**Summary**

Title: Comparison and optimization of investigative algorithms on the example

abstract game Yinsh

In the master's thesis we are dealing with investigative algorithms with which

it is possible to find solutions for abstract desktop games. We pick you abstract

the game Yinsh, which is part of the famous project Gipf. The aim of the master's thesis is to find

and implement the smart player of the game Yinsh, who is able to beat it already

existing smart players implementation, and at the same time it is competitive in the game

against an experienced human actor. As an underlying method, we select an algorithm

Minimax, for which we assemble several evaluation functions, whereby

These are upgraded. At this point, we also analyze the presence of constants

in the evaluation function and find the optimal combination of them. Basic

Minimax due to the large binding factor of the Yinsh tree is not satisfactory

solution. Depending on the nature of the game, we propose several optimizations of the Minimax method,

we implement, test and evaluate them. We are also implementing

the Monte Carlo tree search algorithm, the success of which is compared

with the Minimax algorithm. Based on research and developed algorithms

we form a final solution. We also suggest possible improvements and further

work. Our final solution brings very good results: convincingly overcomes

most of the existing implementations, but it is also capable of overcoming the very experienced

A human actor.

**Key words**

artificial intelligence, investigative algorithms, project Gipf, Yinsh, Minimax

**Chapter 1**

**Introduction**

The GIPF project consists of six abstract desktop games manufactured by Chris Bryant.

This is a type of full-information game for two players. Project

represents a major challenge in the field of artificial intelligence, since the resilience is contained

games, according to Heula and Rothkrantza, a complex problem [7]. Games

the Gipf project is played with simple figures that lay on the systeem

playing board. The purpose of the project is to create a framework of various games, whereby

each game has slightly different rules and figures (s) with which it is

it is possible to compose new types of games [17]. On the board game Board Game Geek

[19] The Gipf Project Games were rated as part of the best abstract games

very good: Yinsh is currently in second place, fourth on Tzaar,

on the fifth Dvonn. The other (Z`ertz, Gipf, P¨unct) are ranked below the 45th place.

There are some studies in the Gipf project. Most materials exist

for the game of the Dvonn: thus Mauss in his diploma work [12] dealt with

with the implementation of Monte-Carlo Tree Investigation, and Kilgo is in his

Research [9] made progress in the implementation of the Minimax algorithm for the game

Двонн. Dvonner [21] and Holtz [22] are advanced smart implementation

player of the game Dvonn. Waltz [16] is described as powerful in the article

the game program Tzaar, using advanced versions of investigative

methods for cutting Alfa-Beta trees and investigating with proofs

proof-number search). In her PhD thesis is a successful implementation

the smart player of the game Gipf created Wentink [17]. All the games of the project

Gipf has the implementation on the Board Space Web site [20], where

smart game players of the Gipf project have been described as being overwhelming primarily for

Experienced players. The only exception here is the Dvonn, the last version of it

robots on the portal also challenge the experienced players.

The problem we are researching is finding the best approach for implementation

smart player of the Yinsh game according to its complexity. About this

we will also be interested in how the knowledge and strategies of the experienced players can be integrated

in our implementation. We also want to test our implementation, and

otherwise, both on existing implementations and against the human actor. Dela,

which pose the greatest challenge to us, are:

• The implementation of David Petr [23], which is its open source implementation

the smart player of the game Yinsh also downloaded to the online version

games,

• Smart players (dumbot, weakbot, smartbot) on the well-known Board Board

Space,

• The implementation of Thomas Debrai, who is with his Yinsh bot in the tournament

Yinsh bots reached third place.

In the article Solving games: Dependence of applicable solving procedures [7]

the author also points to the severity and theoretical value

games of the Gipf project. Yinsh is defined as the most complex game in terms of

resilience. Yinsh has many possible starting positions: for the initial layout

there are 7.9 × 1014 Possible Positions. Also taking into account

the symmetry of the gaming panel, this number is not significantly reduced.

The structure of the master's thesis is as follows: in the chapter Yinsh we will be detailed

describe the game Yinsh and its rules. Below (Chapter 3) follows the description

all the methods and algorithms used during the master's course

work met and implemented. We will also describe how we are using these methods

applied to the game Yinsh. In the Optimization section, we will describe methods that

we have implemented them in order to speed up the thinking of our smartphone

player. The following is a description of the implementation (Chapter 5), where we explain it

selecting tools for creating a program, and describing the connection between the server

work and user interface of the game. The results will give us insight

in the competitiveness of our program - we will describe how smart we are

the player cuts off against already existing smart players, and also against less and

more experienced human personals Yinsha. We will be the existing smart players

also described. In the Conclusion section, we will evaluate the results and indicate the possibilities

for further work.

**Chapter 2**

**Yinsh**

The rules of the game are summarized according to the official rules of the game Yinsh [27].

**2.1. Components**

The necessary components for playing are:

• gaming plates,

• 5 white and 5 black bracelets,

• 51 zetons (on one side black, on the other white).

**2.2 The goal of the game**

Players start each game with their five rings placed on the gaming

plosci. The player may remove the ring from the playing board if five zettons reach

its colors in a row. The winner is the one who first removed three of his rings from

gaming plates. It is therefore necessary to collect three types of five zetons to win

its color.

**2.3 The course of the game**

**2.3.1 Layout**

A player starts a game with white rings. Players alternate their positions

rings on the cut-off of the slot playing area.

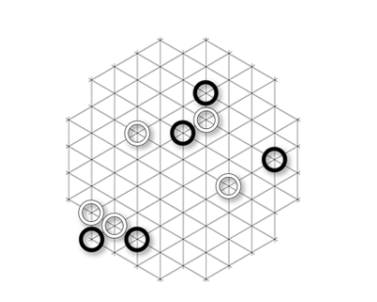


Figure 2.1: An example of the condition of the playing board after the setup of the rounds. Image source:

http://www.boardspace.net/yinsh/english/rules.htm.

**2.3.2 Shifting of the circles**

Each move begins by placing a zeton of its color in a ring of its color.

Then move the player to the following rules:

• When the ring moves, the zeton remains at the starting position of the ring (Fig

2.2)

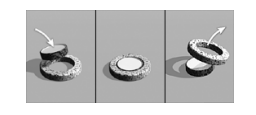


Figure 2.2: Moving a ring. Zeton remains in the previous position of the ring,

the ring moves. Image source:

http://www.boardspace.net/yinsh/english/rules.htm.

• The ring always moves in the straight line and on the free spot of the panel.

• A ring can skip one or more free places.

• A ring can skip one or more zetons of any color, whereby

they must be placed sequentially, i.e. no free places in the line.

If the ring crosses the zetone, it must be in the first free place for

Zithens in the line.

• The ring can skip first the free places and continue the leap

Through the zetone, but after the last zeton in the line jump to end.

• The ring must not skip other rounds.

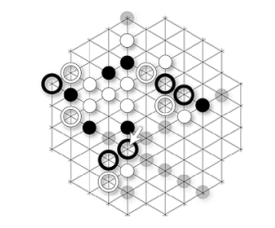


Figure 2.3: The picture shows the allowed movements of the marked black ring. Vir

images: http://www.boardspace.net/yinsh/english/rules.htm

**2.3.3 Turning the zeton**

If the ring during the move skips the zetone, all skipping kicks turn

(change the color). This does not apply to the zeton, which at the beginning of the move is placed in

moving ring, because it was not skipped.

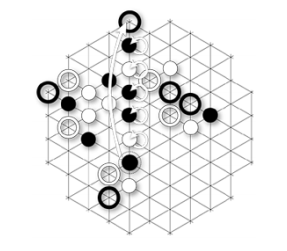


Figure 2.4: The picture shows the movement of the marked black ring.

Marked zettons that change color after turning. Image source:

http://www.boardspace.net/yinsh/english/rules.htm.

**2.3.4. Formation of the type and removal of the ring**

As part of the game, the player's goal is to form a series of consecutive five zetons

its color. With this achievement, the player removes a series of five zetons from the gaming

also removes any ring of its color.

If the player reaches more than five zettons in a row, he can remove it arbitrarily

five consecutive zetons.

When playing, it is possible for a player to get two types of five-way with one move

Zetones. If the line does not intersect, the player removes both types, and also eliminates them

two rings. If the queue is sequenced, the player removes any type (the other will be

after removal of the first incomplete).

The player can form a series of five zetons in his opponent's move

colors. In this case, the opponent will remove the tokens and the ring and continue

with your move.

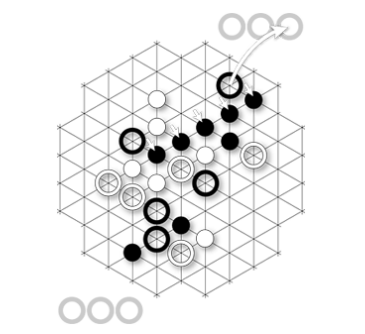


Figure 2.5: The image shows the formed series of five black zetons.Vir images:

http://www.boardspace.net/yinsh/english/rules.htm.

**2.3.5 The end of the game**

The game is over in the following cases:

• The player gets 3 rings and wins.

• The player with his last move acquires his opponent and his opponent's third row. V

this is the case where the player who made the move wins the match, because it was his

type removed earlier.

• If all the tokens were placed on the playing board before forming three

types of players, wins the player with a larger number of acquired rounds. If

they have accumulated the same number of rounds, the outcome is outstanding.

**2.3.6 Characteristics of the game**

The game Yinsh has some special features that help us in finding her software

solutions represent the greatest challenge:

• Unlike games of type n-v-type, it is necessary to collect as much as Yinsh

Three types of five zetons. It is necessary to determine the appropriate equilibrium between

victory and one set of five zetons.

• The initial layout is dynamic. Players are set up rings alternately

at any place of the gaming board. Unlike games like for example

lady and sah, is in general in the game Yinsh the initial layout at

Every game is different. This prevents the player from using predefined ones

initial moves.

• The game contains the turning of zetons, which means that the game state can be in

completely change the moment.

**Chapter 3**

**Investigative algorithms**

**3.1 Minimax**

Minimax is the most common algorithm for determining the best move from the current one

state of play. It is very useful in simple two-player games, where

players play against one another, all of which are the following possible moves

known to both actors (ie, playing with complete information).

Minimax is based on the construction of a search tree,

where each node represents a possible move. For each node

it is possible to calculate a value that estimates the power of goodness. Regarding

the best possible move, which follows the current one, can be selected from these values

state of play. In this, you need to pay attention to the alternating nature of games for two

player (two-player games), as each player chooses the best move

for him (and constitutes a loss for the opponent).

Minimax can use two strategies: the first is investigated the entire investigative

tree, to leaf nodes, which illustrate the final state

games (win or defeat). The second strategy is based on the limitation of the investigative one

trees according to a predetermined depth.

Minimax is an algorithm that uses depth research

search), while maintaining the minimum or maximum value of the node

the successors of a given node. When reaching the final node (sheet), it is calculated the value of this node by means of an evaluation function. This value is propagated

up the tree up to the root. On the levels where we are in line

smart player, the maximum of the value of the successor knot is selected; for

the opponent takes the minimum. Propagation through the tree up is therefore

alternating: maximizing the minimum and minimizing the maximum. With others

In words, we assume that each player will play the gesture that is for him

most promising [8]. The operation of the algorithm is shown in Figure 3.1.

We explain the timing of the Minimax algorithm. If it's maximum

depth of the tree m, and from each node, the next steps are possible, then

is the time complexity of the algorithm O (b

m). Spatial complexity is known

O (bm) for an algorithm that generates all actions at a time, or O (m) for an algorithm,

which generates only one campaign at a time. For depth searching it requires

only saving the current path from root to leaf, along with the remaining unreparated ones

neighbors of all knots on the go. When the node is expanded, it can be

removed from memory as soon as all of its descendants have already been researched.

For real games, such time complexity is completely impractical, a

the algorithm serves as a basis for mathematical analysis of games and for more practical ones

algorithms [15].

Pseudocode 1 shows the general Minimax algorithm summarized by [24].

**3.2 Alpha-beta cutting**

The main problem with the Minimax algorithm is that the number of investigated nodes is investigated

grows exponentially with the depth of the tree. The field of exponential investigation does not

we can remove it, but there is a way to effectively halve the exponent.

It's possible to get exactly the Minimax value without having to investigate each

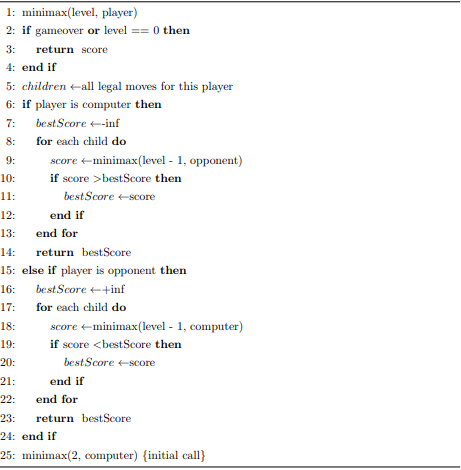
nodes of a tree. The algorithm that allows this is cutting alpha-beta. Enables

obtaining the exact Minimax value, but at the same time discarding the investigation of branches that do not

they can influence this value [15].

Let's take a look at Figure 3.2.

Algorithm 1 The pseudocode of the Minimax algorithm



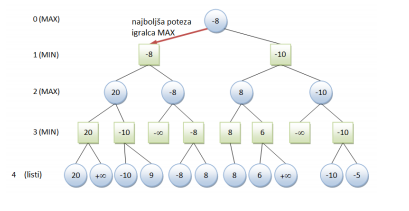


Figure 3.1: The image shows the operation of the Minimax algorithm. Blue knot

(circles) represent the opponent's moves, and the green node (squares) is moves

smart player. At the last level (the leaves) it propagates on the stars

value of the child with a minimum value. On the penultimate level, the stars

pass the maximum value.

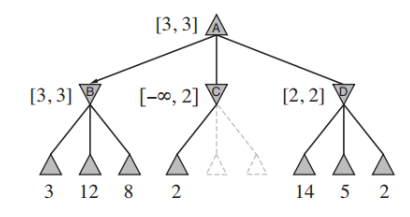


Figure 3.2: Simulation of the Minimax algorithm by cutting alpha-beta.

Undiscounted nodes in the image are called x and y. The root value

Undiscounted nodes in the image are called x and y. The root value

node is known:

Minimax (root) = max (min (3, 12, 8), min (2, x, y), min (14, 5, 2))

= max (3, min (2, x, y), 2)

= max (3, z, 2)

= 3

(3.1)

We see that the value of z in equality (3.1) is less than or equal to 2, which means,

that x and y are not actually interested in us and they are the final value

Minimax algorithm is independent of their value.

Let's try to argue in a more natural way: we have a knot n at any one

the depth of the investigative tree where the player can move. If

the player has the better option m on the star node n or else anywhere in

tree; then it will never be visited in the real game (Figure 3.3).

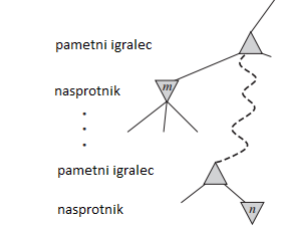


Figure 3.3: Simulation of alpha-beta algorithm. If m is better than n for the current one

player, then in the game we will never visit n.

Alpha-beta investigation during the investigation updates the values ​​of α and β and

so it ensures cutting the remaining branches on the current node if it is a value

of the current node is weaker than the current α or β for MAX or MIN.

α thus applies to the best (highest) value found anywhere in the

investigating tree on the way to MAX.

β refers to the best (least) value found anywhere in the investigation

tree on the way to MIN.

Pseudocode 2 shows the Minimax algorithm, upgraded by cutting alphabet,

summarized by [24].

**3.3 Using the Minimax algorithm in the Yinsh game**

When applying the Minimax method to the Yinsh game, we had to answer

The following questions:

• What will be the evaluation function for the game state? What are the parameters?

did you take into account when evaluating the gaming board?

• What will be the depth of the investigation?

• What will be the output condition from the recursion?

• What information will we store in the tree node?

Evaluation function was developed gradually. We started with completely

simple features, which we gradually upgraded to power

and a more representative function, which means that the parameters will be rendered

real game. With the complexity of the function we also adjusted the depth of the investigation

and partly also the output condition from the recursion.

**3.3.1. The types of evaluation functions**

**1. Number of zeton v**

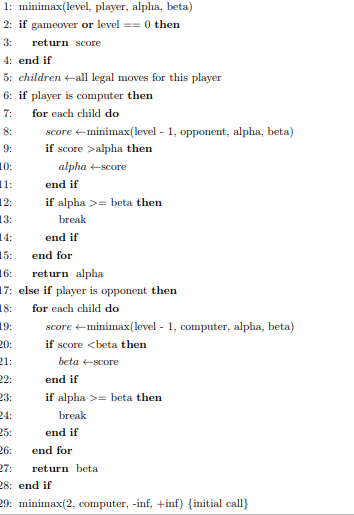
Although the number of zetons in our color does not necessarily mean dominance (because it increases

the risk that the opponent turns his tokens into his own color), however

this is an important factor in the game. That's why it was our first, the simplest

the solution is as follows:

Algorithm 2 Algorithm Algorithm Alfa-Beta algorithm



At the depth of the five investigative trees, we will close the investigation

the successors of the node (the exit condition) and evaluate our game

flat according to the following equation:

score (node) = numberOfM arkers (AI) -numberOfM arkers (opponent)

(3.2)

We also conclude the investigation with each collected type of five zetons (optional

colors). Then we estimate the node with 100 points, if there are five zetons

reached the smart player. If the five zetons reach the opponent, the nod

We allocate -100 points.

Despite the simplicity of the solution, we found that this is not bad. Dana hevristika

has enabled smart player to play. We're testing

they found that the solution is already capable of overcoming an inexperienced human being

a player who is able to create and defend his species and turn the species

opponent.

However, we are aware of the shortcomings of the solution: only the number is taken into account

of its own ketones, but does not examine whether these tokens are meaningfully formed

in species. It does not check whether these species are adequately protected. It's also about

hesitant heuristics: the player can acquire his own type, but this allows for

win the opponent in the next move. The solution therefore does not take into account the thesis

victory or defeat; it checks only the intermediate collected types of five zetons.

**2. Number of ketones in a row**

The other way is somewhat more complex. We develop a simple algorithm,

which checks all the lines of the gaming panel and evaluates each type of ketone.

The algorithm in the individual line is the individual zetone, and also the type

two, three, four or five ketones. Of course, the length of the series is necessary

weighted accordingly.

Let's take a look at Figure 3.4. Each subsequent token brings in a higher value in turn

the current line. The number of zetons is estimated at 1, 3, 9,

27 and 81 (from 1 to 5 zettons in a row), always starting to get in

the same zetone. If the number of zetons is in the order of n, then the species is evaluated by

equation (3.3).

score = ½\*(3^n - 1)

One zeton brings one point, two zettons bring four points, three

The zetons bring 13 points, stir up 40 points and five zetons 121 points. Smart

player plays in white. First, let's look at the white zetone in

vertical lines. We find five times a zeton (five points), and once

forty zettons in a row, which is 40 points, altogether 45 points.

Similarly, we make two diagonals. We get 13 and 11 points. Common

The total point for a white player on the board is 69 points.

Similarly, we calculate the points for the opponent and leave them out

tock. So we get to the final value of the panel, 57 points. On such

The way we evaluate the game board at the following events: when we arrive

to the depth of the four investigative trees, or at the discovery of five zetons

(we estimate before we remove zetones).

The latter way, he says, is a much improved version of the smart player.

He is able to beat even the most experienced players, but we still notice

There are some shortcomings. Here too, we are not checking the protection of ours

Zetons, i.e. whether the opponent can turn them in the next turn. We're noticing

also the occurrence of the withdrawal of our bots. Because we assess only the state of the whole

we do not reward the record and the actor, especially when the five ketons are collected,

A smart player skillfully gains a very strong state of the gaming board in his own

favor. It has many combinations of four zettons, but it sometimes happens that it has

the opportunity to get to the five zetons, and his opponent for the withdrawal

this option breaks down.

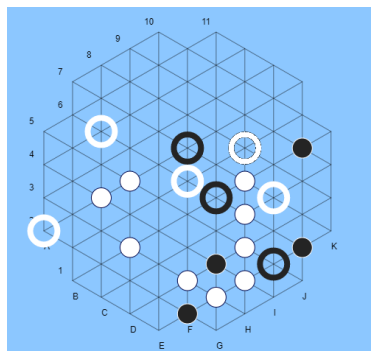


Figure 3.4: An example of a gaming board.

**A combination of points for assessing the types of tokens**

Within the described function, we made a minor analysis for the choice of the combination

points that we give to the types of zetons. We chose honey

by the following combinations for 1, 2, 3, 4, 5 of the zetones 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

During each pair of combinations we performed 20 automated games, at

Every smart player is playing with his rating scale. Izmed

these twenty games were started by A player 10, the other 10 games started by the player

B (changing colors of players). Games were randomly set up initially

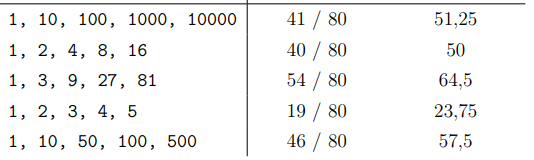
Setting up the circles. In total, we played 200 games, where it is

each type of combination was played in 80 games. The results are collected in

Table 3.1.

Table 3.1: Comparison of the combinations of constants to evaluate the types of cetons

Combination Number of wins Percentage (%)



On the basis of the above results, we decided to be a smart player

equipped with a combination of points 1, 3, 9, 27, 81 for 1, 2, 3, 4, 5 cetons in

type. If the species is longer than five zetons, each next zetone is estimated

with 81 points. For some species, we should not consider ourselves to be superior,

because they do not illustrate the goal of the game, which is the achievement of five tokens in a row.

**3. Poor Evaluation Function**

The third version is a combination of the previous two. It works on the next one

way:

• At the reached depth, the four investigative trees are assigned to the node

points according to the evaluation function described in the previous section.

• When the five ketons are collected, the investigation ends and the node

with 10,000 points (five intelligent player zetons) or

-10000 points (five zetons of the opponent). Number 10000 here

is the largest possible number that can be evaluated

the playing board (-10000 is the smallest). By doing so, it is always,

When we have a chance, we pick up five zetons. Also, always when

We have an opportunity, we can prevent this from our opponent. In case it is

Possibly both, we decide for the possibility that brings us together

tock.

The game of a smart player with a positive evaluation function works better

- Evaluation function ensures an appropriate assessment of the plate, 10000 points

On five ketones, he ensures that the player always takes his five

Zetones and does not exclude this. Of course, it can happen to pick up his own

for example, it would be more meaningful to prevent the opponent from winning.

Another disadvantage is that a smart player here can not be evaluated

protecting his zetons.

**4. A more advanced evaluation function**

Let's take a look at Figure 3.5. If we calculate using the previous evaluation function

The number of points, we find that in row J we find three consecutive zetons and

Are two consecutive zettons in black. But each player can clearly see,

that the ring that locates these two types can reach the next line in a row

five (even six) black zetons. We update this from our findings

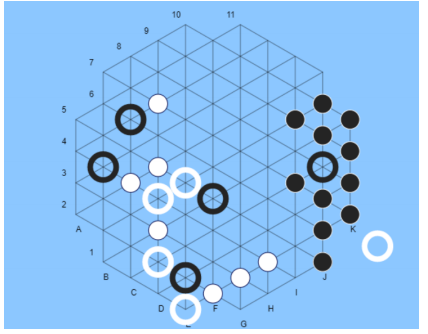


Figure 3.5: An example of a gaming board.

Function: If a ring of the same color appears in the line of successive zetons,

We do not interrupt the search lines, but we evaluate the ring in a row by halved

value according to the place of the order in a sequence. In our house

In case the line is long, there are six places. As we mentioned before, sit down

The zetone is evaluated in the same way as the heel. Thus, the line of circles between J5 and J10 would be

is estimated using the equation value (3.4), which is more indicative of

the actual power of the J5-J10 type.

score = 1 + 3 + 9 + 0.5 · 27 + 81 + 81 (3.4)

Here, too, we evaluate the nodes at the depth of four. In addition to the update in

Evaluation function itself, in the current version we also update the evaluation

types of five zetons:

Once the line of five zetons is reached, we arrive at the result of the evaluation function

Points according to the following criterion:

• 100 points at the reach of five smart player cents or 10000 points

at the victory of a smart player. Here we want to weigh the victory appropriately

in comparison with the collected series of five zetons.

• -50 points at the reach of five opponent's zetons or -10000 points at

win the opponent. Here we also weight the opponent's victory and only

a single line of five zetons. For the line of five zetons here

We assign -50 and not -100 points, as we could expect depending on

the previous indent.

With this parameter we can control aggressiveness or defensiveness

our player. With the current dot assignment is our kind

five zetons more weighted than the type of five zetons of the opponent,

which means that the player plays us a bit more aggressive. In reverse

the smart player would play a more defensive role. By the end

empirical analysis, such allocation of points proved to be somewhat

better than the allocation of 100 and -100 points and the allocation of 50 and -100 points

(defensive play).

At this stage, we focus on the stability of the type of zeton. It is clear that

the type of four zetons on line K in Figure 3.5 is protected. This means,

that the white opponent can not endanger the next line of the black type (i.e.

turn zetones). Sometimes it happens that the species is not protected and enables it

the opponent to reverse the part of the species. We therefore decide to reward stability

types:

• 100 points approximate to knots belonging to the branch at which points

successive knots grow with depth. In such a way we reward

sequence of moves that are in favor of our player.

• 100 points are lost to knots belonging to the branch on which they are

reduced by more than 100 points during the course of two successive nodes

(which means that the opponent turned 4 of his zetones).

With an advanced method of evaluation we came to a very powerful version

our smart player, who is solving the problems we encountered in

the first three versions. Details of how good our evaluation is,

are described in the Results section.

**3.3.2. Depth of the investigative tree**

The depth that we achieved in the types of smart player we played was five

(1st player type) or depth of four (the other three types). These are the depths that

we achieved them prior to the optimization of the algorithm and are time dependent

investigation. Considering that we also tested the games against a human player,

the time of thinking a smart player to move should be moderate: in our own

In the case, this was somewhere up to 30 seconds, sometimes much less, sometimes even longer.

After optimization, this time decreased considerably (about 5 seconds). More about this,

How we optimized the time of the investigation, and thus increased the depth, is described

in the Optimization section.

**3.3.3 Tree node structure**

The information that our knots carry is as follows:

• Player Index (1: White, 2: Black)

• The status of the playing board (white / black zones / rings)

• The position from which the current player played a move and thus came to

state of the board

• The position to which the current player moved and thus came to

state of the board

• The next currently optimal node (child)

• Stability of the nozzle (yes / no)

• Points of the node (calculated according to the evaluation function for the gaming state

panel in the current state)

**3.4 Treasure hunting Monte Carlo**

Monte Carlo Tree Search is a method

to find the optimal decision in a given domain. On a random basis

sampling collects results and gradually builds an investigative tree. Scasoma

the algorithm becomes better in evaluating the node and consequently finding the best

knot. The method proved to be very effective in the field of artificial

intelligence, especially in the domain of games and planning [3, 5].

The MCTS consists of four phases (Figure 3.6), which are repeatedly determined

Number of iterations (usually limited to time) [14]:

1. Choice

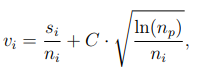
At this stage, from the root node, we select one of the nodes of the offspring,

using the UCB equation (3.5). From the selected node again

we select one of the offspring and repeat the procedure until it is done

we come to a nod, which is not fully expanded (ie all his children

Have not been added to the tree) [14].



We explain the UCB Upper Confidence Bounds (3.5) (source:

[13]). Equation is used in the framework of the confidence-building strategy

Upper Confidence bounds for Trees (UCT) tree [3]. In equation

it determines the number of points of the child, and where the win was awarded

with 1 point, defeat with 0 points. No count

and np are applied to the number

visits of the child i or stars p. The number C is a constant that regulates

the relationship between exploration and exploitation

exploitation, and is empirically determined in practice [13].

2. Expansion

From the last selected node of the previous phase, the nodes are randomly selected

(child) L which is not added to the tree [14].

3. Simulation (playout)

From the node L, the algorithm simulates a game that begins in the state of the node L,

ending with a win or a defeat (end of the game). The moves can be random,

but they can more effectively reflect actual gaming and thus reach the algorithm more

reliable results, but it is necessary to know that a more complex simulation can be

it reduces the number of simulations per second and so at the same time

we achieve less MCTS iteration [14]. A compromise is needed

between these two parameters [2].

4. Backpropagation

At this stage, the result R is propagated from the starting sheet L to the root

nodes to update all nodes between them [14]. Update

Number of knot visits and points of the knot (but only when the simulated game is won).

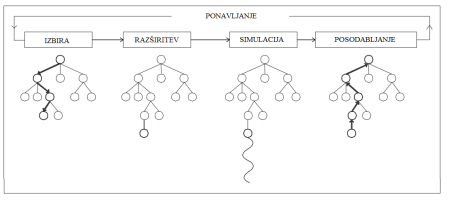


Figure 3.6: Phases of the algorithm Monte-Carlo tree search.

**3.5 Using the MCTS algorithm on the Yinsh game**

To apply the Monte Carlo tree search method to Yinsh's game

it was necessary to slightly transform the structure of the node that we used

with the Minimax method. The structure is as follows:

• Player Index (1: White, 2: Black)

• The status of the playing board (white / black zones / rings)

• The position from which the player is currently playing has taken a move and thus came to

state of the board

• The position to which the current player moved and thus came to

state of the board

• Connection to the star node

• List of nursing children

• Number of node visits

• The number of nests won

The time to run was limited to 10 seconds, which means a different one

The number of iterations relative to the initial state from which the move moves. Initially

Simulations last longer last longer than the end, when the result is already

for example 2: 2. Let's mention the specific numbers: the number of iterations in 10 seconds

in the first stroke after setting up a ring, it is on average between 3000 and 4000

the Monte-Carlo tree investigation. In the 10th move, there are a number of iterations

Already over 10,000, and in the last strokes it is up to 500,000.

**3.5.1 Empirical analysis of the constant C**

Let's look at equation (3.5). First part si/ni focuses on exploiting better

estimated moves, while the second part

C:\Users\Kartik Kumar\Pictures\16.PNG

dedicated to exploring new,

No knots have been explored [3].

The choice of the C constants was determined empirically in our program, whereby

we tried the following values:

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

We performed automatic gaming, and we tried 10 games for

any combination of constants. The highest victory has brought a value of 2.0, so we are

in the continuation always used this constant

**3.5.2. Select the type of simulation**

As already mentioned, a more advanced type of simulation can be more reliable

results. Winands and Nijssen [14] in their study mention -great

-greedy playouts, where the probability of simulation moves

makes it random. Otherwise, a heuristic that reflects is used

actual game.

We chose simple heuristics, which is actually taken from the first

the type of evaluation function mentioned in section 3.3. A player who is in the simulation

on the line, so with probability 1 - chooses a move that in the next step

brings the largest number of zetons.

This is heuristics, which has proved to be relatively well in the Minimax method

and at the same time it is completely unpretentious.

In automatic testing, the version of the MonteCarlo tree search

with advanced, full simulations, it turned out to be somewhat better than the whole version

random simulations (19 wins of a total of 30 games). Therefore, for further

We use the simulation version with the simulations.

**3.6. Setting up the initials**

Unlike classic abstract games such as chess and lady, Yinsh plays

variable initial layout. In fact, at the beginning it is possible to millions

different layouts [7], which in our opinion means that it is at this stage

senseless moves are calculated using methods such as Minimax and MCTS. We believe,

that a strong start-up strategy is particularly helpful to the experienced strategy

players of the game Yinsh. Thomas Debray, in the description of his smart player,

that it is important, in particular, that the player remains movable and so is a ring

does not place it on the same line, but at the same time keeps them near the center of the playing board,

which according to professional players determines central dominance [26].

**3.6.1 Setting up the rings against the opponent**

Before testing, we perform an analysis of the game of opponents. We come here

to the following conclusions:

• A cowardly opponent is important to Yinsh's clever player

Advantage: man often practices a commodity strategy, which means that

you can build an inflexible net that can not be turned on the edge (Figure 3.7).

While in a general man can not think about the depth of research

in the case of a commodity strategy, it can think deeper than

computers. Along with the layout of the bracelets you can provide a good foundation

for setting up such a network. If both players know this strategy, you are

it can prevent the player from making important things by setting up the bets

the edge on the edge of the board.

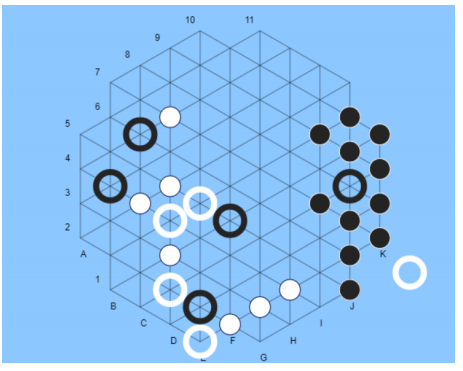


Figure 3.7: An example of a situation that is quite common in Yinshu: the player is on the verge

build a network that can not be turned.

• If the opening of the opponent's rings is based on our layout

always the same, the smart player must provide us with a slight chance of being random

with this opponent being prevented from developing the perfect combination with time

the starting positions of the circles according to our layout and thus acquire

The advantage is that the player does not have a strong strategy for the layout. That's true

especially for the human actor, because we can not make it

analyze it in advance, as we can do for the existing versions of Yinsha.

• The implementation of David Petr is already in the game against a human player

proved to be very dominant if one does not use commodity strategies.

Using a commodity strategy, a person can overcome without major difficulties

smart player David Petr. So let's try to use this fact

even in the implementation of our smart player.

Because of the described different behavior of opponents, which will be used as

test against our smart player, we decide to implement two strategies

for layouts:

1. In the case of a human opponent, we want to introduce a coincidence in the setting,

at the same time, we want to prevent the opponent from building a commodity network. Z

network construction often tries to play the bots on Boardspace,

so in these two cases we decide on the rings of the smart player

to be placed under the following rules:

• If possible, place the ring in the immediate vicinity of the last post-

the opponent's rally set. The position has up to six possible adjacent ones

позиций. Because of the cabbage, the position is chosen randomly on the basis of dynamics.

By placing in the vicinity of his opponent's circles (at least

partly) we do not want to play a strong commodity strategy. By chance

we prevent him from developing the perfect combination of the initial ones

position of the hoops and thereby acquires a significant initial advantage.

• If this is not possible, place a ring as much as possible in the middle of the game

plosce. By doing this, we maintain our motility and central domination

gaming plates.

2. In the case of the players of David Peter and Thomas Debray, we notice that

our opponent allows us to smoothly set the circles on the edge. That's why

here we try to integrate the commodity strategy into our player: if a player

place a ring on the edge, place it next to it. If a player does not

puts it to the brink (as happens in the implementation of David Petr),

place 2 rings on the same edge, the rest are as close as possible to the middle. A ration

on the edge with common powers, they can build a strong network on the edge of the game

and they are scribbled. However, it's good to have some bucks on

center, so keep the other rings in the middle of the playing board.

**3.7 Removing a ring**

After each accumulated type of ketone, it is necessary to remove the ring from the playing board

its color. When removing a ring, you need to be very careful, because one

The smaller the ring means less control over the board. They also have

rings are an important function of blocking opponents and squatting of our species.

For the removal of circles, we develop a simple algorithm that chooses the least

a useful ring depending on the current state of the game.

For each ring we do the following: remove the ring from the playing board

R. On the playing plate without the R, we look for the most likely move of the opponent.

This is a move that brings the maximum point to the opponent (MIN). So we have

for each removed ring, the number of points that this removal would bring us.

So, we choose a ring that brings us after the removal and move of the opponent

maximum point.

**Chapter 4**

**Optimization**

We have already mentioned that the Minimax method is timely very complex. In the chapter

Therefore, optimization is mainly focused on the methods we have used in

to improve the timing of our smart player. These methods

they can speed up the time of thinking, and at the same time they can help achieve it

bigger depths of the investigative tree.

In describing the optimization, we will focus primarily on the Solving article

games: Dependence of applicable solution prodecures [7], which describes Yinsh's game

as one of the most complex games in terms of resilience.

We evaluated the above article by testing these methods. At the same time

we have proposed some of our methods that we have integrated into the implementation

smart player. We will also mention methods that are not possible in Yinshu

to implement because of its different nature.

**4.1 The range of the game Yinsh**

The extent of the game-tree is

Usually, it's estimated by the average number of moves to move to the position and the average

Number of all game moves. At the Othello game, the expansion factor is so

it's 10, and the average number of moves is 58. The estimated size of the investigative

The Othello game tree is 1058 [7]. The spread factor at chess is 35

the average number of moves is 70 [11], which means that the size of the investigative one

trees 35^70

.

For the Yinsh game, we reached the approximate value in the following way: for

For example, we took 10 automated games between the advanced and advanced smart

players. We followed the number of all moves and the number of all the following

possible positions from the current state. Expansion factor based on

This empirical analysis is 47; The number of total moves is 51. Estimated

The size of the investigative tree of the game Yinsh is thus 47^51, which is quite a lot

greater than the size of the game Othello and slightly less than the game sah. Because

of such a degree, of course, investigation to the final state of the game is not possible,

therefore, testing is usually limited by the depth of search (horizon).

**4.2 Collection opening**

Despite many evaluation and investigation improvements, many algorithms

programs are always showing deficiencies, especially in the opening stages

the very games, since programs often do not have a satisfactory strategic planning.

In order to solve this problem, there are often used collections of openings where they are

stored sequence of moves that establish a good initial strategy [4].

Unlike games such as chess and lady, Yinsh plays a dynamic game

opening. This is due to the installation of a ring that is not known in advance

(as is known, for example, the initial layout of the saha).

If you would like to apply the method of collecting openings to the Yinsh game,

They should have stored initial moves for each initial combination of positions

obrockov. However, since these combinations are millions, the collection of openings is optimistic

We abandon the method

**4.3 Transposition table**

A large number of nodes that show can appear in the investigative game tree

identical state of the game [7]. Repeatable states can be caused by

transpositions - various permutations of successive moves that come to the same

state of the board.

Assume that White has the action a1 on which Black can answer

with b1, an independent action on the other side of the gaming board a2 may be followed by the action

b2. Then the sequences of moves [a1, b1, a2, b2], and [a2, b2, a1, b1] concatenate in the same state

gaming plates. That's why, if it's possible, it's handy to remember what it is

was determined on this position when it was last investigated, as this is how we save

re-investigate the same sub-tree [15] and thus reduce

the size of the investigative area. For this reason, they have sahovski smart

players are usually stored a transposition table that is implemented in

the form of a larger hash table and stores position information,

which have already been researched [25].

Use of transposition tables can have a dramatic effect (doubling

investigative depth at chess), but it is necessary to know that it can be stored

Transposition tables are impractical, if we evaluate millions of knots at

second. At this site, it is necessary to decide which nodes to store and

which does not [15]. Information stored in a transposition table for each

nodes should be as small as possible. This ensures more stored nodes

to the same unit of memory [7].

**4.3.1 Application of transposition tables to the game Yinsh**

In article [7], the claim that the turning of ketones makes transpositional ones

tables are useless. This is partly because of its dynamic nature

the initial layout and the fact that every move is added to the gaming board a new one

Zeton, we rarely come to exactly the same state of the game board.

But it is necessary to mention the influence of transposition tables on the investigative tree

the current moves, where it often happens that at the same level we get to

the same condition. Let's take a look at Figure 4.1. Suppose the white player moves

from A4 to A5. This is followed by a shift of black that moves from C6 to C7. The following is a move

white, which moves from C4 to C5.

If a white player turns his moves and first moves from C4 to C5, after the pre-

Miku black (C6 to C7) moves from A4 to A5, the result is identical.

The diagram of the investigator tree knot is shown in Figure 4.2

Following this analysis, it is also clear that such moves are at least in such a state

gaming plates, a lot. It is only necessary to mention that the same situations can be

it only comes at the same level of the investigative tree.

Therefore, in the implementation of the smart player of the game Yinsh we introduce transposition

tables that appear in the form of spreading tables, with one sprinkler

The table used for moves at the same level of the current investigative tree.

The repeatable vehicles are simply not added to the investigative one

tree.

The state of the board is stored in the form of a set of characters that represent

arranged positions of all circles and zetons according to the following rule:

rings of white; rings of black, zetones of white, zetones of black.

For the current situation this series would be A4C4F6F7G5; C3C6E7G4I8; B5F8F9 ;.

**4.4 Sorting moves**

The effectiveness of cutting alpha-beta depends primarily on the order of investigation

knot. Let's take a look at Figure 4.3. We can not cut off trees (a) and (b)

none of the successor to the Node D, because he was the worst successor (from the point of view

MIN) generated first. If the third successor of the node D was generated first,

we could omit the investigation of the other two knots (cutting). Sorting the knot

of course it can not be done perfectly - if that were possible, it would be sorting

enabled the smart player to play without error [15].

If this were possible, the alpha-beta algorithm could only be explored by O (b^(m/2)) the knot

(instead of O (b

m), which is the complexity of the basic Minimax).

This means that the divergence factor b would decrease to √b – v at the same time,

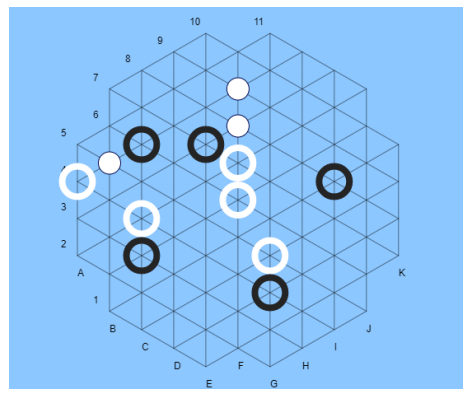


Figure 4.1: The example of the status of the playing board

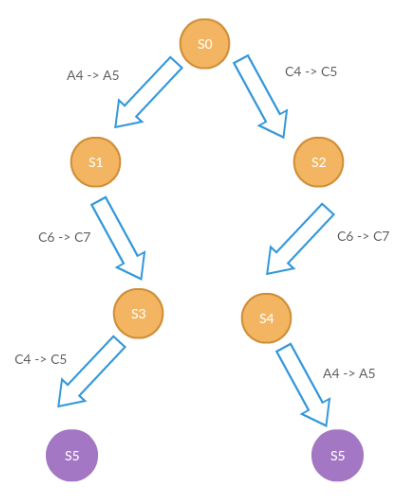


Figure 4.2: Schematic of a simple investigative tree representing a transposition.

The initial node represents the state of the game shown in Figure 4.1. Po

both of the displayed paths reach the same state of the game board. On the picture for

simplicity does not display all the possible knots that are derived from the current one

state of the board.

considers that the divorce factor would be reduced from 35 to 6. If successors

given in random order, the number of searched nodes of order O (b^(3m / 4)).

At chess, a simple sorting function is already sufficient to reach the investigated ones

only around O (b^(m/2)) the knot.

In sorting, we can also help with the search for places that already exist

proved to be very good in the past. This can mean the previous move, you can

then investigating the current knots in the past. A common way to achieve it

the latter is with iterative deepening, which means that we move in depth

gradually - we increase the depth only when we have studied all nodes

of the current tree level [15].

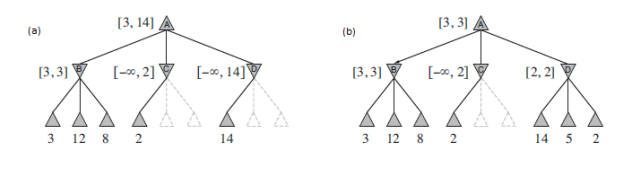


Figure 4.3: Investigative tree with cutting alpha-beta

We've also integrated our moves into our smart player:

When searching the node, we first create all the successors, and then we assign them

points after the same evaluation function with which we evaluate the final nodes.

The sidekick is then sorted - on the MAX player's move, the player moves down

MIN growling. We assume that the state at the lower level will be an indicator

conditions at a slightly higher (deeper) level. Because we first understand

more promising nodes (from the point of view of the current player), we can cut off more

such branches that will surely bring a smaller number of points to the current player

and therefore they do not even care about us. This is made possible by the nature of the alpha-beta algorithm,

described by equation (3.1). Despite that sorting and evaluating all the knots

additional complexity, we find that the technique significantly reduces the space

and the time of the investigation. We will describe the details of the optimization in the section

4.7

**4.5 Select the level according to game development**

Yinsh is a type of game where the size of the investigative tree is reduced by the course

games (end-games). In the game, the tree's branch factor is reduced,

because the number of the next possible moves during the game falls. That's it, for example

the average number of the next moves for the first move of the game is 80. At the end

moves this number is reduced to 25. This happens because of the reduced number

circles and due to the fullness of the playing board, which limits the player to the movement

obrockov.

We decided to use this information to build our own

of the investigative tree. Without the methods described in the Optimization section, we have

in the exemplary time (below 10 s), they reached four levels of investigation. With all the described

optimization, and using information on the reduction of the investigated

During the game we make an empirical analysis for the depth of the investigation. Smart

We determine the player the following depths:

• Depth 4, if the number of game moves is less than 15 and the number of successors

given condition is greater than 60.

• Depth 5, if the number of heirs of a given state is less than 60.

• Depth 6, if the number of heirs of the given state is less than 15.

**4.6 Horrors and opportunities**

In games often there is a situation where we are endangered by a sudden defeat or

Sudden victory (in chess this position is a sah, which means attack on the king). Taksne

situations can significantly reduce the investigative space, as the opponent must

update your priorities and, first and foremost, prevent your opponent's victory.

Even in the Yinsh game, this situation can occur. If in Figure 4.4 a black player

White does not prevent the installation of the fifth token in a row, it will be in the next move

won the third term and won.

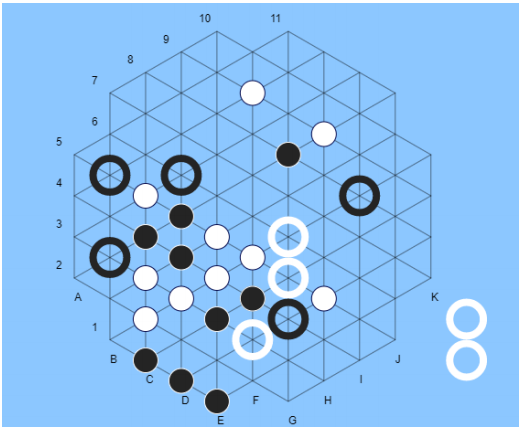


Figure 4.4: An example of the white player's ability to win in the next move.

At the same time, it threatens the black player to be defeated.

In article [7], the use of threat and winning strategies is estimated as almost

unusable, because they happen very rarely. Yinsh wants the three picked up

rings, and the player with a new type weakens his position as he is with

it reduces mobility less by one, and also loses five zetons

its color.

Despite this, we do not agree with the assertion that the strategy is sudden

winning or defeating ineffective. In our advanced evaluation function, mentioned

in section 3.3.1, this is taken into account, giving 10,000 points for victory

and -10000 points for defeat; for the collected five zetons we evaluate the node by 100

points, and the opponent's five -50 points. This makes us better

balance between the accumulated type of betting and winning or defeating. For example

we achieve that with the result 2: 1 we are more focused on the opponent

to block the opponent's victory, and reaching our kind is here

secondary importance. If the opponent can win the next move, he will

we try to prevent it. If we are not at risk, we focus more on it again

to acquire the types of our player.

**4.7 The impact of optimization methods on investigation**

In this section, we will be interested mainly in how the optimizations above are

have influenced the speeding up of our smart player's performance. We were interested

how the integration of sorting moves, transpositional, will be influenced by a certain state of the game

tables and a combination of both optimization methods. Also,

what is the state without any optimization methods. We will be interested in the time of implementation,

the level of investigation, the number of searched nodes and the number of calls for evaluation

functions.

The last parameter was added for the following reason: if nodes are searched

tree leaf, we calculate only the estimate of the current state of the gaming board and that

we return. If the nested tree is not a tree leaf, the sorting and

Stability estimates calculate the estimates of all children of the node according to the same formula,

as the situation in the sheets is assessed. This is the time difficulty of examining the leaves

much less than the time required to investigate stars who need to be evaluated

all your children. The number of searched knots is insufficient

to explain the duration of the investigation, so the numbers of calls were given in the tables

Evaluation functions.

Let's take a look at Figures 4.5, 4.6 and 4.7. The black player has the potential in these three situations

88 (Figure 4.5), 68 (Figure 4.6) and 20 (Figure 4.7) moves. Testing level in testing

these three states are four, and in the latter case we are for transparency

Measurements with a depth of five were added. Tables 4.1, 4.2, 4.3 and 4.4 summarize,

how much time he needed an individual approach to determine the next best

moves.

With optimizations, in any case, a significant reduction in the number is achieved

explored nodes, which also means a significant increase in implementation time. V

In the case of the situation in Figure 4.7, in the actual game, you can also explore the depth

Six, because the implementation time would be about 4 seconds.

From the tables it is also clear that the investigation tree is running out of time

games: we see that in the initial stages of the game there were a number of investigated nodes of the tree

depths without optimizations above 300000. Number of studied tree nodes

The depth of four in the intermediate stages is about 40,000. The number of studied knots

the tree of depth in the final stages is about 2000.

As the best optimization tool, we evaluate the sorting of moves with help

Advanced Evaluation Functions. Despite the greater complexity of sorting, it is from

the tables clearly show that the sorting of moves allows a significant reduction in the investigative one

trees. First-level sailors can do this to a great extent

indication of the nodes at higher levels.

Transposition tables also reduced the size of the investigative tree,

but to a lesser degree. Thus, in figure 4.7, the number of studied nodes is at the level

five with the help of transposition tables decreased from 35070 to 23646. It's time

it decreased from 2876ms to only 2630ms. Transposition tables have therefore shown

Reduction of the investigative tree, but the time of research is for the sake of it

the complexity of tables has not significantly decreased.

We performed the measurements on an Intel (R) Core (TM)

i5-4200U CPU @ 1.60GHz 2.30 GHz.

**4.7.1 The influence of optimization methods on the node structure**

It was because of the application of optimization methods to our smart player

it is necessary to partially change the structure of our node. We are so on the rest

properties are added to the list of all the successors of the current node. To the taxi

We were able to ensure the sorting of moves.

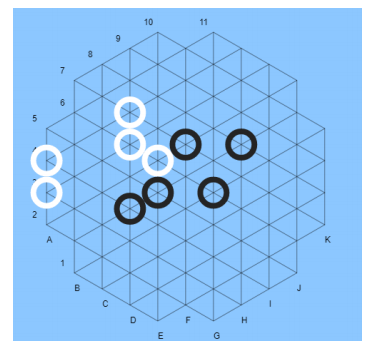


Figure 4.5: An example of the state of the game where the black player has the next move possible

88 moves.

Table 4.1: Analysis of Optimization Methods for Figure 4.5. The depth of the investigative

trees are 4.

Approach Implementation time[ms] Number researched knot Number calls Eval. Func

No optimizations

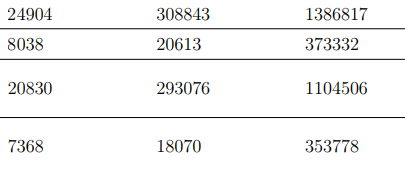
Sorting

Transpozicijske

tables

Combination

approaches



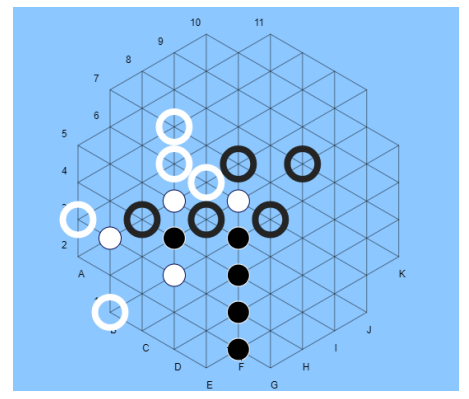


Figure 4.6: An example of a game state where the black player has the next move possible

68 moves.

Table 4.2: Analysis of optimization methods for the image 4.6. The depth of the investigative

trees are 4.

Approach Implementation time[ms] Number researched knot Number calls Eval. Func

No optimizations

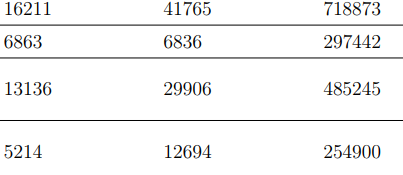
Sorting

Transpozicijske

tables

Combination

approaches



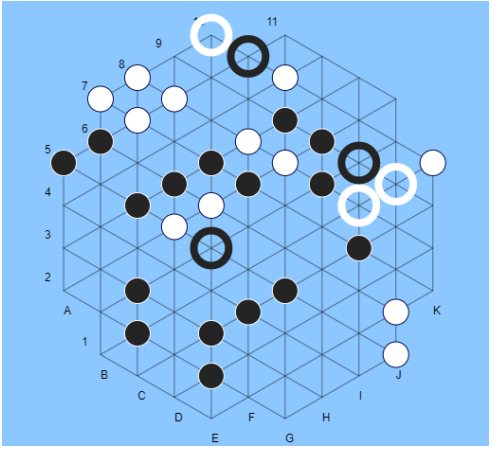


Figure 4.7: An example of a state of play where the black player has the next move possible

20 moves.

Approach Implementation time[ms] Number researched knot Number calls Eval. Func

No optimizations

Sorting

Transpozicijske

tables

Combination

approaches

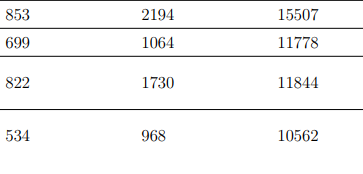


Table 4.3: Analysis of optimization methods for the image 4.7. The depth of the investigative

trees are 4.

Table 4.4: Analysis of optimization methods for the image 4.7. The depth of the investigative trees are 5.

Approach Implementation time[ms] Number researched knot Number calls Eval. Func

No optimizations

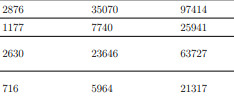
Sorting

Transpozicijske

tables

Combination

approaches



**Chapter 5**

**Implementation**

The rules of the game and the smart player were implemented in the programming language

Java (version 8). For testing against a human opponent

we also implemented a simple user interface (Figure 5.3), made in

the Javascript and HTML programming language. Our back and forth part communicate

using HTTP requests sent in the form of data in JSON

format (figure 5.1). Depending on the move on the user interface, the backend

part obtains one of the three types of commands in JSON format and answers with

a package of similar shape (Figure 5.2). When set up, it will be dispatched

information about the player and the desired position of the ring. When moving, information is transmitted

about the player, the position of the rally that has moved and the position on which

the ring has moved. When collecting five zetons, information about the

the player, the position of the rally that has moved, the position on which the ring is

moved, and the list of five zetons and eyebrows removed. Here it is

once emphasize that the player can create two independent lines with one move

five zetons.



Figure 5.1: Picture of communication between the main and the back of the application.



Figure 5.2: Examples of commands in JSON format that served as communication

between the back and the end part.

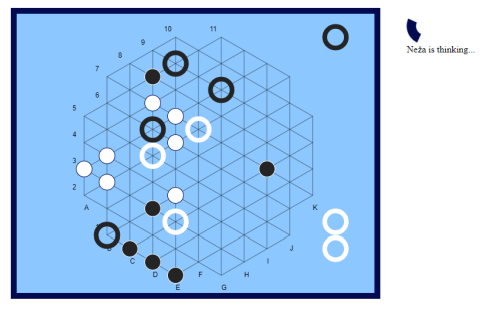


Figure 5.3: Image of the user interface. In addition to the gaming board, they are on the side

shown rings collected. On the right side there is also a loader that rotates in

The time of thinking a smart player.

**Chapter 6**

**Results**

**6.1 Comparison of advanced, generous and MCTS**

**variants of a smart player**

Before testing a smart player against existing opponents and humans

Find out which one of us is the strongest version of a smart player. With each other

we compare three versions:

• Minimax with a positive evaluation function

• Minimax with advanced evaluation function

• Monte Carlo Treasure Investigation with Simulated Simulations

During each combination of methods we play 30 automatic games, alternately

we change the colors of the players (white and black). The results are shown in

Table 6.1. It is clear that Minimax with advanced evaluation function is ours

the most powerful version, so we decide that this version will appear in the testing

against computer and human opponents

Table 6.1: The results of a comparison of three smart player variants

Players Result

Advanced method: Warm method 18: 12

Advanced method: MCTS 28: 2

Descriptive method: MCTS 28: 2

The Monte-Carlo tree search method has been successfully applied

Already on a series of games, such as Go, Amazons and Lines of Action [1]. Good

has also proved for games with incomplete information, such as Scrabble,

Bridge or the Scotland Yard [3]. But there is always a large number of traditional ones

games where the Minimax method is more appropriate. Such games are, for example, sah and

lady. MCTS builds a tree that focuses primarily on the most promising

paths, but it is not suitable for large-scale investigation areas

Number of traps on lower levels of the tree. Such situations are common in chess and

They need a precise and tactical game to avoid defeat. MCTS due

can easily be overlooked or underestimated by their randomness and way of assessment

important move. On the other hand, MCTS can be shown in domains like

for example Go, where the traps are rare or do not occur until the last stages

games [1].

Yinsh is a symmetrical game with complete information for two players and is of type

zero-sum game. Yinsh, too, for turning the ketone

It contains a lot of traps. The game is similar to that of chess, so you can

the above explanation satisfactorily explains the greater success of the Minimax algorithm

against the MCTS algorithm.

**6.2 Portal Boardspace**

The Boardspace portal is a popular online gaming portal. Maybe

is to play against other human opponents, but there are also some pa- game players. In addition to famous games such as chess, lady and Go are on the portal

also all the games of the Gipf project.

There are three smart players of the Yinsh game added to the portal: dumbot, weakbot

and smartbot. Their author is David Dyer, and the actors are implemented with

by searching alpha-beta with real-time increasing depth of search

games. They are described as very good against inexperienced players, but overwhelming

against experienced players [20].

With the username missRobot we registered on the portal and with

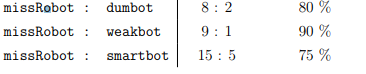
with the help of an implemented smart player using the Minimax method

with advanced evaluation function, played 10 games against dumbot and weakbot

and 20 games against smartbot. The results are as follows:

Table 6.2: The results of a smart player against the bot on Boardspace

Players Result Percentage of wins



After playing games, we climb to the current top of the scale of success

in the game Yinsh (Figure 6.1).

6.3 Implementation of David Petr

David Peter is a German physicist who is implementing his smart player role

Yinsh posted online [23]. He implemented his smart player with

upgrading the alpha-beta algorithm, Negascout. Negascout requires good sorting

move and speed up the operation of general alpha-beta. With poor sorting

The function of the algorithm is poorer than the investigation of alpha-beta. Algorithm was brought

good results for computer players saha. The advantage of the algorithm is above all

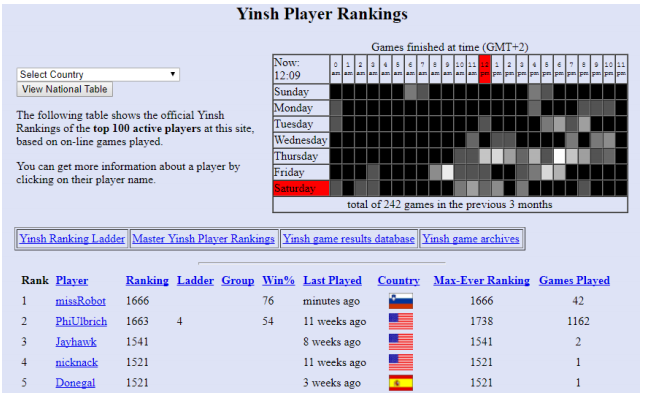


Figure 6.1: The success of our smart player on Boardspace.

that, despite the rush, it does not jeopardize the accuracy of the returned result and thus returns

the same result that would be returned by the Minimax algorithm [18].

Using the installation of the bits mentioned in section 3.6.1, and rewarding

stability of the species, we encourage our player to start building the type of ob

goods. Since the implementation of David Petre on the creation of the species responds to the line

too late, we can convincely overcome this strategy of the player.

Implementation of set-ups and gameplay for both players

they do not contain randomness, so the game is unique and always repeat the same

moves. And depending on playing this game, we can say that the player is convincing

better, because it defeats with a score of 3: 0 (Figure 6.2)

**6.4 Implementation by Thomas Debrai**

Smart player Thomasa Debraya participated in the Yinsh tournament in 2008

and won the third place. In the first place was Pim Nijssen, researcher at

Maastricht University with a number of important research in the field of artificial and

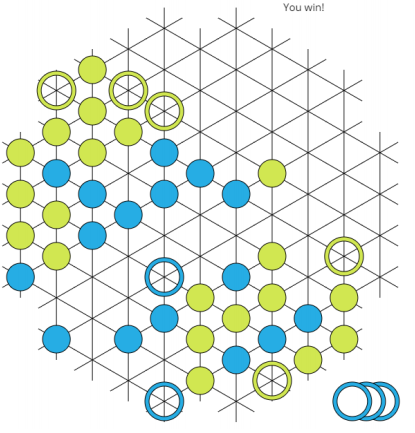


Figure 6.2: The victory against the implementation of David Petr.

telegence. T. Debray also obtained a master's degree during the tournament

at Maastricht University.

The third-place program is available on its website

[26], which also describes heuristics that were taken into account in the evaluation function:

• The number of collections collected. -

• Number of successive ketones in a row related to a ring. -

• The distance between the rings and the center of the playing board. Setting up the bonds

in the center of the playing board, according to Debra, it means more dominance

and mobility on the playing board.

• Distance between the tokens and the center of the playing board.

• The difference between the number of smart player and opponent's cents.

The weights of the heuristics were optimized using an evolutionary algorithm

playing against the earlier versions of the same smart player.

Smart player T. Debraya presented us with the greatest challenge. Anyway

we see the disadvantage we mentioned earlier: the initial layout

the circles allow the seamless layout of the sleeves on the edge. Here's also a layout

rounds on both sides are unique, but in player T.Debraya it does not matter

some coincidence is involved. That's why we played 10 games against the player and from them

they won 5. All the games were fairly balanced and ended with the result

3: 2.

**6.5 A human opponent**

We also tested the game against human players of various ages and experiences.

Inexperienced players who know the rules can build their own types and turn

their opponent was defeated already in the initial stages of program development (first type

Evaluation functions in Chapter 3). The other type of players are those with more experience

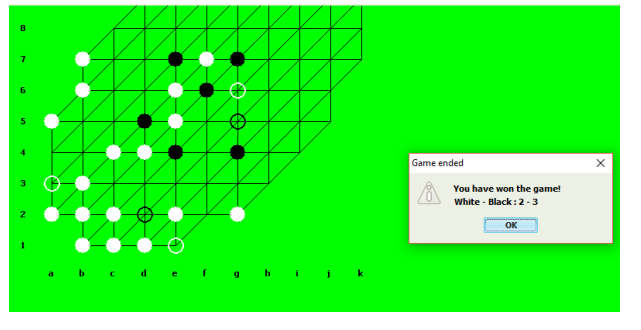


Figure 6.3: The victory against the implementation of Thomas Debrai.

and strategies, which are not always among the best players of the abstract

games. These people also beat the program without difficulty. The program is shown

as a competitive, but sometimes also a defensive opponent to the most experienced players

Yinsha.

**Chapter 7**

**Conclusion and further work**

In the present part we analyzed the game Yinsh and its characteristics. Implemented

we have a number of different methods for playing the smart Yinsh player.

Gradually we upgraded the evaluation function, which we took into consideration

in particular, the number of collections collected and the number of successive gaming cetins

plosci. We have come up with a powerful evaluation function that together with speed

Optimization Minimax brings more than satisfactory results against existing ones

smart player implementations. The player is also well presented

against a human opponent, where only the most experienced players of the game are cunning

Yinsh, this is only occasionally.

We also implemented a smart player with a tree search

Monte-Carlo, which does not appear to be a competitive opponent. We think it is

The reason is mainly in the nature of the game Yinsh, which is quite similar to chess. The game contains

more traps (turning zetons), but at the same time it is a game with complete information

for two players and simple rules. Minimax is usually for such games

always the most powerful method.

We are noticing our actor and competitive smart players

interesting features.

Players have problems with a balance between two events, which are between

connected to each other:

• Picking up a species or pulling it out. The selected species also means a minor

mobility, because we lose one of the twins. In some cases it is

dragging with the collection of zetons is good, as we maintain mobility and

dominance on the playing board. Removing our type can allow you to

immediately afterwards the opponent wins. In this case, it is essential to first

Let's prevent the opponent from winning. Again in other cases it is essential that

the species is removed immediately, even if it is not protected.

• Picked up or winning (collected 3 types). The game Yinsh has a very atypical

nature: collected can be three types of zetons. It often occurs

question how to evaluate each of these events, what is said

even in the behavior of smart players who sometimes can not judge whether or not

better pick up the line or prevent the opponent from winning.

**7.1 Further work**

We are aware of the weakness of the initial setting of the smart player's circles. We believe,

that there would be a lot of research and consultation with

professional players who could provide us with insights into strategies that would

the game made it more powerful.

Evaluation of the collected species and victory could also be analyzed in more detail.

As mentioned, smart players are hard to determine the balance between whether or not

it's better to pick up some sort of situation or wait for it. It would be for something like that

it is sensible to follow the implementation of Thomas Debra and use evolutionary ones

algorithms for optimizing the parameters of our smart player.

Given that we are limited to playing time when playing with a human opponent,

it would make sense to use the time of thinking of a human actor

for further exploration of the program [6]. Program on your move at a time

thinking of the opponent thus chooses the most likely move of the opponent. From

these moves continue to build the investigative tree. In the case of the correct one

Predicting the game of an opponent thus extends the time of thinking. If it moves

is not predicted correctly, the tree is devalued and the investigation begins again.

It would also be worth examining the Minimax pathology in the Yinsh game. Sometimes

too much depth of investigation results in less reliable results. Theoretical

Namely, the analyzes mention that by increasing the depth it also increases

A sum caused by a heuristic function with which we evaluate the leaves of a tree

[10].

Due to the variable nature of the Yinsh game, it would be good to use the method

Quiescence search. When looking for a hibernation

the emphasis is mainly on troubled knots. These are nodes for which

there is a high probability that the estimated value of the gaming board will be in the next one

step change significantly. Such a nodal algorithm is investigating more deeply

like a quiet knot. In this way, we can get rid of the effect of horizontality

(horizon effect). This can occur at fixed depth investigation.

The leaf can be evaluated very well, but in the next step it can be

the value changes significantly. The algorithm of such a more restless position is being investigated

deeper than quiet positions [17].