



MASTER'S THESIS OF

Arjen Bos (9204318)

SUPERVISED BY

Ida Sprinkhuizen-Kuyper

Henk Goeman

CONTENTS

Preface	iii
Introduction	iv
List of Figures	vii
Glossary	ix
1 Game Description	2
1.1 Material	2
1.2 Notation	3
1.3 Rules	6
2 Game Characteristics	13
2.1 Complexity	13
2.2 Symmetrical variants	18
3 Creating A Computer player	21
3.1 Representation	21
3.2 Search Algorithm	25
3.3 Search Optimization	28
3.4 Evaluation Function and Genetic Optimizing	35
3.5 Using Neural Networks	40

4 Endgame of Cover	43
4.1 One vs One	43
4.2 Breakthrough	52
5 Game Viewpoints	56
5.1 Interesting Positions	56
5.2 Interesting Strategies	59
Conclusions	62
Appendices	65
Appendix A: The origin of Cover	66
Appendix B: Board diagrams of Cover-4 "One Vs One"	68
Appendix C: Board diagrams of Cover-4 breakthrough	70
Appendix D: WWW-Interface	74
Appendix E: Cover-Kernel	77
Appendix F: GIF-library	79
Appendix G: Cover play kit	82
Bibliography	86

PREFACE

This report has been created as a conclusion of my research for my master's thesis. This thesis would not be here without the help of the following persons.

Dr. Sprinkhuizen-Kuyper and Drs. Goeman for their support, effort and supervision, mr. Twigt for designing the game and sharing his experiences with me, Jeroen van Jaarsveld for correcting the language in this report. Furthermore, I would like to thank my girlfriend for her continuous support, stimulation and patience.

Finally, I would like to thank the more than 3000 users of Internet who tried the game and shared their experiences with me.

Arjen Bos

November 1995

INTRODUCTION

A manager of a large game-producing company said that if Chess was not known yet and someone would present it to him, he would not consider producing it. Now, you may think that he hasn't got feeling for the market, but he gave another explanation. The market for new board games has changed and purely strategic board games are not very popular now. The chance of a new strategic board game being profitable is just too low and therefore, they are not a very well investment.

Still, people create new board games. One of these new games is "Cover".

In this thesis, this new board game "Cover" is investigated from an artificial intelligence approach. The goal of this research is finding the possibilities for a computer to play "Cover" and discovering the problems one has to overcome when trying to find the theoretical value of the game. The theoretical value of a game is the outcome of the game, with the players of the game playing optimally.

During the research, many areas of artificial intelligence have been encountered. Clearly game theory was one of these areas, since a game was researched. Besides game theory, other areas like neural networks and genetic algorithms were encountered in this research. Neural networks were examined for their suitability to use as a technique to find a next move for a position. Genetic algorithms were used as an optimizer for the created computer player of Cover.

Besides artificial intelligence, other aspects of computer science were part of the research. One of these areas is about computer networks. The theory about computer networks was used to create and maintain a server on Internet, which was used for the World Wide Web

interface of Cover¹. For the same reason, a library was created which could manipulate graphic files. These graphic files were used in the WWW interface. More information about these areas can be found in the appendices of this thesis.

The research on Cover started with an investigation on the rules of Cover and the game theory. Soon after this investigation a basic computer player was designed based on the alphabeta search techniques. Other search techniques like best-first searches, proof-number search, conspiracy-number search and dependency-based search were also considered as basic search technique but were not chosen at this point. During the research on Cover, the basic computer player was improved with several optimisation techniques like null-move search and a variant of principal continuation. Neural networks were also a part of the research on how to optimize the search technique used in the computer player of Cover. The evaluation function of Cover was also improved continuously during the research. The improvements on both the evaluation function and the used search technique were made with the outcome of the further investigations on Cover done during the research. These investigations are investigations on the endgame stadia and on possible winning playing strategies. The following endgame stadia were investigated. The first is the "One vs One" endgame in which each player has only one stone left. The second is the breakthrough endgame in which a player has broken through the defence of the opponent.

The thesis is divided into 5 chapters. In chapter 1, the rules of the games are explained with the possible variants. Furthermore in this chapter, some possible notations for the game are described, among which, the notation that is used in this thesis. In chapter 2, some mathematical characteristics of the game are described, like complexity and symmetrical boards. Chapter 3 describes the design and implementation of a computer player for the game. In this chapter, the choices, which one has to make while creating a computer player with the examined options, are described. Per choice, the used option is explained with the argumentation for using this option. Chapter 4 describes how some

¹ The interface can be found on URL: <http://artemis.wi.leidenuniv.nl:7171/cover/>

endgame databases for the game were created and used. In chapter 5, some interesting positions and strategies are described.

At the end of the report a conclusion is given with some recommendations about further examination of "Cover".

LIST OF FIGURES

Figure 1: An example of a Cover-4 position	2
Figure 2: The original notation	4
Figure 3: The diagram notation	4
Figure 4: The delta notation	5
Figure 5: The initial boardposition of Cover-4	6
Figure 6: All possible moves	7
Figure 7: Shooting a stone	9
Figure 8: Sketch of the computation of the state-space of Cover-4	15
Figure 9: Sketch of the game-tree computation of Cover-4	17
Figure 10: Complexities of the Cover variants	17
Figure 11: Game complexities compared	18
Figure 12: Symmetrical variants of a boardposition	20
Figure 13: Properties of the board of Cover	22
Figure 14: Pseudocode of standard alphabeta	26
Figure 15: Alphabeta search on a game tree	27
Figure 16: Quiescence search	29
Figure 17: The influence of the ordering of moves on alphabeta search	30
Figure 18: Pseudocode of optimized alphabeta	33
Figure 19: Sure stone captures	37
Figure 20: Selection of players in a genetic algorithm	38
Figure 21: Genetic creation of evaluation parameters	39
Figure 22: Optimized evaluation parameters	40
Figure 23: An "One v One" endgame position: [I09]x[J08], blue wins	47
Figure 24: A complex "One vs One" endgame position: [H09]x[I10], blue wins	47
Figure 25: An "One vs One" endgame position: [H10]x[J10], blue wins	49

Figure 26: An "One vs One" endgame diagram	50
Figure 27: Statistics on the "One vs One" endgame database	51
Figure 28: Position [A08]x[A11] with red to move is won by blue after 38 moves. . .	51
Figure 29: Breakthrough function	53
Figure 30: The breakthrough endgame diagram of F08	55
Figure 31: Position 1: [H08]x[J10]	56
Figure 32: Position 2: [B05,G07,H04]x[D03,F08,I02,D10]	57
Figure 33: Position 3: [B02,I11]x[A03,J10]	58
Figure 34: Position 4: [C02]x[B05]	59
Figure 35: The page creation cycle of the WWW interface	75
Figure 36: The MS-Windows Interface	77
Figure 37: Overview of the kernel of Cover	78
Figure 38: Structure of a GIF-file	79

GLOSSARY

Boardposition	The set of stones of both players with their locations on the board. A boardposition is notated as ["Blue stones"]x["Red stones"]. E.g. [G01, E03, C05, A07]x[K05, I07, G09, E11].
Breakthrough	A stone has broken through the opponent's defence and has a straight path to the opponent's homebase without the opponent being able to obstruct.
CGI-bin	Common Gateway Interface binary, a program that produces HTML files for a WWW-server.
Distance	The distance from the location of a stone to the homebase of the opponent, measured in moves to make.
Endgame	Phase in the game that is near the end of the game.
Field	Location in a board-diagram that represents the location at which a stone can be placed.
Game-tree	The tree which contains all possible courses of a game. The moves of the players are the branches of the tree and the positions are the nodes of the tree. One position can correspond to multiple nodes in the tree since different combinations of moves can lead to the same position.

Homebase	Each player has a homebase which is located in a far corner of the board. A player has to capture the opponent's homebase to win the game.
Horizontal distance	The distance of the location to the homebase of the opponent, measured in the horizontal lines between the location and the homebase.
HTML	Hypertext Markup Language, a text document format as used in WWW.
Internet	World wide computer network.
Move	The displacement of a stone by one of the players. A move is notated as "From"- <i>"To"</i> <i>"x"</i> <i>"Hit"</i> . If no stone was hit, the part which appoints the hit is omitted. E.g. A04-B05, C05-D06xH09.
Notation	The way of describing the locations of a stone, the moves and the (board)positions.
Ply	A ply is a technical term used for a move.
Position	The current situation in the game which includes the boardposition, the current player to move and the relevant previous moves. If no current player is specified with a position, the blue player is the player to move.

Shooting	Moving a stone in the length direction of the stone and by that hitting a stone of the opponent on its weak side. The latter stone is removed from the board.
State-space	The number of different positions that can be reached in a game.
Symmetrical variants	Boards which result when a board is mirrored in the length-axis of the board or when mirrored over the colors of the stones and in the breadth-axis.
Theoretical value	The theoretical value of a game is the outcome of the game from the initial position with both the players of the game playing optimally.
Threat	Possibility to shoot a stone.
URL	Universal Reference Link, a pointer to an address on internet/WWW.
Vertical distance	The distance from a location on the board to the center of the board, measured in vertical lines between a location and the center vertical line.
World Wide Web	Graphical interface of Internet.
WWW	World Wide Web

MAIN

DOCUMENT

1 GAME DESCRIPTION

Cover is a strategic game, played by two players, red (dark) and blue (light). The players move their stones across the board, with the goal of reaching the opponent's homebase. During the game, a player can shoot the stones of the opponent which are removed from the board when hit.

1.1 MATERIAL

Cover is played in several variants. The most common one is Cover-4 in which both players have four stones. Other logical variants are Cover-2, Cover-6, and Cover 8 with respectively 2, 6 and 8 stones per player.

The number of stones determines the size of the board. The size of the board of Cover-X is $Y \times Y$ where $Y = 3 + 2 * X$. For example: Cover-4 is played on an 11x11 board.

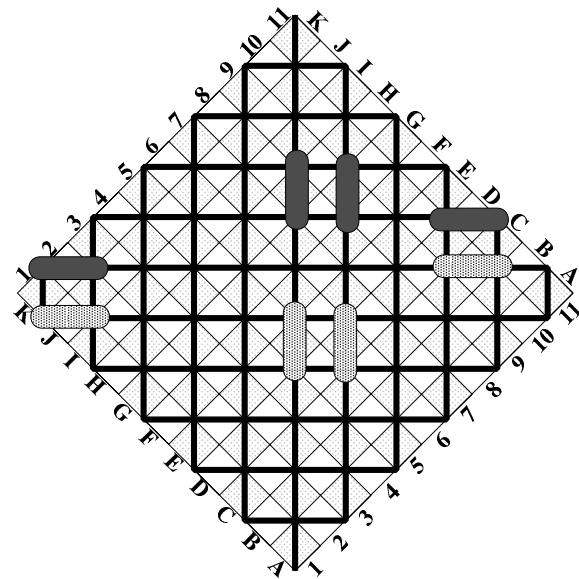


Figure 1: An example of a Cover-4 position

A board of size $Y \times Y$ contains Y^2 fields on which a stone can be placed. Cover-4 with the size of 11x11 has therefore 121 fields c.q. locations where a stone can be placed.

The board is placed in a diagonal direction. At the top and the bottom of the board are the homebases of the red and blue player. In figure 1, A01 is the homebase of the blue player and K11 is the homebase of the red player. The board also contains a frame of lines over which the stones can be moved.

Each stone is always located on a line between two adjacent intersections of lines. In the diagram, each location on which a stone can be placed is called a field. In each field a diagonal line is drawn. This is the line on which a stone can be placed. The direction of a stone can either be in a horizontal or in a vertical direction depending on the location of the stone. The vertical locations have a gray background in the figures. The horizontal locations have a white background.

The stones have an oblong shape. The long side is the weak, vulnerable side. The short side is the attacking, aggressive side. A stone can attack an opponent's stone when its aggressive side is pointing to the weak side of the opponent's stone.

1.2 NOTATION

During the research three variants of notations for the board were created. Below, these notations will be described with their advantages and disadvantages.

ORIGINAL NOTATION

This notation is based on the lines on which the stones are moved. The vertical lines are marked with digits, the odd on the right side, the even on the left side. The horizontal lines are marked with letters.

The notation for the position in the example on the right is: [BD.0, C.35, CE.1, A.68]x[FH.0, A.35, AB.1, A.24].

The advantages of this notations are that it is easy to see on which line a stone is located and what the direction of the stone is. The disadvantages are that there is no simple relation between adjacent locations and that the notation has a large amount of redundancy.

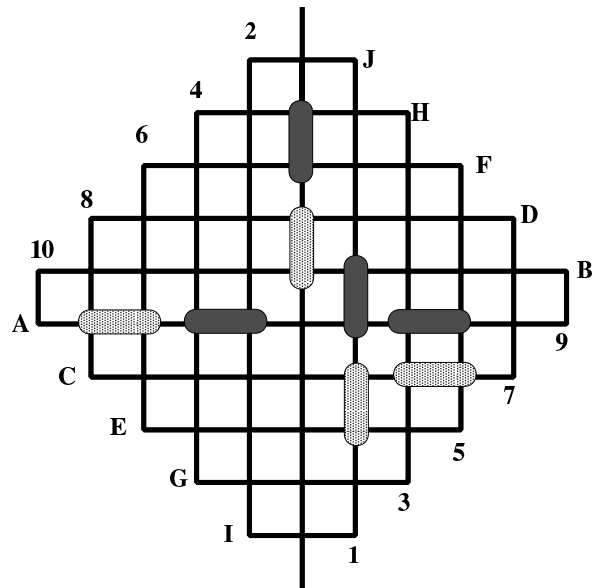


Figure 2: The original notation

DIAGRAM NOTATION

This notation is the one used in this thesis. The notation is based on the diagram that can be placed over the board so that every location on which a stone can be placed is covered by a field in the diagram². The diagonal line through a field is a line on which a stone can be placed. The representation of a location on the board consists of two parts. The first part marks the left diagonal and is represented in letters. The second part marks the right diagonal and is represented by two digits.

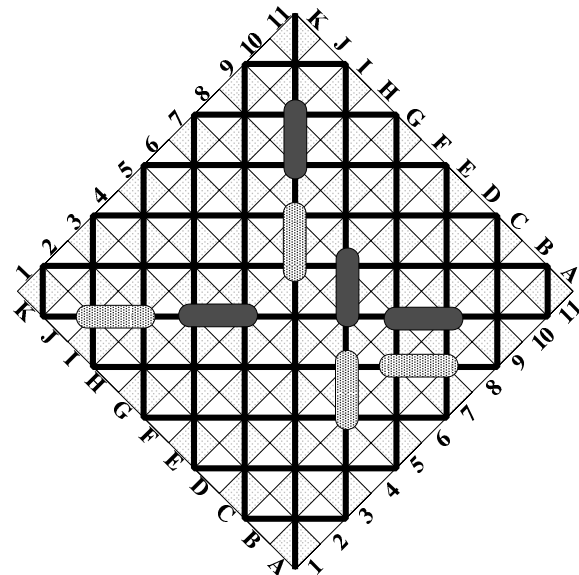


Figure 3: The diagram notation

² Art-lovers may see a resemblance between the diagram and the painting of P. Mondriaan called 'Losangique with grey lines'.

The notation for the position in figure 3 is: [I02,C05,B07,G07]x[G04,I09,E07,C08]. The advantages of this notation are that there is no redundancy and a large amount of regularity. Almost every possible property of the board is easy to calculate or can be seen immediately. This makes this representation an ideal representation for a computer program. The fact that it is hard to see the coordinates of a location without an assisting diagram is the main disadvantage.

DELTA NOTATION

This notation is a variant of the original notation and is based on the distance of the fields and lines to the center of the board.

The notation is like an axes system. The location in the center is marked 0/0, where the first number represents the value of the X-axis and the second number the value of the Y-axis. Each vertical location contains two even numbers, while each horizontal location contains two odd numbers. For instance,

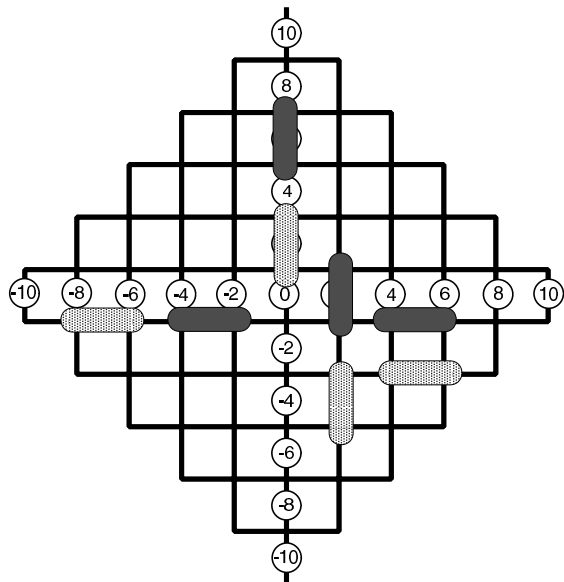


Figure 4: The delta notation

the four horizontal locations around the center of the board are [1/1, 1/-1, -1/-1, -1/1].

The notation for the position as given in the above example is:

$[-7/-1,0/2,2/-4,5/-3]x[-3/-1,0/6,2/0,5/-1]$.

The advantages of this notation are that relations between locations and other properties like on which line a stone is located and the direction of stones are also easy to see. This

notation is very suitable for human players, but is less efficient than the diagram-notation since it also denotes each intersection of lines, e.g. 2/5 is not a location.

1.3 RULES

INITIAL POSITION

Initially, the stones of a player are placed next to each other. Between two stones, one vertical line is kept empty. Between the red and the blue stones are two empty horizontal lines. The player with the blue stones begins. This means that for Cover-4 the initial boardposition is [G01,E03,C05,A07]x[K05,I07,G09,E11], with [G01,E03,C05,A07] being the locations of the blue stones and [K05,I07,G09,E11] being the locations of the red stones. This position is shown in figure 5.

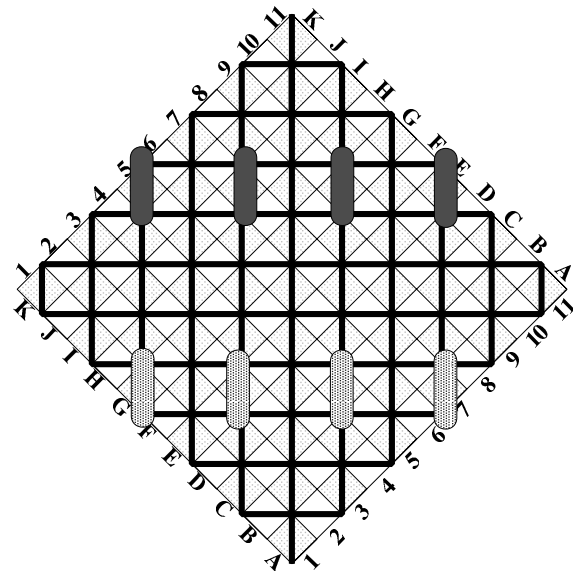


Figure 5: The initial boardposition of Cover-4

MOVING

A stone has at most eight locations to which it can move. These locations can be reached by making one of the following moves as seen from the viewpoint of the blue player:

- *moving forward*
E.g. J05-K06 or E01-F02
- *moving backwards*
E.g. K10-J09 or B06-A05
- *moving left*
E.g. H09-I08 or A03-B02
- *moving right*
E.g. H07-G08 or B10-A11
- *rotating +90 or -90 degrees around one of the two intersections of a location*

- E.g. A08-A07 or K01-K02
- H11-I11 or I03-H03
- E10-E11 or F06-F05
- E08-D08 or D04-E04

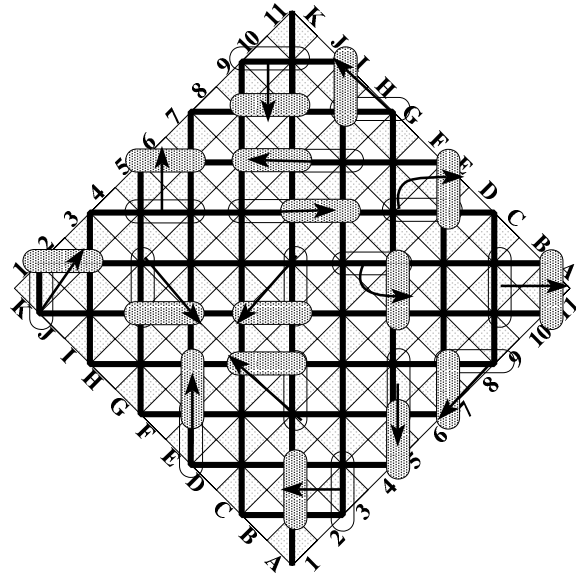


Figure 6: All possible moves

The following restrictions apply for a move:

- *The destination of a move may not be occupied*
A stone may only be moved to a location if both the intersections of this location are empty. The only exception to this rule is when the stone which is placed on one of the intersections of the destination will be shot in this move.
- *The destination of a move may not be the homebase of the current player*
A stone may not be placed on the homebase of the current player, since the other player would not be able to reach this location.

- *Boardpositions may not be repeated*

Several variants of this rule are known of which the most common ones will be described below. This rule restricts the possible moves by not allowing a move when it leads to a boardposition that is identical to the boardposition as a number of moves earlier in the game. The following variants are found for the number of moves between which the same boardposition is not allowed:

- a) any boardposition earlier in the game. This means no boardposition may be repeated.
- b) within 4 moves. This means that when one player has moved a stone forward and in the next turn returns it to its old location, the other player is not allowed to return her stone to its previous location.
- c) within 8 moves. This means that more complex situations which bring back a boardposition earlier achieved are not allowed.

Variant "b", which excludes only the direct forward and back movement by both players, seems the best option. It is simple and influential enough to prevent the simple draw which is clearly the theoretical value of Cover without this rule. Variant "a", and also variant "c", restrict moves too much and are too complex for a human player to use in a match. Unless mentioned otherwise, variant "b" is used in this thesis.

This rule determines what the definition of the current position will be. With variant "b", the last three moves are part of the current position since one of the possible next moves may now be illegal because of the last three moves.

SHOOTING

A stone of the opponent can be shot by making a move in the length direction of the stone. The stone that was hit is removed from the board.

The following rules apply for shooting:

- *Shooting is voluntary*
A player does not have to shoot other moves are available.
- *The shot is in the moving direction.*
A player can shoot by moving the stone in the length direction of the stone.
The shot is made in the same direction.
- *A stone can only be hit on the weak side.*
A stone is only shot and removed from the board if it is hit on the weak vulnerable side. Otherwise the shot is just blocked.
- *A shot has infinite length.*
A shot has the reach of the entire board if it is not blocked by another stone.
- *A shot never goes past another stone on the shooting line.*
A shot is blocked when it reaches a stone on its track. The blocking stone may be any stone of both players. Only a stone of the opponent must be removed when hit.

A possibility to hit a stone is called a threat. In the diagram on the right the following threats are available:

- *A07-B08xE10*
The blue stone on A07 can shoot the red stone on E10 by moving to B08.
- *K02-J03xE07*
The red stone on location K02 can shoot the blue stone on E07 by moving to J03.

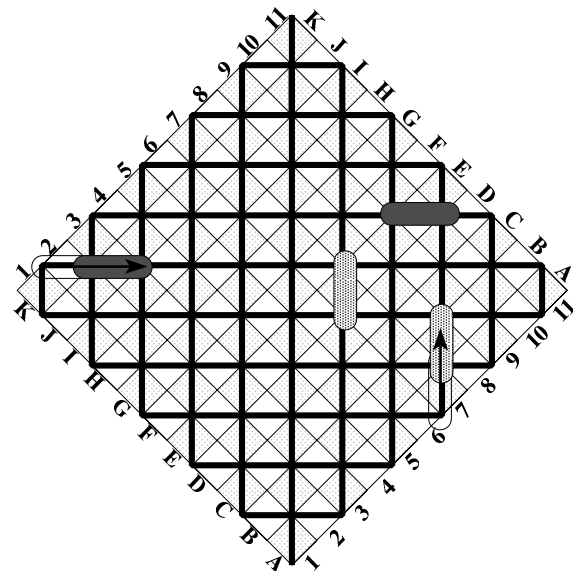


Figure 7: Shooting a stone

GOAL

The goal of the game is to reach the homebase of the opponent. The player that reaches the opponent's homebase first is the winner of the game. A player has also won when the opponent is out of stones or when the opponent is not able to perform any legal move in her next turn.

1.4 PHILOSOPHY

The game Cover is designed by N. Twigt. This section contains his viewpoint on the game Cover.

N. Twigt says the following about the philosophy of Cover:

HIERARCHY AND CONTEXT

Cover symbolizes a modern way of thinking about the concepts of "hierarchy" and "context". Especially hierarchy is very fundamental. It has to do with the oldest question in the world: who eats who. Or: am I stronger than you, or are you stronger than me? Or: do I have a better car, a more rich or beautiful partner, etc. Cover represents a slow, but nevertheless far-reaching change in our way of thinking about hierarchy.

VILLAGE-HIERARCHY

Up to about 100 years ago, people lived in a village-hierarchy, dominated by the burgomaster, the schoolmaster, and the policeman. Your place in the hierarchy was more or less determined by birth. If your father was a carpenter, you would be a carpenter too. The same idea is found in the Chess game. In Chess, the strength of each piece is defined by the moves it can make. A queen for example, can make many moves and is therefor

stronger than a knight or a bishop. The pawn is the weakest piece. If a pawn makes many moves, it reaches the other side and is promoted. "Strategy" is another good example of a game where the pieces all have a predetermined hierarchy.

CITY-HIERARCHY

In modern society, most people live in a "city-hierarchy". In cities people live in many independent hierarchies. Each profession has in fact a hierarchy of its own. In the city, our village carpenter is manufacturer of furniture. He may start working with his hands, and be promoted until he becomes the manager. He may even begin another career and become a major or the president of the country. In the modern school system, each child is confronted with city-hierarchy. One will be good in science, another may be good in languages or in sports. The context, or the class, or the place and time, exposes his strong or weak sides. Later, the sportsman may earn as much and be as influential as the scientist or the politician, so their status is more or less the same.

HIERARCHY OF PRODUCTS

The same principle of strong and weak sides we find in products. In the old days we bought the products (the car, the house, etc.) that matched our place in hierarchy. Now we buy the product that matches the way we use it. If you live in the city, it is very unpractical to drive a big Mercedes or a Rolls, even if you can afford it. The weak side of a big car is exposed when you try to find a parking place.

HIERARCHY ON THE BATTLEFIELD

There is a remarkable resemblance between the way we think about hierarchy and the way we organize ourselves on the battlefield. On the traditional battlefield, many ranks were present. The same principle is found in Chess. In the first and second world war, the

battlefield was for the soldiers. This principle is found in Draughts and Go. Modern battles are mainly fought from tanks, ships and aeroplanes. These machines are a bit like Cover pieces: vulnerable from the sides, aggressive at the front and shooting over a long distance. Tactically a Cover game looks and feels a little like a battle with tanks or ships.

DEVELOPMENT

It is not to be expected that the "village" type of hierarchy will eventually disappear. Especially in armies and other fighting situations, where decisions have to be executed rapidly, the classical "village" type of hierarchy is very practical. In "economical" situations, where we try to gain as much as possible from the strong sides of products and people, the new "city" type of hierarchy will develop itself. Cover is the game that will help us think accordingly. People and products have many strong and weak sides. In Cover the pieces have only 2 strong and 2 weak sides.

For more information about Cover, please contact the inventor³:

NTO

Kerkplein 5

1621CX Hoorn

The Netherlands

Tel. (0031) (0)229-470726

³ See appendix A for the background story about the design of Cover.

2 GAME CHARACTERISTICS

2.1 COMPLEXITY

The complexity of a game is a measurement for the solvability of a game. In general it can be said that the lower the complexity, the easier it is to solve the game. The complexity of a game is denoted in two different measures, the state-space complexity and the game-tree complexity. In this section, these two measures, as defined by Allis (Allis 94), will be described, together with an estimation of these values for Cover.

STATE-SPACE COMPLEXITY

The state-space complexity is the number of different positions that can be reached in a game without considering the symmetries of a board. For Cover, the current position depends on three factors: the boardposition, the current player to move and the last three moves, since the repetition of a boardposition is not allowed within four moves. The complexity of these three factors can be calculated separately and can be multiplied afterwards to get the total estimated state-space complexity. The precise calculation of these three numbers is described below, followed by an example.

The complexity of the current player to move is two, since both players can have the turn. More difficult is the total number of boardpositions. This number is estimated as follows:

In each boardposition, each player owns one to four stones. The number of boardpositions for a certain combination of X blue stones and Y red stones can now be compared with picking X locations out of the total number of locations and picking Y locations out of the

remaining locations. These two factors have to be multiplied to get to total number of boardpositions with X blue stones and Y red stones. Summing this number for all possible combinations of X blue stones and Y red stones results in an upperbound of the total number of boardpositions. This estimation can be optimized by considering the fact that once a stone is placed on a location, most other surrounding locations cannot be used anymore. The number of locations on which the next stone can be placed can now be reduced with the average number of occupied locations per placement. A more precise upperbound of the number of boardpositions can now be calculated. Now, we have the following situation. A total number of $X+Y$ stones must be placed on F locations with an average location occupation per placement of O . This means that the first stone can be placed on F locations, the next stone on $(F-O)$ locations and the third stone on $(F-2*O)$ and likewise for the rest of the stones. The number of possible placements per stone have to be multiplied. This total must be divided by the number of combinations of ordering of $X+Y$ stones in X blue and Y red stones. Summing this last number for all combinations of X and Y gives a better upperbound for the total number of boardpositions.

Finally, the number of possible previous moves must be calculated. Per stone, at most eight moves can be made per turn. An upperbound for the total number of possible previous moves can now be estimated by multiplying the possible number of moves for the last three moves.

Multiplying the above three factors gives an upperbound for the total state-space complexity for Cover.

Figure 8 lists the algorithm used to calculate an upperbound for the state-space of Cover-4.

"The number of stones in this Cover variant"
 $CoverSize := 4$

"The size of an axis of the diagram of the board"
 $BoardSize := 3 + CoverSize * 2$

"The number of field in the diagram"
 $NrFld := BoardSize^2$

"The number of corner fields (A11, K01 in Cover-4)"
 $CornerFld := 2$

"The number of homebases (A01, K11 in Cover-4)"
 $HomeFld := 2$

"The number of remaining fields on the side of the board"
 $SideFld := (BoardSize - 2) * 4$

"The number of remaining fields"
 $NormalFld := NrFld - HomeFld - CornerFld - SideFld$

"The average number of occupied field per placed stone. Per type of field, multiply the number of fields with the number of fields they occupy and sum these factors. This sum has to be divided by the total number of fields to get an average"

$$AvgOccupied = \frac{NormalFld * 7 + SideFld * 5 + CornerFld * 3 + HomeFld * 4}{NrFld}$$

" $t_{i,j}$ is the number of boardpositions with i blue stones and j red stones"
 $i, j := 1..CoverSize$
 $m := 1..2 * CoverSize$
 $t_0 := 1$
 $t_m := t_{m-1} * (NrFld - (m - 1) * AvgOccupied)$

"Multiply the number of boardpositions with the complexity of the player to move and the complexity of the possible previous moves to get to total number of positions with i blue stones and j red stones $a_{i,j}$ "

$$a_{i,j} = \frac{2 * t_{i,j} * (8 * i)^2 * (8 * j)}{i * j}$$

"The upperbound of the state-space is the sum of each combination of i and j."

$$\sum a_{i,j} = 1.041 * 10^{18} \quad \text{State-space of Cover-4} \leq 10^{18}$$

Figure 8: Sketch of the computation of the state-space of Cover-4

GAME-TREE COMPLEXITY

The game-tree complexity is the minimal number of nodes that must be evaluated to determine the game theoretic value with a full width search. This number can, in general, be calculated very easily. The average number of possible moves should be taken to the power of the number of the average game length.

For Cover, the calculation of the game-tree complexity is not very easy since the game is very young, and there is not much information about factors as game length and average number of legal moves per move. Therefore these numbers have to be estimated, which makes the game-tree complexity a rather rough estimation.

The following numbers are based on the games that were played during my research. All numbers are estimated rather low to obtain a lowerbound for the game-tree complexity. The average possible moves per stone is estimated on 5, that means that of the 8 possible directions, every stone has three directions to which it can not move. The average number of stones per player during the game is estimated on 50% of the number of stones at the begin of the game, since stones can be shot during the game. The average game length is set to about 10 times the initial number of stones per player.

The game-tree complexity can now be calculated by taking the estimated average number of possible moves per move to the power of the estimated average of the number of moves.

"The average number of possible moves per turn"
 $Moves = 5 * 4 * 1/2$
 "The average game length"
 $Depth = 4 * 10$
 "The lowerbound of the game-tree"
 $Moves^{Depth} = 10^{40}$

Figure 9: Sketch of the game-tree computation of Cover-4

The estimated game-tree complexity is considered a lowerbound, since all factors of the game-tree complexity are estimated rather low.

COMPARISON

Calculating the game-tree and state-space complexities for the different variants of Cover produces the following table.

Complexity	Cover-2	Cover-4	Cover-6	Cover-8	Cover-10
State-space	10^9	10^{18}	10^{27}	10^{36}	10^{45}
Game-tree	10^{13}	10^{40}	10^{70}	10^{104}	10^{139}

Figure 10: Complexities of the Cover variants

Comparing these numbers to the numbers known from other games as given by Allis (Allis 94), results in the diagram of figure 11.

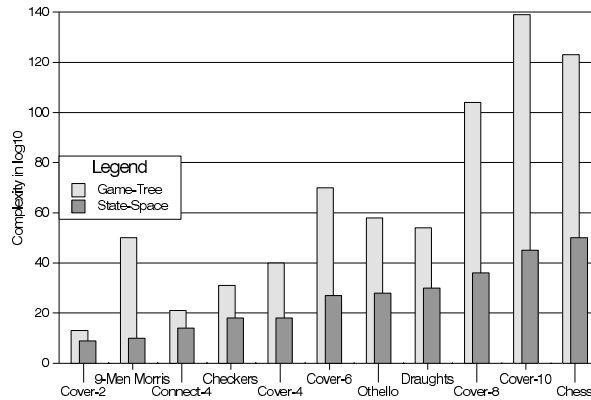


Figure 11: Game complexities compared

The diagram shows that the complexity of Cover-4 is about the same as Checkers and Cover-6 comes in the neighborhood of Othello. Since these are both unsolved games, it is not likely that Cover will be solved in the near future. Cover-2 has a low complexity and can therefore be solved very easily in the future when researched.

2.2 SYMMETRICAL VARIANTS

A board of a game can be mirrored into several axes in such way that the resulting boards are functionally identical. These symmetrical variants are useful, since it reduces the number of unique positions. Symmetrical variants can be treated equal in, for example, a game-tree search. In the implementation of Cover, the symmetrical variants are used in the breakthrough database and the endgame database.

For the board on which Cover is played, four symmetrical variants can be found, which are listed below followed by an example:

- *The original board*

The board itself.

- *The original board mirrored in the length-axis*

Every stone on the board is mirrored in the length-axis over the board. The turn and color of the stones stay the same.

- *The original board mirrored in the breadth-axis*

Every stone on the board is mirrored in the breadth-axis. The stones exchange owners. Blue stones become red stones and red stones become blue stones. The turn also changes. This can be compared with switching the viewpoints of the players.

- *The original board mirrored in both axes*

The stones are both mirrored over the breadth-axis and the length-axis. The color of the stones and the current player to move change also.

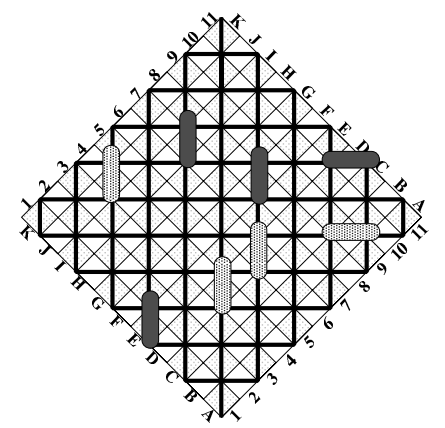
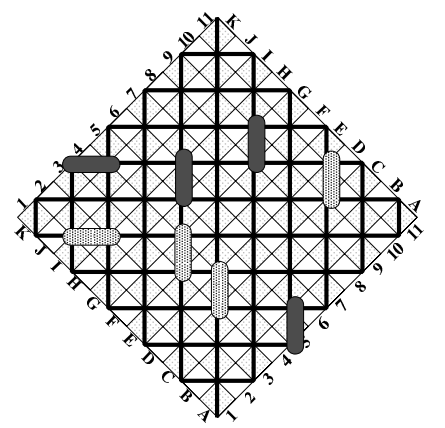
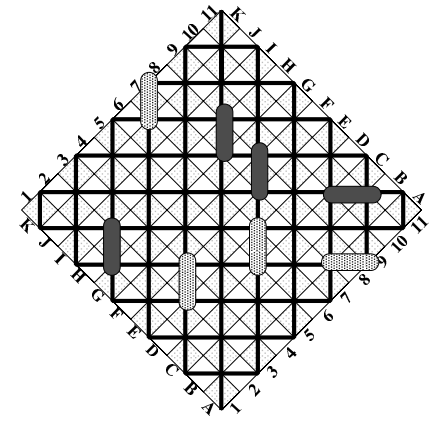
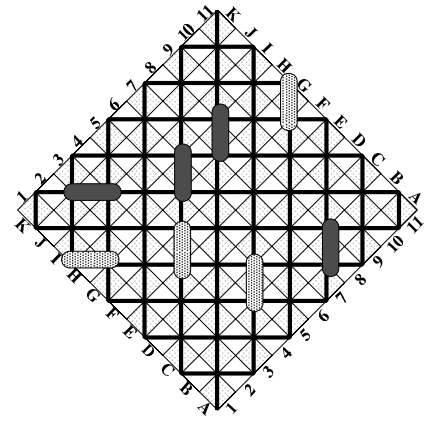
<p>Original board</p>  <p>[J04,D04,D06,B09]x[E01,I07,F08,D11] (Blue player's turn)</p>	<p>Board after length-axis mirroring</p>  <p>[D10,D04,F04,I02]x[K04,H06,G09,A05] (Blue player's turn)</p>
<p>Board after breadth-axis mirroring</p>  <p>[K07,E03,D06,A08]x[H02,H08,F08,C10] (Red player's turn)</p>	<p>Board after both axis mirroring</p>  <p>[H01,F04,C05,G11]x[J03,H06,H08,B08] (Red player's turn)</p>

Figure 12: Symmetrical variants of a boardposition

3 CREATING A COMPUTER PLAYER

When a computer player is designed, several decisions should be made regarding different areas. These areas can roughly be divided into three separate parts:

- *Representation*
- *Search algorithm*
- *Evaluation function*

The choices and options in the areas that were investigated are described in the following sections.

3.1 REPRESENTATION

The first kind of decisions that has be made when creating a computer player is about the representation of the game.

For Cover, there was no efficient representation known yet, so one of the first things that had to be done was the creation. The result of this research was the YxY diagram, which covers all locations on the board. This representation also was the basis of the diagram-notation as described in chapter 1.

Some of the important properties of the board are described below, together with the method of deriving them from the diagram:

- *Size*

The size of the board depends on the variant of Cover, i.e. the number of stones. The size of the board of Cover-X is $Y \times Y$ where $Y = 2X + 3$.

Example: Cover-4 is played on an 11x11 board.

- *Locations*

Each location on the board is represented by a field in the diagram and each field in the diagram represents a location.

- *Direction of a location*

Adding the coordinates of a location gives the direction of a location. If the sum is odd the direction is horizontal, if the sum is even the direction is vertical.

Examples: A03 \approx (1,3) is vertical, D07 \approx (4,7) is horizontal

- *Side of the board*

The first coordinate of a location on the left side of the board is larger than the second coordinate of the location. The opposite is true for location on the right side of the board.

Examples: The location C06 \approx (3,6) is on the right side, and the location I02 \approx (9,2) is on the left side.

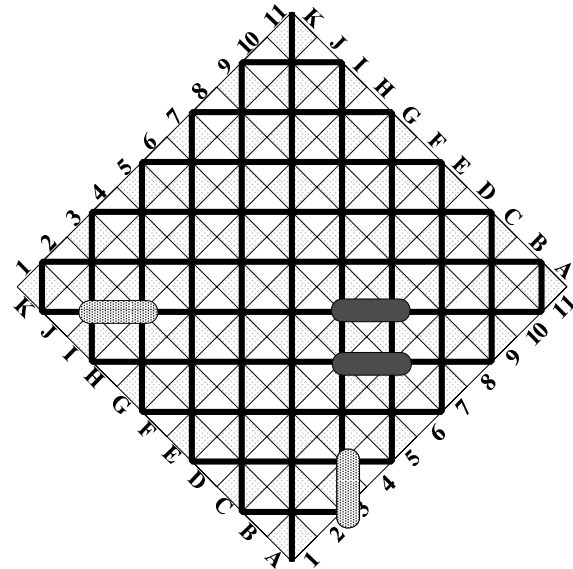


Figure 13: Properties of the board of Cover

- *Neighbor locations*

The neighbors of a location on the board are the neighbors in the diagram. This means that the locations to which a stone on a location can be moved to are very easy to determine because they are the neighbors in the diagram.

Example: Neighbors of C06 \approx (3,6) are $\{(3,5) \approx C05, (3,7) \approx C07, (2,6) \approx B06, (4,6) \approx D06, (4,7) \approx D07, (2,5) \approx B05, (4,5) \approx D05, (2,7) \approx B07\}$

- *Overlapping locations*

Overlapping locations on which no other stone may be located are also very easy to detect in the diagram since they are a subset of the neighbors of a location. For vertical locations, the neighbors with delta (1,-1) and (-1,1) are not overlapping and for horizontal locations the neighbors with delta (1,1) and (-1, -1) are not overlapping.

Example: In the example above C06 \approx (3,6) is a horizontal location and therefore (2,5) \approx B05 and (4,7) \approx D07 are the non-overlapping neighbors of C06.

- *Distance to homebase*

The distance to the blue homebase is the maximum of the two coordinates minus 1 and represents the number of moves needed to reach the homebase. The distance to the red homebase is Y minus the minimum of the two coordinates, where Y is the size of the diagram YxY.

Example: The distance of C06 to the blue homebase is $\max(3,6)-1 = 5$.

The distance of I02 to the red homebase is $11-\min(9,2) = 9$.

- *Horizontal lines*

The board of Cover has several horizontal lines. With the diagram representation, it is easy to calculate on which line a location is located by just adding the coordinates. Only the locations with an odd horizontal line are located on an existing line. Even horizontal lines are the imaginary lines between two existing horizontal lines.

Examples: The horizontal line of D07 $\approx (4,7)$ is 11. The horizontal line of I02 $\approx (9,2)$ is also 11. The locations are therefore located on the same horizontal line. The horizontal line of C06 $\approx (3,6)$ is 9 which means that this location is lower on the board than D07 and I02.

- *Vertical lines*

The board of Cover has several vertical lines. With the diagram representation, it is easy to calculate on which vertical line a location is located by taking the absolute value of the subtraction of the first coordinate from the second coordinate. Only the locations with an even vertical line are located on an existing line. Odd vertical lines are the imaginary lines between two existing vertical lines.

Examples: The vertical line of C06 $\approx (3,6)$ is 3. The vertical line of D07 $\approx (4,7)$ is also 3. The locations are thus located on the same vertical line. The vertical line of A03 $\approx (1,3)$ is 2 which indicates, since it is absolute smaller, that this location is closer to the center line of the board than the locations C06 and D07.

- *Shooting lines*

The shooting lines of stone on a horizontal location are locations which lie on a vertical line that is one more or one less than the vertical line of the stone itself. Vice versa for stones on a vertical line.

Examples: A03 is a vertical location on the horizontal line $3-1 = 3$. Therefore it can shoot stones on the horizontal lines 2 and 4. Stones C06 and D07 are located on the horizontal line 4 and are therefore on the shooting line of A03. Only the stone C06 can be shot.

For efficiency reasons an extra data structure in the form of an array of stones was kept which enabled easy determination of the location of the stones. Together with the diagram, they are the game representation that was used during the investigation. See also appendix E.

3.2 SEARCH ALGORITHM

Several game-tree search algorithms can be used for traversing the state-space. The most important candidate seems to be alphabeta so the other techniques are compared to alphabeta for their suitability for using for Cover.

ALPHABETA SEARCH

One of the most used search algorithms is alphabeta search (Knuth 1975). Alphabeta has many advantages, the little amount of internal memory required, many optimization techniques and its general suitability. A disadvantage is that a lot of optimization is needed to make it efficient. Still alphabeta seems the most suitable and was therefore chosen as the best algorithm in this stage of the research of Cover.

Standard alphabeta search is based on the following technique. The game-tree is traversed in a depth first order for a fixed depth. Each time, when the fixed depth is reached, the evaluation function is called to calculate the evaluation value for the current position. During the traversal of the game-tree two values are maintained, the lowerbound α and the upperbound β of the evaluation values of the currently traversed tree. The evaluation function returns values from the viewpoint of one of the two players. One player tries to reach a position with a high evaluation value, while the other tries to reach a position with a low evaluation value. By maintaining the limits of the evaluation values, cut-off in the traversal of the game-tree can be made. When at a certain point of traversal, an evaluation value is retrieved for a position which is higher than the current maximum, the player with the goal to maximize the evaluation value will know that the other player will decide to take another direction in the tree on a higher level. Further examination of the tree in this level is therefore useless.

```
procedure AlphaBeta(depth, alpha, beta)
  if ((depth = 0) OR (IsEndGame())) then
    return Evaluation(depth); /* Search no deeper */

  /* Alphabet */
  help = alpha;
  GenerateMoveList();

  for all moves do
    aMove = GetMove(s); /* Get Move x */
    if (not IsIllegalMove(aMove)) then /* If legal move */
      PlayMove(aMove);

      temp = -AlphaBeta(depth, -beta, -help);
      UndoMove();

      if (temp > help) then
        help = temp; /* New Best Move */
        AddPath(aMove);

      if (help >= beta) then
        return help; /* Cut-Off */

  return help;
```

Figure 14: Pseudocode of standard alphabeta

Figure 15 shows how a game-tree is traversed with alphabeta search. The first player is the player who tries to maximize the evaluation value.

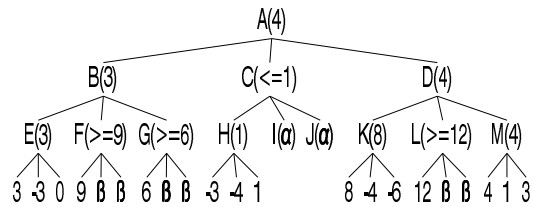


Figure 15: Alphabeta search on a game tree

By the time that the children of E are investigated, B knows that the value 3 can be reached by choosing E. When the first child of F is investigated, F knows

that its final value will be greater or equal than 9. But since B is a minimizer, B will prefer E over F. So further examination of F is useless. The same for G. By the time that the children of H have been investigated C knows that its value will be smaller or equal than 1. But since A is a maximizer, A will prefer B over C, and again further examination is pointless. Finally after all options have been investigated A knows that the best option is choosing D. D will choose M and M will choose the child with the value 4.

ALTERNATIVE SEARCH TECHNIQUES

The following other algorithms have been researched for their suitability for searching the state-space of Cover but have been rejected.

- *Best-first search*

Many best-first search algorithms have been developed over the years. Some examples are SSS* (Stockman 79), A* and B* (Berliner 79). All these techniques require a large amount of internal memory and some knowledge about the game. Since there is only little knowledge available of Cover and the branching factor of Cover being rather high, these technique seem to be less suitable.

- *Proof/Conspiracy-number search*

Conspiracy-number search (McAllester 88) and proof-number search (Allis 94) are both rather new techniques which try to find the theoretical value of the game-tree. They have the main advantage of being very efficient on non-uniform game-trees. The large amount of memory necessary is a disadvantage. These techniques do not seem to be very suitable either, since the game-tree of Cover is very uniform.

- *Dependency-based search*

Dependency-based search (Allis 94) is efficient in games with sequences of threats. This technique does not seem very suitable either, since Cover does not have many game winning threats.

Note that the research for an appropriate search technique was done in an early stage of the research on Cover. Techniques that were not suitable at this early stage may be useful when more information about Cover is found. An example is proof-number search. In chapter 4, techniques will be described that make the game-tree more irregular. With these techniques, proof-number search might be successful. See also the conclusions and recommendations at the end of this report.

3.3 SEARCH OPTIMIZATION

Many optimization techniques are known for the standard alphabeta search. Some of the most common ones are described by Winston (Winston 77), Rich and Knight (Rich 91) and Van den Herik (Herik 85). The following section describes the optimization techniques that have been researched.

QUIESCENCE SEARCH

Evaluation functions can only give a reliable result when a position is quiet, that means that no threats are available. One way to accomplish this is with quiescence search. Quiescence search is a method of continuing the search until a quiet position is found. This makes the examined game-tree more irregular like the figure 16 shows.

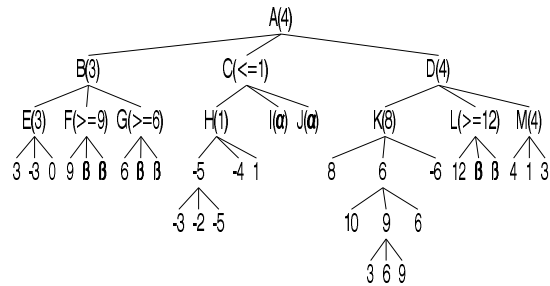


Figure 16: Quiescence search

Quiescence is used in the following places of the implementation of the computer player:

- *Breakthrough*

When a stone has just broken through, it might still be hit in the next turn. So one move search deepening was performed to check if the breakthrough was real.

- *Shooting a stone*

When one player has just shot a stone it might be that the other player strikes back in the next turn. To check whether that is the case the search was deepened with one move.

ORDERING OF THE MOVE LIST

The ordering of the moves is of major importance in the alphabeta algorithm, since the cut-offs will be bigger if the best moves are examined first. Figure 17 shows that a better ordering causes less nodes to be evaluated, the figure uses the same game-tree as shown in figure 15. In figure 15 only two subtrees were cut-off, where in figure 17, four subtrees are cut-off.

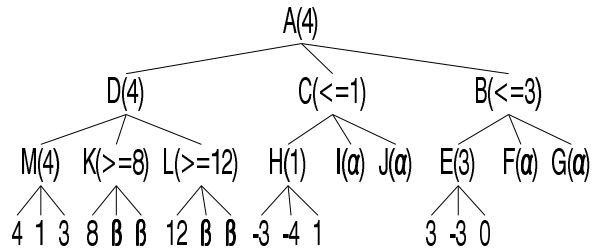


Figure 17: The influence of the ordering of moves on alphabeta search

For Cover, the order of the moves is determined by looking at the goal of the game. It is obvious to see that forward moves are better than backward moves since the stones will be closer to their objective. Also moves that may strike another stone can create major cut-offs and should be examined in an early stage.

Taking all this information together creates the following order of the moves:

- *Vertical stones*
 Forward, backwards, left, right, rotations
- *Horizontal stones*
 Left, right, forward, backwards, rotations

The directions are relative to the board as described in chapter 1.

Apart from the order of the moves, the order of the stones may be important. The following order was chosen: the first stone to examine is the last moved stone since that stone was the best to move the last time. The remaining stones are examined in a non-specific order.

One decision about the order remains being the order between the stones and moves per stone. Should all moves of one stone be examined first before examining the moves of the next stone or should the moves be ordered by direction. So first, the first move of all the stones than the second, etc. The second option was chosen because, in this way, the striking moves are examined first.

PRINCIPAL CONTINUATION

After a search algorithm has determined the best move, it contains information over the complete best path in the game-tree to a certain depth. Using this path to create killer moves is called principal continuation.

For Cover, a variant on principal continuation was used. Instead of only using the best path at the next turn, it was also used in every sub-tree during the search, hoping that the best move, the last time will be a killer move this time. This was implemented in the ordering of the move list. The ordering causes the best move of the previous search to become the first move to be evaluated in the current search.

NULL MOVE

The null-move strategy (Goetsch 1990) is a kind of forward pruning. It is based on the idea of a player skipping a turn to see what the opponent is going to do and to examine if the opponent has a threat.

With the null-move strategy, a player makes one more move to see if the position of the opponent is worse. If the position didn't become worse, the original move was not very well and further examination of this game line seems useless.

This technique was used with the variant as described by Plenker (Plenker 1995). This variant uses a verification for the situation in which zugzwang occurs. When a position is cut-off by the null-move strategy, a smaller alphabeta search is performed to check if the cut-off was correct, because it might be the case that the null-move gave misleading information. This can be the case, because, although illegal, making no move could be the best possible move and was therefore found as best move. The implementation of zugzwang is simple and is shown in figure 18.

```

procedure AlphaBeta(depth, alpha, beta)
  if ((depth = 0) OR (IsEndGame())) then
    return Evaluation(depth); /* Search no deeper */
  /* Null Move */
  if (depth > TEMPO) then
    temp = -AlphaBeta(depth - TEMPO - 1, -beta, -beta+1);
    if (temp >= beta) then /* Zugzwang Verification */
      temp = AlphaBeta(depth - TEMPO - 1, beta-1, beta);
      if (temp >= beta) then
        return temp; /* Verified Ok */
  /* Alphabet */
  help = alpha;
  GenerateOrderedMoveList();

  for all moves do
    aMove = GetMove(s); /* Get Move x */
    if (not IsIllegalMove(aMove)) then /* If legal move */
      PlayMove(aMove);

      if (strikemove ) then /* Quiescence */
        dodepth=1;
      else
        dodepth=depth-1;

      temp = -AlphaBeta(dodepth, -beta, -help);
      UndoMove();

      if (temp > help) then
        help = temp; /* New Best Move */
        AddPath(aMove);

      if (help >= beta) then
        return help; /* Cut-Off */

  return help;

```

Figure 18: Pseudocode of optimized alphabeta

OTHER OPTIMIZATION TECHNIQUES

The following optimization techniques were investigated but rejected:

- *Focus neural networks*

A focus neural network is a network that determines which moves should be examined in the game-tree (Moriarty 1995). Such a network gets the current position as input. The problem with Cover is that the current position in Cover is very complex because the history of a game is a part of the current position. The neural network should therefore be very large to learn this and that means a lot of training. Forgetting the history leads to extra problems since the network must give different answers for the same input, which is of course impossible. These outputs must therefore be regarded as noise by the network. The network will need extra training to learn to recognize this noise. Also because Cover without the history-rule is a clear draw, a network trained without the history will most likely be training to play for the draw. It therefore seems that this kind of a neural network approach is not very suitable for Cover.

- *Killer heuristic*

Some moves create a lot of cut-offs in the game-tree. These moves are called killers. These killers can be used in the future to cut-off the game-tree again. This is called a killer heuristic. Since in the current implementation, the principal continuation was expanded to a kind of killer heuristic, it seemed unnecessary to create an additional killer heuristic.

- *Iterative deepening*

Iterative deepening is the system that uses the alphabeta algorithm for a better arrangement of the moves. Before each alphabeta search, a search halve the depth is performed to order the list of moves. The main disadvantage is that it is not always faster than the normal alphabeta search. Furthermore, there was no need yet to control the time spend thinking. Therefore this technique was not used for Cover.

3.4 EVALUATION FUNCTION AND GENETIC OPTIMIZING

An evaluation function is a function which gives a valuation to a position from the viewpoint of one player. The goal of an evaluation function is to predict the outcome of a game given a position. Since this is impossible without knowing the theoretical value of the game, the outcome is an approximation with the help of properties of the current position. The properties on which the evaluation value of a position is based can be divided into two categories:

- *Primary properties*

Properties which are based on one element of a position. E.g. a stone.

- *Secondary properties*

Properties which are based on multiple elements of a position and their relations. E.g. a breakthrough.

In the following sections, some primary and secondary properties of a position that were used in the evaluation function of Cover will be discussed.

PRIMARY PROPERTIES

The following properties are primary properties and were used as part of the evaluation function:

- *Stones*

The more stones a player has, the better the position of the player.

- *Distance*

The number of steps needed to reach the opponent's homebase per stone. The closer the stones are to the homebase of the opponent, the closer the player is to the goal of the game.

- *Horizontal distance*

The number of horizontal lines to the opponent's homebase. Since the rules of Cover allow the hitting of stones, it is good to threat important locations so that the opponent cannot place its stones on these locations.

- *Vertical distance*

The number of vertical lines to the center of the board. The center of the board is the straight line between both the homebases. This line is also the shortest route to a homebase across the board. For this reason, it is important to cover this line.

- *Location threats*

The number of locations that are attacked. The more locations that are covered by a stone the better its position.

SECONDARY PROPERTIES

The following secondary properties could be used as part of the evaluation function:

- *Stone threats*

The number of stones that are attacked. Threatened stones are about to be lost and must be defended. This reduces the opponent in its movability.

- *Breakthrough*

A stone has a straight path to the opponent's homebase. See chapter 4.

- *Sure captures*

Pattern of stones that leads to the capturing of a stone without the opponent being able to prevent it. The example in figure 19 shows two general variations. The blue stone on J04 can strike both the stone on K06 and H03. Even if it is the red player's turn, the red player cannot prevent the loss of a stone except by shooting the blue stone. The second

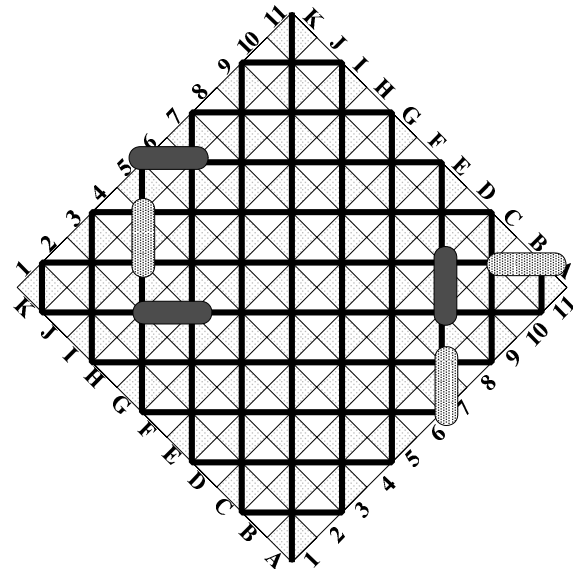


Figure 19: Sure stone captures

situation concerns the blue stones on A07 and B11 and the red stone on C09. The red stone is lost if the red player cannot strike the stone on B11.

GENETIC OPTIMIZING

The different evaluation function properties are multiplied with the evaluation parameters. The sums of these multiplications form the evaluation value of a position. The quality of the evaluation can be upgraded by optimizing the evaluation parameters.

The evaluation parameters for Cover were optimized with a genetic algorithm (Goldberg 1989). Ten sets of evaluation parameters were put together as players into a pool. Each player played twice against each other player, each player starting once. The

winning player got two points the losing player got no points. In case of a draw, both players got one point. The points gained by a player after playing the complete competition was the fitness value.

After playing, two of these players were selected for mutation and replacement. This is done as follows: Two players are picked with the roulette wheel selection method (Michalewicz 94). Two random numbers were generated between 1 and 180. (180 is the total number of points that can be gained by the players, 10 players playing 2 times against the 9 other players, with 2 points to give away per game, but since each game contained two players, the total value must be divided by 2). Per random number, a player is selected if it is the weakest player of which the sum of the fitness values of the player and all weaker players is higher than the random number.

Ten players played their matches against each other resulting in the following table.

Player	1	2	3	4	5	6	7	8	9	10
Fitness	3	4	5	14	16	20	22	30	30	36
Sum	3	7	12	26	42	62	84	114	144	180

The 2 random numbers are 94 and 152.

For number 94, player 8 is selected since 114 is the lowest sum which is bigger than 94.
 For number 152, player 10 is selected since 180 is the lowest sum which is bigger than 152.

Player 8 and 10 are selected.

Figure 20: Selection of players in a genetic algorithm

The values of the new players were created as follows:

The new player 1 gets all the parameters with an odd order from the selected player 1 and all the even ordered from the selected player 2. Vice versa for the new player 2. After the recombination of the parameters, the parameters are changed by adding a uniformly chosen

number drawn from an interval between -10% and +10% of the parameter value. In genetic algorithms, this is called mutation. In figure 21, an example is given of the recombination of players.

Parameter	1	2	3	4	5
Selected player 1	10	16	50	40	9000
Selected player 2	21	3	55	123	2000
New player 1 before mutation	10	3	50	123	9000
New player 2 before mutation	21	16	55	40	2000
Random mutation player 1	0	0	-3	-8	231
Random mutation player 2	1	0	1	-2	-20
New player 1	11	3	51	121	9231
New player 2	22	16	56	38	1980

Figure 21: Genetic creation of evaluation parameters

The genetic algorithm was stopped when the corresponding parameters of all the players lie within 10%.

Executing the genetic algorithm on the evaluation parameters resulted in the following evaluation values. A full breakthrough and a winning sequence found with the "One vs One" endgame database are regarded as a winning position, see also chapter 4. The points are calculated for both the players and subtracted. The result must be negated to retrieve the evaluation value as seen from the viewpoint of the other player.

Maximum in the table means the maximum number of lines or the maximum distance.

Winning position	9000
Number of stones	250 per stone
Location threats	18 per location
Horizontal distance	15 * (maximum - steps)
Vertical distance	15 * (maximum - steps)
Line breakthrough	13 * (maximum - lines)
Distance breakthrough	20 * (maximum - steps)
Depth	1 per search ply

Figure 22: Optimized evaluation parameters

3.5 USING NEURAL NETWORKS

In the last few years, several good computer players were created with the help of neural networks. Some examples are TD-gammon which plays backgammon (Tesauro 1993) and an implementation of Othello created by Moriarty (Moriarty 1995). In this chapter, the possibilities to use neural networks for creating a computer player for Cover are described.

TECHNIQUES

The most common techniques to use neural networks in the game-theory are as:

- *Evaluation function*

The network is used as evaluation function which can be used during the search in the game-tree. This technique is used in TD-gammon. The main disadvantage of using a neural network as evaluation function is the fact that it must generate a real value instead of a boolean value, since the search algorithm must be able to compare different positions. The results of such networks have been discouraging so far.

- *Focus network*

The network now determines which nodes of the game-tree should be pruned and which nodes should be examined further. This technique is used by Moriarty. The main advantage except gaining time is the fact that the neural network may learn the mistakes of the evaluation function. A good neural network excludes the moves which lead to these situations.

NEURAL NETWORKS FOR COVER

Both the techniques as described in the previous section have one thing in common: they use the current position as input for the network. This fact is also the main problem for using neural networks in Cover. It is not very easy to define the current position in simple terms in Cover, since the history of the game is of major importance for the current position. Normally, the input of a network consists of boolean values like whether or not a location is occupied. For the history, there is no simple solution to represent it in boolean values.

It seems to be difficult to feed the complete position as input to the neural network. One possibility is to feed only the current board-position to the network and forgetting about the history of the game. This is only possible for the focus-network technique because an evaluation function needs the complete position information. The focus network will have to interpret the different output possibilities as noise. But even if the history is forgotten, the input layer still contains at least 242 nodes, 121 locations times 2 colors. Per location and color, the input value of the network represents whether a location is or is not occupied by a stone of the color red/blue.

4 ENDGAME OF COVER

One of the most common used tactics to increase the performance of a computer player is with the help of endgame strategies (Nelson 1993). An endgame is a stadium of a game in which game is almost finished. Teaching a computer player endgame strategies can be done by giving the player extra knowledge of how to recognize an endgame and how it should play when an endgame stadium is reached.

The following two endgame stadia have been investigated:

- *One vs One*

In this stadium, each player has only one stone left.

- *Breakthrough*

A player has a straight path to the opponent's homebase. The player can take the shortest route to this homebase without the opponent being able to prevent it.

Below, these two endgame stadia will be described more thoroughly, together with the method of how they were used in the implementation in the computer player of Cover.

4.1 ONE VS ONE

The typical endgame to analyze for Cover, although rare, is the "One vs One". In this endgame, each player has only one stone left. This situation is rare because most of the times when Cover is played, the game is decided before this stadium is reached.

The recognition of this endgame stadium is trivial, since each player has only one stone. The strategy of how to play is more difficult and needed more research. The result of this research is described below.

DEVELOPING A PLAYING STRATEGY

The most common way to define the playing strategy of an endgame is to build up a database with all positions of the endgame. Per position is, for instance, the best move to be played listed in the database. When this stadium is reached, the computer player can look up the current position in the database to see which move it has to play. The creation of a database seemed also the best way to define the playing strategy of "One vs One", since there is no simple playing strategy.

DESIGNING THE END-GAME DATABASE

The first problem of the creation of an end-game database for Cover is that the history of a game is part of the current position. The endgame database should give the playing strategy for all positions, but since identical boardpositions with a different history can give different results, they should both be listed in the database or another solution should be found. Listing all possible positions including the ones that only differ for their history seems a waste of space and effort, since this would generate a lot of positions which most of the time can be treated identically. Therefore, other techniques were investigated which treated positions with identical boardposition but with a different history, as equal as possible.

The solution that was used, was forgetting the exact history and only storing in the database if the history was used. When accessing the database, the computer player can now recognize the positions for which the history will be important and can verify the

information of the database. The best possible move as recommended by the database might not be possible anymore or could be illegal because of the history of the current game.

The next problem is: which data should be stored in the database, knowing that it must be enough to determine the best move to play per boardposition. Just storing one move is not enough since the best move may differ because of difference of the history of a boardposition. The solution used, was to store the number of moves till the end of the game. From each position, the next move to play can now be found by performing a one move deep search, one of the now reached positions should have the same game result but it is reached in one move less.

With the technique above, the following information should be registered per boardposition:

- *History importance*

The flag which indicates if the history of the game is important.

- *Result*

The outcome of the game if played optimally, either blue or red wins or it is a draw.

- *Depth*

The number of moves till the game is finished.

With this information and technique, only one problem remains: the recognition of the moments when the history is important. This was solved by the way the database was created.

CREATING THE DATABASE

With the use of the above technique, the number of possible positions to store in the database is reduced since the history can be forgotten. The number is equal to the number of unique boardposition times the number of possible players to move. Each stone can be placed on 121 locations, and so an upperbound of $121 \times 120 \times 2 = 14520$ is found. With the help of the symmetrical variant of the Cover board, this number can be reduced to about 8000 positions. Also, the current player of a position can be extracted from the information to store with the symmetrical variants. The reductions were implemented by always converting a board to its symmetrical variant in which the blue player has to move first and with the blue stone located on the right side on the board. With these reductions, only a small amount of positions remains and so every position can be given an absolute location in the database.

The database was created with a bottom up method which traversed all legal positions with one stone each. First all the positions which were won by one of the players were listed in the database. The now solved positions are positions in which one of the players has captured the homebase of the opponent.

Now all positions which reached another winning listed position within one move were put in the database. This step was done until no more newly solved positions were found.

Example: at some point, the position [I09]x[J08] with red to move will be solved since red has no move which prevents blue from winning. Blue can always make a move which is closer to winning than the current position. For example: J08-K08, I09-I10. After this move, the current position is [I10]x[K08] which was already in the database. The next step for blue will be to move to J11. Finally blue will move to K11, at which point it will have won the game.

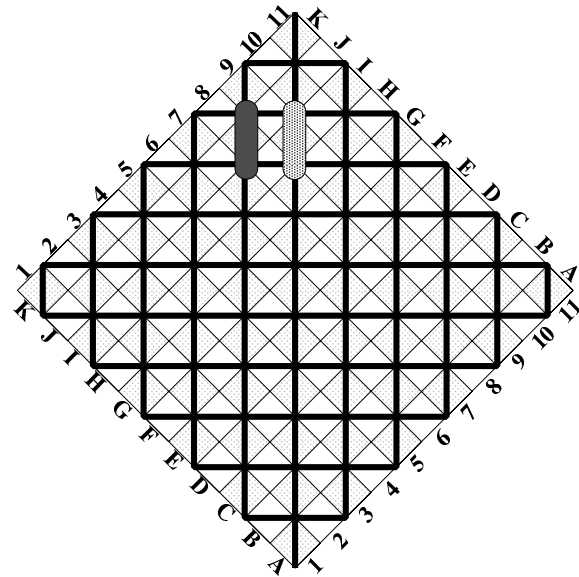


Figure 23: An "One v One" endgame position: [I09]x[J08], blue wins

All positions that were found after these steps did not use the history of the game to win and will not need the history in the future to win the game. The next step is searching for positions which will be solved after a four move search. These positions use the history and should be marked as using the history.

An example of such a position is [H09]x[I10]. After a four move search, the position [H09]x[J10] is found as optimum. The blue stone moves to I08. Red moves to J09. Blue moves back to H09. After this move, red is not allowed to move back to I10 and must therefore move to J10. The position [H09]x[J10] is

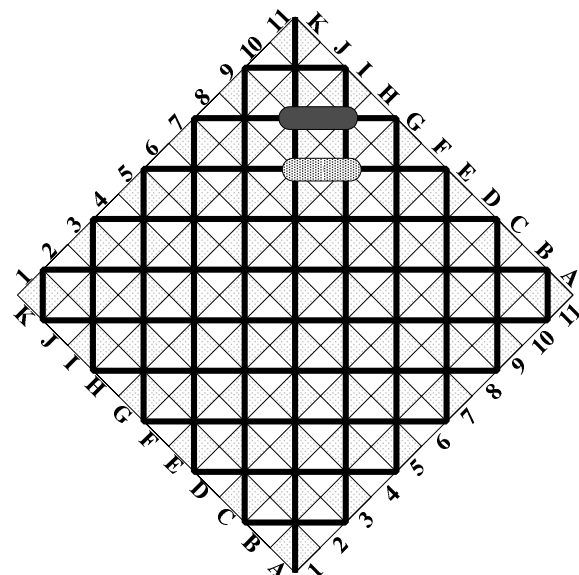


Figure 24: A complex "One vs One" endgame position: [H09]x[I10], blue wins

a position which is already in the database as won by blue and therefore the position [H09]x[I10] can be put in the database as won by blue.

After no more positions are found, the remaining positions can be marked as a draw, since all solved positions are found.

USING THE DATABASE

The created database is now used as follows:

When a computer player has to move, it looks up the current position in the end-game database. If it finds that the history is important or the game is a draw, it searches four move deep. When the position is a draw, the 4-ply search is performed in the hope to find a position that leads to winning the game. Such a position may be found because of the history of the current position, which might forbid the opponent to play the optimal move. When the current position uses the history, it should be checked with this 4-ply search if there is still a solution for the current position. If the history was not important a one move search can be performed to find the best move.

Perform the alphabeta search for the given move depth. The evaluation function searches the database with the new current position. If, in this position, the history is important, the evaluation will perform a new four move search to verify the answer. Otherwise, the retrieved value for the achieved position is correct and can be returned.

This process ends because the number of steps to the end of the game is a fixed number which will only become smaller. By using the moves to go till the end of the game, in the evaluation function result, the alphabeta window can be used to search. E.g. the current position needs 17 moves to the end of the game. Searching one move deep must find positions which need 16 moves to the end.

Figure 24 shows the position [H10]x[J10] with blue to move. This position is won by blue in 13 moves. Blue will however need the boardposition repetition rule to win the game.

An example of an optimal game play would look as follows:

H10-I11, J10-I09
 I11-H10, I09-J09
 H10-H11, J09-K10
 H11-I10, K10-J11
 I10-I09, J11-I11
 I09-J10, I11-H10
 J10-K11

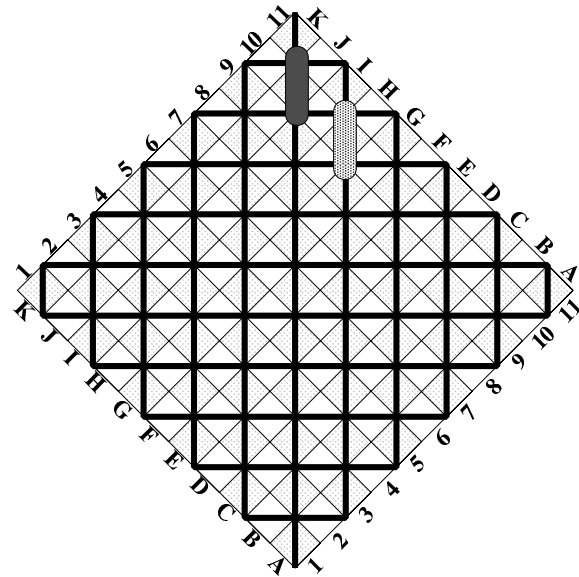


Figure 25: An "One vs One" endgame position: [H10]x[J10], blue wins

Searching the game-tree with four move deep search results in finding the position [H10]x[J09] with blue to move which is won in 9 without using the history, so there is no need to verify the found position. If the history was important a new four move search was started from position [H10]x[J09] to see if it found a position which was won by blue in 5 moves.

RESULTS OF THE "ONE VS ONE" DATABASE

The results are listed in the form of board diagrams. These diagrams represent multiple boardpositions. In these boardpositions, the blue player can have multiple stones. The red player has always only one stone.

The diagram can be seen as the board on which Cover is played. The diagram has the same size as the Cover board. The locations in the diagram are the same as the locations on the board, but rotated 45 degrees counter-clockwise. The diagram represents the board

from the viewpoint of the blue player. The blue player is also the first player to move. This can be done without the loss of any information because a boardposition has a functional identical symmetrical variant with the red player to move. See also chapter 2.2.

An example of such a board diagram is shown in figure 26.

A board diagram shows the result of the positions with one blue stone and one red stone where the location of the blue stone is fixed. The location of the red stone varies and can be on any legal location.

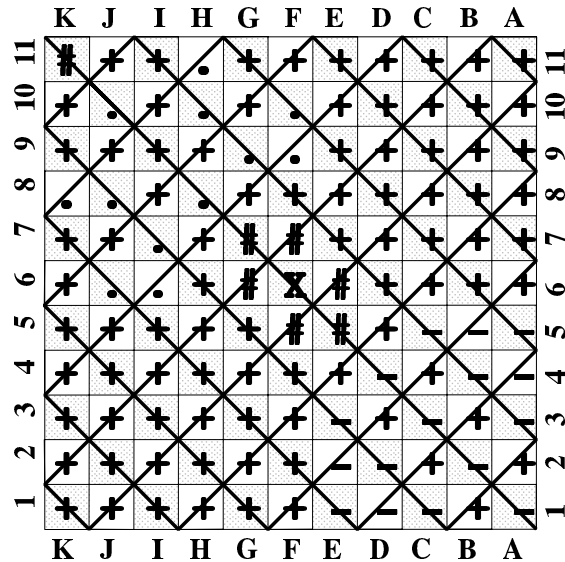


Figure 26: An "One vs One" endgame diagram

The following signs are used in the diagram:

- *X* = The location of the blue stone
- *#* = A location where no stone may be located.
- *+* = If the red stone is located here then the blue player can win.
- *-* = If the red stone is located here then the red player can win.
- *.* = If the red stone is located here then the game is a draw.

The diagram should be read as follows. The blue player has a stone on the location marked with an X, in the example F06. The sign on a certain location L is the result of the position where the red player occupies location L and the blue player the location marked with an X. E.g. The position [F06]x[H08] is a draw since H08 is marked with an ".". In appendix B, the board diagrams can be found of the "One vs One" database.

STATISTICS

In this section, some statistics are given about the "One vs One" database.

Estimated positions: (see chapter 2.1)	27768	Positions won by blue	9075
Legal positions:	27294	Positions won by red	9075
Different symmetrical variants: (see chapter 2.2)	6860	Positions draw	9144
Longest game line:			38

Figure 27: Statistics on the "One vs One" endgame database

An example of a position with an optimal game line of 38 is [A08]x[A11] with red to move. Below the position is shown together an example of the game line.

MOVES

```

-----, A11-A10
A08-A07, A10-A11
A07-B08, A11-B11
B08-C08, B11-C10
C08-C07, C10-C11
C07-D08, C11-D11
D08-E08, D11-E10
E08-E07, E10-E11
E07-F08, E11-F11
F08-G08, F11-G10
G08-G07, G10-G11
G07-H08, G11-H11
H08-H09, H11-I10
H09-I08, I10-J09
I08-H09, J09-J10
H09-H08, J10-I11
H08-I09, I11-H11
I09-J09, H11-G10
J09-K10, G10-F11
K10-K11
    
```

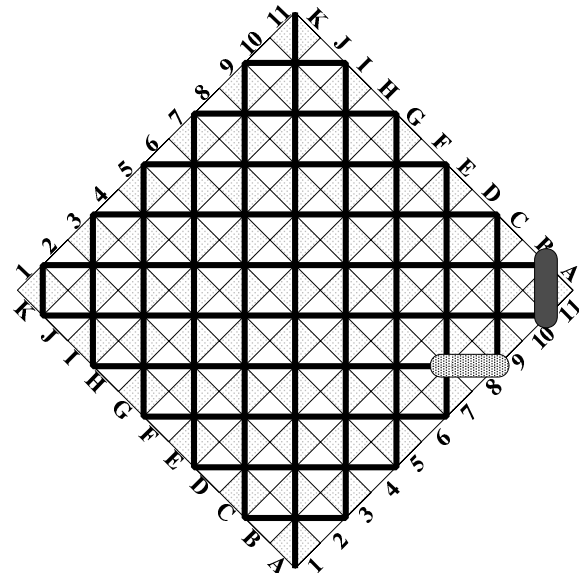


Figure 28: Position [A08]x[A11] with red to move is won by blue after 38 moves.

4.2 BREAKTHROUGH

Another endgame stadium is when one of the players has a stone which is unreachable for the stones of the other player and can walk to the homebase of the opponent without any obstruction. This stadium is called a breakthrough, since one of the stones has broken through the defence of the opponent. The problem with the "One vs One" endgame was the playing strategy, while the recognition of the stadium was easy. For a breakthrough, it is the other way around. The problem is now the recognition of the stadium. The playing-strategy is simple because the computer player can follow the shortest path to the homebase.

CREATING THE BREAKTHROUGH DATABASE

The creation of the breakthrough database was similar to the creation of the "One vs One" database. For every combination of location and stone color, the set of locations had to be found with the property that, if none of these locations is occupied by the opponent, there is a breakthrough.

For every location on the board, the set of locations which prevent a breakthrough is determined. The history can be ignored since a stone that is broken through will never go backwards.

This database is also created in a bottom-up method. First, all direct winning position are put in the database as being a breakthrough. Next every position which can reach a position which is already in the database is added to the database. The attacking player has the restriction that the shortest route has to be taken to the opponent's homebase. That means, the player may only move forward, sideward to the center and only rotate once if the initial location of the stone was horizontal. The defender was allowed to move in any direction. Like with "One vs One" this is done until no more positions are found.

DEFINING FUNCTION OF A BREAKTHROUGH

While examining the print-out diagrams of the database a clear pattern came to sight.

Consider the following two functions.

- *LocationDistance(X, Y)*

LocationDistance is the number of steps a stone on location X has to make to reach homebase Y.

Example: In Cover-4 the locations I11, I10, I09, J09, K09 have the same locationdistance to the homebase of red.

- *HorizontalLine(X, Y)*

HorizontalLine is the number of lines between a stone on location X and homebase Y.

Example: In Cover-4 the locations D01, C02, B03, A04 have the same HorizontalLine to the homebase of blue.

The following function, defines the breakthrough of a stone on location X to the opponent's homebase Y and estimates the locations which prevent a breakthrough.

If LocationDistance(X, Y) is smaller than FieldLine(Z, Y) for every Z occupied by the opponent
and
if HorizontalLine(X, Y) is smaller than HorizontalLine(Z, Y) for every Z occupied by the
opponent then
the stone on location X is broken through.

Figure 29: Breakthrough function

The following adjustments can be made to this function to make it even more precise:

- *The stone may not be possible to shoot. This check is only necessary if the player with the possible breakthrough has not the turn*
- *For even locations X , the horizontal distance may be equal to the defenders horizontal distance if the attacker has to move.*

This function with above adjustments was used in the evaluation function to detect breakthrough for the created computer player for Cover.

RESULTS OF THE BREAKTHROUGH DATABASE

The results are listed in the form of board diagrams. These diagrams represent multiple boardpositions. In these boardpositions, the blue player can have multiple stones. The red player has always only one stone.

The diagram can be seen as the board on which Cover is played. The diagram has the same size as the Cover board. The locations in the diagram are the same as the locations on the board, but rotated 45 degrees counter-clockwise. The diagram represents the board from the viewpoint of the blue player. The blue player is also the first player to move. This can be done without the lose of any information because a boardposition has a functional identical symmetrical variant with the red player to move. See "Symmetrical variants".

An example of such a board diagram is shown in the figure at the right.

The diagrams show the set of locations which prevent a breakthrough of a stone on a certain location.

The following signs are used in the diagram:

- *X* = A location of a blue stone which might be broken through
- # = A location where no stone may be located
- + = A location that does not prevent a breakthrough if occupied by a red stone
- - = A location that does prevent a breakthrough if occupied by a red stone
- . = A location that does prevent a breakthrough but which is not covered by the extended breakthrough function.

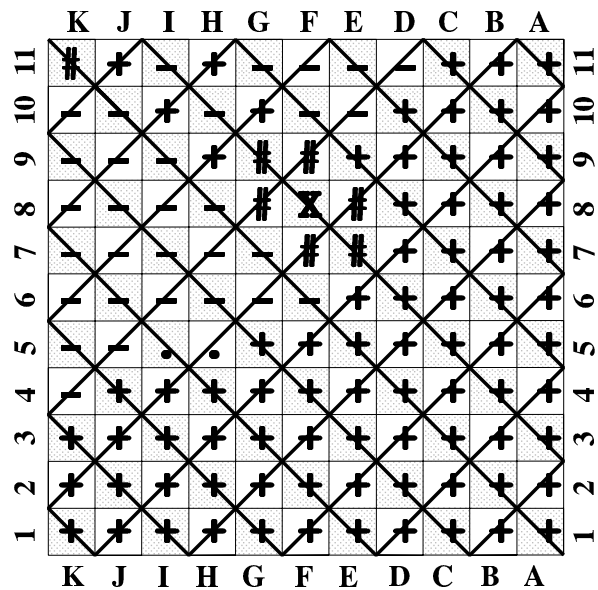


Figure 30: The breakthrough endgame diagram of F08

The diagram should be read as follows. The blue player has a stone on the location marked with an X, other blue stone are ignored. In the example this stone is located on F08. The sign on a location L is the result of the position where the red player occupies location L and the blue player the location marked with an X. E.g. The position [F08]x[F06] is not a breakthrough for the stone on F08 since the sign of F06 is -.

In appendix B, the board diagrams can be found of the breakthrough database.

5 GAME VIEWPOINTS

In this chapter, some interesting positions and strategies will be described that were found during the research on Cover.

5.1 INTERESTING POSITIONS

POSITION 1

Take the following position: Cover-4 with [H08]x[J10] and blue to move. Although blue seems to have an advantage, this position is a draw. The draw can be explained as follows:

The positions [H08]x[J10], [H11]x[J09] and [G10]x[I10] with blue to move are positions which both are in the same draw sequence. If one of the positions is reached, the other can be retrieved also by the defender. This means that the positions are a draw.

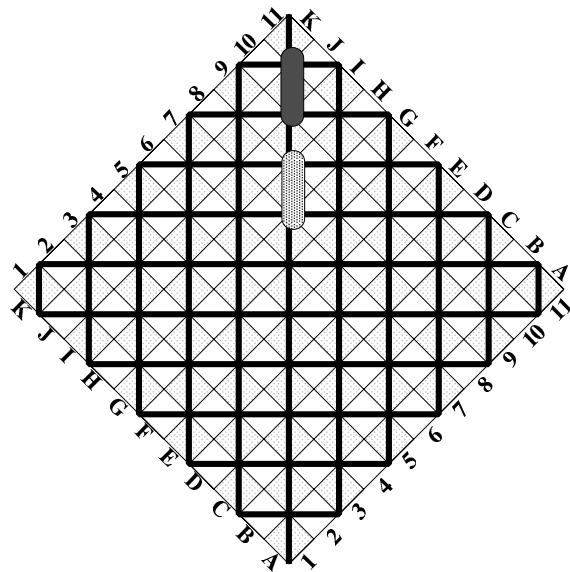


Figure 31: Position 1: [H08]x[J10]

The way to play as defender is as follows. Always keep the intersection of the positions J10, J09, I10 and I09 occupied and keep the direction the same as the attacking player. When this is not allowed or when the attacker moves backwards or to far to the side, the

defender should stay as close as possible and preferably on the same vertical line. This position is now shifted and will also be a draw.

A good variant is:

H08-G09, J10-I09
 G09-H10, I09-I10
 H10-I11, I10-I09
 I11-H11, I09-J09 (*the move of blue was forced*)
 H11-G10, J09-I09 (*the move of blue was forced again*)
 G09-G10, I09-I10 (*I09-J09 is not very wise*)
 G10-H09, I10-J09 (*the start of boardposition repetition rule*)
 H09-I08, J09-I10 (*red moved back, so blue may not*)
 I08-H08, I10-J10 (*back to the beginning situation*)

POSITION 2

Take the following position: Cover-4 with [B05,G07,H04]x[D03,F08,I02,D10] with blue to move. This position is won by the blue player, which seems odd because red has one stone more than blue and red is closer to the opponent's homebase.

The solution can be seen as follows. The first move is the most important one. Two moves seem to be given the best changes. We will try them both.

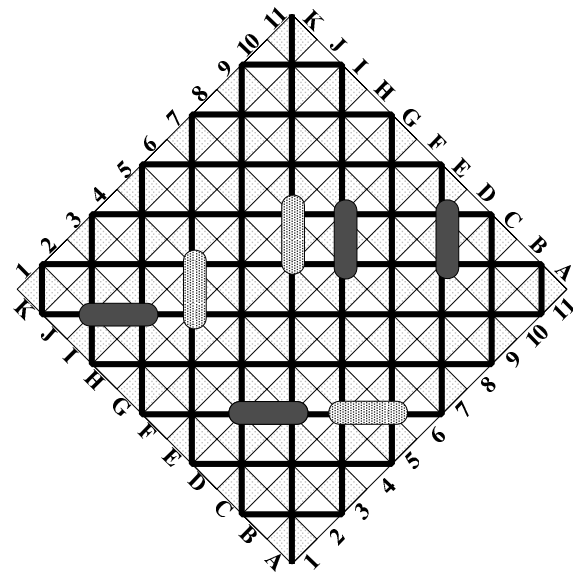


Figure 32: Position 2:
 [B05,G07,H04]x[D03,F08,I02,D10]

First, we try G07-F06 striking the threatening stone on D03.

G07-F06, F08-E07 (*Red strikes back*)
 F06-G07, D10-E10 (*Red may not move back, red threatens blue*)
 G07-H07, E07-F06 (*Blue must keep attacking, red defends*)
 H07-I06, F06-G05
 I06-J05, I02-H03 (*Blue stone hit, blue has lost for sure*)

This combination does not lead to success.

Try G07-H08 in order to make a breakthrough.

G07-H08, D03-C02 (*red attacks too, blue must strike*)
 H08-G07, D10-E11 (*red's only good move, blue cannot attack*)
 B05-C06

Since the stone E11 cannot rotate without being hit and because any other blue stone can capture the blue stone on G07, blue has won.

POSITION 3

Take the following situation: Cover-4 with [B02,I11]x[A03,J10]. Both players have one attacking stone and one defending stone. This position is clearly a draw since none of the stones can be rotated or it will be shot.

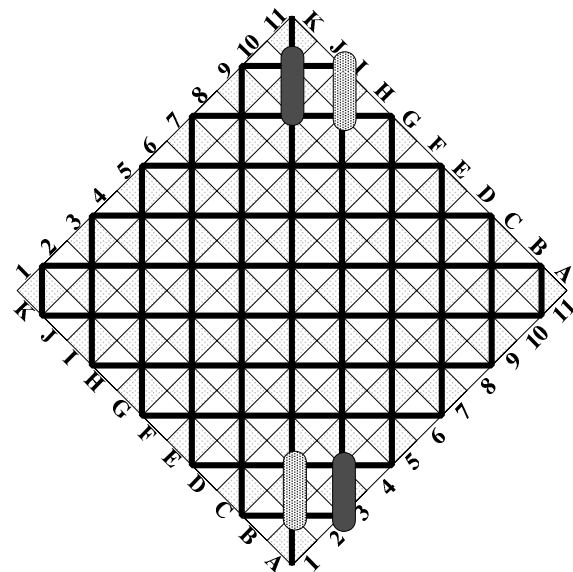


Figure 33: Position 3: [B02,I11]x[A03,J10]

POSITION 4

The following position shows that the history can be of major importance. The position is [C02]x[B05]. Blue moves first: C02-D03. The most obvious move for red now seems: B05-A04, but it is not the best move. After the move of red, the blue player will move back: D03-C02. Blue now moved back so red may not. This is a big problem for red since all other moves will put the red stone in a hitable location for blue. Red has lost.

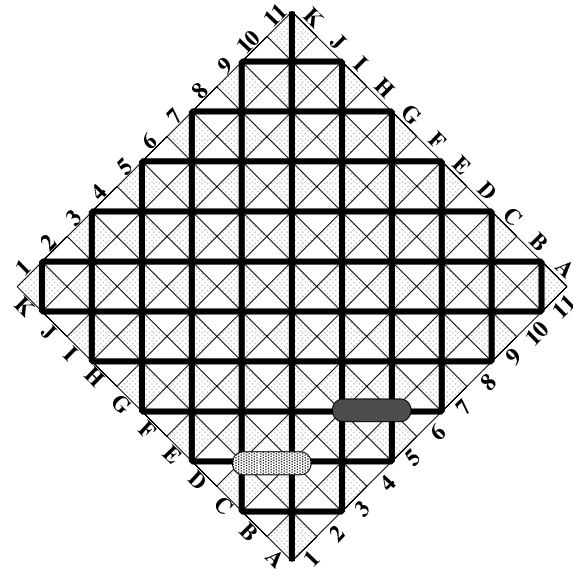


Figure 34: Position 4: [C02]x[B05]

5.2 INTERESTING STRATEGIES

During my research on Cover, some interesting strategies were discovered. The most interesting ones are described below.

KEEPING INITIAL FORMATION

The player which plays this strategy tries to keep the initial formation of her stones. Further more she tries to keep her stones on the same vertical line as her opponent and on the horizontal line. Doing so, she tries to prevent that a stone of the opponent rotates or breakthrough the line of defense.

An example of how this strategy is played is listed below. The red player is playing the described strategy.

MOVES

C05-D06, G09-F08
E03-F04, F08-G09
F05-E05, I07-H06
G01-F02, H06-I07
E05-F06, I07-H08
D06-E07, K05-J06
F06-G05, J06-K05 (*not H08-I07*)
G05-H06, J05-K04 (*Red continues*)
H06-G05, H08-I07
F02-G03, K04-K05

Red has now his old formation back.

Although this strategy seems very tight, it is my believe that it can be broken if all stones are facing right against each other. Also, the strategy seems not very suitable for the blue player, since that is the starting player.

BLOCKING THE HOMEBASE

In this strategy, the player puts three stones on the horizontal line before the homebase. For the blue player these locations are C01, B02 and A03. Doing so, the player prevents that the opponent can reach her homebase. There are a few extra advantages when playing this strategy. The three blocking stones are very hard to shoot since they can only be shot from two locations. Another advantage is that the positions becomes very hazardous for the opponent, since she is not able to rotate on the center vertical lines of the board. In other Cover variants this can even be extended by blocking an earlier line with e.g. five stones (E01, D02, C03, B04, A05).

The best strategy that I found against this strategy is striking the free stone(s) without losing stones yourself. Most of the times this can be done. After striking these stones, the blocking stones are forced to move and the position that was defended so well is not weakened.

CONCLUSIONS

The game Cover has certainly potential as a new strategic board game. The charm of the game lies in that in spite of her simple rules, it has many interesting positions and game-developments.

The rules of the game are effective enough to make a game an interesting one. The only recommendation I have about the rules is to change the boardposition repetition rule. The inventor of the game uses this eight boardpositions variant of this rule. I found that too complex to play and would propose to change this rule to the variant which only forbids the move which brings back the same boardposition as four moves ago. This modification makes the game more simple.

Another recommendation is about the notation of the game. The notation as proposed by the inventor is not very logical. I created to other notations. The diagram notation is used in this thesis and is very efficient but not very easy in use. The delta notation is less efficient but is very consistent and seems to be the best notation.

The complexity of Cover depends, naturally, on the variant of Cover. Cover-4 seems to have the same complexity as Checkers while the complexity of Cover-6 is in the neighborhood of Othello. Cover-2 has a rather low complexity, even lower than nine men morris. Since nine men morris is solved, it is expected that Cover-2 can be solved in the near future.

The complexity of Cover can be reduced by using the symmetrical variants of the Cover board. Each board has three different symmetrical variants making it together a

mathematical group 4. The variants can be created by mirroring the board in horizontal direction or in vertical direction.

Since most variants of Cover are too complex to solve, a computer player was created to play Cover. Several search algorithms were investigated for the computer player. Alphabeta search seemed the best at this stage of investigation on Cover. For alphabeta search several optimizing techniques were investigated. The most important techniques used, are advanced move ordering, null move strategy, quiescence and a variant of principal continuation.

The evaluation function that was created makes use of primary and secondary properties of a position. Some used primary properties are the number of stones and their location on the board. Secondary properties that are used are breakthrough and stone threats. The evaluation function parameters were optimized with a genetic algorithm using the roulette wheel method.

During the creation of a computer player, neural networks were investigated for their use in the evaluation function or as search optimizer. In both situations, neural networks did not seem to be a good method. The main reason for that is the fact that it is difficult to give the history of a position as input to a network in a compact and clear form. Neural networks were therefore not used in the implementation of the computer player for Cover.

Besides the research on the entire game, researches on parts of the game were performed, e.g. several endgames of Cover. The "One vs One" endgame is an endgame in which both players have only one stone. This endgame was solved and contained a lot of draws. Another researched endgame is the breakthrough. This endgame is reached when one of the players has a stone that is broken through the defence of the opponent and has a straight path to the opponent's homebase. For every possible location of a stone, the locations are registered which prevent the breakthrough of this stone. Besides a

breakthrough database, an estimation function for the breakthrough was created to define which situations are considered a breakthrough.

Using this endgame makes the game-tree more irregular because the outcome of a game can be seen earlier in the game. This opens the opportunity to use more sophisticated search techniques like proof-number search. This is my primal recommendation, trying to solve Cover with proof-number search and the use of the breakthrough database.

APPENDICES

APPENDIX A: THE ORIGIN OF COVER

This section contains the story about the development of Cover as told by N. Twigt

On a Sunday morning, I dreamt that I was playing chess with Jan Timman. It was a simultaneouslike scene and Jan played black. I looked at my position and I could not believe that it looked like I would win. I am not an extremely good player. When Timman arrived at my board again, he moved his black knight with a graceful curve to a field, a move which I had overlooked. 'Aaah', went through my mind: 'that was the move that I had not foreseen.' I felt a strange kind of relieve because the riddle of the chess position was solved and the balance of power was restored again. Then something strange happened. Timman did not put down his knight, but held it a little bit slanting above the board. The board changed into honey and the knight sucked it up like a vacuumcleaner! After this incident, I woke up. I knew that something 'diagonal' (the knight movement) was passed through to me and started to experiment with wooden sticks on a chessboard and on a draughtsboard. On the same day, I had the rules for Cover ready except for a few details.

A long time ago, I wondered if Draughts, Chess and Go were the only possible strategic games on a simple board. How would it be to invent Draughts, if it had not been invented yet? Not very special, according to my feelings. And Go. It is a good example of a game that needs a lot of time to become the 'cult' that it now is. And what if the game of Chess was not invented yet? As an inventors kick and for it is originality I find it a real 'killer'. But assume that the inventor of one of these games had been discouraged by the idea that all games were invented yet? I always kept believing the idea that there must be other games like these that were not invented yet. Because of my dream, I put the sticks immediately on two equal colored fields of the chessboard. Now, I looked at the movement possibilities and the confrontations between the stones.

I had sown about twenty pieces before I discovered that I would not need all of them. I first switched to a draughtsboard and later to a computer. With a computer, I could experiment faster with different pieces, board shapes, and board sizes. The pieces looked a bit like submarines or little animals with a head and a tale. I saw that a diagonal board gave a clear goal. The idea of a strong head and a weak flank was obvious, just like the unlimited shooting distance. The pieces do not shoot each other in a man to man fight like in former days, but they shoot each other like gunmen. In the first instance, pieces were only allowed to walk forward, but soon I found out that it was more fun to walk both forward and backwards.

My ideal, as a designer, was to bring back the game to it is utmost simplicity: as less stones and rules as possible and also an as direct game as possible. I discovered that an even number of pieces was more interesting than an odd number of pieces. With an odd number of stones, the center-line is occupied immediately from the beginning. A board with lines is also more quiet than an enlarged draughtboard with fields. Also with the notation, you will see that Cover lets itself play over the lines. The Cover pieces may have all kind of shape as long as the stones are oblong. It could be tanks or ships, but finally the most abstract and simple shape conquered. In my environment, the meanings were extremely different and distinct. One wanted angular stones while the other liked cylindrical pieces better. In the future, many different variations of Cover will be invented, just like with Chess. As for a starter, I looked for the simplicity. My goal was that the game ought to be a confrontation between people and not an event on the board. The board and the pieces should be as clear as possible and not to intrusive or turbulent. The game was not allowed to distract the players.

And so it became.

COVER: THE WORLD IN ONE GAME

```
[F09, BLUE]
# - - - - - - + + +
- - - - - # # + + + +
- - - - - # X # + + + +
- - - - - # # + + + +
- - - - - + + + + +
- - - - - + + + + +
- - . . . + + + + +
- + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
[E08, BLUE]
# - - - - - - +
- - - - - - +
- - - - - # # + + +
- - - - - # X # + + +
- - - - - # # + + + +
- - - - - - + + + +
- - - - - - + + + +
- - . . . + + + + +
- - + + + + + + + +
- + + + + + + + + +
+ + + + + + + + + +
[G11, BLUE]
# - - # X # + + + +
- - - - # # + + + +
- - - - - + + + + +
- - - - - + + + + +
- - - - - + + + + +
. . . + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
[F10, BLUE]
# - - - # # + + + +
- - - - # X # + + + +
- - - - - # # + + + +
- - - - - + + + + +
- - - - - + + + + +
- - . . . + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
[E09, BLUE]
# - - - - - - + +
- - - - - # # + + + +
- - - - - # X # + + + +
- - - - - # # + + + +
- - - - - - + + + +
- - - - - - + + + +
- - . . . + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
```

```
[D08, BLUE]
# - - - - - - - +
- - - - - - - + +
- - - - - # # + + +
- - - - - # X # + +
- - - - - # # + + +
- - - - - - + + + +
- - - - - - + + + +
- - . . . + + + + +
- + + + + + + + + +
+ + + + + + + + + +
[C07, BLUE]
# - - - - - - - -
- - - - - - - - -
- - - - - - - - +
- - - - - - # # + +
- - - - - - # X # +
- - - - - - # # + +
- - - - - - - + + +
- - - - - - - + + +
- - . . . + + + + +
- - + + + + + + + +
[B06, BLUE]
# - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - # # +
- - - - - - # X #
- - - - - - - # #
- - - - - - - - +
- - - - - - - - +
- - . . . + + + + +
[A06, BLUE]
# - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - # #
- - - - - - # X
- - - - - - - #
- - - - - - - - -
- - - - - - - - -
[E11, BLUE]
# - - - - # X # + + +
- - - - - # # + + +
- - - - - - + + + +
- - - - - - + + + +
- - - - - - + + + +
- - - - - - + + + +
- - . . . + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
```

```
[D11, BLUE]
# - - - - - # X # + +
- - - - - # # + + +
- - - - - - + + + +
- - - - - - + + + +
- - - - - - + + + +
- - - - - - + + + +
- - . . . + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
[D10, BLUE]
# - - - - - # # + + +
- - - - - # X # + + +
- - - - - # # + + +
- - - - - - + + + +
- - - - - - + + + +
- - - - - - + + + +
- - . . . + + + + +
+ + + + + + + + + +
+ + + + + + + + + +
```

APPENDIX D: WWW-INTERFACE

One of the major interfaces I created for Cover is an interface in the form of a World Wide Web (WWW) URL. URL is a Universal Reference Link and defines an address on WWW. WWW is the graphical interface of Internet, a world-wide network. On WWW people can put their documents which can be read by other users of internet. There are two kinds of ways to supply these documents. Just supplying the static document or by making a program that when it is called, creates a document. Such a program is called a Common Gateway Interface binary (CGI-bin).

I choose for WWW as environment since it has some major advantages. The ones that I found the most important are:

- *Platform independence*

Since WWW pages can be read on any type of platform, only one version of the program should be made.

- *Large amount of users*

Internet has a large amount of users. Since Cover is a new game, it is important that a lot of people can get to know the game Cover. Furthermore I could get a large amount of data which could be used for determining the average game length and see a large variation of play styles.

- *Simple Graphical Environment*

WWW pages may contain graphics and simple user input. Therefore WWW is an easy method to create a graphical environment and user interface.

Except for these advantages, It was also a challenge to create a more complex interface with the tools of WWW and Netscape.

The interface created for Cover consists of four major parts:

- *Intro page*

On this page the player gets a small introduction to the game. Furthermore the player can choose the variant of Cover she likes to play and which player starts the game. After these decisions are made, the game can begin by

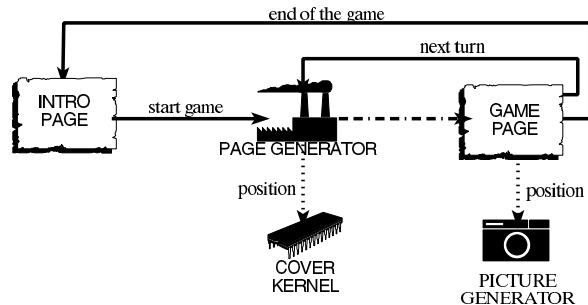


Figure 35: The page creation cycle of the WWW interface

giving the page-generator the command to create the first page.

- *Page generator*

The page generator is a CGI-bin which generates the pages that the player gets to see during the game. The input of the page-generator is the current position of the game together with, if necessary, the move that the user entered. The page generator passes these information through to the Cover-kernel, which will calculate the next computer move. With this move and the current position of the game, the generator creates a new page. If it is the users turn to move, the page contains all legal moves from which the user can choose the next move. Otherwise the page just contains a continue button so the user can see which move he has made.

- *Cover-kernel*

The Cover-kernel is the library with the implementation of the game and the computer players. The Cover-kernel is called by the page generator to calculate the moves of the computer player. For details about the Cover-kernel see appendix E.

- *Picture generator*

The picture generator is a CGI-bin which generates a picture of the current possession of the board. This picture is part of the page as created by the page generator. The input is the current position of the game. The output is a picture in GIF format. The picture generator reads the GIF files of the board and the stones and combines this GIF files to a new multi-page GIF-file. This GIF-file is transformed to a single page GIF file which is the output of this program. For more information see appendix F.

Together, these four parts form the interface of Cover. The interface can be found on URL: <http://artemis.wi.leidenuniv.nl:7171/cover/>

APPENDIX E: COVER-KERNEL

The Cover-kernel is the implementation of the game Cover along with some computer players for this game. The kernel is used for several interfaces which were build upon this kernel. The interfaces which were created upon this interface are:

- *WWW-interface*

The most important interface in the form of HTML-pages and CGI-binaries. For more information about this interface see appendix D.

- *MS-Windows interface*

A Visual Basic implementation of the game used as a stand-alone, graphical playing environment. It contains the playing board together with two lists with the moves of both players.

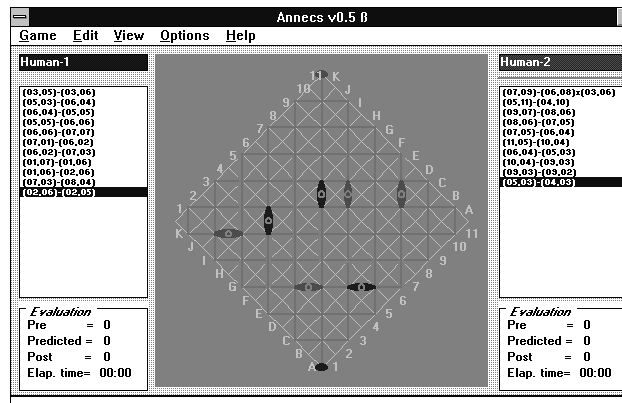


Figure 36: The MS-Windows Interface

- *MS-DOS interface*

A C++ implementation used for research and development of the kernel. The kernel is created in C++ and is highly device independent. Below the separate classes of the kernel will be described together with their most important features.

- *Board*

The implementation of the Cover-board. This class maintains the position of the stones and the history of the game. The board also verifies the correctness of the moves.

- *IsoBoard*

This class is an inheritor of Board. The difference between Board and IsoBoard is that IsoBoard returns always a same symmetrical boards for all positions that are identical apart from the symmetry of the boards.

- *Player*

The implementation of a standard player. This class has a reference to the game of which it is part of.

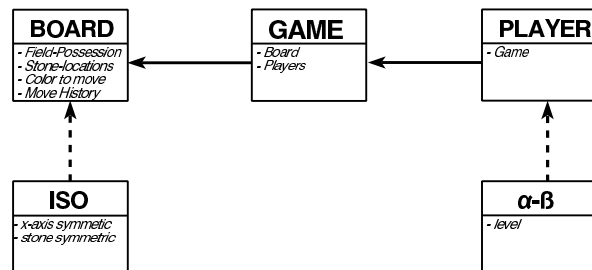


Figure 37: Overview of the kernel of Cover

- *α-β player*

This class is an inheritor of Player and contains the implementation of the computer player.

- *Game*

This class is the implementation of the Game.

The Cover-kernel contains about 4000 lines of code.

APPENDIX F: GIF-LIBRARY

The GIF-library is a C++-library that was created by me to manipulate GIF-Files. A GIF-file is a file representing a picture and is short for Graphic Interchange Format and is described by CompuServe (CompuServe 90).

The most important parts of a GIF-file and their relation are described below.

- *Header*

Contains a GIF-signature and the version identifier.

- *Logical Screen*

Contains, for instance, the height, weight and number of colors of the picture.

- *Color Table*

Contains color-codes as used in the picture.

- *Graphic control extension*

Contains, for instance, data about which color must be interpreted as a transparent color.

- *Image describer*

Contains, for instance, the offset and size of an image block in the total picture. Each GIF-file contains at least one of these blocks. A GIF-file that contains multiple image blocks is called a multi-page GIF-file.

- *Image data*

Contains the compressed data of the picture.

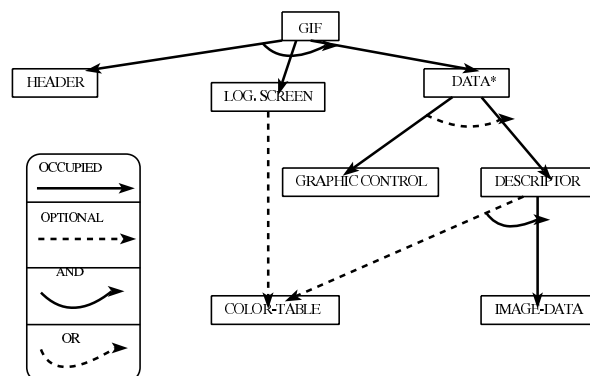


Figure 38: Structure of a GIF-file

The classes in the GIF-library follow the structure of the GIF-file. The classes that represent the parts of the GIF-file are all an inheritor of the "GIF-Part" class. This class handles the in- and output of the GIF-data.

The main functions of the class "GIF-picture" are the following:

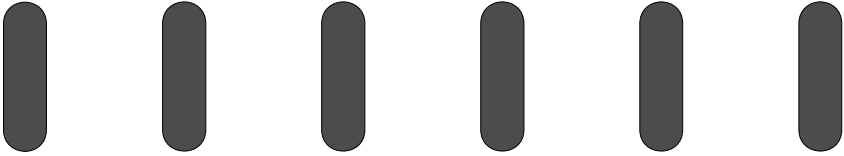
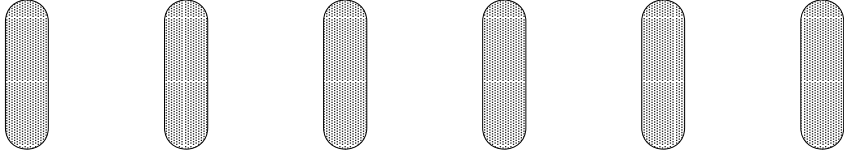
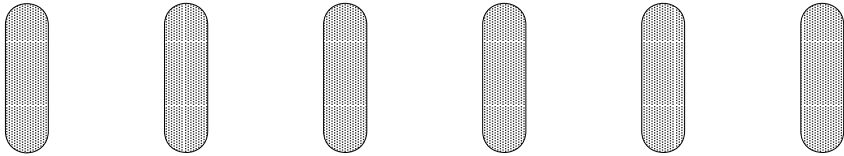
- *Load*
Loading a GIF-file from disk to memory
- *Save*
Saving the GIF-file from disk to memory.
- *Insert*
Insert another GIF-file in this GIF-file as new image blocks at a certain location.
- *Flatten*
Changes a multi-page GIF to single-page GIF-file. This is done as follows: the different image data blocks are decompressed and combined to one, single bitmap. This bitmap is compressed again to a new image block which replaces the multiple image blocks.

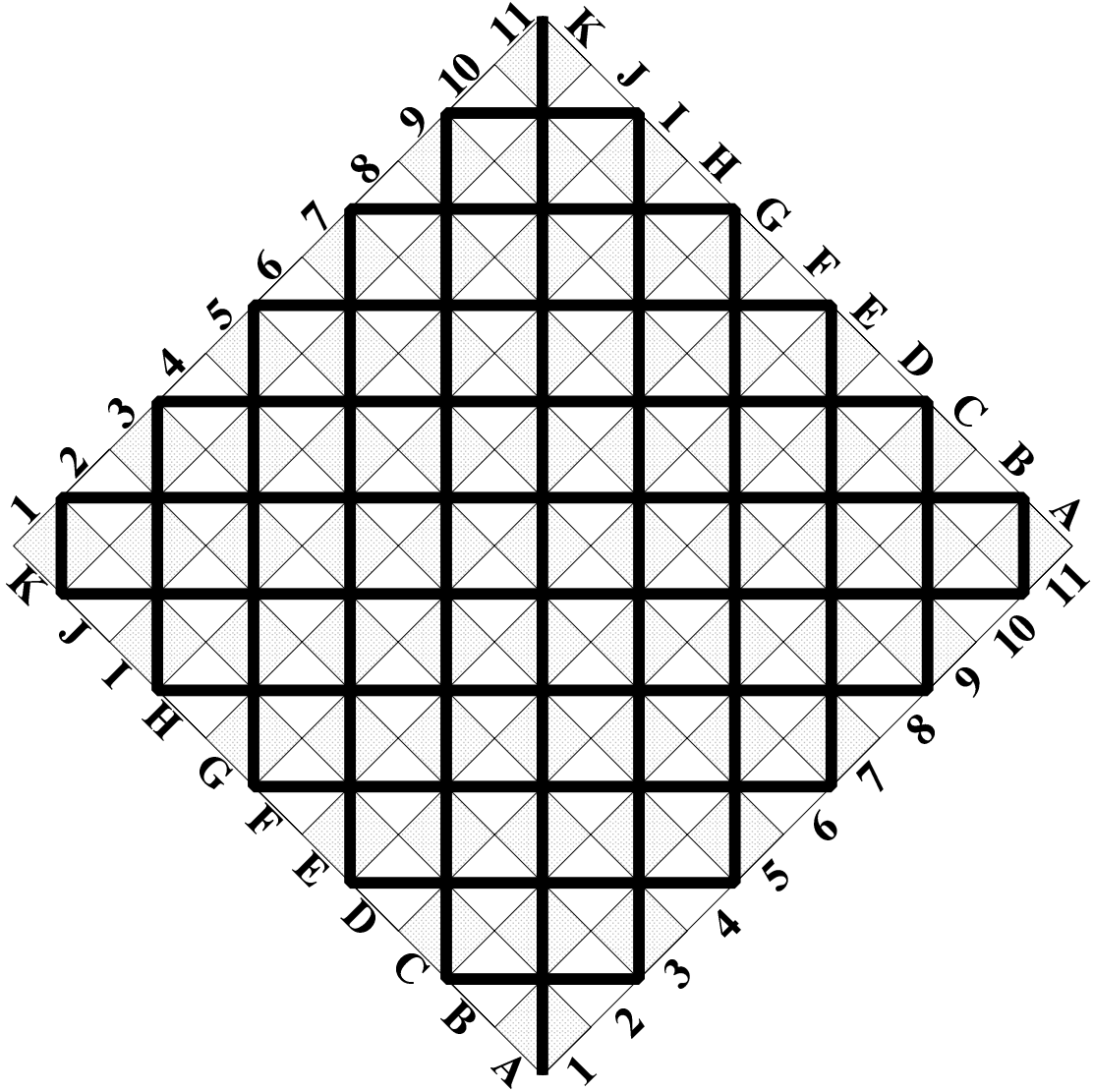
The flatten function is very important since most GIF-viewers can not handle multi-page GIF-files, for instance, Netscape. This function is also the most complex function since it handles the complete compression and decompression of a picture together with the feature of combining multiple pictures to one. This feature is used to create a picture of the board with the stone as they are located on the board. On the picture of the board, several pictures are placed on their correct locations. In the flatten function, these pictures are combined to one picture by copying the pictures of the stones into the picture of the board, while ignoring the background of the stones since that must be regarded as transparent.

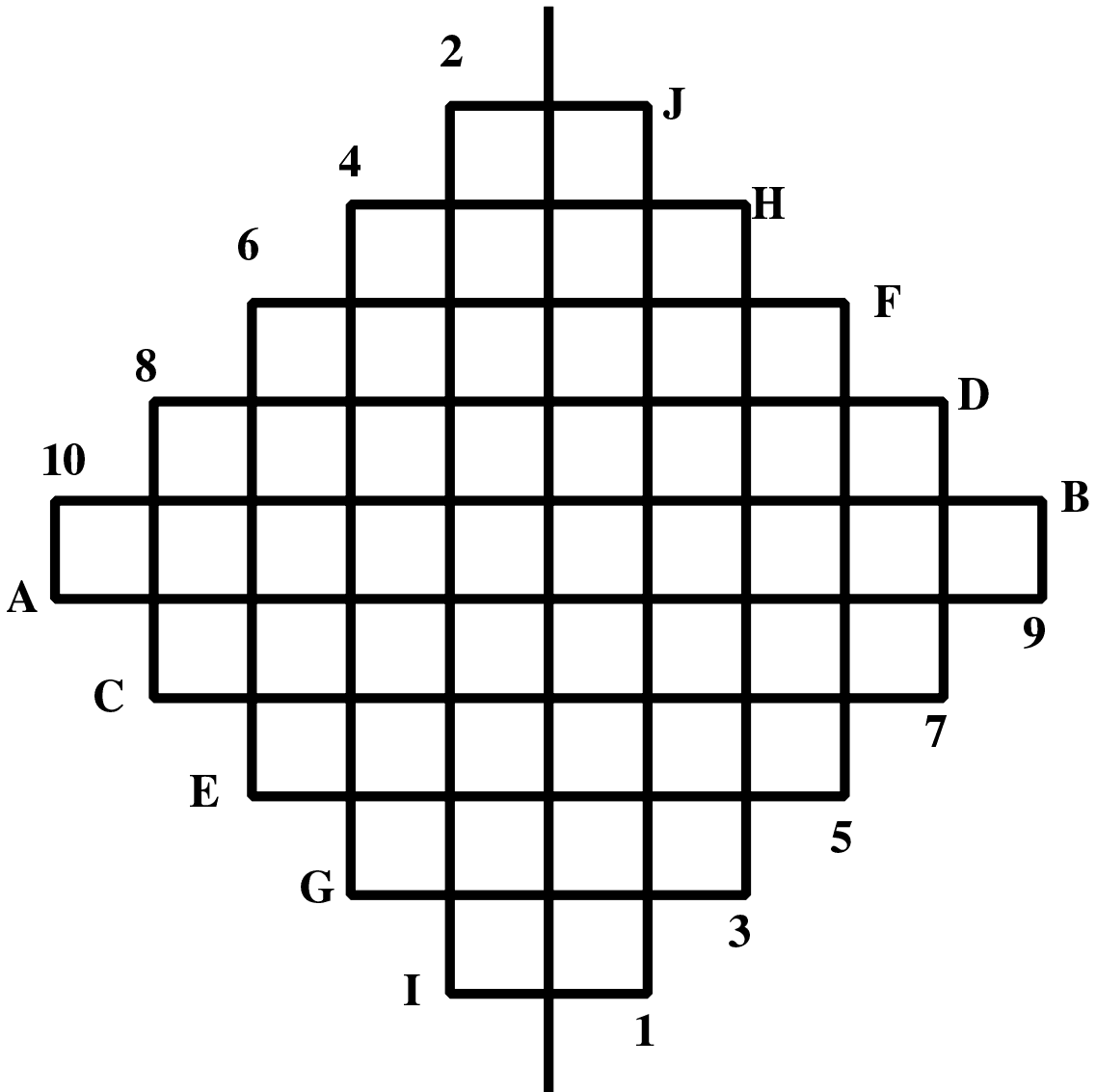
If, in future versions of Netscape, multi-page GIF-files are supported, the flatten function will be superfluous, since the combining of multi-page GIF-files to one picture will be done on screen.

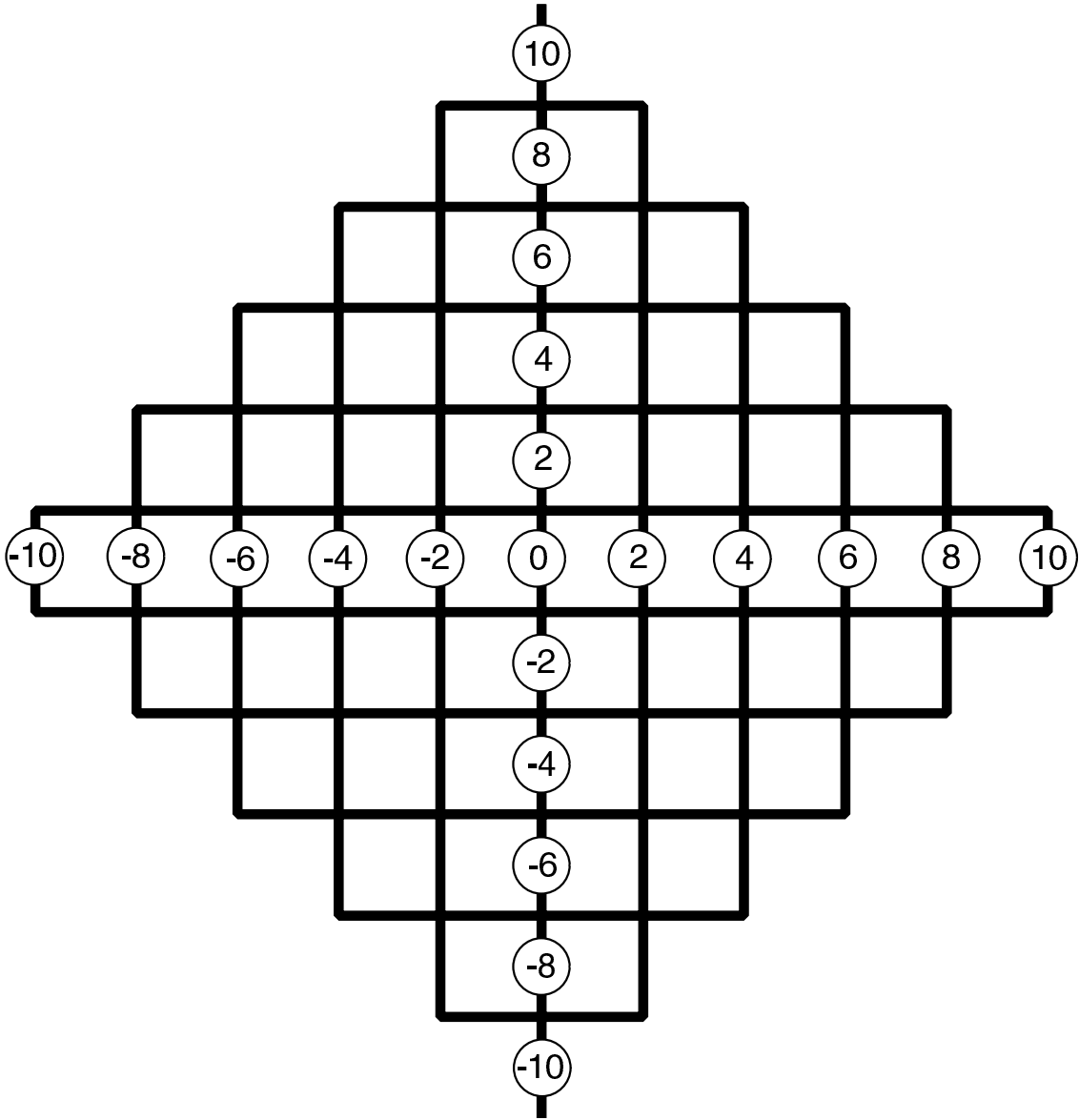
The GIF-library contains about 1500 lines of code.

APPENDIX G: COVER PLAY KIT









BIBLIOGRAPHY

Allis, L.V, 1994, Searching for Solutions in Games and Artificial Intelligence, *Thesis Maastricht*.

Berliner, H.J., 1979, The B* Tree Search Algorithm: A Best-First Proof Procedure, *Artificial Intelligence, Vol. 12*.

Compuserve Incorporated, 1990, Graphic Interchange Format Programmers Reference, <http://www.w3.org/hypertext/WWW/Graphics/GIF/spec-gif89a.txt>.

Goetsch, G., Campbell, M.S., 1990, Experiments with the Null-Move Heuristic, *Computers Chess and Cognition, Springer-Verlag*.

Goldberg, D.E., 1989, Generic Algorithms in Search and Optimisation and Machine Learning, *Addison-Wesley*.

Herik, H.J. van den, 1985, Computerschaak, Schaakwereld, Kunstmatige Intelligentie, *Academic Service*.

Knuth, D.E. and Moore, R.W., 1975, An Analysis of Alpha-Beta Pruning, *Artificial Intelligence, Vol. 6*.

McAllester, D.A, 1988, Conspiracy Numbers for Min-Max Search, *Artificial Intelligence, Vol. 35*.

Michalewicz, Z., 1994, Genetic Algorithm + Data Structure = Evolution Programs, 2nd ed., *Springer*.

Moriarty, D.E., Mikkulainen, R., 1995, Evolving Neural Networks to Focus Minimax Search, *AAAI-94*.

Plenkner, D., 1995, A Null-Move technique imperious to ZugZwang, *ICCA june 1995*.

Rich, E., Knight, K., 1991, Artificial Intelligence 2nd ed., *McGraw-Hill*.

Stockman, G., 1979, A Minimax Algorithm better than Alpha-Beta?, *Artificial Intelligence, Vol. 12*.

Tesauro, G., 1993, TD-Gammon, A Self Teaching Backgammon Program, Achieves Master-Level Play, *AAAI Technical report FS-93-02*.

Winston, P.H., 1977, Artificial Intelligence, *Addison-Wesley*.