2 and a Lowinverse at a4-a5-b4-b5. Now let us suppose that White plays e6. Now it is Blacks turn. He cannot play a1, because of the Claimeven, and he cannot play b4, because of the Lowinverse. So clearly something is wrong with the rules he chose.

How can we detect that these two rules cannot be combined? We have to look at the a-column. If we suppose that at a certain time a1 is played by White, Black will answer a2. Now the a-column comes available up to a3, since a4 is already part of the Lowinverse. The number of new squares is therefore 3, which is odd. This means that filling the Claimeven would turn the Zugzwang. That means, that either the Zugzwang was right when the Claimeven is filled, but wrong for the Lowinverse, or, as we have seen, the Zugzwang was wrong for the Claimeven and would turn right for the Lowinverse. In both cases we lose one of our rules and therefore these two cannot be combined. (Note that Black still can win by playing b4. He should have used a Before instead of a Lowinverse to refute a5-d5).

In general two rules can be combined if the number of squares coming available after one rule is filled in, is even. That way, we are sure that the Zugzwang will not change after a rule is filled. This observation is the key to checking which rules can be combined.

Now suppose we have a Lowinverse at a2-a3-b2-b3 and a Claimeven at a5-a6. Can these two be combined? To see they can, we first note that the first rule filled in is the Lowinverse, since a2 must be played Before a5. As soon as a2 or b2 is played by the opponent and resp. a3 or b3 is answered, the squares of the Lowinverse in both columns come available. In this case a2-a4 and b2-b6, a total of eight squares, which is even. This shows that a Claimeven can be used above a Lowinverse, while a Lowinverse cannot be used above a Claimeven.

§ 7.2. Zugzwang-Independent Rules

Not all nine rules depend on the Zugzwang. The Vertical for instance does not. The only demand of the Vertical is that as soon as the opponent plays the first square, the controller of the Zugzwang should play the other. If at some time during the game the controller wants to play the lower square himself, that is fine, too. Therefore a Vertical can be combined with all other rules, as long as the sets of squares of the different rules are disjoint. If that is the case, one can be sure that the demands of both rules do not interfere. There are even some positions in which it is clear that disjointness of squares is not necessary. Suppose we want to combine a Vertical and a Before, while the squares of the Vertical are equal to one of the Verticals of which the Before consists. In that case both rules have the same demands for the squares and therefore they can be combined without trouble. This is, however, useless. The Vertical which is part of the Before solves all problems which are solved by the Vertical alone, and therefore if the Before is used, it is not necessary to also use the Vertical. For this reason this combination is not allowed, like some other similar combinations.

The Baseinverse does not depend on Zugzwang, either. Like the Vertical it can therefore be combined with every other rule, as long as the sets of squares are disjoint. Although it can be combined with a Baseclaim sometimes, since the sets of squares are not disjoint, this is not interesting, as in the case of the combination of the Vertical and the Before, and therefore not allowed.

This gives us a second observation which is important in order to find out which rules can and which cannot be combined: a Zugzwang independent rule can (at least) be combined with another rule, if the sets of squares are disjoint.

§ 7.3. Combinations of Zugzwang-Dependent Rules

In exactly the same way the Vertical can be combined with the Before sometimes, although they share some squares, the Claimeven can sometimes be combined with an Aftereven, Before or Specialbefore. This again is not useful, since these rules solve all the problems which the Claimeven could solve. Some other combinations can be useful, however. Suppose we have two Afterevens, which share a Claimeven. This combination will generally solve more problems then each of them alone. Since there are no problems in combining them, this is allowed. The same result holds for a combination of a (Special)Before and an Aftereven. If they share a Claimeven, they can be combined. They cannot, if a Vertical of one of them intersects a Claimeven of the other.

This gives rise to the following observation: two rules of the type Aftereven, Before or Specialbefore, can be combined if they are column-wise disjoint or equal. This eliminates the possibility of an intersecting Vertical and Claimeven in one of the columns.

The last interesting combination is the Lowinverse and (Special)Before. If a Claimeven part of the Before intersects with the Lowinverse, or can be found beneath the Lowinverse, we cannot combine them. But what happens if a Vertical of the Before equals the two squares of one of the columns of the Lowinverse? Since the Lowinverse ensures a Vertical in both columns, these two can be combined. For this reason we can combine a Lowinverse and a (Special)Before if they are column wise disjoint or equal and no Claimeven part of the (Special)Before is lower in a column than a Lowinverse part (as was shown in section 1).

§ 7.4. Collection of Possible Combinations

The next table shows which rules can be combined. For each combination a code is given, which denotes a constraint which must hold to allow them to combine. The following abbreviations are used: CL = Claimeven, BI = Baseinverse, VE = Vertical, AE = Aftereven, LI = Lowinverse, HI = Highinverse, BC = Baseclaim, BE = Before, SB = Specialbefore.

	CL	BI	VE	AE	LI	HI	BC	BE	SB
CL	1								
BI	1	1							
VE	1	1	1						
AE	1	1	1	3					
LI	2	1	1	1&2	4				
HI	2	1	1	1&2	4	4			
BC	1	1	1	1	1&2	1&2	1		
BE	1	1	1	3	2&3	1&2	1	3	
SB	1	1	1	3	2&3	1&2	1	3	3

1. Combination allowed if the sets of squares are disjoint.
2. Combination allowed if no Claimeven is below the inverse.
3. Combination allowed if sets of squares are column wise disjoint or equal.
4. Combination allowed if the sets of squares are disjoint, and the set of columns used by the inverses are disjoint or equal.

N.B. (i). If two constraints are given, both constraints must be satisfied.
N.B. (ii). For the Specialbefore, the two special squares are said never to equal the squares of the other rule. This means that they can only be combined with another Before, Claimeven or Aftereven if they share exactly some Verticals or Claimevens.

8. How to Apply the Strategic Rules.

In this chapter the positions are described in which one of the players can try to prove that a certain result can be obtained by correct play. A game is said to be played correctly, if both players make each move (one of) the best move(s). The result of a correctly played game is therefore always equal to the game-theoretical value of Connect-Four.

§ 8.1. Strategic Rules for Black

We have developed a set of rules which can be used to show that certain potential threats can be refuted. Since some of the rules depend on Zugzwang, it is important that the person who applies them, is in control of the Zugzwang.

As we have seen, Black is in control of the Zugzwang, as long as White has not created an odd threat. Now suppose White has created an odd threat, which gives him control of the Zugzwang. What would happen if we still tried to apply the rules? Is it possible that in that case we would get a result from this application of rules which cannot be trusted, or can we be sure that the result we find is correct?

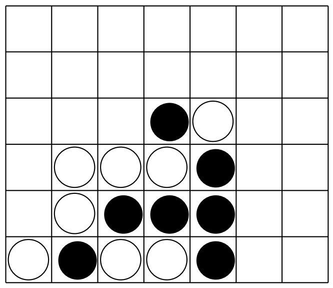
If the threat White has is good, there will generally be no set of rules which can refute that threat. In that case we do not conclude anything for Black. If we can find a way to refute the threat, as well as all other potential threats, we have found that White is not really in control of the Zugzwang.

From this we can conclude that we do not have to check if Black is in control of the Zugzwang, before we try to apply the rules. If Black is not in control, the rules will not yield any result, in which case no harm is done by trying.

The above reasoning shows that we do not have to be able to check in a given position who is in control of the Zugzwang. This may be a difficult task.

§ 8.2. Strategic Rules for White

Since White normally does not control the Zugzwang, he needs an odd threat somewhere to take control. Let us look at a position (diagram 8.1) where he has such a threat, to show what happens.



Black to move

Diagram 8.1.

Black is to move, and White has an odd threat at a3. Therefore the number of squares which will not be filled in, in the a-column is 5. This gives White control of the Zugzwang on the remainder of the board. If he can refute every potential threat using the strategic rules, he can be sure that at some time, Black is forced to play a2, giving White a3. In that way Black will not get any square higher than a2 in the a-column. For this reason, White can use as an extra rule, that no groups which need a square in the a-column, above a2, can be completed by Black.

The following set of rules will solve all White's problems for the position of diagram 8.1:

Claimeven: b5-b6, c5-c6, f1-f2, f3-f4, f5-f6, g3-g4, g5-g6. Baseinverse: d5-e5.

It is easy to see that this way all vertical problems are solved. For the other potential threats, the used solution is given:

c2-f2, d2-g2 : Claimeven f1-f2.
b5-e5, c5-f5, d5-g5 : Baseinverse d5-e5.
b6-e6, c6-f6, d6-g6 : Claimeven d5-d6.
d2-g5 : Claimeven f3-f4.
b6-e3 : Claimeven b5-b6.
c5-f2, d4-g1 : Claimeven f1-f2.
d6-g3 : Claimeven f3-f4.

Now let us suppose that the man at a1 is not played yet. Does this mean that White does not control the Zugzwang yet? To see that this is not the case we look at diagram 8.2.

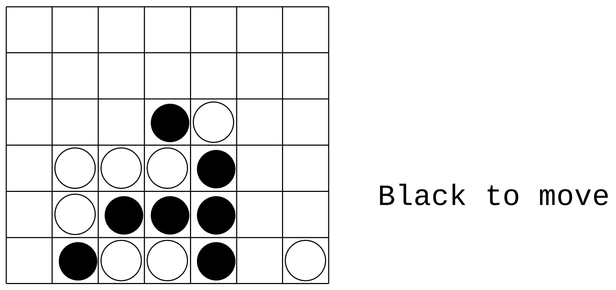


Diagram 8.2.

In this position the set of rules of diagram 8.1 can still be used. If we now look at the squares which can be played before Zugzwang is important we get the following list: a1, b4, c4, d5, d6, e5, e6, g2. This is an even number of squares, which could have been proven also by observing that each strategic rule releases an even number of squares as soon as it is filled in. The threat at a3 takes an odd number of squares, which gives an even number of free squares with Black to move. Now we cannot tell which player is going to take a1, but this is not important. This decision can be made by Black. If Black does not want to move at a1, he does not have to. White then has to fill that square at last, while it is an odd square.

This observation shows, that in the column where White has an odd threat, he normally gets the odd squares up to and including his odd threat. If the playable square is odd, however, Black could decide to take that square. If he takes that square, he can be forced to eventually take the even square above it, giving White all odd squares in the column, except for the playable odd square.

In this case Black cannot use the fact that he can get a1 and a2, and therefore this position is still won for White.

In general, White can use the strategic rules on the remainder of the board, while he gets at least all odd squares in the column of his odd threat, except for the lowest odd square if it is directly playable.

If White has more than one odd threat, he can choose one as he likes, and then let the other odd threat be part of the remainder of the board (which normally allows the opponent to fill in that threat).

§ 8.3. An Example Threat Combination for White

Let us look at the position of diagram 8.3.

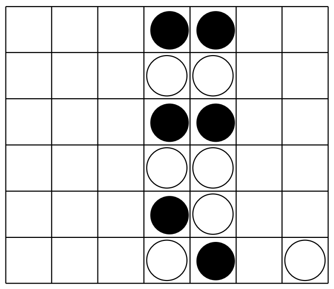


Diagram 8.3.

White has two groups, d1-g4 and d3-g3, which both are filled half. These two groups can together ensure White of getting an odd threat. This can easily be seen, by looking at what happens if White starts filling up the g-column. Even if Black will put in some men himself, White will get at least one of g3 and g4, which gives him an odd threat at f3. If Black has not yet played f2, this will become a serious problem for him. Now suppose Black has played f2 already. In that case White answers directly f3, giving an odd threat at g3 which is serious, if g2 is not yet played. Therefore let us suppose that as soon as White plays g2, Black plays f2. Now White answers f3, and Black must answer g3, giving White the victory at g4.

As the above reasoning shows, White can always get an odd threat, or win. Therefore he can get hold of the Zugzwang and try to solve all problems on the remainder of the board. But what to say about the f- and g-column? We cannot be sure yet where the odd threat will be situated, and therefore we have to take reckon with both variants. One in which the f-columns is filled, and one in which the g-columns is. Which squares can White claim in these columns, in both variants?

First we suppose that White has used the rules on the left part of the board. This might result in situations where White must answer with a move in the left part of the board, if Black plays there. It might also result in a situation where White cannot make any move in that part of the board. We have to consider all these situations.

We let White use the following strategy: He takes odd squares in the f-column whenever he can, and if he is forced to play somewhere in these columns (since he cannot play in the left part of the board) he takes the lowest empty square in the g-column. If Black plays in the g-column, White answers in the g-column. We now look at all different variants:

1. Black plays f1 (diagram 8.4).

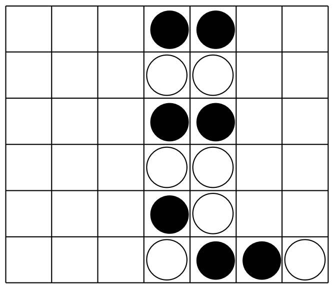


Diagram 8.4.

White cannot play f2, since he wants to get the odd squares of the f-column. He cannot play at the remainder of the board, therefore he plays g2. If Black now plays f2, White answers f3, and wins at g3 or g4. In this case Black gets f1, f2 and f4 or g3, but not both. If Black plays g3 after g2, White plays g4 and the odd threat at f3 is created. Now the g-column can be considered as part of the remainder of the board. Let us suppose that White uses Claimevens for the remainder of the g-column, then he gets g6 as well. In this case Black gets f1, g3 and g5.

2. Black plays g2 (diagram 8.5).

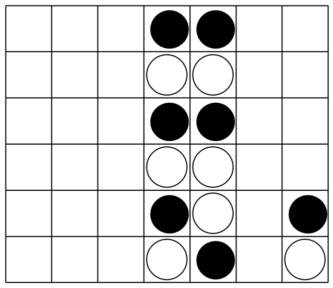


Diagram 8.5.

White now answers g3. He has created an odd threat at f3 and can once again consider the remainder of the g-column as being part of the remainder of the board, and using Claimeven, gets g6. In this case Black can get f1, f2, g2, g4, g5.

3. f1 is played before g1 (diagram 8.6).

In the first two variants, the odd threat was created at f3. It is also possible that the threat is created at g3. To see what happens we look at a variant in which the White man of g1 is replaced by one at f1.

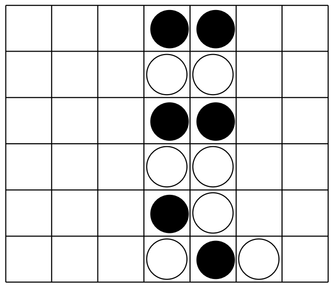


Diagram 8.6.

Suppose Black plays f2, White then answers f3 and an odd threat is created at g3. This threat is however stronger than a normal threat, since White can play g2, if he wants to, since after Black plays g3, White still can win at g4. Therefore White cannot get Zugzwang problems, for which reason he can go on taking the odd squares in the f-column, giving Black only even squares in that column. In this case Black at most gets f2, f4, f6, g1, g3.

Suppose Black plays g1 instead of f2. Then White plays g2, and we get variant 1 again, except for the bottom men.

Conclusion: In all variants, White gets the odd squares of the f-column, except maybe for the lowest, directly playable square. He gets all even squares above g4 (that is only g6). Furthermore, he gets one out of every two squares in the g-column, starting from the first playable square up to and including g4. (This only ensures that Black cannot get a vertical group in the g-column.) Last, but not least, Black cannot complete a group which needs both a square in the range f3-f6 and one in the range g4-g6, since before that time, White will have won.

§ 8.4. Threat Combinations in General

The above reasoning and conclusion shows that even in this particular diagram, it is not easy to see what White at least gets, and which groups cannot be completed by Black. For the interested reader we state in this section which conclusions can be used by White in general if a double threat occurs. These results are not proven here, since they rely entirely on variant analysis as has been done in the former section. Once the diagram is clear, the validity of the conclusions in this section are not difficult to understand.

First we must define in which position we can talk about a threat combination. This definition is not difficult:

A threat combination consists of two groups, which both are filled with two men. One group needs two odd squares, while the second group needs one of the two squares of the first group, and another even square, directly above, or beneath the second odd square of the first group. The square which both groups share should not be directly playable.

As this definition shows, there are two different kinds of threat combinations. One has the even square above the odd square, like in the g-column of diagram 8.3. The other kind has the even square below the odd square, as in the g-column of diagram 8.7 (groups d3-g3 and d5-g2).

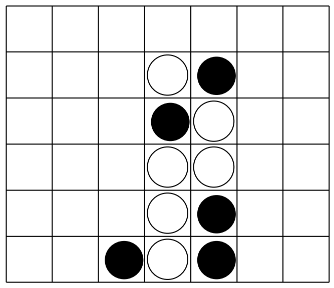


Diagram 8.7.

Since these two kinds are different in the way they can make claims about squares White can occupy, these are dealt with separately.

1. Even square above odd square.

The following claims can be made by White;

1) Black gets no odd squares in the crossing column (except for a first directly playable square).
2) Black will not get both a square above the crossing square and a square above the odd square in the other column. This is caused by the fact that after both the crossing square and the odd square in the other column are played, White must already have won the game. Therefore above at least one of these squares, no moves are made.
3) Black will not get both the square above the crossing square and the odd square in the other column.
4) If the odd square in the other column is playable, the highest square in the crossing column which can be taken by Black, is the square directly above the crossing square.
5) If the first empty square in the crossing column is odd and the odd square in the other column is not directly playable, a Baseinverse can be used on the lowest squares of both columns, giving one of the two squares to White.
6) In the other column each move of the opponent will be answered, giving (at least) one out of every two points to the solving player up to and including the odd square in that column. If rule 5 is applied, this observation starts one square higher.

2. Odd square above even square.

This type of threat combination can (again) be split in two different variants, depending on the fact whether the even square in the other column is directly playable or not. First we look at the rules which hold if the square is not directly playable:

The following claims can be made by White:

1) Black gets no odd squares in the crossing column (except for a first directly playable square).
2) Black will not get both a square above the crossing square and a square above the odd square in the other column. This is caused by the fact that after both the crossing square and the odd square in the other column are played, White must already have won the game. Therefore above at least one of these squares, no moves are made.
3) If the first empty square in the crossing column is odd and the odd square in the other column is not directly playable, a Baseinverse can be used on the lowest squares of both column, giving one of the two squares to White.
4) In the other column each move of the opponent will be answered, giving (at least) one out of every two points to the solving player up to and including the odd square in that column. If rule 3 is

applied, this observation starts one square higher.

If the even square is directly playable, the following claims can be made:
1) Black gets no odd squares in the crossing column (except for a first directly playable square).
2) Black will not get both a square above the crossing square and a square above the odd square in the other column. This is caused by the fact that after both the crossing square and the odd square in the other column are played, White must already have won the game. Therefore above at least one of these squares, no moves are made.

9. Implementation

A program has been written which can be used to determine the value of a position, using the nine strategic rules described earlier. This program has been called ’VICTOR’. This name is chosen since the program will always win playing White (as will be shown in chapter 13) (Incidentally, it is also the authors first name).

§ 9.1. Purpose of VICTOR

In chapters 5 to 8 we have developed a set of strategic rules, of which we have shown how these can be used to determine the game theoretical value of a position. VICTOR is an implementation of these rules, achieving one of the goals of the project: a Shannon C-type strategy program to play Connect-Four.

§ 9.2. Evaluation of a Position

A position in which White is to move, can be evaluated with Black as controller of the Zugzwang, while a position with Black to move can be evaluated with White as controller. In the last case the evaluation should be done on a part of the board, without one (odd threat) or two columns (threat combination). For this reason, the position evaluator gets a description of the board, in which for each square is set if it can be used in the evaluation. All empty squares necessary for one of the nine strategic rules, should lie in the part of the board which is set.

The evaluation starts with finding all possible instances of the nine strategic rules. These rules are tried on every part of the board. For each position where a rule can be applied it is then checked if application of the rule will solve at least one real problem. To do this, all 69 possibilities to connect four men are checked to see if they can still be completed by the opponent.

Three numbers: rows, columns and possible groups are the constants describing the dimensions of a board. All other constants are calculated, starting from these three (such as the number 69 for the standard 7 x 6 board). This way we can use VICTOR for other board sizes as well.

Each application of one of the rules which solves at least one problem is stored. These applications are called solutions. This part of VICTOR results in a list of solutions, where each solution is stored as a struct. The struct consists of fields describing the type of rule (Claimeven, Baseinverse etc.) and the squares involved. Furthermore for each solution we have a list of the groups which are solved by the solution. These lists are at most 69 groups long.

We also make a list for each problem, which contains (pointers to) the solutions can be used to solve the problem. We have calculated an upper bound for the maximum number of different solutions, as function of the number of rows and columns of the board.

After all solutions have been found, it should be checked which solutions can work together, and which cannot. For this reason the solutions are seen as nodes of a undirected graph. Two nodes (solutions) are connected if and only if they cannot be used simultaneously. The connections are stored in a adjacency matrix. In order to fill the adjacency matrix, the type of the solution and the squares involved are important. As soon as the adjacency matrix is filled, it is not important any more to know which squares are used by a solution, or what kind of solution it is. At that time we are only interested in which problems can be solved by a given solution and with which other solutions it can work together. The adjacency matrix is a normal square array.

The last part of the evaluation consists of finding a set of nodes (solutions) which can be used to solve all problems, while no two solutions are connected. If we see the problems as nodes, too, and we connect a solution and a problem if the solution solves the problem, and no problems are connected, we can solve it as a pure graph problem:

Given are two sets of nodes, S(olutions) and P(roblems). We try to find an allowable (in graph theory: independent) subset C(hosen) of S, with the property that P is contained in B(C) (the set of all neighbours of nodes in C).

This problem can be solved using a simple backtracking algorithm. The following recursive function illustrates the working of the algorithm.

```
FindChosenSet(P, S)
{
    if (P == EmptySet) {
        Eureka() /* We have found a subset C which meets all constraints. */
    } else {
        MostDifficultNode = NodeWithLeastNumberOfNeighbours(P);
        for (all neighbours of MostDifficultNode) {
            FindChosenSet(P - { MostDifficultNode },
                          S - AllNeighboursOf(ChosenNeighbour));
        }
    }
}
```

The recursive function gets as parameters a set of problem nodes P which have to be solved, and a set of solutions which still can be used. If the set of problems is empty, all problems are solved, and we can stop (indicated by Eureka()). Otherwise we find the most difficult problem to solve, which is chosen to be the problem which has the least number of solutions which can be used to solve it. For this problem we try all of his neighbours, which are solutions, to solve the problem. Trying a solution consists of marking the problem as solved, and making sure that all solutions which are connected to the chosen solution cannot be used to solve an other problem. This is done by passing the appropriate sets to the next call of the function. If a problem has no solutions any more, or none of the solutions work, the function ends without success, causing it to backtrack.

If a set of solutions is found for a given position, these solutions show the plan which has to be followed to play the game until the desired result (win for White, or at least a draw for Black) is reached.

With a minor modification it is possible to sometimes detect a win for Black instead of at least a draw. If we find a set of solutions of which one is an Aftereven, we can be sure that Black will win the game. With this modification, the same search procedures can be used to do this. Currently, this modification is not implemented.

§ 9.3. Bottleneck of Algorithm

The recursive search for the independent set is potentially NP-complete. Given the fact that the number of solutions on the 7 x 6 board can be as much as several hundreds, one could expect that the recursive search would be the bottleneck of VICTOR during execution. This is not the case. The amount of time spent to find the solutions is in general responsible for over 80% of the used CPU-time, including the building of the adjacency matrix, which is rather expensive. The recursive search takes normally less than 10% of the CPU-time. This is probably due to the fact that in most positions there are some groups which permit only one or two solutions. After chosing these, the possible number of solutions for the other problems decreases, giving not much possibilities to backtrack. This fact is illustrated best by looking at the test results which show that the evaluation of a position on the standard board never takes more than a few CPU seconds.

§ 9.4. Some Data on the Program

VICTOR was written in the C programming language. The source code consists of about 5000 lines of C-code and an additional 1000 lines of comment. VICTOR consists of several modules. For each strategic rule, one module is made. The way these modules interact makes it easy to add new strategic rules, with only minor modifications in existing code. This is easily understood, by looking at section 9.2. Only the first part of the algorithm uses information about the different kinds of solutions. As soon as the adjacency matrix is build, these differences are no longer used. For a new solution type, we first need to find all instances which can be used to solve problems. This part is completely independent of the other solution types. Next the adjacency matrix must be build. Here information is needed to detect if two solutions can be used simultaneously. That part of VICTOR uses a table, as described in chapter 7. Modification of the table just consists of adding another row of possibilities, and implementing possible new conditions which must be checked. This has be done several times since the first evaluation function was written. Section 6.10 describes the way the nine rules where found, implying that the evaluation function has changed through time.

The evaluation part of VICTOR uses an adjacency matrix to keep track of the connected nodes of the graph. Due to this large matrix, the evaluation part uses about 2 Mb at run time.

The time needed to evaluate a position depends roughly on the number of men in the position. The larger the number of men, the shorter it takes to evaluate a position. Evaluation time lies somewhere between 0.01 and 10 seconds.

10. First Results.

Using VICTOR, consisting solely of the evaluation function (and some nice I/O routines), tests were made. The results of these tests are presented in this chapter.

§ 10.1. Small Boards

The evaluation function was also used for boards smaller than the standard 7 x 6 board. All boards used had an even number of rows, since rules depending on Zugzwang make use of that fact.

It is not difficult to check that all rules derived in chapters 5, 6, 7 and 8 are independent of the board size. The only important aspect of the board size is the fact that Zugzwang should not change after filling up a column. Therefore the number of rows must be even.

For six boards, 4 x 4, 4 x 6, 5 x 4, 5 x 6, 6 x 4 and 6 x 6, VICTOR was used to find a good answer for Black after the first move of White. In all six positions the program found a good (i.e., leading to a draw) answer for Black against every first move of White. After VICTOR had found similar results on a 6 x 8 board, we looked into the reason why VICTOR was sure about getting at least a draw. This resulted in finding the strategic rules which can be used on all boards of 6 columns or less and an even number of rows (see chapter 2, section 3).

As an example, we give the set of rules VICTOR found for an empty 6 x 6 board:

Before	a1 a2 b1 b2 c2 c3 d2 d3
Before	a3 a4 b3 b4 c4 c5 d4 d5
Before	b1 b2 c2 c3 d2 d3 e1 e2
Before	b3 b4 c4 c5 d4 d5 e3 e4
Before	c2 c3 d2 d3 e1 e2 f1 f2
Before	c4 c5 d4 d5 e3 e4 f3 f4
Baseinverse	c1 d1
Claimeven	a5 a6
Claimeven	b5 b6
Claimeven	e5 e6
Claimeven	f5 f6

If we look at the Claimevens which are part of the Befores, we see that Claimevens are used at: a1-a2, a3-a4, a5-a6, b1-b2, b3-b4, b5-b6, e1-e2, e3-e4, e5-e6, f1-f2, f3-f4 and f5-f6. These are equivalent with the rule which says that in the columns a, b, e and f, Black should play follow-up.

The Baseinverse at c1-d1 shows the fact that after the first answer in the middle column, the other middle column should be played. The verticals c2-c3, c4-c5, d2-d3 and d4-d5 show that after the Baseinverse is played, Black has to answer each move in a middle column in that same column.

It should be clear that, given enough time, VICTOR would have found a similar set of rules for any 6 x (2n) board.

§ 10.2. The 7 x 4 Board

The strategy which has been described in chapter 2 does not work on a board with more than six columns. On these boards it is not possible for Black to refute all horizontal combinations in a row, with only one move, as could be done on a board of 6 columns or less. If, however, White does not play in the middle column with his first move, Black has the opportunity to refute all horizontal groups in the first row with only one move. Therefore it is not very surprising, that if White does not plays his first man in the middle column on a 7 x 4 board, Black has an easy path to at least a draw. Since the strategy Black can use in that case will be explained for the 7 x 6 board, we will omit it here.

What happens if White plays 1. d1, .. ? In that case VICTOR could not find a strategy to give Black at least a draw.

At this time we had to start suggesting a best move for Black and see if after all answers of White, Black could find a drawing strategy.

This approach can be used repeatedly: For all positions VICTOR cannot solve, we suggest a best move for Black, and see if all positions after answers of White, can be solved. In this way it was possible to show that Black can get at least a draw on a 7 x 4 board, in all variants, evaluating only about thirty positions.

§ 10.3. The Standard 7 x 6 Board

Since the search tree which was build manually for the 7 x 4 board was rather small, the same approach was tried for the 7 x 6 board. Like on the 7 x 4 board, the only move White can make to have a chance to win the game, is 1. d1. All other moves give Black the possibility to use a simple strategy to achieve an easy draw. The strategic rules used by VICTOR after 1. a1 or b1 or c1 to show that Black can reach a draw are given in appendix B. It is easy to see that the solutions found can be extended to all boards of the form 7 x (2n), for n a natural number.

Since the search tree became very large after 1. d1, a manual approach could not be successful. For this reason this approach was automated. The techniques used are described in the next chapters, as well as the results found in this way.

11. Automated Analysis.

As has been stated in chapter 10, the position after 1. d1 on the standard 7 x 6 board was not yet clear. The search tree starting from this position was also too large to search manually. For this reason an automated search needed to be used, which could try to show that starting from a given position, Black could always draw the game, or White could always win. Using such an automated search, several unclear positions could be tested, at the end (hopefully) showing what the value of the game would be after correct play. The used search methods are described in this chapter.

§ 11.1. The Oracle

Let us suppose we have access to an oracle. This oracle is able to tell us in a given position, which move is the best that can be made. Unfortunately, the oracle is not able to tell us what the result will be, if both players play correctly, starting from that position. How can we use the information provided by the oracle?

We can start by giving the initial position to the oracle and ask for the best move. After playing that move, we reach a new position which is also offered to the oracle. Again we play the move suggested by the oracle. We repeat this until the game has ended. Let us suppose that White has won the game. Does this result imply anything? To show it does we suppose that Black does not necessarily have to loose the game. If that is the case, he should have somewhere played another move then he played in the game guided by the oracle. However, that would mean that somewhere the oracle did not suggest the best move, which cannot be true. This shows, that the result of only one game, provided that both players made the best move all the time, is enough to determine what the result will be after correct play.

Unfortunately, we do not have access to a perfect oracle. Let us therefore suppose we only can use an oracle which can tell us which moves are best for White, but sometimes fails to tell us what Black should do. We again play a game guided by the oracle, and end up with White winning. Does this result imply anything? We now are not sure Black played the best moves all the time. Therefore, in all positions where Black had to move, Black still can try some other moves. He might find a path which leads to a draw, or even better, a win. It is not useful for White to try other moves, as long as the moves given, lead to a win. Only if Black forces to go in a new variant, White needs to find the best move again. Fortunately, the oracle will tell him which move to choose.

As long as White keeps winning, it is enough to try all variants for Black if Black is to move, and only try the best move for White. White does not need to change, since he is perfectly happy with the result. If Black would find a path to a draw, which makes Black happy, White is the one who has to try all variants, while Black can stick to only one, as long as that results in a draw.

Therefore it is very valuable to know what the result will be starting from a given position, even if we have to confirm it by trying all variants. We then know for which player we have to try all different moves, and for which player we only have to try the best move. This eases our task considerably.

§ 11.2. Conspiracy-Number Search

It has been shown in the former sections, that large amounts of work can be saved, if we grow only the useful parts of a minimax tree. A simple n-ply tree is therefore of no use. Recently, McAllester suggested a new search method in minimax trees. This new method is called conspiracy-number search.

In this section we will consider only the simple form of conspiracy-number search which has been used for our Connect-Four program. More general descriptions of conspiracy-number search can be found in [7] and [8].

In the next sections we will use different values for the possible results of the evaluation function. -1 is used to mean that Black can at least draw the game, 0 means that a position is still unclear, while the value 1 indicates that White can win the game. Now White wants to get a maximal result (1), while Black tries to get a minimal result (-1). Therefore nodes in the search tree where White is to move are called max-nodes. Nodes where Black is to move are called min-nodes. Note, that for a max-node, it is enough to find one son which has value 1, in order to change the value of the max-node itself into one. To change the value of a max-node into -1, all sons have to have the value -1. For the min-nodes, the opposite is true.

In our search tree a node can have three different values. Two of these, -1 and 1, are values which cannot be changed. As soon as a node has one of these values, we know it is useless to evaluate sons of the node, since it can never change value. If a node has value 0, we can change its value, by evaluating its sons. As we have seen before, it is easier to change the value of a max-node into 1 then into -1. For the first change, we only need to find one son which has the value 1, while for the value -1, we need to change the values of all his sons into -1. One could say that all his sons must conspire in order to change its value into -1. Therefore the conspiracy number for -1 of a max-node is equal to his number of sons if no son has been evaluated yet. As soon as one one or more of his sons has been evaluated and has the value -1, the conspiracy number decreases.

The conspiracy number for the value 1 for a max-node is 1, as long as no son is evaluated with value 1, since we only need to find one son, whose value is 1, to change the fathers value to 1.

In general we have for each node two conspiracy numbers associated with it. One conspiracy number for reaching the value -1 and one conspiracy number for the value 1. These conspiracy numbers are equal to the least number of nodes in the subtree with this node as root, which have to conspire in order to change the value of the father to the given value. As soon as one of the conspiracy numbers becomes 0, the value associated with that conspiracy number becomes the value of that node. At that time the conspiracy number for the other value should have become infinite, since there is no finite number of nodes which could by conspiring change the value of the node.

If we evaluate a new node we have to find the correct conspiracy values. These are very easy to find:

Evaluation returns -1: Conspiracy(-1) = 0, Conspiracy(1) = infinite.
Evaluation returns 0 (Min-node): Conspiracy(-1) = 1, Conspiracy(1) = Number of Sons.
Evaluation returns 0 (Max-node): Conspiracy(-1) = Number of Sons, Conspiracy(1) = 1.
Evaluation returns 1: Conspiracy(-1) = infinite, Conspiracy(1) = 0.

§ 11.3. How to use Conspiracy-Number Search

The only thing we have done so far, is adding some information to each node. Now we should make use of the conspiracy numbers. To see how we can use it, we look at the meaning of the conspiracy numbers. If a conspiracy number is high, this means that we need to evaluate a large number of nodes,
in order to have a chance that the value of the root will change into that value. It is also possible, that while evaluating some of these nodes, we find that we have to evaluate even more nodes, since these nodes and their sons are still unclear.

If a conspiracy number is low, we have a chance that we need only evaluate a few nodes.

We now use the following reasoning: suppose we can find the value of the root by evaluating just a few nodes, it would be a waste to first evaluate many other nodes. If, on the other hand, we need to evaluate a large number of nodes, not much harm is done, if we first evaluate a few others.

This reasoning says that if we have a chance to finish work soon, we should try that. Even if we discover that it does not work, the amount of extra work proportional to the work we have to do afterwards is not much. If, however, we are lucky, we have saved a lot of work.

Applying this to conspiracy numbers, we first look at the conspiracy numbers for both values and decide which result seems easier to achieve. After we have chosen the value which we try to achieve, we look at all sons and pick the one which has the least conspiracy number for that value. We are first going to try to change the value of that node. To do this, we pick the son of his, which has the least conspiracy number, for the chosen value. If this node is still unclear, we pick the son of his with least conspiracy number for the chosen value, etc. Eventually, we reach a node which has no sons yet. This node we expand, and we evaluate all his sons. Using the conspiracy numbers of these sons, we adjust the conspiracy number of the node we just expanded. This might change the conspiracy number of his father, his fathers father, etc. After we have changed the conspiracy numbers of all his ancestors, we start again at the root, repeating the whole procedure. Hopefully, soon the value of the root is changed into -1 or 1, which concludes our search.

§ 11.4. Performance of Conspiracy-Number Search

Although a search tree which is grown using conspiracy-number search, is not a simple n-ply tree, it is very well possible that much more than only the best variant for one player is tried. Let us see what happens in an undesirable position. We assume that the strategic rules are powerful enough to determine the value of any position of at least 10 men, while it cannot find any conclusions for positions with 9 men or less. Given this information, we can easily see what happens to the tree grown by the conspiracy-number search. Since all variants are equally bad, attention will be divided between all variants. This will result in a tree which is roughly equally deep in all variants. In this way we will probably not be able to find any results, since there are a large number of positions containing 9 men or less.

This shows that although conspiracy numbers tend to try promising variants sooner than others, they sometimes cannot distinguish between good and bad variants and will therefore grow a tree which is too wide.

This theoretical observation was confirmed by practical results. First we look at an example which worked fine.

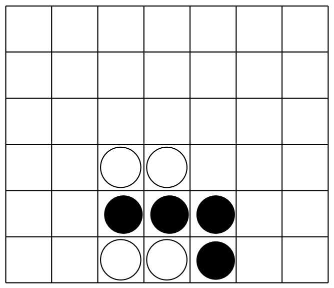


Diagram 11.1.

The position of diagram 11.1 can be won by White, by playing the following variant:

5. a1, b1
6. b2, d4
7. b3, e3
8. e4, ..

The position after the 8th move of White can be shown to be a win for White. All moves made by Black are forced, and therefore the conspiracy number for the value 1 is at all nodes in this variants equal to 1. Therefore this variant is expanded immediately, before a lot of time is spent in other variants where Black has a lot of choice.

In other positions the conspiracy-number search gets worse results.

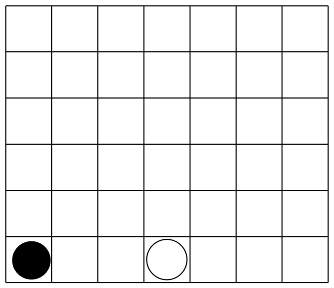


Diagram 11.2.

In the position of diagram 11.2, Black made a very bad first move. Therefore many positions which can be reached from this position can be won by White. Since we can only evaluate positions for White in which White has a real odd threat or a threat combination, most positions will be unclear. From this position therefore a tree is build in which most variants are grown simultaneously. Even after a tree of 10,000 positions is built, no result has been found.

Therefore conspiracy-number search alone, does not guarantee us that the main variants are searched first. The next section shows how we can combine conspiracy-number search and depth first search.

§ 11.5. Conspiracy Numbers as Oracle

Suppose, as in diagram 11.2, that a tree of 10,000 nodes is built and no result has been found for the root yet. In that case all seven sons of the root have conspiracy numbers set. Looking at these conspiracy numbers we can try to get an impression about which variant is the best in order to achieve a win for White, since White is to move in the root position.

There are different ways to choose a son as ’best son’. We could take the node with the lowest conspiracy number for White. We could also take the son with least quotient WhiteConspiracyNumber/BlackConspiracyNumber. An argument in favour of the last criterion is that it is very well possible that the number of nodes in the subtrees are far apart. Therefore it is possible that the conspiracy numbers for one son are 10 and 15 respectively, and for the other 11 and 200. The conspiracy numbers for White are almost equal, but for Black they are far apart. It is clear that the chance that the first son eventually appears to be at least a draw for Black is much higher than the chance that the second variant will be a draw for Black. For that reason we choose the second son as best variant. In general we use the above described quotient to chose a best son.

After a son is chosen, we first determine the value of that son. This is done in exactly the same way as for the initial position: first we build a search tree using conspiracy-number search. If this does not yield a result, we chose the best variant for the player whose turn it is, and start again with a new position. At last we will have determined the value of this son. At that time we go back to his father and evaluate that position again, using the knowledge of this son we have evaluated. If, for instance, this son is won by White, and White was to move in the father position, it is clear that the father position can be won by White. If however, the son can be drawn by Black, we have to try other variants to see if we may have chosen a wrong variant. In this way we will normally only try alternatives for the player which cannot get the result he wants to achieve. Only if we first try a bad variant before the best variant, we do too much work.

The choice for the variant is based on a large search tree, and therefore we can hope that we will most of the time choose the right variant.

§ 11.6. Three-Level Search

The here described search method now consists of three levels to evaluate a position:

The first level consists of the knowledge-based search. A position is evaluated using strategic rules only. The result of this is -1 (at least a draw for Black) 0 (unclear) or 1 (win for White).

The second level consists of the conspiracy-number search. A search tree is built, using conspiracy numbers. Each node is evaluated using the first level search method. This can result in a value for the root (-1 or 1), or the fact that the position is still unclear, even after a large tree has been build (10,000 positions for instance). If that is the case, the conspiracy numbers of the sons of the root are used to determine which son is the best variant.

The third level consists of the Depth First Search. Using the second level search method, each node is evaluated. If the result of the node is -1 or 1, we return to the father of the node. If the result of the node is 0, we use the information about the sons to determine which son to try first.

12. Implementation of Conspiracy-Number Search and Search Tables.

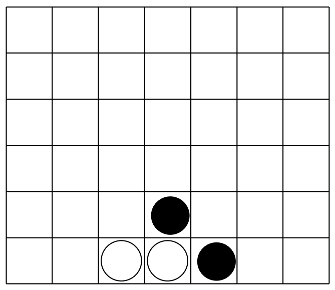
In this chapter the implementation of the conspiracy number search in our Connect-Four program is explained.

§ 12.1. Search Tree in Memory during Search

Conspiracy number search chooses a node to expand, expands it and evaluates its sons. After this has been done another node is chosen, expanded, etc. The nodes which are chosen after each other can be part of different subtrees, and for this reason, we need to have all subtrees in memory, during the search. For most chess programs this is a problem, since the number of positions evaluated during a search is so large, that it is impossible to keep the whole search tree in memory. In the case of VICTOR, we do grow search trees in the order of 10,000 positions, which makes it easily possible to keep the entire tree in memory.

§ 12.2. Transpositions

A node of the search tree is equivalent to a position, while the connections between fathers and sons in the tree determine the way the position was reached. While playing Connect-Four, it is very well possible to reach the same position in several different ways. If we look at the position of diagram 12.1, it is not possible to tell, which of the three given variants was played to reach the position.



White to move

Diagram 12.1.

1. d1, d2
2. c1, e1

or

1. d1, e1
2. c1, d2

or

1. c1, e1
2. d1, d2

Now suppose we make an analysis of the initial position. It is then very well possible that the position of diagram 12.1 will be part of the tree in three different variants. As soon as we have evaluated the position in one of the variants, we do not need to evaluate the position in the two other variants any more, since the value is independent of the way the position is reached.

Therefore we should store the positions in a way which makes it easy to determine how many times the position is son of another father. This way we can prevent to do extra work.

The way we solved this problem is by giving a node potentially more than one father. When we evaluate a node, we first try to find it in the set of nodes we already have. If the node is found there, another father is added to the list of fathers for this node. It is not evaluated, since that has already been done. If the node was not found, a new node is created with only one father, and the node is evaluated.

When the conspiracy numbers of a node are updated, we normally had to pass the information to his father. This time we have to pass it to all his fathers.

The structure we now have is no longer a tree. It is a directed acyclic graph, but we can still look at it, as if it were a tree. The only difference is the updating of all fathers, instead of one.

Note that since we update all fathers if a node's conspiracy values change, the node is actually seen as some copies of the same node. Therefore the node is counted as many times as it is part of a variant which leads to that node. It is therefore possible that both conspiracy numbers of the root are larger than the number of leaves which need to change value, since some of the leaves occur in more than one variant. Since this undesired effect does not change the nature of the conspiracy number search, and can happen in all variants - upgrading on the average all conspiracy numbers a little - it is not removed.

§ 12.3. Combination of Conspiracy-Number Search and Depth-First Search

As has been stated in chapter 11, the result of building a search tree with conspiracy numbers is either the value of the root, or a guide to choose what the best move is from the root position. In the latter case, we build a new search tree from the best son of the former root. This tree will contain a large number of positions which we have evaluated before. It would therefore be a waste of CPU time, to evaluate these again. For that reason search tables are build, which contain all positions for which we have found the correct value so far.

§ 12.4. Search Tables

A search table is a set of positions which have been evaluated. The position and the result are stored in the table. These search tables can be used to check if a position which has to be evaluated already has been evaluated. If the position is found in the search table, the result which is stored there is used, instead of the value obtained by evaluating the position again.

We have changed VICTOR to work with a search table. Each time a search tree is built, for each position it is checked if it is part of the search table. After the new search tree has been built, no matter if the value of the root has been determined, each position in the search tree of which the value is known to be -1 or 1, is stored in the search table, unless it was already there.

Diagram 12.2 shows the advantages of this approach:

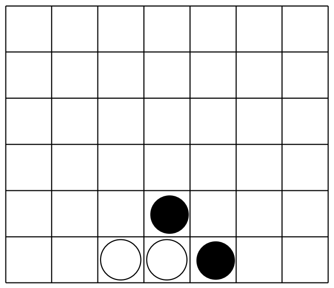


Diagram 12.2.

The position of diagram 12.2, reached after 1. d1, d2. 2. c1, e1, has been shown to be at least a draw for Black, after a long analysis. All positions evaluated during this analysis are stored in the search table, including this position itself, since its value is known.

Now suppose we are going to check the position of diagram 12.3:

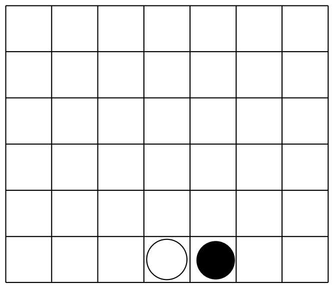


Diagram 12.3.

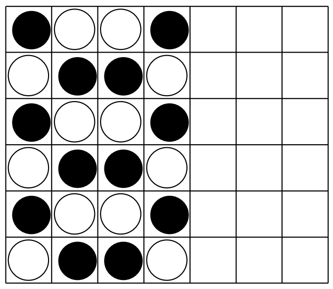
Using the search table, it is immediately found that 2. c1 is not a good move for White, since the answer 2 .., d2 results in a position which is at least drawn for Black. If this position would not have been part of the search table, the whole analysis which has been done for the position of diagram 12.2, had to be repeated.

One could argue that even those positions which are still unclear, should be part of the search table. We have chosen not to do this, since the only thing we gain for such a position is the time needed to evaluate that position. Positions of which we have determined the correct value, may have been solved only after a large search tree has been build starting from that position. Putting such a position in the search table not only saves the time to evaluate that position, but also the time needed to rebuild that whole search tree.

The possibility to find positions in the search tables which are the root of a large subtree, of which the game-theoretical value has already been found, is the main reason to implement the search tables.

§ 12.5. Generalised Positions

If a large part of the board is filled with men, some men are surrounded by men of the other colour. In such a position it is possible that some of the men cannot become part of a winning group any more. To illustrate this, we look at diagram 12.4.



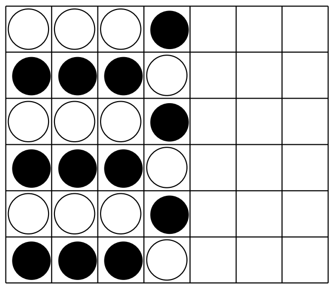
White to move

Diagram 12.4.

If we look at the white man at a1, it is clear that that man cannot become part of winning group any more. Therefore we can say that the fact that that man is White is not relevant in this position. This does not mean that it could as well have been Black. We only claim that the fact that the man is white is not important for any new potential threats. Still it is important that the man has been played there, otherwise the other men in the a-column could not have been played.

We call all men which cannot become part any more of a winning group, in a given position *indifferent*. In the position of diagram 12.4. all men in the a-column and b-column are indifferent, while all the men in the c-column and d-column are not.

The reason why we defined indifferent men will become clear, after we have looked at the position of diagram 12.5.



White to move

Diagram 12.5.

The only difference between the positions of diagrams 12.4 and 12.5 is the order in which the a-column has been filled. Since the rest of the game will be played at the other side of the board, we wonder if the game-theoretical values of both positions are equal.

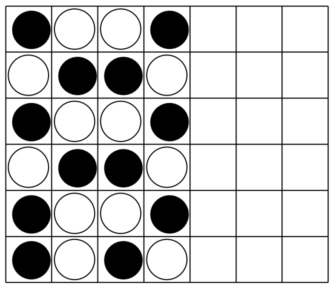
To see, that they must be equal, we look at the set of indifferent men of position 12.5. Just like in the position of diagram 12.4 all men in the a-column and b-column are indifferent, while the others are not.

From the fact that the set of indifferent men of the two positions are equal and all other men are equal, we conclude that the game-theoretical value of the positions must also be equal. To see this, we suppose that play has proceeded in both positions in exactly the same way. Now suppose that one of the players wins as play proceeds in one of the positions. Then the winning group cannot contain any indifferent men, and therefore the same player will have won at the same time in the other position.

In general: since any winning group of four men cannot contain indifferent men, a game which leads to a win starting from one position leads also to a win in the other. In other words, the game-theoretical value of two positions which have exactly the same set of indifferent men and contain equal men at all other squares, must be equal.

This observation can be used in the search tables. Instead of putting a normal position in the search table, we have three kind of men: White, Black and Indifferent. Every time we want to check if some position is present in the search table, we first change the position into a position with indifferent men, and check if that position is part of the search table. This way we can easily use the conclusions about, for instance, the position of diagram 12.4, to find the game-theoretical value of the position of diagram 12.5.

One should note that the fact that, for instance, a White man is indifferent, does not imply, that it could as well have been a Black man. To see this, we look at the position of diagram 12.6.



White to move

Diagram 12.6.

The only difference between the positions of diagram 12.4 and 12.6, are the men at a1 and b1. In the position of diagram 12.4 these men where indifferent. Now we have changed the colours of these men, the man at b1 is no longer indifferent, since it can be part of the group b1-e4. The values of the positions 12.4 and 12.6 are not necessarily the same, since their sets of indifferent men are not equal.

This example shows that an indifferent man cannot be replaced by a man of the other colour. The only thing we know about an indifferent man, is the fact that its current colour is not useful during the rest of the game.

§ 12.6. Symmetry

It should be clear that when the position of diagram 12.7 is evaluated, the result is equal to the result of diagram 12.2.

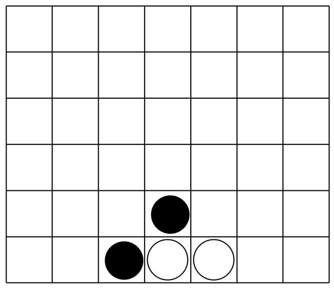


Diagram 12.7.

These two positions are each others mirror image after reflection in the middle column of the board. It is easy to see that positions which are each others mirror image have the same result:

If we look at the standard situation where two human players play against each other, they sit at opposite sides of a vertical framework. They look at different sides of the men which are thrown into the framework. The result of this is, that if one player sees the position of diagram 12.7 the other player sees the position of diagram 12.2!

Since we now know that two positions which are each others mirror image are equivalent, in VICTOR an ordering is defined between positions. For every position given, it is compared using that ordering, with its mirror image. The smallest position according to the ordering is then chosen. In this way, we can be sure we are not working with mirror images of positions we have already evaluated.

The result of this is, that the initial position has only four different sons, since the moves e1, f1 and g1 result in mirror images of the first three moves. Due to the fact that we list all possible fathers of a given son, the initial position gets seven sons in our implementation, of which three have two fathers, which are equivalent.

Although one could argue that this is not the best representation possible, it has some nice features: the concept of search tree is not changed where it was not necessary. If in a given position 7 different moves can be made, we are sure we can find a son where we look for him. We do not have to check if symmetry arises. Symmetry is thus only dealt with at the time a position is evaluated. Afterwards, we do not have to concern ourselves with it.

13. Results for the Standard 7 x 6 Board

In previous chapters various aspects of VICTOR where described. VICTOR has been used for an exhaustive search of the 7 x 6 board, in the hope to find the game-theoretical value for the initial position. As will be shown in this chapter, it was found that White can always win, if and only if he starts playing 1. d1.

§ 13.1. Initial Moves

As has been stated in chapter 11, Black has an easy strategy which gives him at least a draw, which works in all variants of an 7 x (2n) board, provided the fact that White does not start at d1. This strategy is described in appendix B. The only remaining first move for White which gives him a win, is 1. d1. To this move Black has four different answers. All of these have been checked.

§ 13.2. 1. d1, d2

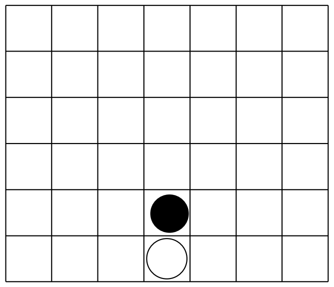


Diagram 13.1.

The position of diagram 13.1 can be won by White. It was easy to show that all other moves than 2. d3 will give Black at least a draw. After 2. d3 we get the position of diagram 13.2.

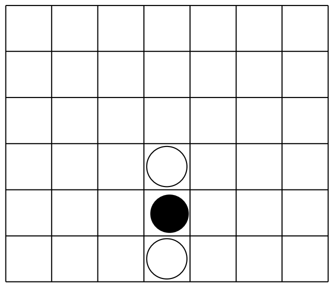


Diagram 13.2.

It did not take much effort to show that each move at the bottom row for Black loses in the position of diagram 13.2. This is mostly due to the fact that White can take the field d4, while he already has d3. Therefore 2. .., d4 is Blacks last chance. That leads to the position of diagram 13.3.

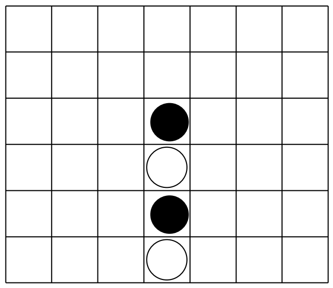


Diagram 13.3.

In the position of diagram 13.3 it was easily shown that all moves at the bottom row lose for White. Therefore White's third move is forced: 3. d5, ... This leads to the position of Diagram 13.4.

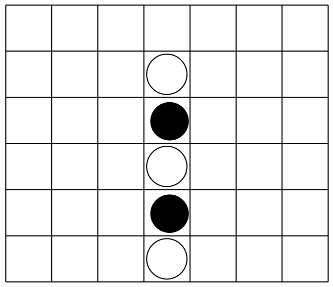


Diagram 13.4.

In the position of Diagram 13.4, it did not take much effort to show that three of the four possible moves for Black lose. The only move which remained unclear for some time was 3 .., b1. This resulted in the position of Diagram 13.5.

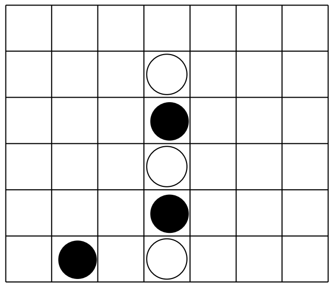


Diagram 13.5.

This time not all White moves were checked. Best move to play seemed to be 4. b2. This position is shown in diagram 13.6. As we will see, 4. b2 will lead White to a win. Therefore it is not necessary to look at other possibilities for White.

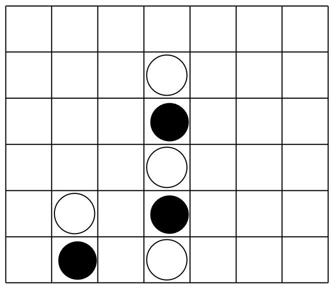


Diagram 13.6.

The position of diagram 13.6 was the most difficult position of the whole variation 1 d1, d2. After a long search, VICTOR showed that White can win from this position. This conclusion showed that 1 d1, d2 leads to a win for White.

§ 13.3. 1. d1, c1

The position of diagram 13.7 results after 1. d1, c1.

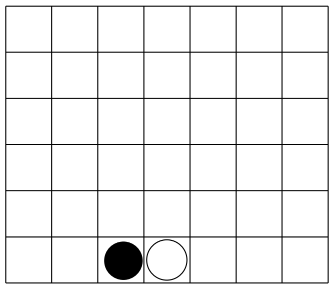


Diagram 13.7.

It was easy to show that White now only has three possibilities to win the game: 2. c2 or 2. f1 or 2. g1. Although VICTOR’s conspiracy numbers indicated that 2. f1 should be tried first, the author thought that 2. c2 was a better choice. After a long search it was shown that 2. c2 gives Black the possibility to at least draw the game. To reach that result, Black must play 2 .., c3, otherwise White still can win. The position after 2 .., c3 is shown in position 13.8.

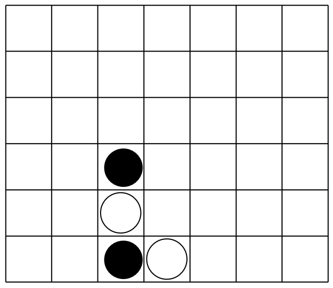


Diagram 13.8.

It was easy to show that most moves in position 13.8 for White do not lead to a win. The only move which seemed promising was 3. c4. This position is show in diagram 13.9

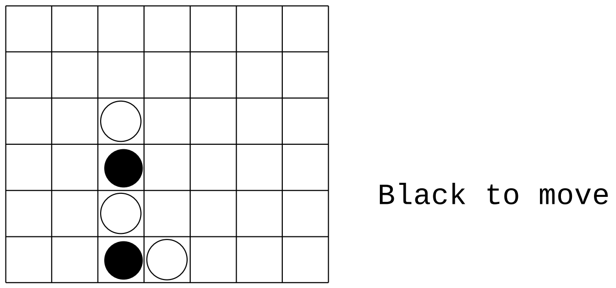


Diagram 13.9.

In the position of diagram 13.9 most moves were easily shown to be lost for Black. The only moves which were not clear yet, were 3 .., d2 and 3 .., g1. Unfortunately, it was shown that after 3 .., g1 there was no move which will give White a win. From this conclusion it followed that the move 2 c2 leads to at least a draw for Black. Therefore an other second move for White had to be tried. VICTOR had already suggested 2 f1 as a best move and this time the author listened to that suggestion. The position is shown in diagram 13.10.

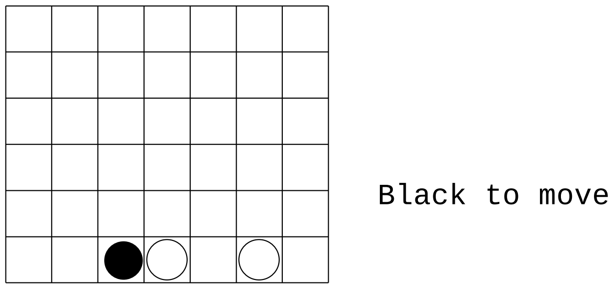


Diagram 13.10.

In the position of diagram 13.10, most moves were easily shown to give White a win. Only 2 .., g1 gave some problems. That position is shown in diagram 13.11.

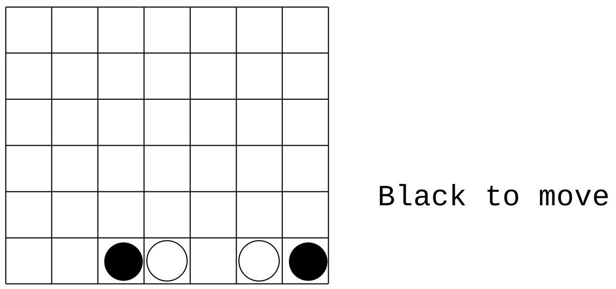


Diagram 13.11.

It was suggested by VICTOR that in the position of diagram 13.11 the best move for White would be 3 f2. It turned out that this move indeed leads to a win for White. For that reason no other White moves were checked. All possible moves for Black were easily refuted, of which only 3 .., f3 took some more time than the others. That position can be found in diagram 13.12.

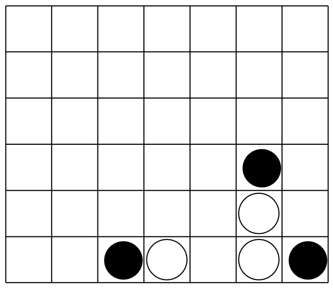


Diagram 13.12.

In the position of diagram 13.12 VICTOR suggested 4. d2 as best move. After this move VICTOR did not have much trouble to show that White can win, independent of Blacks next move.

The fact that 2 f1 leads to a win, while the seemingly most promising move, 2 c2 does not, illustrates the fact that human intuition is not good enough to really understand the game.

§ 13.4. 1. d1, b1

This variation was remarkably easy to solve. White's strategy consists of using his domination at the right wing of the board. The only move for White checked was 2 f1, which was strong enough to win the game. The position after 2 f1 is shown in diagram 13.13.

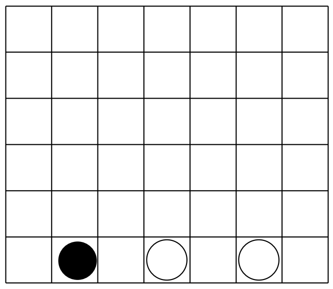


Diagram 13.13.

In the position of diagram 13.13, White threats to play e1, creating two threats at c1 and g1. For that reason Black only has three possible moves: c1, e1 or g1. After 2 .., c1 White can play 3 e1, forcing Black to answer 3 .., g1. The position after these moves is shown in diagram 13.14.

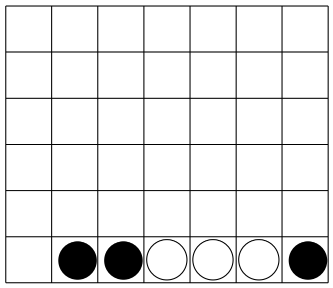


Diagram 13.14.

The position of diagram 13.14 had already been checked and shown to be won by White, after the variation: 1 d1, c1 2 f1, b1 3 e1, g1

If Black plays 2 .., g1 instead of 2 .., c1 White can once again obtain the position of diagram 13.14 by playing 3 e1. We now have shown that the only remaining move for Black after 2 f1, is 2 .., e1. That position is shown in diagram 13.15.

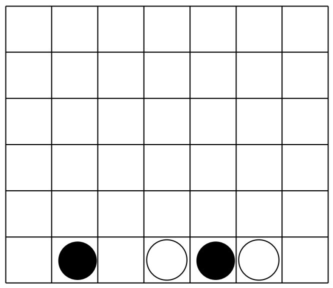


Diagram 13.15.

In the position of diagram 13.15 VICTOR suggested 3. f2 as best move. After a few hours of search it was shown that that move indeed guarantees White a win.

§ 13.5. 1. d1, a1

As was to be expected, Blacks first move at a1 is very bad. White should be able to use the opportunity to create threats at the center of the board. It was therefore not surprising that the first suggested move, 2 e1 ensures White a win.

After 2 e1 Black only has two real possibilities: 2 .., c1 and 2 .., f1. All other moves give White the opportunity to create two threats at the bottom row.

Independent of which of these two moves Black makes, 3 e2 will always ensure White a win. The two possible positions after White’s third move are shown in diagrams 13.16 and 13.17.

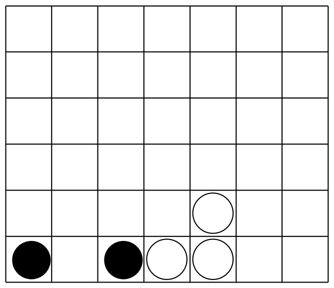


Diagram 13.16.

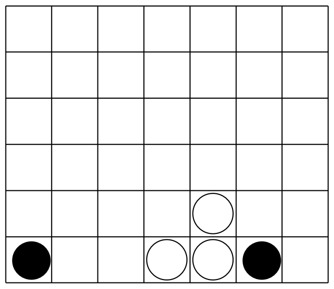


Diagram 13.17.

The fact that the positions of diagrams 13.16 and 13.17 can be won by White, shows that after 1 d1, a1 White can win the game.

§ 13.6. Evaluation of Result

An interesting aspect of some variations, is the fact that in many positions where White can win the game, he must make a forced move. This shows that the difference between a win for White and at least a draw for Black is small. As soon as White makes one sub-optimal move, Black can reach a draw. The 7 x 6 board is the smallest board at which this situation occurs. All smaller boards lead to easily drawn games for Black. It is of course still possible, that the difference between a win for Black and a draw for White on these smaller boards is also very small, although we do not expect that.

Observing the variations which have to be searched, it is clear, that some positions consisting of only a few men can easily be checked, while other variations consisting of 6 or 7 men cause rather difficult problems. This is probably due to the fact, that VICTOR has more trouble with positions where both players seem to have chances. If one player can easily win, it is most of the time detected soon. The more difficult positions are decided by tactical threats, a concept which has to be solved by searching instead of evaluating.

14. Conclusions

In this chapter the project is evaluated. Moreover, the results are placed in the context of the Shannon C-type strategy.

§ 14.1. Evaluation of the Project

Strategic rules have been defined for Connect-Four. These rules are similar to the rules expert players use to determine which move to make. Using an algorithm to combine these rules it is sometimes possible to determine the game-theoretical value of a position. Sometimes these rules cannot be used to find a result.

The algorithm can be used as evaluation function, on any board of m x (2n) column. Application of the algorithm on 6 x 6 and 6 x 8 boards revealed an easy strategy which can be used for Black to at least draw the game on any board with less than 7 columns, and an even number of rows.

Using the same evaluation function it was easy to show that on a 7 x 4 board Black could also draw the game.

The evaluation function was not powerful enough to determine the game-theoretical value of the standard 7 x 6 board. After some search techniques were implemented, it became clear, that the search tree would become very large. Still it was possible, using 2 Sun-4 computers of the Vrije Universiteit of Amsterdam, to show that the initial position can be won by white.

Mostly due to hardware problems and wrong choices of variations by the author more than 1000 hours of CPU-time were used. The four winning variations described in chapter 13, together took about 350 hours of CPU-time.

It is quite surprising that two expert players had the feeling that the game could be at least drawn by Black, if not won, while the search has revealed that White can win the game. It is maybe even more surprising that the smallest board where White can win the game is chosen to be standard board.

§ 14.2. Results of Shannon C-type strategy

One goal of the project was to show that a program can be written using the Shannon C-type strategy. This goal is not only met, it is also shown that the program, VICTOR, can always win playing White. There can be some doubt about the fact if the strategy used is pure Shannon C-type. The evaluation function clearly is, while the search techniques clearly are not. If, however, the number of positions which needed to be evaluated is compared to the upper bound for the number of legal positions (section 2.1), it is clear that the impact of the evaluation function is very large. We are optimistic about the possibilities to reduce the amount of search needed if the knowledge used by VICTOR is extended. We therefore believe that it will be possible to use the results of this project to develop a program which is even more Shannon C-type, which achieves the same as VICTOR developed for this project.

This project has shown that it is difficult to develop a Shannon C-type program, since much knowledge has to be implemented. If, however, enough knowledge is gathered, the program can outperform all other programs and human beings. For a moderately complex problem as Connect-Four, this leads to a program which can determine the game-theoretical value of the game.

§ 14.3. Using the Results for Playing

Each time the game-theoretical value of a position had been determined, the search table built during the search was dumped to a file. A small C-program then extracted the relevant, winning lines, from the search table.

Combination of the winning lines of the relevant variants resulted in a database of about half a million positions. This database can be seen as an opening book for Connect-Four. Using this opening book and the evaluation function for positions not present in the database, VICTOR can play correctly against any opponent, always winning playing White.

Each time VICTOR has to make a move, he checks if any of the sons of the position are present in the data base. If so, one of these sons is chosen. If no son is present, all sons are evaluated. The value of at least one of these sons must be 1, otherwise the database would not be complete. Such a son is chosen. The database is ordered in a way which makes a binary search for a position possible. Therefore, making a move takes at most a few seconds.

15. Future Development

In this chapter some ideas for future development are considered, as well as a message which has been received a few days before this text was printed.

§ 15.1. Suggestions for further Research

It was possible to find the game-theoretical value of Connect-Four, using a combination of Shannon C-type strategy and some search techniques. It can be tried to extend the knowledge implemented, in that way reducing the amount of search needed. Another approach could try to automatically derive the strategic rules, instead of implementing rules suggested by a human expert.

§ 15.2. Future Development

VICTOR will be used to check more variations, to find paths for Black in which an imperfect White player, can easily make a mistake. Using this knowledge VICTOR should be able to win all his White games, some of its Black games, and at least draw most of his Black games, depending on the strength of its opponents. To test its strength, it will compete in the Olympiad organised in London which is planned for August 1989.

§ 15.3. Results Confirmed

A few days before this final text was printed, a message in the USENET newsgroup rec.games.programmer appeared, claiming that Connect-Four can be won by the first player. Author of the USENET article was James D. Allen. Some variations were given, which confirmed our results. Some results presented there were not yet checked by VICTOR, while others (e.g, the results after 1 a1, b1 or c1 which can be used for every 7 x (2n) board) were found by VICTOR and not yet by Allen. Allen made use of a brute force program which checked about 6 billion positions. It was not possible to get more details about variations Allen checked before this text had to be printed.

References

[1] H.J. van den Herik, *Computerschaak, Schaakwereld en Kunstmatige Intelligentie*, Ph.D thesis, 1983.

[2] P. van Diepen and H.J. van den Herik, *Schaken voor Computers*, Academic Service, 1987.

[3] C.E. Shannon, *Programming a Computer for Playing Chess*, Philosophical Magazine, **41**, pp. 265-275, 1950.

[4] D.E. Wilkins, *Using Patterns and Plans in Chess*, Artificial Intelligence, **14**(2), pp. 165-203 (1980).

[5] H.J. van den Herik and I.S. Herschberg, *The construction of an Omniscient Endgame Data Base*, ICCA Journal, **8**(2), pp. 66-87, 1985.

[6] H.J. van den Herik, *Informatica en het Menselijk Blikveld*, Inaugurele Rede, Rijksuniversiteit Limburg, Maastricht, 1988.

[7] J. Schaeffer, *Conspiracy Numbers*, Advances in Computer Chess, D. Beal, 1987.

[8] D.A. McAllester, *Conspiracy Numbers for Min-Max Search*, Artificial Intelligence, **35**, pp. 287-310, 1988.

Appendix A

```
/* This program calculates an upper bound of the number of legal positions
 * on a COLUMNS x ROWS board.
 * This is done using the following observations:
 * The number of men on such a board can vary between 0 and COLUMNS * ROWS.
 * For each number of men, we count the number of ways to distribute the
 * men over the different columns. This is done by first putting a number
 * of men in the first row. Then some men are put in the second row. These
 * men can only be put in columns where a man was put in the first row.
 * This way continuing, we get a decreasing series of numbers, indicating the
 * number of men in rows 1 up to ROWS.
 * If we name these numbers a(1) to a(ROWS), then the number of ways we
 * can choose the a(1) columns to put the men of the first row in, is
 * Chose(ROWS, a(1)). The a(2) men of the second row can be put in this
 * row in Chose(a(1), a(2)) different ways, and so on.
 * The total number of ways to distribute a given number of men over the
 * different columns is multiplied by the number of ways to colour these men,
 * which is equal to Chose(Men, Men/2), since half of the men are white, while
 * the other half is black.
 *
 * N.B.1 In order to be able to work with the large numbers involved, doubles
 *       are used instead of int.
 * N.B.2 In order to speed things up, first all Chose(n,k) numbers are
 *       calculated.
 * N.B.3 Another method, somewhat faster than the one used here, using
 *       a recursive formula was suggested to me by John Tromp.
 */

#define COLUMNS         7
#define ROWS        6
#define MIN(A,B)    ((A) < (B) ? (A) : (B))

double Chose[COLUMNS * ROWS + 1][COLUMNS * ROWS + 1];
```

```
main()
{
        int Men;
        double Total, Extra, Distributions();

        Total = (double) 0;
        InitChose();
        for (Men = 0; Men <= COLUMNS * ROWS; Men++) {
                Extra = Distributions(Men) * Chose[Men][Men/2];
                printf("Total with %d men, is %.0f0, Men, Extra);
                Total += Extra;
        }
        printf("Total number of positions on %d x %d board is at most %.0f0,
              COLUMNS, ROWS, Total);
}

InitChose()
{
        int i, j;

        for (i = 0; i <= COLUMNS * ROWS; i++) {
                Chose[i][0] = (double) 1;
                for (j = 1; j < i; j++) {
                        Chose[i][j] = Chose[i-1][j] + Chose[i-1][j-1];
                }
                Chose[i][i] = (double) 1;
        }
}

double
Distributions(Men)
int Men;
{
        double Distribute();

        return (Distribute(1, COLUMNS, (double) 1, Men));
}
```

```
double
Distribute(Row, UpperBound, Product, MenLeft)
int Row, UpperBound, MenLeft;
double Product;
{
	int i;
	double Sum;

	if ((ROWS - Row + 1) * UpperBound < MenLeft) {
		return ((double) 0);
	}
	if (Row > ROWS) {
		return (Product);
	}
	Sum = (double) 0;
	for (i = MIN(UpperBound, MenLeft); i >= 0; i--) {
		Sum += Distribute(Row+1, i,
				Product * Chose[UpperBound][i],
				MenLeft - i);
	}
	return (Sum);
}
```

Appendix B

In this appendix we look at the set of rules found, after the best answer was made after the first move of White, where White did not play 1. d1. In all cases the set of rules can be shown to be independent of the height of the board, and therefore similar sets of rules can be used for all 7 x (2n) boards.

§ 1. Situation after 1. a1.

After 1. a1, a good answer for Black is 1. .., b1.

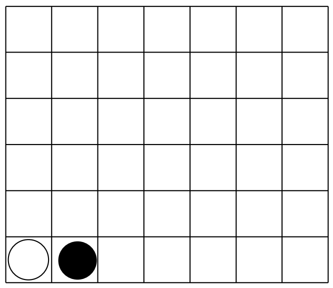


Diagram B.1.

For the position of diagram B.1, the program found the following set of rules:

Before	a2 a3 b2 b3 c1 c2 d1 d2:
Before	a4 a5 b4 b5 c3 c4 d3 d4:
Before	b2 b3 c1 c2 d1 d2 e2 e3:
Before	b4 b5 c3 c4 d3 d4 e4 e5:
Before	c1 c2 d1 d2 e2 e3 f2 f3:
Before	c3 c4 d3 d4 e4 e5 f4 f5:
Before	d1 d2 e2 e3 f2 f3 g1 g2:
Before	d3 d4 e4 e5 f4 f5 g3 g4:
Baseinverse	e1 f1:
Claimeven	c5 c6:
Claimeven	d5 d6:
Claimeven	g5 g6:

If we look at the Claimevens which are part of the befores, we see that Claimevens are used throughout the whole c, d and g column. In the e and f column a Baseinverse is used for the lower two squares and verticals are used for the rest of the e and f columns. In the a and b columns we find verticals, too.

This results in two pairs of columns, where Black gets at least one men in every odd row of these columns, and three other columns where Black gets all even squares, while White gets all odd squares. Since these pairs of columns, a and b versus e and f are divided by another pair of columns, where Black gets all even squares, it is not hard to see that there is not one group White can complete, without giving Black a win first. The reasoning for that is very similar to the reasoning used for the 6 x (2n) board.

§ 2. Situation after 1. b1.

After 1. b1, a good answer for Black is 1. .., c1.

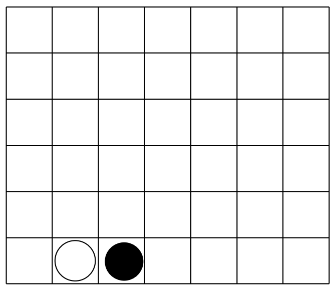


Diagram B.2.

For the position of diagram B.2, the program found the following set of rules:

Before a1 a2 b2 b3 c2 c3 d1 d2:
Before a3 a4 b4 b5 c4 c5 d3 d4:
Before b2 b3 c2 c3 d1 d2 e1 e2:
Before b4 b5 c4 c5 d3 d4 e3 e4:
Before c2 c3 d1 d2 e1 e2 f2 f3:
Before c4 c5 d3 d4 e3 e4 f4 f5:
Before d1 d2 e1 e2 f2 f3 g2 g3:
Before d3 d4 e3 e4 f4 f5 g4 g5:
Baseinverse f1 g1:
Claimeven a5 a6:
Claimeven d5 d6:
Claimeven e5 e6:

As we can see, in the a, d and e column, all even squares are claimed by Black, while the b and c, resp. f and g column form pairs, similar to the pairs of section 1. We can therefore conclude in the same way, that Black can achieve at least a draw on every 7 x (2n) board, if White opens 1. b1.

§ 3. Situation after 1. c1.

After 1. c1, a good answer for Black is 1. .., d1.

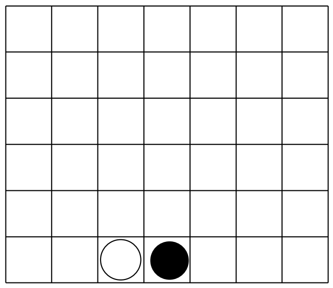


Diagram B.3.

For the position of diagram B.3, the program found the following set of rules:

Before	a1 a2 b1 b2 c2 c3 d2 d3:
Before	a3 a4 b3 b4 c4 c5 d4 d5:
Before	b1 b2 c2 c3 d2 d3 e1 e2:
Before	b3 b4 c4 c5 d4 d5 e3 e4:
Before	c2 c3 d2 d3 e1 e2 f1 f2:
Before	c4 c5 d4 d5 e3 e4 f3 f4:
Before	d2 d3 e1 e2 f1 f2 g1 g2:
Before	d4 d5 e3 e4 f3 f4 g3 g4:
Claimeven	a5 a6:
Claimeven	b5 b6:
Claimeven	e5 e6:
Claimeven	f5 f6:

We can see that in the a, b, e, f and g column all even squares are claimed (except for g6, but we can claim that square, just for symmetry reasons). The c and d column form a pair of columns, giving Black at least one men in all odd rows of these columns. Again we can see that White cannot complete a group, before Black does, and therefore Black achieves at least a draw. Like in the former two sections, this result can be extended to all 7 x (2n) boards.

Appendix C

In this appendix the number of positions found for each number of men on the standard 7 x 6 board is presented in a table.

0	1	22	42121344720
1	7	23	76871042612
2	56	24	141627466344
3	252	25	242619996500
4	1260	26	417750499600
5	4620	27	669746637000
6	18480	28	1073881265400
7	59815	29	1602674216640
8	206780	30	2377951581600
9	605934	31	3277691366670
10	1869840	32	4460016493800
11	5038572	33	5611156155990
12	14164920	34	6893472773880
13	35459424	35	7754703113850
14	91871208	36	8385425017200
15	214864650	37	8164755937800
16	516936420	38	7422505398000
17	1134183050	39	5789554210440
18	2546423880	40	3859702806960
19	5252058812	41	1883902560540
20	11031780760	42	538257874440
21	21406686756	Total	70728639995483