University of Ljubljana
Faculty of ra cunalnistvo and Information

Neza Belej

Comparison and Optimization algorithms investigation on the case abstract games Yinsh

MASTER'S THESIS

MASTERS PROGRAM LEVEL
RA CUNALNI StVO AND INFORMATICS

mentor: Assoc. prof. dr. Polona Oblak

Ljubljana, 2017

Acknowledgments

At the end of the study to mention people who are each in their own way contributed to the fact that my student years was a wonderful and unforgettable. I thank my mentor, Assoc. prof. dr. Polona Oblak. For the first class mentoring BA and MA part, as well as to aid during the entire MA. Thanks for all the tips and discussions. Professional and life.

I thank my family: parents, sister, Barbara, my best friend; and Primoz dear brothers and Peter. My suns to Vito and yaks, because I always embellish the day.

Thanks to your friends, you make sure you are in stressful student years dominated by the beautiful moments: Manca, Blaz, clementines, Alen, Matthew, Veronica Matthew, Jan, Dejan.

Nejc, it's hard to describe how much it means to me your support. Thank you was all the time with me. The good and bad.

Neza Belej, 2017

"Sometimes you'll have the impression That there is nothing but chaos on the board. You line up a few of your pieces

- and the next thing you notice is That They have changed color! The can piece on the board, the not di FFI cult it Becomes to predict What will happen. But Also the Opportunities you can have! So do not worry: you'll get your chance. Just stay alert, and you'll create your own little bit of order and the chaos! "

Contents

List of Abbreviations

, the English abbreviation		Slovenian
DFS	depth- fi rst search	investigate in depth
MCTS	Monte Carlo Tree Search	Monte Carlo tree search
UCT	Upper Con fi dence bounds Trees	for the upper confidence limit for the trees
AI	Arti fi tial Intelligence	artificial intelligence

Summary

address: Comparison and Optimization algorithms investigation on the case of abstract games Yinsh

The thesis is dealing with the investigating algorithms with which it is possible to find solutions to abstract table games. Pick your abstract game Yinsh, which is part of the famous Gipf project. The aim of the master thesis is to find and implement smart player games Yinsh that is able to overcome the existing implementations of smart players, while the competition in the game against experienced human player. As a basic method you choose Minimax algorithm, which assembled more evaluation functions, whereby they are upgraded. At this point, we do also present in person analyzing the constants in the evaluation function, and to find the optimum combination of these. Basic Minimax because of the large trees factor vejitvenega games Yinsh not a satisfactory solution. Depending on the nature of the game proposing several optimization methods Minimax them implement, We tested and we evaluate. Implement a tree algorithm investigation Monte Carlo, whose performance is compared with the Minimax algorithm. Based on the research and developed the algorithms produce a final solution. We also suggest possible improvements and future work. Our final solution brings very good results: a convincing defeat most existing implementations, however, possible to beat the very experienced human players.

Key words

artificial intelligence, investigating algorithms Gipf project, Yinsh, Minimax

Abstract

title: Comparison and optimization of search algorithms exempli fi ed by abstract game Yinsh

In the thesis we analyze That search algorithms are able to fi nd solutions for abstract board games. We choose the game Yinsh, one of the well known games Gipf project. The goal of the thesis is to fi nd and implement a Yinshplaying That program is able to defeat all the available computer programs and is strong against experienced human players. As a base Minimax method we choose, for Which we implement various functions gradually upgrading evaluation. We analyze the constants and Those functions and fi nd the best combinations of Them empirically. Because Yinsh has a large branching factor, basic Minimax is not an optimal solution. Based on the nature of the game Yinsh we propose multiple optimisations of the Minimax method, Which we implement, test and evaluate. We also implement the Monte-Carlo Tree Search method for the game Yinsh ITS and compare performance to the Minimax-based algorithms. Based on the analysed Approaches we compose the fi nal solution. We also propose improvements and future work. Our fi nal solution produces very good results. It Defeats multiple computer programs and is Also experienced strong against human players.

Keywords

arti fi tial intelligence, search algorithms, Gipf project, Yinsh, Minimax

Chapter 1

Introduction

The project includes six Gipfel abstract table games manufacturer Krisa Burma. This is the type of games with complete information intended for two players. The project represents a major challenge in the field of artificial intelligence, as the solvability of the games contained in the opinion of Heule and Rothkrantza complex problem [7]. Games Gipf project are playing with simple fi Guram, which are placed on the hexagonal game board. The purpose of the project is to create a framework of different games, with each game slightly different rules and fi Gure (potential) with which it is possible to constitute new types of games [17]. The web portal Board Game Geek [19] In the context of the finest abstract games were games Gipf project estimated very well: Yinsh game is currently in second place in the Tzaar fourth, fifth Dvonn. Other (Z`ertz, Gipf, P

UNCT) are classified under 45 cities.

The project Gipf there is already some research. More material there is to the game Dvonn: so he Mauss in his thesis [12] dealt with the implementation of the tree to investigate Monte-Carlo, Kilgo was in his study [9] made progress in the implementation of the Minimax algorithm for game Dvonn. Dvonner [21] and Holtz [22], the implementation of advanced smart player games Dvonn. Waltz [16] In the aforementioned article describes as a strong program for a game Tzaar, using an advanced version of investigative techniques *Cutting trees Alfa-Beta* and *investigation of the evidence borders (England. proof-number search).* In his doctoral thesis successfully implement

smart player games Gipf created Wentink [17]. It all games Gipf project implementation could have on the website of Board Space [20], where the smart gamblers project Gipf described as negotiable especially for experienced players. The only exception here is Dvonn, the latest version of the robot on the portal represent a challenge even for experienced players.

The problem under investigation, the search for the best approach to implement a smart player games Yinsh depending on its complexity. In doing so, we will be interested also, how could knowledge and strategies for experienced players to integrate into our implementation. Our implementation, we also want to test, and so on existing implementations as against the human player. The works that we represent major challenges are:

- Implementation of David Peter [23], which is an open source implementation of its smart player games Yinsh also transferred to the web version of the game,

- Smart players (*dumbot, weakbot, smartbot)* the known portal Board Space

- Implementation Thomas Debray, who, with his Yinsh botom of the tournament Yinsh bots reached third place.

The paper *Solving games: dependence of solving Applicable Procedures [*7], the author of, among other things roped on the solvability and the theoretical value of the games Gipf project. Yinsh be defined as the most complex game in terms of solvability. Yinsh already has millions of possible starting positions: the initial formation of rings there are 7.9×10^{14} possible positions. Even taking into account the symmetry of the game board this number is not significantly reduced.

Structure of the master's thesis is as follows: in section *Yinsh* We will be described in detail Yinsh game and its rules. Below (Chapter 3), followed by a description of all of the methods and algorithms that have been during the manufacture of the master part meet and implemented. We will also describe how these methods are applied to the game Yinsh. In chapter *Optimizing* We will describe methods which

we have implemented with the aim of pohitrititi time thinking of our smart player. Below is a description of implementation (Chapter 5), where be explained our selection of tools for the preparation of the program and a description of the connection between the server and the user interface portion of the game. *results* They will give us an insight into the competitiveness of our program - we will describe how the smart player cut us against existing smart players, as well as against less and more experienced human players Yinsha. Existing smart players will also be described. In chapter *Conclusion* the results will be evaluated and referred to the possibility for further work.

Chapter 2

Yinsh

Rules of the game are taken from the official rules of the game Yinsh [27].

2.1 Components

The necessary components to play are:

- board games,
- 5 white and 5 black rings,
- 51 chips (the first black, the second white).

2.2 Aim of the game

Players start the game each with its five rings placed on the game board. The player should remove the ring from the game board, if it reaches five tokens of their color in a row. The winner is the one who first removes its three rings from the game board. To win it is therefore necessary to pick the three types of five chips of their color.

2.3 Confident

2.3.1 Layout

The game starts the player with white rings. The players alternately place the own rings at the intersection of hexagonal playing surface.

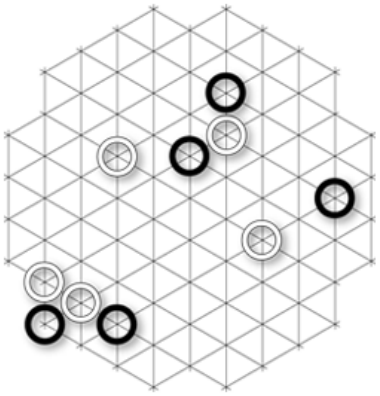

Figure 2.1: For example, the state of the game board by placing rings. Image Source: http://www.boardspace.net/yinsh/english/rules.htm.

2.3.2 Moving rings

Each move starts with a token of their color in your color ring. The ring then moves the player according to the following rules:

- Upon movement of the rings of the token remains in the initial position of the ring (Figure 2.2).

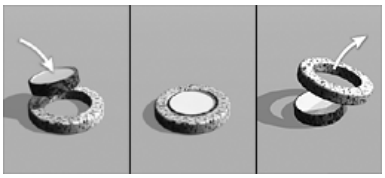

Figure 2.2: Move the rings. Eton remains in the previous position of the rings, ring moves. Image Source: http://www.boardspace.net/yinsh/english/rules.htm.

- Ring always move in a straight line and the vacant seat panel.
- The ring may skip one or more vacancies.
- The ring may skip one or more chips any color, where they must be placed sequentially, ie. no vacancies in the line.

  If you skip token ring must be the first free town of Eton in line.

- Ring can skip the first vacancies and continues to skip over the token, a token must be the last in a line jump to end.
- The ring may not jump over other rings.

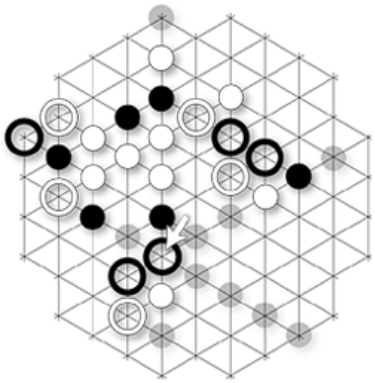

Figure 2.3: The picture shows the permitted movements labeled black rings. Image Source: http://www.boardspace.net/yinsh/english/rules.htm.

2.3.3 Turning chips

If you move the ring to skip tokens, tokens are all able to turn (change color). This does not apply to the token that is at the beginning of the mood moves in a moving ring, as he was not able to.

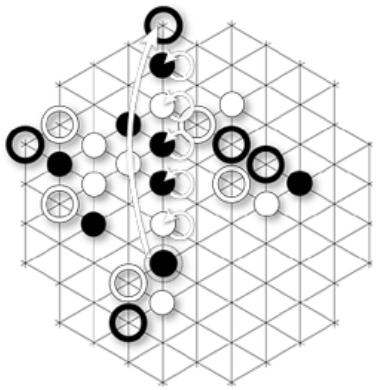

Figure 2.4: The picture shows the displacement of labeled black rings. Mark the chips, which in turn change color. Image Source: http://www.boardspace.net/yinsh/english/rules.htm.

2.3.4 Formation of species and removal of rings

Within the game, players aim to form a series of five successive chips of their color. The achievement of the right kind of five chips removed from the playing surface, it also removes any ring of their color.

If a player reaches more than five chips in a row, you can remove any consecutive top five in chips.

When playing, it is possible that a player with one stroke acquire two types of five chips.

If you type do not intersect, the player removes both also removed two rings.

If you type intersect, the player removes any type (second will after removal of the first incomplete).

The player can use their movement formed a series of five chips into the color. In this case, the opponent is removed and the token ring and continuing its stroke.

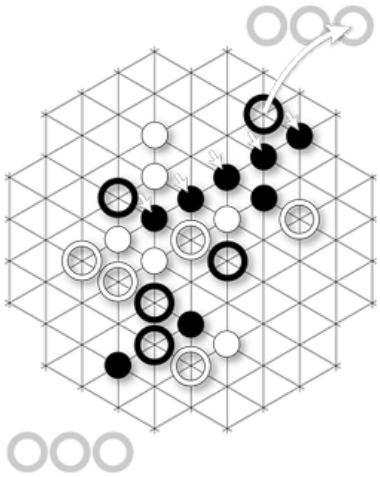

Figure 2.5: The picture shows formed a series of five black zetonov.Vir Image:
http://www.boardspace.net/yinsh/english/rules.htm.

2.3.5 End Game

The game is over in the following cases:

- The player obtains 3 rings and both win.
- The player with the last move and obtain his opponent's third row. In this case, the player wins who made his move, because it was his kind of removed earlier.
- If they were all tokens placed on the board before the formation of the three types of player, wins the largest number of rings obtained.

  If

  They have accumulated the same number of rings, a tie.

2.3.6 Characteristics games

Game Yinsh has some special characteristics that we in search of its software solutions represent major challenges:

- Unlike type games *n-type* Yinsh in the game you need to collect three types of five chips. It is necessary to determine the appropriate balance between victory and reclaimed a series of five chips.

- The initial layout is dynamic. The players alternately place the rings anywhere gameboard. Unlike games such as checkers and chess in general at the game Yinsh initial layout for each game is different. This prevents the player using a predetermined initial moves.

- The game contains turning chips, which means that the state of the game at the moment completely changed.

Chapter 3

The investigating algorithms

3.1 Minimax

Minimax is the most common algorithm to determine the best moves of the current state of the game. It is very useful for simple games for two players, where players play against each other, whereby all of the following possible features of two well-known actors (ie. Play with complete information).

Minimax is based on building *Inquiry trees (England. search tree)* where each *node (England. node)* It represents a possible move. For each node it is possible to calculate the value of which is estimated *power (England. goodness)* moves. Based on these values can be chosen the best possible move, which follows the current state of the game. It is necessary to pay attention to the alternating nature of the games for two players (angl. Two-player games), because any player choosing a move that's best for him (and a loss to the opponent).

Minimax can use two strategies: the first is the investigative searched the whole tree, all the leaves (England. Leaf nodes), illustrating the final state of the game (win or lose). Another strategy is based on limiting the investigative trees against predetermined depth.

Minimax is an algorithm that uses investigating in depth (England. Depth fi rst search) and to maintain a minimum or maximum value of the nodes successors of a given node. Having reached the end node (leaf) is calculated

the value of this node using the evaluation tool. This value propagiramo the tree up, all the way to the root. At the levels where it is the turn of us smart player, you choose the maximum amount of nodes successors; for the opponent to take minimum. Propagation of a tree upward is thus alternating: maximize a minimum to minimize the maximum. In other words, we assume that each player made his move, which he most promising [8]. The operation of the algorithm is shown in Figure 3.1.

Explain the time complexity of the algorithm Minimax. If the max the depth of the tree *m* of each node is possible *b* the following traits, then the time complexity of the algorithm *About (b m).* The spatial complexity of the amounts *O (bm)* the algorithm that generates all campaigns at once, or *O (m)* algorithm, which generates only one campaign at a time. Investigation in depth requires only save the current path from root to leaf, along with the remaining unexpanded neighbors all nodes in the path. When the node is expanded, it can be removed from memory as soon as all of its descendants already studied.

The real game such time complexities completely impractical, but the algorithm serves as the basis for mathematical analysis of games and for more practical algorithms [15].

Pseudocode 1 shows a general Minimax algorithm taken from [24].

3.2 Cutting alpha-beta

The main problem with Minimax algorithm is that the number of test nodes increases exponentially with the depth of the tree. Unfortunately no investigation eksponentnosti It is removed, but there is a way how to effectively halve the exponent. Namely, it is possible to obtain an accurate value Minimax free to investigate each node of the tree. The algorithm that allows the cutting the alpha-beta. It allows to obtain accurate minimax values, but at the same time discarding the investigation branches that can not affect this value [15].

Let's look at Figure 3.2.

algorithm 1 Pseudocode algorithm Minimax

```
1: minimax (level, player) 2: if gameover or Level
== 0 then
3:      return Score 4: end
If
5: children ← all legal moves for this player 6: if Player is
computer then
7:      bestScore ← - inf 8:
        for Each child to
9:          score ← minimax (level - 1, opponent) 10:
            if Score> bestScore then
11:             bestScore ← score
12:         end If
13:     end for
14:     return bestScore 15: else if Player
opponent is then
16:     bestScore ← + INF 17:
        for Each child to
18:         score ← minimax (level - 1, computer) 19:
            if Score <bestScore then
20:             bestScore ← score
21:         end If
22:     end for
23:     return bestScore 24: end
If
25: minimax (2, computer) {initial call}
```

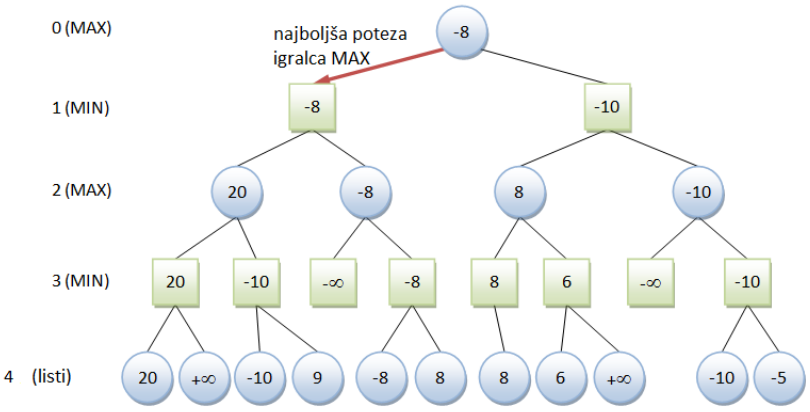

Figure 3.1: The figure shows the operation of the algorithm Minimax. Blue nodes (circles) represent the moves of the opponent, green nodes (squares) but features a smart player. At the last level (leaves) are both parents of the child promotes the value of minimal value. On the penultimate level of the parents transfer the maximum value.

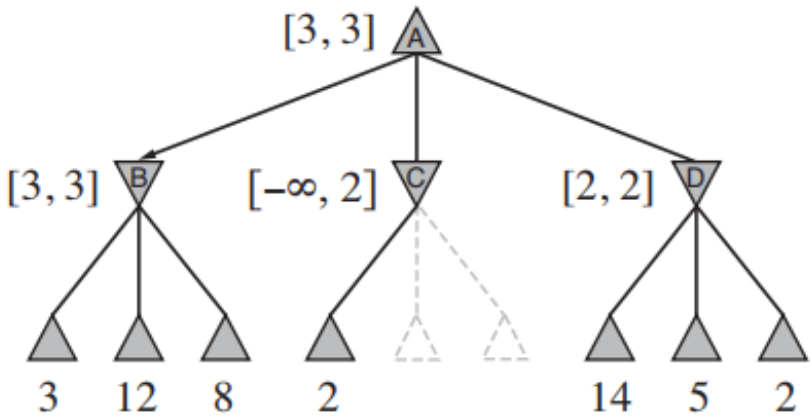

Figure 3.2: Simulation Algorithm Minimax cutting alpha-beta.

Unexplored node in Figure name all x and y. value root

node is:

$$\begin{aligned} \text{Minimax (root)} &= \max(\min(3, 12\ 8), \min(2, x, y), \min(14\ 5\ 2)) \\ &= \text{Max}(3, \min(2, x, y), 2) \\ &= \text{Max}(3, by, 2) \\ &= 3 \end{aligned} \tag{3.1}$$

We see that the value *with* in equality (3.1) is less than or equal to 2, which means that we value x and y not really interested and the final value of the Minimax algorithm independent of their value.

Trying to explain it in a more natural way: we have a hub n at any depth investigative trees, where you can move the player. If the player has a better chance m to the parent node n or anywhere higher in the tree; then n never visited the real game (Figure 3.3).

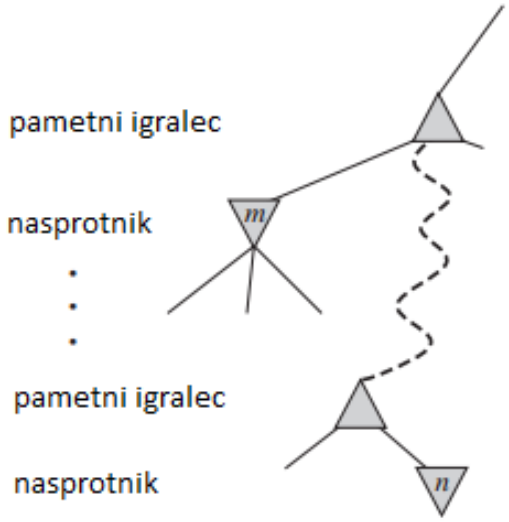

Figure 3.3: Simulation of the alpha-beta. If m best of n for the current player, then the game will never visit n.

Alfa-Beta investigation during the investigation of updated values α and β and to provide for cutting the remaining branches of the current node, if the value of the current node inferior to the current α or β the MAX or MIN.

α it also refers to the best (highest) value is found anywhere in detention tree on the way to MAX.

β refers to a better (less) value is found anywhere in detention tree on the way to MIN.

2 shows a pseudocode algorithm Minimax upgraded by cutting the alphabet, taken from [24].

3.3 Use of the algorithm Minimax in the game Yinsh

Minimax method when applied to the game Yinsh we need to answer the following questions:

- What will be the evaluation function for the state of the game? What parameters will be taken into account when assessing the game board?
- What will be the depth of the investigation?
- What will be the exit condition of recursion?
- What information will be stored in the node tree?

The evaluation function was formed gradually. We started with a completely simple features, which were gradually upgraded into a stronger and more representative function, which means that the parameters are depicted realistically play. The complexity of the functions are adapted to the depth of the investigation and partly an exit condition of recursion.

3.3.1 Types of evaluation functions

First number zetono in

Although the number of chips of our color does not necessarily mean domination (because it increases the risk to the opponent turns our token of their color), it is nevertheless an important factor in the game. Therefore, it was our first, simplest solution as follows:

algorithm 2 Pseudocode algorithm Alpha-Beta

```
1: minimax (level, player, alpha, beta) 2: if gameover
or Level == 0 then
3:      return Score 4: end
If
5: children ← all legal moves for this player 6: if Player is
computer then
7:      for Each child to
8:          score ← minimax (level - 1, opponent, alpha, beta) 9:
            if Score> alpha then
10:             alpha ← score
11:         end If
12:         if alpha> beta = then
13:             BREAK
14:         end If
15:     end for
16:     return alpha 17: else if Player opponent
is then
18:     for Each child to
19:         score ← minimax (level - 1, computer, alpha, beta) of 20:
            if Score <beta then
21:             beta ← score
22:         end If
23:         if alpha> beta = then
24:             BREAK
25:         end If
26:     end for
27:     return beta 28: end
If
29: minimax (2, computer, -inf, + inf) {initial call}
```

At the depth reached five Inquiry tree conclude the investigation successors node (exit condition) and evaluate our game board according to the following equation:

score (node) = numberOfMarkers (AI) - numberOfMarkers (opponent)

(3.2)

Investigation also completed on each collected a series of five chips (any color). Then hub evaluated by 100 points if he scored five tokens smart player.

If five chips to reach opponent node

-100 points assigned.

Despite the simplicity of solutions, we found that this is not bad. Dana heuristics enable smart playing computer player. In testing, we found that the solution is already possible to overcome the human inexperienced player who is able to create and defend their kind, and inverted types opponent.

However, we are aware of the shortcomings solutions: solution takes into account only the number of your chips, not scans, however, whether these tokens makes sense formed in kind. Do not verify that this type are adequately protected. It is also ravenously heuristics: the player can obtain their type, as this allow the opponents win the next move. The solution is therefore not taken into account the weight of victory or defeat; checked only intermediate species collected five tokens.

Second The number of chips in a row

The second method is slightly more complex. We develop a simple algorithm that checks all the lines gameboard and evaluate each type of chips. The algorithm in a single line for instances of the token, as well as types of two, three, four or five chips. Of course, the length of the species should be properly weighted.

Let's look at the picture 3.4. Each subsequent tokens in a row bring greater value of the current row.

The number of chips is estimated by the values 1, 3, 9,

27 and 81 (from 1 to 5 of tokens in the queue), where we start counting always the same token.

If the number of chips in a row *n* then a series of assessments by

Equation (3.3).

$$score = 1 - \frac{1}{2(3^n - 1)} \tag{3.3}$$

One chip so bring one point, two tokens bring four points, three tokens bring 13 points, four 40 points and five chips 121 points. Smart player plays white colors. First, look at the white token in a vertical line. Five found one chip (five points), and again after four tokens in a row, which is 40 points, a total of 45 points. Similarly done by the two diagonal lines. We were obtained 13 and 11 points. The total of points for the white player on the panel is therefore 69 points. Similarly calculate the points for the opponent and deduct them from our points. Thus we come to the end boards, 57 points. This way we evaluate the board at the following events: when it comes to the depth of four investigative trees or the discovery of five chips (estimate before removing the token).

The latter mode indicates an already much improved version of the smart player. It is possible to overcome the more experienced players already, but still notice some shortcomings. Again, we do not investigate the protection of our stack, ie. or the opponent can move to the next turn. Notice the occurrence of *delaying* our bot. Since we estimate a condition of the entire panel and the player does not reward particular having collected five tokens, smart player took a very strong win state of the game board to your advantage. It has a lot of combinations of four pieces, but it sometimes happens that has the ability to get to five chips, but with an opponent for delaying the possibility of ruin.

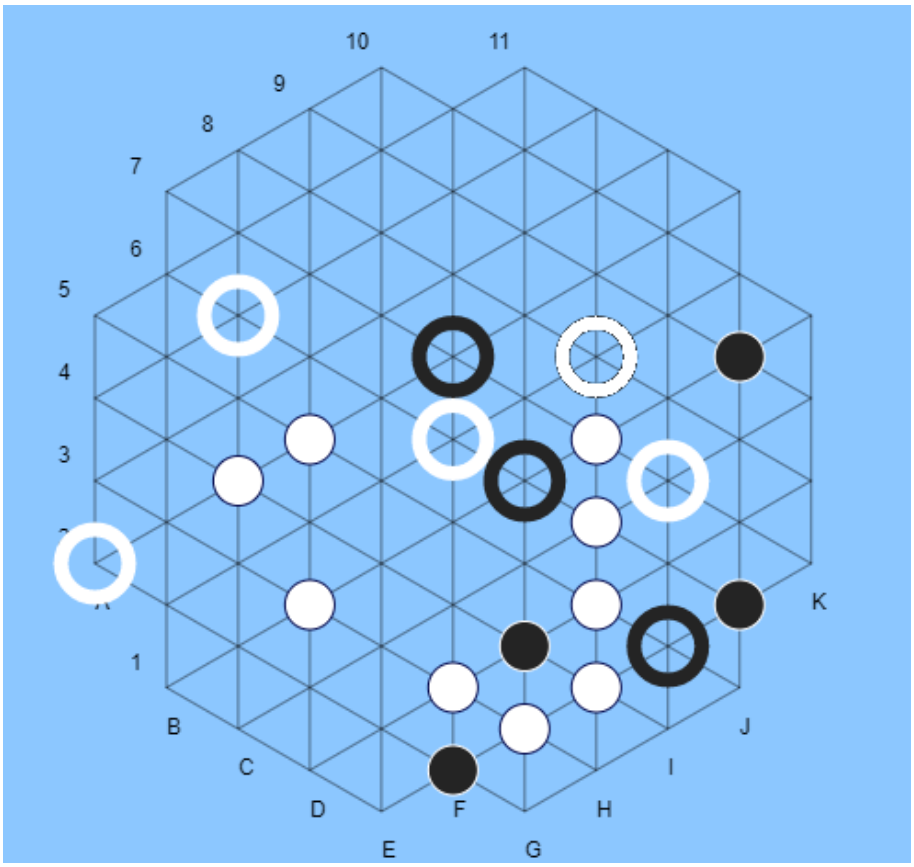

Figure 3.4: For example, the game board.

The combination point for assessing the types of chips

In the context of the functions described are made less analysis for the selection of a combination of points awarded by the types of chips. We chose from the following combinations of 1, 2, 3, 4, 5 chips in a row:

- 1, 10, 100, 1000, 10000
- 1, 2, 4, 8, 16
- 1, 3, 9, 27, 81
- 1, 2, 3, 4, 5
- 1, 10, 50, 100, 500

Between each pair combinations were carried out 20 automated games, where each player play smart with your grading scale. Of these twenty games started by the 10 player A, the other 10 games he started player B (changing colors of players). Games are given randomly placed an initial set of rings. Altogether, we therefore conducted 200 games in each type of combination was involved in 80 games. The results are summarized in Table 3.1.

Table 3.1: Comparison of combinations of constants for assessing the types of chips

combination	Number of wins Percentage [%]	
1, 10, 100, 1000, 10000	41/80	51.25
1, 2, 4, 8, 16	40/80	50
1, 3, 9, 27, 81	54/80	64.5
1, 2, 3, 4, 5	19/80	23.75
1, 10, 50, 100, 500	46/80	57.5

Based on the above results, we decided to smart player is equipped with a combination of the points 1, 3, 9, 27, 81 for 1, 2, 3, 4, 5 chips in a row.

If the species is more than five tokens each subsequent token assessment

with 81 points. The longest species would not be exaggerated to evaluate better, because it does not illustrate the object of the game, which is the achievement of five chips in a row.

Third Greedy evaluation function

The third version is a combination of the previous two. It works as follows:

- Having reached a depth of four investigative tree node allocated points according to the evaluation function described in the preceding paragraph.
- Having collected five chips investigation ends and nodes assessed by 10000 points (five tokens smart player) or
  -10000 points (five chips opponent). Number 10000 here
  It represents the maximum possible number, to which it is possible to evaluate the board (-10000 is smaller). Thereby achieving that whenever we have a chance to pick five tokens. Also, whenever we have the opportunity to prevent this opponent. In the case where it is possible both to consider the possibility that a joint delivers more the points.

Play smart player with consuming evaluation function works much better
- evaluation function to the proper assessment of the appliance, 10000 points with five chips makes sure that the player always picks his five tokens and do not delay. Of course it can happen to pick up your type, you could for example make more sense to prevent the opponent the victory. Another drawback is that the smart player here does not know how to evaluate the protection of its chips.

4th Advanced evaluation function

Let's look at the picture 3.5. If to the earlier evaluation function calculated number of points, we find that in a series of J is found three successive token and two successive token in black. But each player can clearly see that ring that separates these two types can in turn reach the next series of five (all six) black chips. From these findings we update our

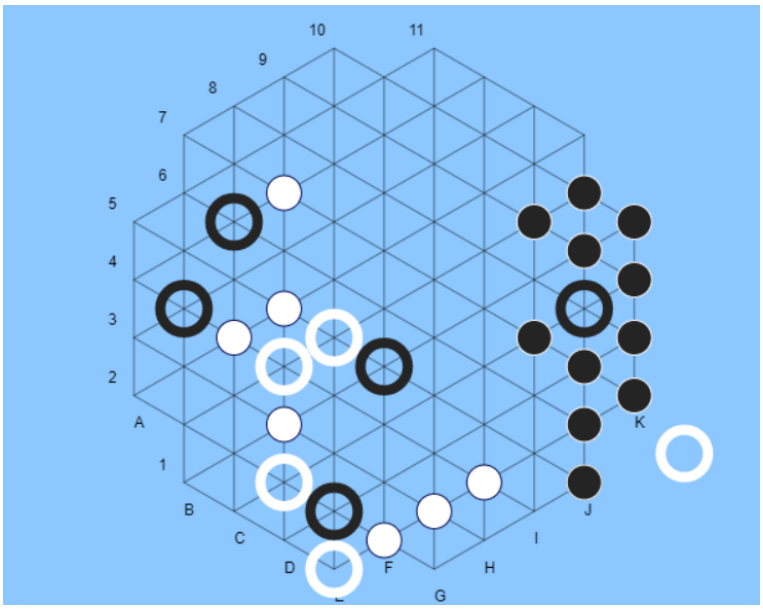

Figure 3.5: For example, the game board.

function: if the line of successive chips occurs ring of the same color line of investigation is not interrupted, but the ring is in line with the estimated value halved in relation to the city ring in sequence. In our case, the line is long as six cities. As already mentioned, the six chips the same as in the fifth. Thus, distance between the rings and J5 J10 was assessed using the value of the equation (3.4), which is already indicative of the actual power of type J5 J10.

$$score = 1 + 3 + 9 + 0.5 \cdot 27 + 81 + 81 \tag{3.4}$$

Here, too, we evaluate nodes at a depth of four. In addition to updates on their own evaluation function in the current version update the assessment of the five types of chips:

When we reach the line of five chips, the result of the evaluation function and adding points according to the following criteria:

- 100 points at a range of five chips smart player or 10000 points

winning smart player. Here we want to win properly weighted in comparison with reclaimed series of five chips.

- -50 points at the range of five chips opponent or -10000 points when defeating an opponent. Here, too, the weights of the opponent and win only a single line of five chips. The line of five chips here and do not assign -50 -100 points than would be expected in relation to the previous bullet.

This parameter can be controlled *aggressiveness* or *defensiveness* our player. With the current allocation of points is our series of five chips weights more than a series of five chips opponent, which means that our player plays a slightly more aggressive. The opposite situation would be smart to play a more defensive player. After a short empirical analysis, this allocation of points turned out to be slightly better than the allocation of 100 and -100 and award points and 50 points -100 (defensive play).

At this stage we focus on stability kinds of chips. It is clear that the type of four chips on the line C in Figure 3.5 protection. This means that white opponent in the next turn black type will not jeopardize (ie. To turn the chips). Sometimes it happens that the species is not protected and allows the opponent to turn part of a series. We decided to reward therefore the stability of the species:

- 100 points are added to the nodes that belong to the branch, at which point the consecutive nodes increase with depth. In such a way to reward a sequence of moves that are in favor of our player.
- 100 points are deducted nodes belonging to arms, where they were during the trip points for two consecutive nodes decreased by more than 100 (which means that his opponent turned our 4 chips).

The advanced mode of evaluation we have come already to a very strong version of our smart player who solves the problems to which we have encountered in

the first three variants. Details of how good is our evaluation, as described in chapter Results.

3.3.2 The depth investigative trees

Depth, which was achieved by these types of Smart player was five (type 1 player) or depth of four (the other three types). This is the depth that we have achieved just before optimization algorithm and depending on the time of investigation. Given that the game was also tested against a human player, the time thinking smart player on the move should be moderate: in our case it was somewhere up to 30 seconds, sometimes much less, sometimes more. After optimization of the time has decreased significantly (about 5 seconds). Learn how we optimize the time of the investigation and thereby increase the depth is described in section Optimization.

3.3.3 Structure of the nodes of the tree

Information to be borne by our nodes are as follows:

- The index of the player (1: white, 2: black)
- State of the game board (positions white / black chips / rings)
- Position from which the current player made his move and thereby came to the state of the game board
- Position, which has moved to the current player and thus came to the state of the game board
- Following the current optimum node (child)
- The steady state node (yes / no)
- Vertices node (calculated according to the evaluation function for the state of the game board in its current state)

3.4 Tree investigate Monte Carlo

Tree investigate Monte-Carlo (England. Monte-Carlo Tree Search) is a method for finding optimal decisions in a given domain. On the basis of random sampling results collected and gradually built investigative tree. Eventually algorithm is getting better in the evaluation of nodes and consequently finding the best node. The method has proven to be very effective in the field of artificial intelligence, especially in the domain of games and planning [3, 5].

MCTS consists of four phases (Figure 3.6), which are repeated a certain number of iterations (usually the search is limited in time) [14]:

1. Select (England. Selection)

   In this phase of the root node is selected one of the nodes of offspring, using the equation UCB (3.5). From the selected node again, choose one of the offspring and the process is repeated until reaching a node that is not fully widespread (ie. All of his children have not been added to the tree) [14].

$$in_i = with_{and} + C \cdot \sqrt{\frac{\ln(n_p)}{n_{and}}}, \qquad (3.5)$$

   Explained the equation UCB (England. Con fi dence Upper Bounds) (3.5) (Source: [13]). The equation used in the context of the strategy ceiling confidence tree UCT (England. Con fi dence Upper bounds for Trees) [3]. In equation

   with and determines the final score of the child *and,* which he was awarded a win 1 point, defeat 0 points. Number n_{and} and n_p refer to the number of child visits *and* My parents and *p.* ˇ number C is a constant that smooth Nava relationship between research (England. exploration) and yields (England. exploitation), and, in practice, be determined empirically. [13]

2. Extension (England. Expansion)

   From the last of the selected node and the previous phase randomly selected node (a child) of the L, which is not added to the tree [14].

3. Simulation (England. Playout)

From L nodes algorithm simulates a game that starts in the state of node L, but at the end of the victory or defeat (at the end of the game). Gestures can be random, but may better reflect the actual playing and so the algorithm achieves more reliable results, but it is necessary to remember that the more complex simulations reduces the number of simulations per second and thus at the same time reaches a minimum iterations MCTS [14]. It is necessary to define a compromise between these two parameters [2].

4. Updating (England. Backpropagation)

At this stage the propagated result R from the initial leaf L to the root node and thus to update all the nodes between them [14]. Update the number of visits to the node and point node (but only when the victory of simulated games).

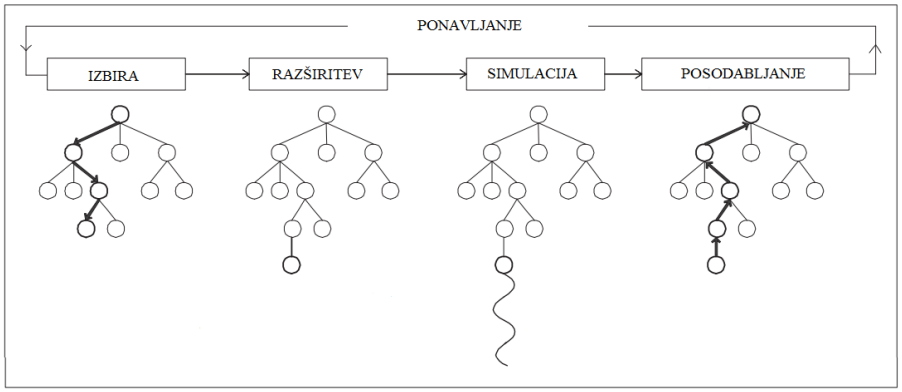

Figure 3.6: Phase algorithm tree search Monte-Carlo.

3.5 Application of the algorithm in the game MCTS Yinsh

The administration method of the tree investigation Monte Carlo to play Yinsh had to be somewhat reformed structure node that was used in the Minimax method. The structure is as follows:

- The index of the player (1: white, 2: black)

- State of the game board (positions white / black chips / rings)
- Position from which the current player made his move and thereby came to the state of the game board
- Position, which has moved to the current player and thus came to the state of the game board
- Link parents hub
- List of child nodes
- Number of visits nodes
- Number of victories hub

Time to move implementation was limited to 10 seconds, which means a different number of iterations of the initial situation, from which it move. Namely, at the beginning of the simulation take longer time than more towards the end, when the result is already example 2: 2nd To mention a specific number: the number of iterations in the 10 seconds in the first turn after installation rings averages between 3000 and 4000 iterations tree investigation of Monte Carlo. 10. In turn, the number of iterations already over 10,000 in the last moves as well as the 500,000th

3.5.1 An empirical analysis of the constants C

Let's equation (3.5). First part *with and* $\frac{}{n_{and}}$ concentrate on using better estimated strokes, while the second part $\frac{\ln(n_p)}{n_{and}}$ dedicated to researching new, are not investigated nodes [3].

choice of constants C We have in our program determined empirically, whereby was tried with the following values:

0.5, 1.0, 1.41, 1.8, 2.0, 2.5.

We carried out automatic playing games, where we try after 10 games for each combination of constants. Most victories gave the value of 2.0, so we have always used this constant.

3.5.2 Choosing the type of simulation

As already mentioned, more advanced type of simulation leads to more reliable results. Winands and Nijssen [14] in his study mention -pozresne simulation (England. -Greedy playouts), where the probability gesture simulation tions make randomized. Otherwise, the use of heuristics, which reflects the actual game.

We opted for simple heuristics, which is actually taken from the first type of evaluation functions mentioned in section 3.3. A player who is in turn the simulation, so with probability 1 - selects move him to the next step bring the greatest number of chips.

This is a heuristic, which is relatively well proven already in the Minimax method is also time completely unpretentious.

In the automatic testing of the version tree investigation with naprednejsemi Montecarlo simulations proved slightly better than the version with a completely random simulations (19 wins out of 30 games). Therefore, for further testing using the version of the -pozresnimi simulations.

3.6 The initial formation of rings

Unlike traditional abstract games such as chess and checkers, the game Yinsh variable initial formation. In fact, at the beginning of the possible millions of different layouts [7], which in our view means that at this stage pointless gestures count by methods such as Minimax and MCTS. We believe that the strong initial installation helps mainly good strategy game Yinsh experienced players. Thomas Debray in the description of your smart player states that it is particularly important that the player remains flexible and thus does not put rings on the same line, but keep them close to the center of the game board, which, according to professional players decided centrally dominance [26].

3.6.1 Setting rings according to the opponent

Prior to testing, carried out an analysis of gaming opponents. In doing so, we come to the following conclusions:

- Ăgainst human opponents is a smart player Yinsh important advantage: the man often practiced *edge strategy,* which means that you're on the verge of building an indestructible network that can not be reversed (Figure 3.7). While in general a man can not think of depth exploration of the computer, in the case of boundary strategy could be considered deeper than the computer.

  Already a ring can provide a good foundation to set up such a network. If both players know this strategy, you You can prevent the formation of rings that one important player Equipment supremacy on the edge of the game board.

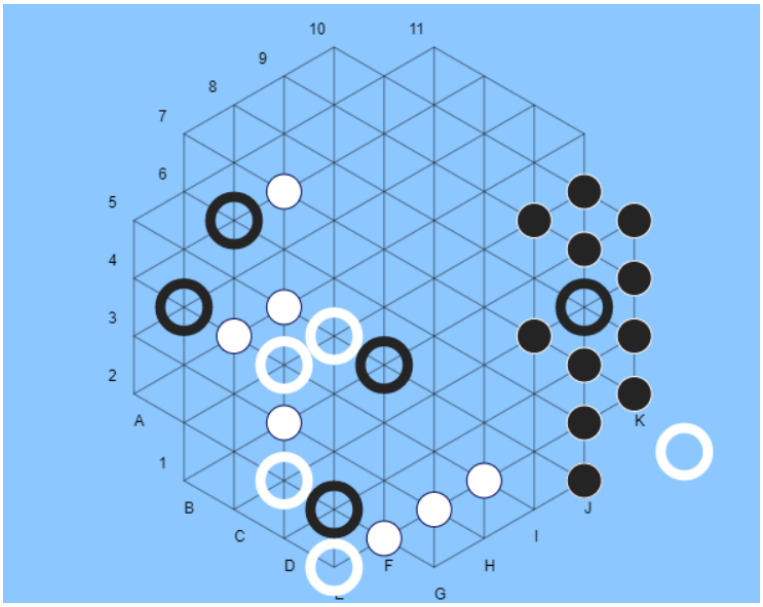

Figure 3.7: An example of a situation that is quite common in Yinshu: player're on the verge of building a network that can not be reversed.

- Ĭf the initial formation of rings opponent Given our layout remains the same, it must be smart player to provide a slight randomness, as

this opponent makes it impossible for the time to develop the perfect combination of initial positions rings according to our layout and gain an advantage, because we are a player does not have a strong strategy. This is particularly true for the human player because of its layout can not be analyzed in advance, as this can be done for the existing version Yinsha.

- Implementation of David Peter was already in the game against a human player proved to be very dominant, if man does not use boundary strategy. Using the edge strategies can a man who eventually wins overcome a smart player David Peter. Therefore, try this fact also be used in the implementation of our smart player.

Because describes the different behavior of opponents, which will be used as a test against our smart player, we decided to implement two strategies for a ring:

1. In the case of human opponents wish to introduce randomness in a while an opponent we want to prevent the construction of the boundary network. The construction of the network often try to play a bot mention Boardspace, therefore, in these two instances ring smart player decided to ask the following rules:

    - Ǐf possible, set the ring directly next to the last empty rings opponent. Position has up to six possible adjacent positions. The desire to dynamism position is selected at random. By placing the appliance adjacent rings with opponents (at least partially) to prevent playing strong edge strategy. By coincidence prevent him to gradually develop the perfect combination of initial positions rings and gain an important initial advantage.

    - Ǐf this is not possible, place the ring which is placed in the middle of the game board. By maintaining our flexibility and centrally dominate the game board.

2. In the case of players David Peter and Thomas Debray see that our opponent allows a smooth ring's edge. Therefore, here we try to edge strategy to integrate our player: If a player puts the ring on edge, place it next.

   If a player does not ring puts on edge (as it happens in the implementation of David Peter), put 2 rings on the same edge, and the rest as close to the middle. Rings on the verge of acting together, to build a strong network at the edge of the game board and when the protect. However, it is good to have some rings in the middle, and the other ring is placed on the middle of the game board.

3.7 Removing rings

After each collected a series of chips from the game board need to remove the ring of their color. In removing the rings have to be very attentive, because one ring less means less control over the board. In addition, the rings have an important function of blocking an opponent and protect our species. To remove rings developed a simple algorithm that selects the least useful ring according to the current state of the game.

For each ring is done as follows: with the game board is removed the ring *R*. On the game board without rings *R* Find the most probable move of the opponent. This is a move that the opponent brought more points (MIN). So we have removed every ring a number of points that we can bring this removal. Therefore choose a ring that allows the removal of the opponent and turn bring more points.

Chapter 4

Optimizing

We have already mentioned that the method Minimax time very complex. In Chapter optimization is therefore focused primarily on methods that have been used for improving the temporal results of our smart player. These methods can speed up the time thinking, and can help to achieve the maximum depth investigative trees.

In the description of the optimization we will focus on article *Solving games: Applicable dependence of solving prodecures [*7] that the game Yinsh described as one of the most complex games in terms of solvability.

We evaluated the said article by testing these methods. At the same time, we proposed some of his methods, which were integrated into the implementation of the smart player. We will mention the methods used by the Yinshu can not be implemented due to their different nature.

4.1 dimensions of the game Yinsh

Dimension investigative trees games (angle. The size of the game-tree) is usually estimated by potentisation average number of moves to the position and the average number of plays of the game. With the game Othello enlargement factor also is 10, the average number of moves is 58. The estimated size of the investigative trees game Othello is so 10^{58} [7]. The enlargement factor in chess is 35,

average number of plays is 70 [11], which means that the size of investigating the tree 35^{70th}

The game Yinsh we came to approximate values as follows: for example, we have taken 10 of automated games between greedy and advanced smart players. We monitored the number of moves and the number of the next possible positions of the current situation. Enlargement factor on the basis of this empirical analysis is 47; number of traits in common amounts 51. Estimated dimension investigative trees games Yinsh thus amounts to 47^{51} which is much greater than the dimension of the game Othello and slightly smaller than play chess. Because of this magnitude, of course, to investigate the final status of the game is not possible, therefore, they tend to limit the search depth (England. Depth of search; horizon).

4.2 Opening Series

Despite many improvements Evaluation and investigation of algorithms for many programs still shows weaknesses especially in the opening stages of the game, because programs often do not have a satisfactory strategic planning. To solve this problem are widely used database openings, which are stored in a sequence of moves that force good initial strategy [4].

Unlike games such as chess and checkers, the game Yinsh dynamic opening. This is due to the layout of rings, which is not known in advance (such as, for example, a known initial Saha).

Ǐf you would like applied to the method of collecting opening game Yinsh should have saved the initial moves for each starting hand positions rings. However, since these combinations of millions, the opening of the collection as an optimization method omitted.

4.3 transposable table

At trial the tree game may appear a large number of nodes that appear identical to the state of the game. [7] By reproducible conditions can be caused by

intrachanges - various permutations of successive moves, which come to the same state of the game board.

Suppose that the White available campaign a_1 which can black answers with b_1 independent action on the other side of the game board a_2 It may be followed by action b_{Second} Then the sequence moves [a_1 b_1 a_2 b_2] and [a_2 b_2 a_1 b_1] They stop in the same state of the game board. It is therefore, wherever possible, be convenient to remember what has been deliberated on this position when it was last inspected, since thereby saving re-investigation of the same sub-trees [15] and thus reduce the size of the search space. For this reason, a smart Chess players usually saved transpozicijsko table, which is implemented in the form of increased atomization table (angl. hash table) and stores information about the positions that have already been studied [25].

The use of transposable tables can have a dramatic effect (a doubling of investigation depth in chess), it is necessary to know that you can save transposable table impractical, if we evaluate millions of nodes per second. At this point, it is necessary to decide which nodes to store and what not. [15] Information, stored in a transposable table for each node should be as small as possible. In this way the number of nodes stored in the same unit of memory [7].

4.3.1 Applying transposable tables to play Yinsh

The paper [7] is written argument that turning chips makes transposable table unusable. This wider view partly true, as a result of the dynamic of the initial layout and the fact that every gesture added to the board in November token, we rarely get to exactly the same state of the game board.

But it is necessary to mention the impact of transposable tables on remand tree current moves, which happens often, that at the same level we get to

the same sheet. Let's look at figure 4.1. Suppose that the white player moved from A4 to A5. Followed by displacement of black, which is moved from a C6 to C7. Below is a shift of white, which moves from C4 to C5.

If white player reverses its stroke and is moved first from the C4 to C5, after displacement of black (6 to 7) is moved from A4 to A5, the result is the same. Scheme investigating nodes of the tree is shown in Figure 4.2

According to this analysis, it is also clear that such a move, at least in this state board games, a lot. It should only be noted that the same situation may occur only on the same level investigative trees.

Therefore, the implementation of a smart player games Yinsh introduced transposable tables that appear in the form of a spray tables, where one table is used for a sputtering moves at the same level of current investigative trees. Nodes that are repeated, simply do not add more in the investigative tree.

State of the game board is stored in a string of characters that represent a regulated position of rings and chips according to the following rule:

white rings, rings of black, white token, token black.

For the current situation, this series was *A4C4F6F7G5; C3C6E7G4I8; B5F8F9 ;.*

4.4 Sorting moves

Cutting Performance of alpha-beta is dependent mainly on the order of investigating node. Let's figure 4.3. Tree (a) and (b) can not cut it any successor node D, because it was the worst successor (in terms of MIN) is generated first.

If it was the third successor node D is generated first,

You could miss investigating two other nodes (cutting). Sorting nodes of course, can not be made perfect - if it would be possible to sorting allows to smart player plays without error. [15]

If this were possible, the algorithm alpha-beta can be explored only *About* (b_m $\overline{2)}$ nodes (instead *About* (b_m), which is the complexity of the underlying Minimax).

This means that the branching factor b decreased to $\sqrt{b}$ - Chess This mandate

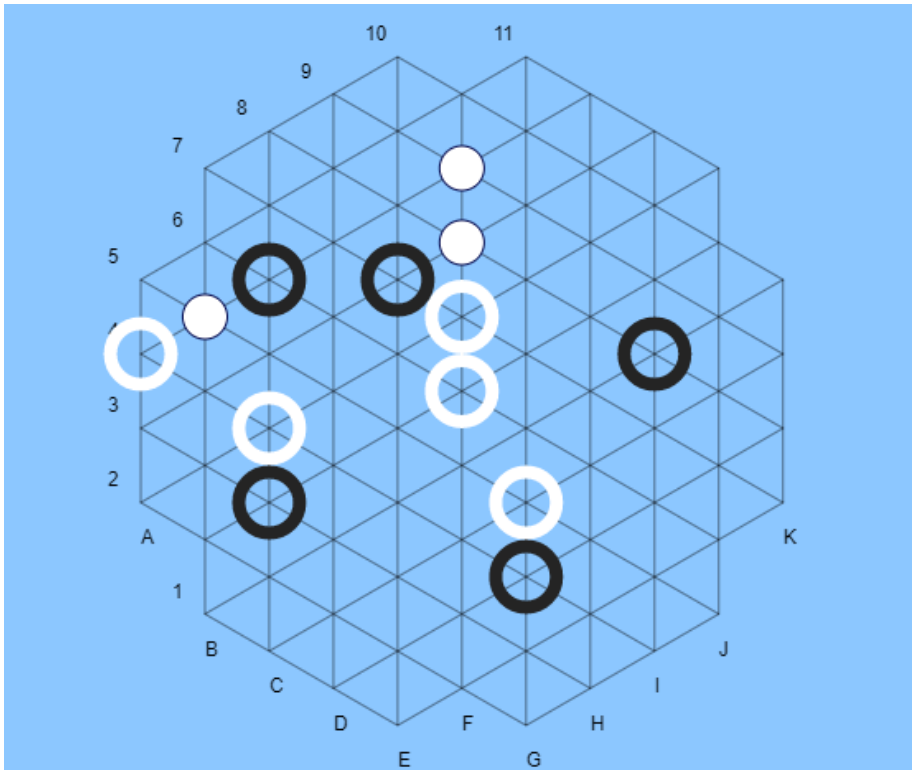

Figure 4.1: For example, the state of the game board.

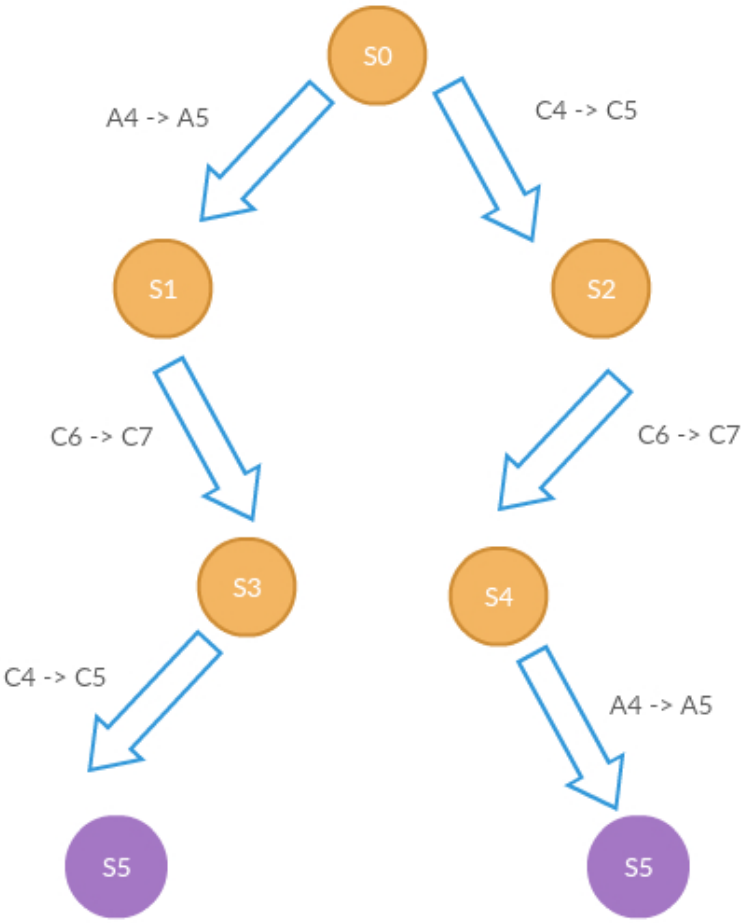

Figure 4.2: The scheme simple investigative tree, which represents transposition. Start node represents the state of the game, shown in Figure 4.1. According to both the displayed paths lead to the same state of the game board. The figure for simplicity does not display all possible nodes, resulting from the current state of the game board.

It considers that the branching factor is reduced from 35 to 6 If the followers are reviewed put in a random order, the order number of nodes examined *About (b* 3 *w* /4).

In chess, the mere function of sorting enough to reach investigated only about *About (b* *m*

2) node.

When sorting, you can also help by searching the sites that have already proven to be very good in the past. This may mean relieved move, you can search the current node in the past. A common way to achieve the latter, the *iterative deepening,* which means that the depth gradually moving - depth are increased only when we've been looking for all nodes in the current tree level [15].

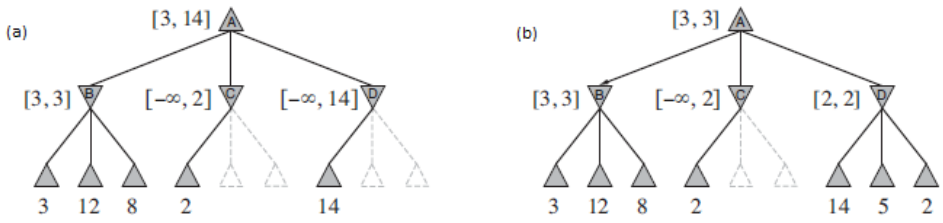

Figure 4.3: Investigative tree cutting alpha-beta

Editing features are also integrated into our smart player: Upon investigating nodes first create all descendants, then they are assigned points according to the same evaluation function, which is estimated finishing nodes. Nodes are then ranked - the player moves to low MAX, the MIN player moves upwards. We expect namely that the situation on the lower level status indicator is slightly higher (deeper) level. Because first of all we are looking more promising nodes (from the perspective of the current player), we can cut more branches such that the current player will certainly bring fewer points at us and therefore not even interested. This allows the nature of the alpha-beta is described by equation (3.1). Despite the fact that sorting and evaluating all the nodes brings additional complexity figure, that the technique significantly reduces the space and time of the investigation. Details optimization will be described in section

4.7.

4.5 Selection level in relation to the development of the game

Yinsh the type of game where the size of the smaller trees investigating the course of the game (England. End-games). In the game, because it reduces the branching factor of the tree, as the number of the next possible moves during the game falls. For instance, the average number of moves following the first stroke play with 80 for the final moves this number is reduced to 25. This is due to the reduced number of rings and the charge due to the game board, which restricts the movement of the player rings.

We decided that we will use this information to build our investigative trees. Without the methods described in section Optimization, we are in a timely manner (under 10s) achieved four levels of investigation. With all the optimizations described, and using the information on the reduction of the investigated trees during the games do empirical analysis of the depth of investigation. Smart player depth to determine the following:

- Depth 4, if the number of moves the game less than 15 and the number of followers of a given state of greater than 60th
- Depth 5 if the number of followers of a given condition is less than the 60th
- The depth of 6, when the number of followers of a given condition is less than the 15th

4.6 Threats and opportunities

The games often come to a situation where we are threatened with a sudden loss or sudden victory (in chess this is position *chess,* which is an attack on the king). Such a situation can significantly reduce the search space, because the opponent to update its priorities and primarily to prevent the opponent the victory. Also in the game Yinsh may apply to this situation.

If the figure 4.4 black player white does not prevent the installation of the fifth tokens in a row, will next turn won the third ring and both won.

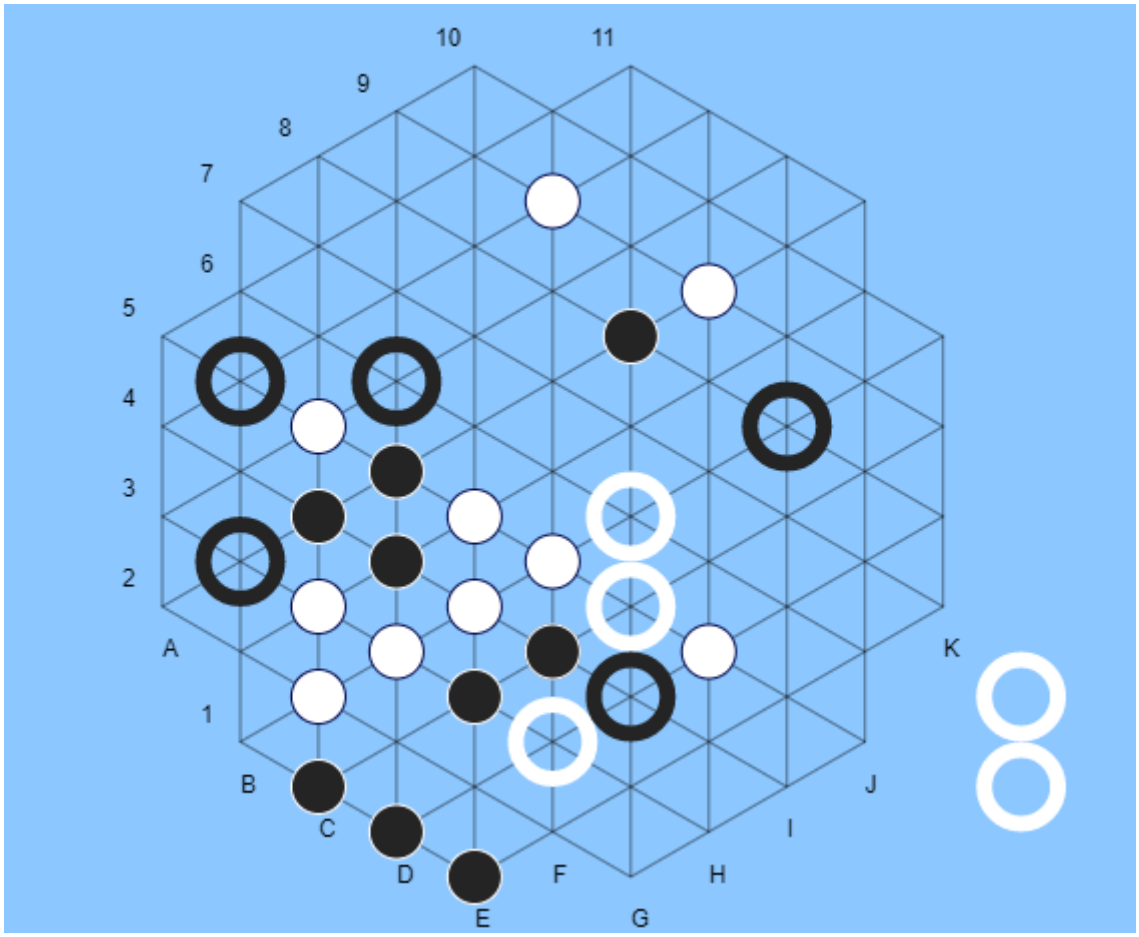

Figure 4.4: For example, the white player opportunities to win in the next turn. At the same time it poses a threat to the black player to defeat.

The paper [7] use strategies threat and win assessed as almost useless, because it happens very rarely. Yinsh requires collected three rings, as well as the player with a new kind of weakens your position, because he crimp band with one less reduced mobility, as well as the loss of five chips of their color.

Despite the fact that this is the case, we disagree with the assertion that the strategy sudden victory or defeat ineffective. In our advanced evaluation function mentioned in section 3.3.1, and considering that confers 10000 points for a win and -10000 points to defeat; collected for five chips hub evaluated by 100

vertices at the opponent row five but -50 points. Thereby achieving a better balance between reclaimed series of chips and victory or defeat. For example, we can achieve that when we are on the achieved result 2: 1 on the opponent more focused on blocking an opponent victory, achievement of our species is of secondary importance here.

If your opponent can win in the next turn, he does trying to prevent. If we are not in danger, but again focusing more on getting kind of our players.

4.7 Effects of optimization methods to investigate

In this chapter we will primarily interested in how the above optimization affect the acceleration of the operation of our smart player. We will also examine how to determine the state of the game affect integration sorting features, transposable tables and a combination of both optimization methods. Also indicate what the situation is, without any optimization methods. We were interested to CAS implementation, the level of investigation, investigated the number of nodes and the number of call evaluation functions.

The last parameter was added for the following reason: If the searched node *Gazette* tree, compute only the assessment of the current state of the game board and return it.

If a node is not searched Journal of trees, due to the needs sorting and evaluation of stability calculated estimates of all child nodes of the same formula to assess the situation in the leaves. Thus, the time complexity of the investigation leaves much smaller than the temporal complexity of the investigation of the parents who must evaluate all of their children.

The number of the investigated node by a lack of information for the interpretation of the duration of the investigation, so we made the tables is the number of call evaluation functions.

Let's look at Figure 4.5, 4.6 and 4.7. Black player has in these three states of possible 88 (Figure 4.5), 68 (Figure 4.6), and 20 (Figure 4.7) moves. The level of investigation to test these three states is four, in the latter case, we add the interest of transparency, the measurements of the depth of five. Tables 4.1, 4.2, 4.3 and 4.4 summarize

How much time is needed each approach to determine the next best move.

With optimizations in any event, achieving a substantial reduction in the number of explored nodes, which means a significant acceleration of the implementation time. In the case of the situation of Figure 4.7, in the actual game can be explored to a depth of six, since the execution time would be about 4 seconds.

The table clearly shows the reduction in investigating the tree during play: we can see that in the initial stages of the game the number of nodes of the tree depth investigated four without optimization of the 300,000th The number of explored nodes of the tree a depth of four in the intermediate stages is about 40,000th The number of explored nodes tree depth of four in the final stages of around 2000th

As the best means of optimizing estimate sorting features Through advanced evaluation tool. Despite the increasing complexity of sorting the tables clearly show that sort of moves to facilitate substantial reductions investigative trees. Nodes on the first level can largely indicator nodes at higher levels.

Even transposable tables are reducing the size of trees investigating, but to a lesser extent. Thus, in Figure 4.7 the number of explored nodes at level five through transposable tables dropped from 35 070 to 23646th time It decreased from 2876ms to 2630ms only. Transposable table therefore shows a decrease in investigative trees, a time of the survey is due to the complexity of the tables is significantly reduced.

The measurements were performed on a computer with a processor *Intel (R) Core (TM) i5-4200U CPU @ 1.60GHz 2.30 GHz.*

4.7.1 Influence of optimization methods to structure nodes

Due to the administration of optimization methods on our smart player, it was necessary to partly change the structure of our nodes. So we have to rest properties added to the list of all the successors of the current node. In this way we can ensure the sort of moves.

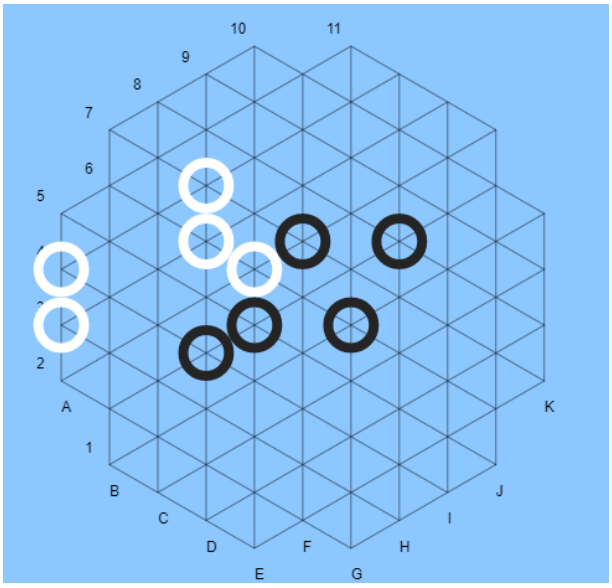

Figure 4.5: For example, the state of the game where the black player in the possible next move 88 moves.

Table 4.1: Analysis of optimization methods to figure 4.5. Depth investigative trees amounts to fourth

The approach	Execution time [ms]	The number of explored nodes sc	The number of call evaluation functions
without optimizations	24,904	308 843	1386817
sorting	8038	20613	373 332
transposable table	20,830	293 076	1104506
A combination of approaches	7368	18,070	353 778

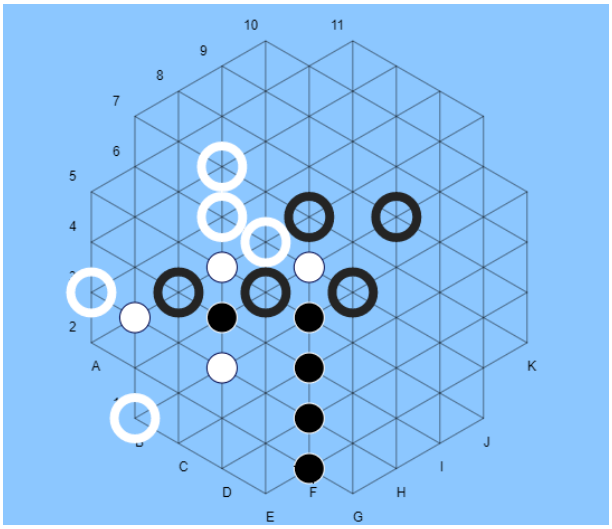

Figure 4.6: For example, the state of the game where the black player in the possible next move 68 moves.

Table 4.2: Analysis of optimization methods to figure 4.6. Depth investigative trees amounts to fourth

The approach	Execution time [ms]	The number of explored nodes sc	The number of call evaluation functions
without optimizations	16,211	41,765	718 873
sorting	6863	6836	297 442
transposable table	13136	29906	485 245
A combination of approaches	5214	12,694	254 900

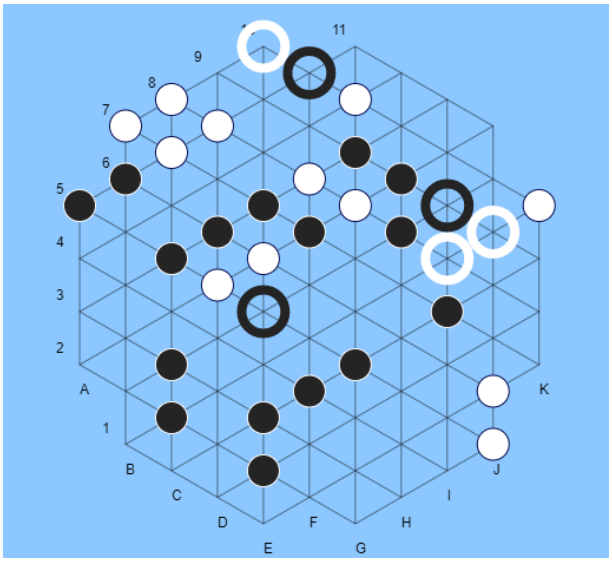

Figure 4.7: For example, the state of the game where the black player in the possible next move 20 moves.

Table 4.3: Analysis of optimization methods to figure 4.7. Depth investigative trees amounts to fourth

The approach	Execution time [ms]	The number of explored nodes sc	The number of call evaluation functions
without optimizations	853	2194	15507
sorting	699	1064	11778
transposable table	822	1730	11,844
A combination of approaches	534	968	10562

Table 4.4: Analysis of optimization methods to figure 4.7. Depth investigative trees amounts to fifth

The approach	Execution time [ms]	The number of explored nodes sc	The number of call evaluation functions
without optimizations	2876	35,070	97,414
sorting	1177	7740	25,941
transposable table	2630	23,646	63,727
A combination of approaches	716	5964	21317

Chapter 5

Implementation of

Rules of the game and a smart player had been implemented in the Java programming language (version 8). For the purpose of testing against a human opponent, we implemented a simple user interface (Figure 5.3), made in the programming language Javascript and HTML. We back-end and the front-end communicating with the help of the HTTP requests, which are sent in the form of data in JSON format (Figure 5.1). Given the move to the user interface, back-end part of obtaining one of three forms of commands to format JSON and answers with a package similar forms (Figure 5.2). When setting rings are also transmit information about the player and the position preference rings. Upon moving to send information about the player, the position of the ring, has moved and the position in which the ring is moved. When collecting five chips to send information about a player position rings, which is moved, the position to which the ring is moved and removed from the list of five chips and rings. Here again we emphasize that a player with one stroke creates two independent series of five chips.

Figure 5.1: Figure communications between the front and back part of the application.

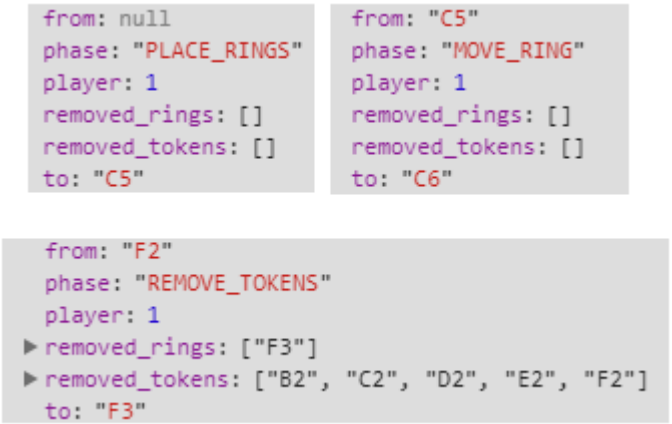

Figure 5.2: Examples of commands in the format of JSON, which serves as a communication between the back-end and the front part.

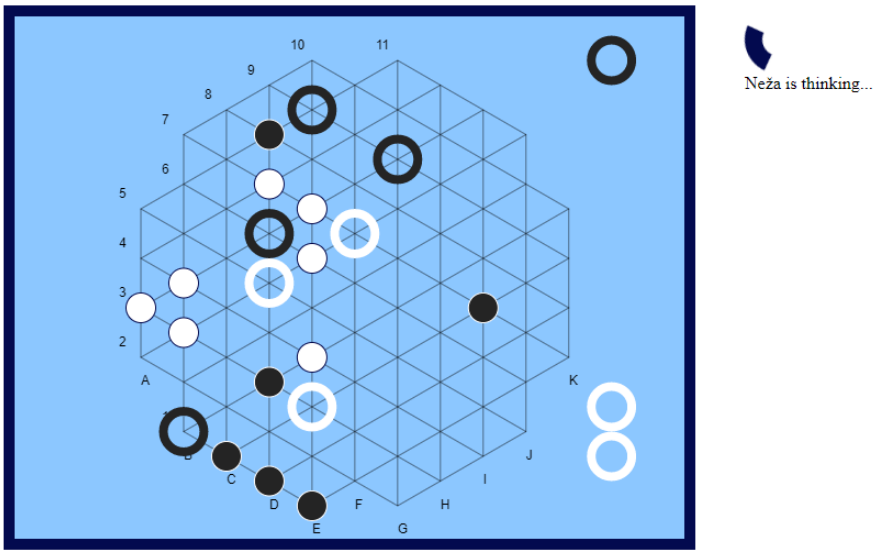

Figure 5.3: Figure user interface. In addition to the game board as shown on the side of collected rings. On the right side is the loader, which rotates at the time thinking smart player.

Chapter 6

results

6.1 Comparison of advanced, greedy and MCTS version of the smart player

Prior to testing smart player against current opponents and human figure, which is our strongest version of the smart player. We compare three different versions:

- Minimax by consuming evaluation function
- Minimax with advanced evaluation functions
- The tree search Monte-Carlo with -pozresnimi simulations

During each combination of methods odigramo 30 automatic games, with players alternating instituting colors (white and black). The results are shown in Table 6.1. It is clear that the Minimax with advanced function evaluation version of our strongest, so he decides that this version appeared in the test against computer and human opponents.

Table 6.1: The results of the comparison of the three versions of the smart players

player		Result
Advanced method: After	serious method	18: 12
Advanced method: MCTS		28: 2
Follow Serious method: MCTS		28: 2

Tree method of investigating the Monte-Carlo was already successfully applied to a range of games such as Go, Amazons and Lines of Action [1]. Well it turned out well in games with incomplete information, such as Scrabble, Bridge or Scotland Yard [3]. But there are still a large number of traditional games, which is the preferred method of Minimax. Such games are, for example, chess and checkers. MCTS builds a tree that mainly focuses on the most promising way, but still not suitable for investigating premises contain a number of *traps* At the lowest levels of the tree. Such situations are common in chess and require precision and tactical play to avoid defeat. MCTS because of its randomness and method of evaluation can easily be overlooked or underestimates the importance of the move. On the other hand, it may MCTS statements in domains such as Go, where they fall rare or do not occur until the final stages of the game. [1]

Yinsh is symmetric game with complete information for two players and the type of victory-defeat (England. Zero-sum game). Even Yinsh for turning chips contain many pitfalls. This game is for these properties similar to chess, so above reasoning can satisfactorily explain the improved performance of the algorithm Minimax algorithm to MCTS.

6.2 Portal Boardspace

Portal Boardspace is a popular web portal for playing table games. It is possible to play against other human opponents, added a few PA-

to turnover gamers. In addition to well-known games such as chess, checkers and Go are also mention all the games Gipf project.

The portal added three smart players play Yinsh: *dumbot, weakbot* and *smartbot.* Their author is David Dyer, players are implemented using alpha-beta search with real-time scaling depth investigation during the game. They are described as very good against inexperienced players, a negotiable against more experienced players [20].

With username *missRobot* We have registered on the portal and through the implemented smart player who uses the method of Minimax with advanced evaluation function played 10 games against *dumbotu* and *weakbotu* and 20 games against *smartbotu.* The results are as follows:

Table 6.2: Results smart player against the botom mention Boardspace

player	Result The percentage of victories	
missRobot: dumbot	8: 2	80%
missRobot: weakbot	9: 1	90%
missRobot: smartbot	15: 5	75%

Games played so climb on top of the current scale success in the game Yinsh (Figure 6.1).

6.3 Implementation of David Peter

David Peter by German fi zik is its implementation of a smart player games Yinsh posted online. [23] His smart player is implemented by upgrading the alpha-beta, *Negascout.* Negascout requires a good sort of moves and thus speeds up the functioning of the alpha-beta. When bad sorting algorithm function is worse than the investigation of alpha-beta. The algorithm has brought good results for computer gamers chess. The advantage of the algorithm is mainly

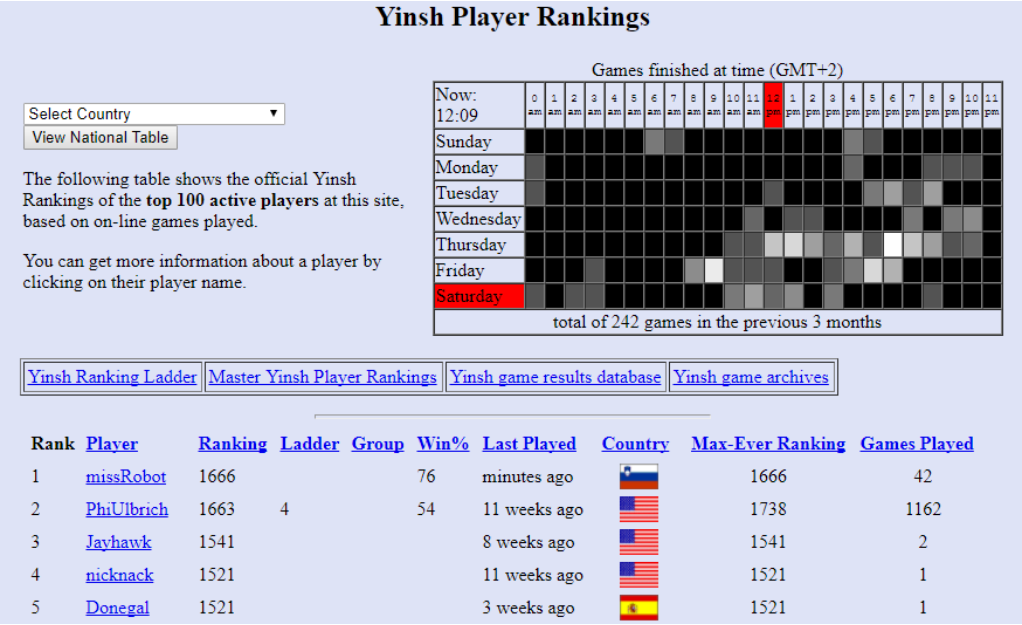

Figure 6.1: The success of our smart player on the portal Boardspace.

This, in spite of speedup does not compromise the accuracy of the returned result and also returns the same result that would be returned Minimax algorithm [18].

With the help of installing rings mentioned in section 3.6.1, and rewarding kind of stability, we encourage our players to begin to build the kind of edge. Since the implementation of David Peter to create a kind of on the sidelines respond too late, you can use this strategy player convincingly overcome.

Implementation of setting rings and single gameplay for both players does not contain randomness, so the game is unique and always repeated the same moves. But regardless of playing this game, we can say that we are convincingly better player, because it defeats with a score of 3: 0 (Figure 6.2).

6.4 Implementation of Thomas Debray

Smart player Thomas Debray in 2008 Yinsh participated in the tournament and won third place. In the first place was Pim Nijssen, a researcher at Maastricht University with a series of important research in the field of artificial in-

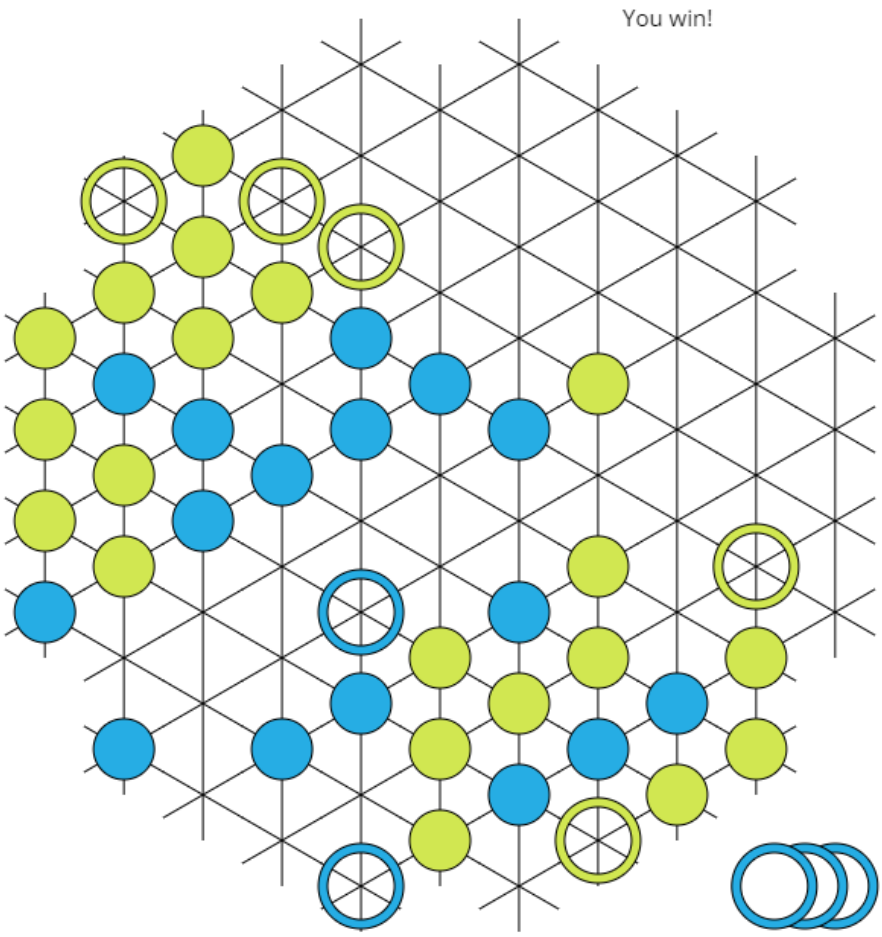

Figure 6.2: Victory against David Peter implementation.

teligence. T. Debray was during the tournament gained master's degree at Maastricht University.

The program, which reached third place, is available on its website [26], which describes the heuristics, which were taken into account in the evaluation function:

- Ǎmount collected rings.
- Ǎ The number of consecutive tokens in a row associated with a ring.
- The distance between the rings and the center of the game board. Positioning rings in the middle of the game board, according to Debray, it means greater flexibility and dominance on the game board.
- The distance between the token and the center of the game board.
- The difference between the number of chips a smart player and the opponent.

Weights identified above heuristics were optimized using an evolutionary algorithm by playing against earlier versions of the same smart player.

Smart player T. Debray he represented the greatest challenge. However, the observed deficiencies, which was mentioned earlier: the initial formation of rings allow the smooth formation of rings at the edge. Also here is a rings on both sides of the unique, but the player T.Debraya is still some randomness involved. Therefore, we are against the player played 10 games and won 5 of which all games were quite balanced and completed with a score of 3: 2nd

6.5 human opponents

The game was also tested against human players of different ages and experience. Inexperienced players who know the rules, know how to build their own kind and inverted opponents were defeated at the earliest stages of program development (the first type of evaluation functions in Chapter 3). The second type of players are those with more experience

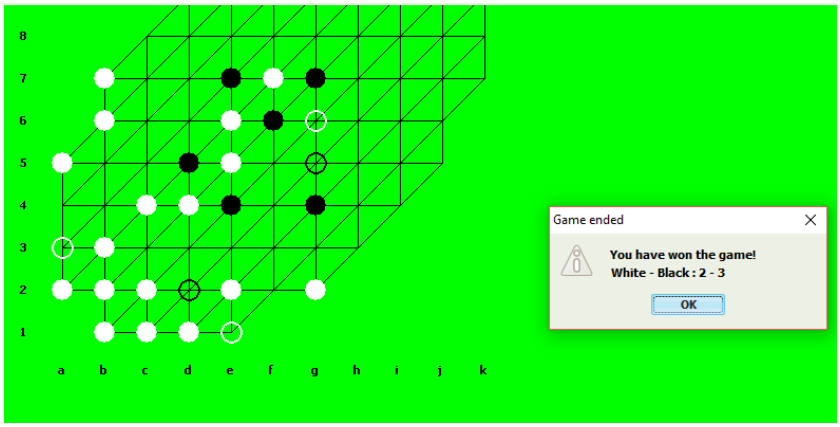

Figure 6.3: Victory against the implementation of the Thomas Debray.

and strategies that are not always among the best players abstract games. These people program without problems defeated. The program is featured as competitive, but sometimes even the most experienced players pushover Yinsha.

Chapter 7

Conclusion and future work

In the present work, we analyzed the game Yinsh and its characteristics. We implemented several different methods for playing a smart game player Yinsh. Gradually, we upgraded the evaluation function, keeping in mind especially the number of collected rings and the number of consecutive tokens on the game board. We came to the strong evaluation function, which together with traffic optimizations MiniMax delivers more than satisfactory results against existing implementation of a smart player. The player is also good statements against a human opponent, where he was only a piece of the most experienced players in the game Yinsh, but only occasionally.

We implemented a smart player with tree searching Monte-Carlo, but it appears as a competitive opponent. We believe that the reason lies mainly in the nature of the game Yinsh, much like chess. The game contains several traps (turning chips), but it is a game with complete information for two players and simple rules. For such games it is usually Minimax still the most powerful method.

In the behavior of our player and playing smart players observe interesting characteristics.

Players have problems with balance between the two events, which are interconnected:

- Picking kind or delay. Species collected at the same time means less

mobility, because the loss of one of the rings. In some cases, delaying collection of chips well as maintain mobility and dominance on the game board. Removing our species may provide that immediately after winning opponent. In this case, it is essential that first victory prevented an opponent. In other cases, it is essential that the species be removed immediately, even if it is not protected.

- Collected type or wins (collected 3 types). Game Yinsh has a very atypical nature: collected can be three types of chips. Often the question arises how to evaluate each of these events, which is also reflected in the behavior of smart players, who sometimes do not know how to judge whether it is better to pick the type of victory or prevent an opponent.

7.1 Further work

We are aware of the weakness of the initial layout of rings smart player. We believe that it would be here required a lot of research and consultation with professional players, which could provide insight into strategies that would make the game even stronger.

More information would be able to analyze the evaluation of the collected species and victory. As we mentioned, the smart players not determine the balance between whether it is in a better situation to pick the type or by the wait. To do so would be wise to follow the implementation of the Thomas Debray and use evolutionary algorithms to optimize the parameters of our smart player.

Given the fact that we are limited to the duration of playing with human opponents, it would make sense to use the time thinking of the human player to further explore the program [6]. Program after its move at the time thinking opponent also chooses the most probable move of the opponent. From this moves forward starts to build investigative tree. In the case of correct predictions game opponent as you extend the time thinking.

if moves

did not anticipate correctly, the tree values, investigation starts again.

It would be good to check the pathology Minimax Game Yinsh. sometimes

namely excessive depth investigation leads to less reliable results. Theoretical analysis namely mention that increase as the depth of a growing suspicion caused by heuristic function, which we evaluate the leaves of the tree. [10]

Due to the changing nature of the game Yinsh would be a good idea to use a method of investigating sleep (angl. Quiescence search). When investigating a standstill, the emphasis is mostly on *Restless* node. These are nodes for which there is a high probability that the estimated value of the game board in the next step has changed significantly. Such nodes algorithm investigates deeper than *quiet* node. In this way, we can get rid of the effect of a horizontal (angle. Horizon e ff ect). This may occur when investigating the fi ksno depth. Namely sheet can be estimated very well, but the next step, this value can be significantly changed. The algorithm also more unstable position investigate deeper than the still positions [17].

References

[1] Hendrik Baier and Mark HM Winands. Monte-Carlo Tree Search and minimax hybrids. and *Computational Intelligence and Games (CIG) 2013 IEEE Conference on,* pages 1-8. IEEE, 2013.

[2] Unprotected Belej. Analysis methods of investigation in the case of game Scotland Yard. Bachelor thesis, University of Ljubljana, Faculty of Computer Science, 2015.

[3] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M-hole Time, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener Diego Perez Spyridon Samothrakis, and Simon Colton. A survey of the Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI and games,* 4 (1): 1-43,, 2012.

[4] Michael Buro. Improving Heuristic mini-max search by supervised learning. *Arti fi tial Intelligence* 134 (1-2): 85-99, 2,002th

[5] Guillaume Chaslot Mark HM Winands, and H Jaap van den Herik. Parallel Monte Carlo tree search. *Computers and Games* 5131: 60-71, 2008th

[6] James Gillogly. *Performance analysis of the technology chess program.* PhD thesis, Carnegie-Mellon University, 1978th

[7] Marijn JH Heule and Leon JM Rothkrantz. Solving games: dependence Applicable Procedures of solving. *Science of Computer Programming,* 67 (1): 105-124, 2007th

[8] Jones mtime. *Arti fi tial Intelligence: A Systems Approach.* Upper Saddle River, New Jersey: Pearson Education, Inc. p. 167, 2015.

[9] Paul Kilgo and Dennis Stevenson. Dvonn and game-playing intelligent agents, 2012th

[10] Mitja Lovage, Ivan Bratko, and Matthew Gams. Why minimax works: An alternative explanation. and *Proceedings of IJCAI,* pages 212-217, 2005th

[11] T Anthony Marsland. A review of game-tree pruning. *ICCA journal,* 9 (1): 3-19, 1986th

[12] Fr'ed'eric J URGEN Mauss. *Einsatz des Monte-Carlo Tree-search-Algorithmus im 2-Personen-Strategiespiel Dvonn.* Technische Universite at Berlin, 2012.

[13] J Pim AM Nijssen and Mark HM Winands. Enhancements for multi-Player monte-carlo tree search. and *International Conference on Computers and Games* pages 238-249. Springer, 2010.

[14] Pim Nijssen and Mark HM Winands. Monte Carlo tree search for the hide-and-seek game Scotland Yard. *IEEE Transactions on Computational Intelligence and AI and Games* 4 (4): 282-294, 2012th

[15] Stuart Russell and Peter Norvig. *Arti fi tial Intelligence: A Modern Approach (3rd Edition).* Jones & Bartlett Learning, 2010th

[16] Tom' as Valla and Paul Vesel` y. Waltz: a strong tzaar-playing program. and *Workshop on Computer Games* pages 81-96. Springer, 2013.

[17] Diederik Wentink. *Analysis and Implementation of the game Gipf.* PhD thesis, Universiteit Maastricht, 2001.

[18] Algorithm: Negascout. [https://equilibriumofnothing.wordpress.com/2013/10/15/algorithmnegascout/; Visited August 5, 2017].

[19] Board game geek. [Https://boardgamegeek.com/; Visited August 5 2017].

[20] Board space: A place for board games. [http://www.boardspace.net/english/index.shtml; Visited August 5, 2017].

[21] Dvonner. [http://matztias.de/spiele/dvonner/dvonnere.htm; Visited August 5, 2017].

[22] Holtz. [Http://holtz.sourceforge.net/; Visited August 5, 2017].

[23] An html5 version of the board game yinsh written and Haskell / haste. [Https://github.com/sharkdp/yinsh/; Visited August 5, 2017].

[24] Playing against computer with ai. [Http://www.ntu.edu.sg/; visited August 5, 2017].

[25] Transposition boards. [Https://chessprogramming.wikispaces.com; Obiskano August 5, 2017].

[26] Yinsh: Heuristics. [Http://www.netstorm.be/content/2/7/; Visited fifth August 2017].

[27] Yinsh Rules. [Http://www.boardspace.net/yinsh/english/rules.htm; Obiskano August 5, 2017].