Journal of Intelligent Learning Systems and Applications, 2019, 11, 15-31
http://www.scirp.org/journal/jilsa
ISSN Online: 2150-8410
ISSN Print: 2150-8402

Research on Different Heuristics for Minimax Algorithm Insight from Connect-4 Game

Xiyu Kang[1*], Yiqi Wang[2*], Yanrui Hu[3*]

[1]Department of Computer Science, Beijing University of Technology, Beijing, China
[2]Department of Information Technology and Engineering, Dalian Polytechnic University, Dalian, China
[3]Department of Computer Science, Pennsylvania State University, University Park, PA, USA
Email: kangxiyu@163.com, 13728997679@163.com, tonyemail2017@gmail.com

How to cite this paper: Kang, X.Y., Wang, Y.Q. and Hu, Y.R. (2019) Research on Different Heuristics for Minimax Algorithm Insight from Connect-4 Game. *Journal of Intelligent Learning Systems and Applications*, **11**, 15-31.
https://doi.org/10.4236/jilsa.2019.112002

Received: January 20, 2019
Accepted: March 4, 2019
Published: March 7, 2019

Abstract

Minimax algorithm and machine learning technologies have been studied for decades to reach an ideal optimization in game areas such as chess and backgammon. In these fields, several generations try to optimize the code for pruning and effectiveness of evaluation function. Thus, there are well-armed algorithms to deal with various sophisticated situations in gaming occasion. However, as a traditional zero-sum game, Connect-4 receives less attention compared with the other members of its zero-sum family using traditional minimax algorithm. In recent years, new generation of heuristics is created to address this problem based on research conclusions, expertise and gaming experiences. However, this paper mainly introduced a self-developed heuristics supported by well-demonstrated result from researches and our own experiences which fighting against the available version of Connect-4 system online. While most previous works focused on winning algorithms and knowledge based approaches, we complement these works with analysis of heuristics. We have conducted three experiments on the relationship among functionality, depth of searching and number of features and doing contrastive test with sample online. Different from the sample based on summarized experience and generalized features, our heuristics have a basic concentration on detailed connection between pieces on board. By analysing the winning percentages when our version fights against the online sample with different searching depths, we find that our heuristics with minimax algorithm is perfect on the early stages of the zero-sum game playing. Because some nodes in the game tree have no influence on the final decision of minimax algorithm, we use alpha-beta pruning to decrease the number of meaningless node which greatly increases the minimax efficiency. During the contrastive experiment with the online sample, this paper also verifies basic characters of the minimax algorithm including depths and quantity of features. According to

*These authors contributed equally to this work and should be considered co-first authors.

the experiment, these two characters can both effect the decision for each step and none of them can be absolutely in charge. Besides, we also explore some potential future issues in Connect-4 game optimization such as precise adjustment on heuristic values and inefficiency pruning on the search tree.

Keywords

Heuristics, Minimax Algorithm, Zero-Sum Game, Connect-4 Game

1. Introduction

Minimax algorithm has already achieved significant success in area of game including chess, backgammon and Connect-4. What's more, as people keep reinforcing search algorithms and machine learning technologies in AI, it has developed superhuman intelligence. For example, Deep Blue, designed by IBM, beat world chess champion Garry Kasparov in 1997 [1]. Moreover, as minimax was being constantly studied by people through self-playing, Google's DeepMind Company designed state-of-the-art intelligent player Alphazero.

However, AI still does not play Connect-4 game in a super optimal way. Connect-4 was first solved by James Dow Allen, and independently by Victor Allis in 1988 [2]. Even though they introduced a knowledge-based approach and several winning strategies, ways to optimize the search algorithms were not taken into consideration. Thus, it would be nice to increase the efficiency and functionality of search algorithms by a certain function. In such cases, heuristics to optimize the minimax algorithm would be desirable.

Connect Four, known as Captain's Mistress, is a two-player connection game on a 6×7 board first published by Milton Bradley in 1974. The complexity of this game is so high that we can see it more clearly how heuristics optimizes minimax in Connect-4. In order to become a strong Connect Four player, there are two ways to play Connect Four: defensive and aggressive. Defensive AI prevents its opponent from winning, whereas aggressive AI makes every possible move to connect four in a row ahead of its opponent; this paper discuses a program that relies on the "aggressive" way [3].

Furthermore, this paper mainly applies minimax with alpha-beta pruning to play Connect Four. Minimax computes minimax values of each following node, and uses backtracking to find out the best move [4]. The two players are called MAX and MIN separately. MAX makes moves to maximize its score while MIN tends to minimize MAX's score. The minimax algorithm predicts the state of the board ahead of time in order to make the best move.

Minimax search algorithm is good at predicting its opponent's move and then beating it, but the runtime of minimax is always an issue. In order to shorten its runtime, this paper applies alpha-beta pruning to minimax. Since time is too limited for minimax to look at every node in the game tree, the main goal of alpha-beta pruning is to increase minimax's efficiency by pruning any unnecessary

move that has no influence on making the final decision [4].

However, even though runtime is shortened, it still takes too much time for minimax to reach peak optimization levels in some cases. For example, when playing Connect Four game, computers lack the ability to search the bottom value of game tree in an optimal speed. In order to speed up the progress of making the best move, heuristic functions are applied in this paper. Heuristic functions determine which branch to follow by sorting the alternatives in each branch-step based on available information [5]. Since minimax picks the highest value that heuristic generates, heuristic becomes so essential that every subtle change in it can alter the final outcome of the move.

Therefore, this paper compares how two heuristics improve the functionality of minimax. Since the effectiveness of heuristic is influenced by depth of searching and number of features, this paper also compares how depth of searching and number of features improve heuristics.

This paper is organized as follows. In Section 2, we review a brief history of minimax and some current research on game areas. In Section 3, we review the methodology of minimax in Connect-4 and introduce two heuristics with detailed explanation of features. What's more, we explain methods to assess experiments. In Section 4, we conduct three experiments and analyze the relationship among functionality of heuristics, depth of searching, and number of features. Finally, we conclude the paper with a discussion of future in Section 5.

2. Related Work

Minimax algorithm basically comes from the "Minimax theorem", which was proposed by John von Neurmann in 1928. At first, minimax theorem was used in zero-sum game with two players knowing all moves that have taken place so far. However, John von Neumann improved and extended the minimax theorem that involving imperfect information games with more than two players, published this result in [6] 1944 (written with Oskar Morgenstern). Later, A. Wald extended Von Neumann theory that M and N (finite dimension simplices) are allowed to be subset of certain infinite dimensional linear fields. Further, Sion Maurice greatly generalize Von Neumann theory in [7] by unifying two arguments (disjoint convex sets by a hyperplane and yields the theorem of Kneser-Fanor or a fixed point theorem and a yields Nikaidó's result) by proving a minimax theorem for a function that is quasi-concave-convex and appropriately semi-continuous in each variable.

From the work [6] generalized by von Neumann and Morgenstern, they proposed theories of rational choice for a special class of situations in which a person is faced with choosing between various alternatives, and where the person knows that the outcome of his choice is in part a function of how another person, who is presumed to be equally rational, chooses. Following the previous researchers steps, T. Parthasarathy in 1970 states in [8] that a particular class of games has a mixed value, provided that at least one of the players has a strategy

that is restricted to absolutely continuous distributions with respect to the Lebesgue measure.

Basically, minimax theorem was first used in the game field with zero-sum games and still played an important role to solve basic modern games such as tic-tac-toe and connect-N currently. The progress of minimax to play an optimal game starts with a groundbreaking paper. In 1950, Claude Shannon published [9], which first put forth the idea of a function for evaluating the efficacy of a particular move and a "minimax" algorithm which took advantage of this evaluation function by taking into account the efficacy of future moves that would be made available by any particular move. Later on, after several years attempts and efforts, machine Deep Blue successfully defeat the chess grandmaster Kasparov in 1997.

In addition, minimax can also be applied to different fields' problems. In a zero-sum game with two players, there is always a specific and rational strategy for player to reach the optimal income, which regard as a core content of game theory [10]. This strategy can be used in financial and economic field when the decision maker tries to optimize its own profit. Besides, combined with other algorithm and mathematical method, minimax is also involved in network design [11], robotic fields [12] and public electing field [13].

3. Methodology

3.1. Connect-4 Game

Connect-4 game is a chess game on a board of 7 vertical columns of 6 squares each. Two players make their moves in turn till 4 men are connected horizontally, vertically or diagonally. Once a man is put in one of the columns, it will fall down to the lowest unoccupied square in the column.

In our research, we designed an intelligent program to seek for the best move, using Minimax and heuristic search. The main steps for solving the best move are summarized in Figure 1.

Therein, the Minimax and the heuristic functions are expanded in the following sections.

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Figure 1. Pseudo-code of minimax.

3.2. Minimax

Minimax is used in artificial intelligence for decision making. In most cases, it is applied in turn-based two player games such as Tic-Tac-Toe, chess, etc. In our Connect-4 chess game, Minimax aims to find the optimal move for a player, assuming that the opponent also plays optimally.

In Minimax, there are two players called Max and Min. Starting with Max trying its first move, Minimax algorithm will try all the possibilities of combination of Max's and Min's move. When either one wins or the game comes to a draw, an evaluation value of the board will be given to indicate the situation of the board. If some features on the board are in favor of Max, a positive value will be given to that feature. Otherwise, a negative value will be given. The final evaluation value is the summation of all the values of features. Max will choose the maximum evaluation value and Min will choose the otherwise. Eventually, Max will decide the best move. A descriptive figure is shown below.

Figure 2 is a possible game tree. When it is Max's turn, he has two possible moves. It can either result in the left situation or the right situation in the second layer of Figure 2. Then, it is Min's turn. Under each of the two situations, Min also has two possible moves. And it will result in four possible situations which are terminal nodes that indicate the search stops because of either the game ends or the maximum search depth is reached. Then, the evaluation value will be given according to the features of the situation. Evaluation of the situation will be discussed in section 3.3. The numbers in the circles of the third layer are the final evaluation value. Min in the second layer will choose the minimum number from its children. Therefore, −20 and −10 are chosen. Then, Max in the first layer will choose the maximum number from its children. Therefore, −10 is chosen. Eventually, Max will decide that the right path is the best choice.

The flowchart for Minimax algorithm is illustrated as following Figure 3.

In our research, this algorithm is specifically for Connect-4 chess game. To decide which column to play, it creates a game tree by trying out all the possibilities

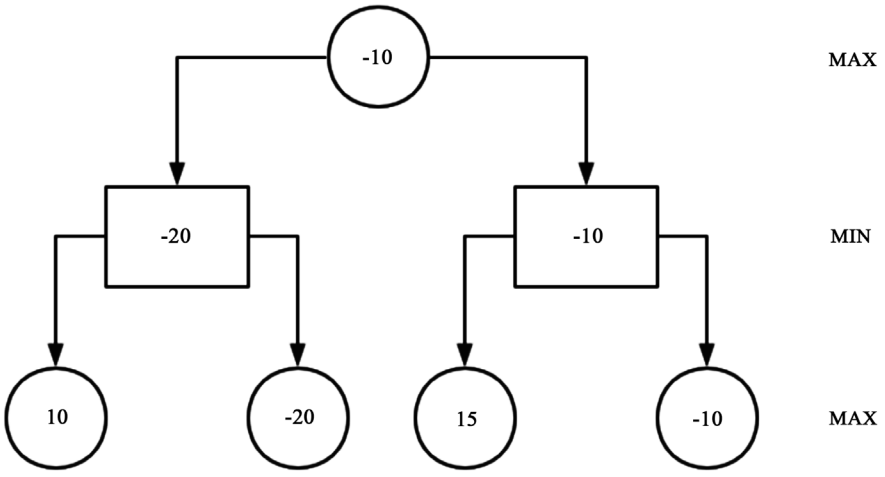

Figure 2. A possible game tree.

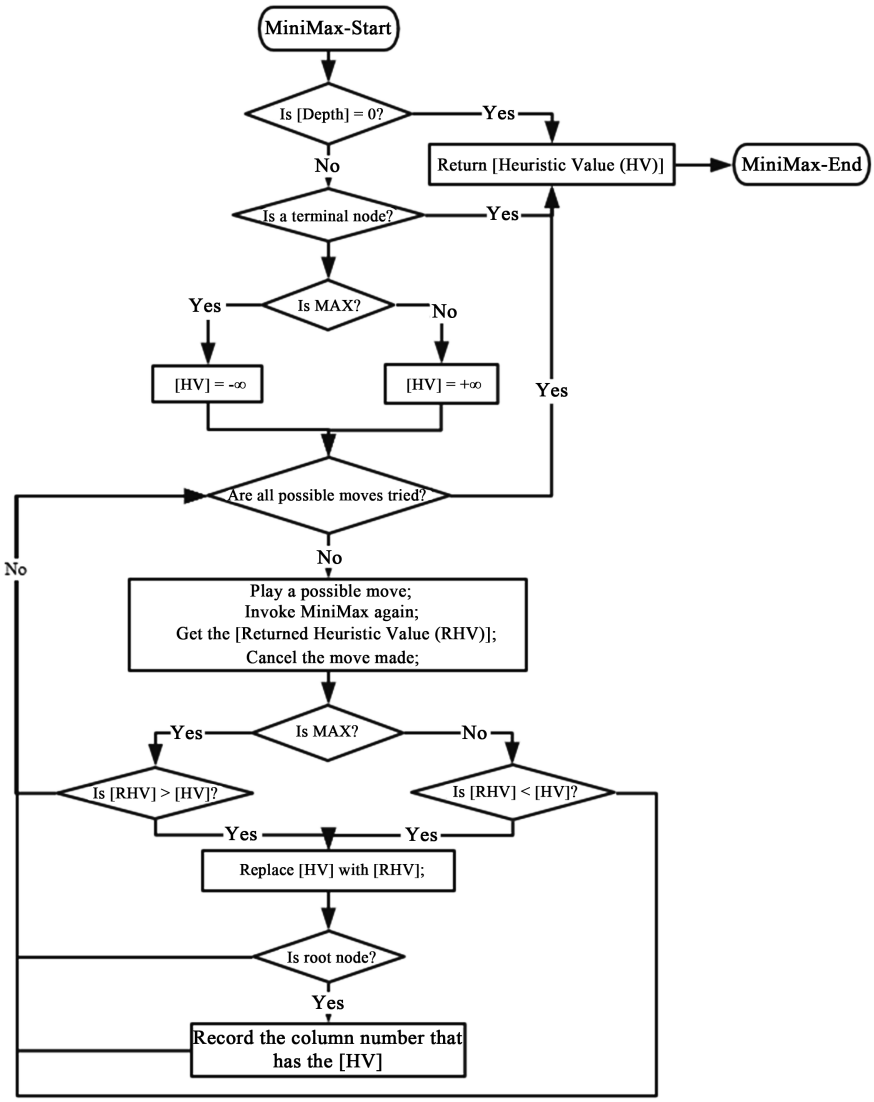

Figure 3. Flowchart for minimax algorithm.

of combination of Max's and Min's moves. Each node of the game tree is initially given a heuristic value. If it is Max node, it's given minus infinity. If it is Min node, it's given infinity. When the pre-set depth is reached or the game ends, the heuristic value will be returned to its father node. The father node compares the returned value and its value and chooses the bigger one. When the root node receives a returned heuristic value, it will record not only the bigger heuristic value but also the column number corresponding to the value. After the root node tries out all the seven columns, the best column is recorded. Hence, the algorithm finds out the best move.

However, this algorithm is time-consuming and space-wasting. **Figure 4** demonstrates the inefficiency of the algorithm.

When the game tree reaches its terminal nodes in the third layer, heuristic values are given to the nodes. Min on the left in the second layer chooses the lowest value (−10) and returns it to the father node. The father node chooses the

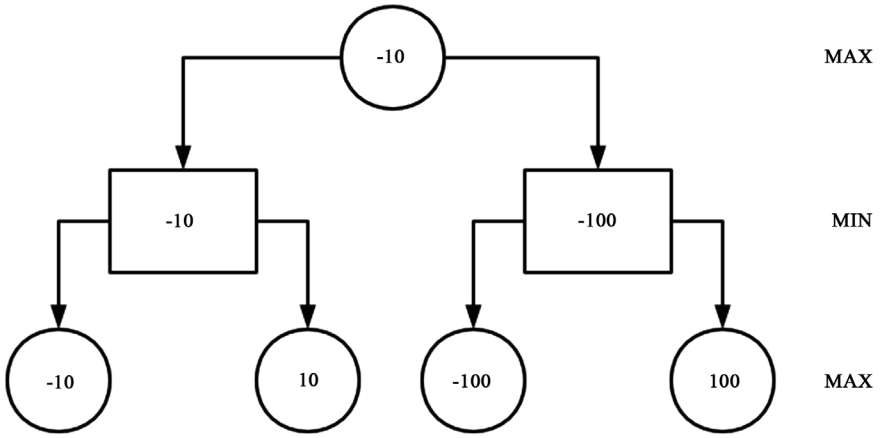

Figure 4. A possible game tree.

bigger one from the returned value (−10) and its initial value (−∞). After the right Min node receives its left child's value (−100) and finds out it is lower than the value of Max in the first layer, it should not search its right child, because the root node will never choose the value returned by the right Min node. Therefore, the algorithm can be optimized in such a way which is called alpha-beta pruning.

3.3. Heuristics

Heuristic function is used in Minimax for evaluation of the current situation of the game. The final decision made by Minimax largely depends on how well the heuristic function is. Therefore, designing a reasonable heuristic function is paramount. In our research, we designed a heuristic function for Connect-4. To evaluate the current situation of the game, the heuristic function firstly looks for different features on the board and then gives them proper values. Finally, the heuristic function returns a summation of all the values of features on the chess board. We also introduced another heuristic function in [14]. It evaluates the board in a different way. It doesn't look for features on the board. Instead, it evaluates each square on the board and gives them a proper value. We want the two heuristic functions to fight against each other so that we can assess them.

First, we will introduce the way for locating a chessman on the board. Then we will discuss the two heuristic functions separately in Section 3.3.1 and Section 3.3.2.

Figure 5 shows the way we locate the chessman on the board. For example, the red chessman on the left bottom corner is located as (a, 1).

3.3.1. Heuristics-1

In our research, we look for 4 kinds of different features from the board. They are listed in **Table 1** below.

For these 4 features, we give them different values listed in **Table 2** below.

Now we will expand on the four features.

Feature 1 shows that one of the two players wins the game. The winner has absolute advantage over the game. Therefore, Infinity is given to the feature to

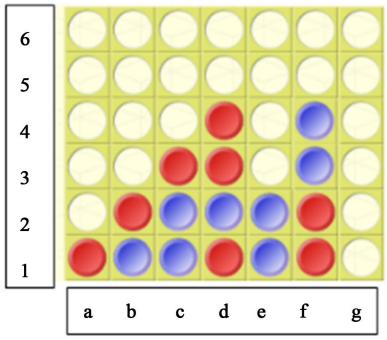

Figure 5. A typical way to locate the chessman.

Table 1. Different features on board.

Feature 1: Absolute win
Four chessmen are connected horizontally, vertically or diagonally as shown in **Figure 5**: (a, 1)-(b, 2)-(c, 3)-(d, 4).
Feature 2: Three connected chessmen
Three chessmen are connected horizontally, vertically or diagonally as shown in **Figure 5**: (c, 2)-(d, 2)-(e, 2) or (c, 1)-(d, 2)-(f, 4).
Feature 3: Two connected chessmen
Two chessmen are connected horizontally, vertically or diagonally as shown in **Figure 5**: (b, 1)-(c, 1).
Feature 4: Single chessman
A chessman that is not connected to another same chessman horizontally, vertically or diagonally as shown in **Figure 5**: (d, 1).

Table 2. Values assigned to different features.

Feature 1	Infinity		
Feature 2	A move can be made on either immediately adjacent columns.	Infinity	
	A move can only be made on one of the immediately adjacent columns.	900,000	
	A same chessman can be found a square away from two connected men.	900,000	
	A move can be made on either immediately adjacent columns.	50,000	
Feature 3	A move can only be made on one of the immediately adjacent columns. (The value depends on the number of available squares along the direction till an unavailable square is met.)	Number of available squares	Values
		5	40,000
		4	30,000
		3	20,000
		2	10,000
Feature 4	In column d	200	
	In column a or g	40	
	In column b or f	70	
	In column c or e	120	

indicate the winner's absolute win. In Figure 5, (a, 1)-(b, 2)-(c, 3)-(d, 4) are connected, which indicates the red player's win.

Feature 2 is the occasion when a player have three men connected. We generally give this feature a score of 900,000 or Infinity because there is probably going to be a winner. For example, if three men are connected horizontally, its horizontally adjacent squares have three possibilities listed below.

Figure 6 shows the availability of the three connected men's immediately adjacent squares. In the Left board, (c, 1)-(d, 1)-(e, 1) are connected. (b, 1) and (f, 1) are available. The red player's win is unstoppable. Therefore, this feature will be given Infinity. In the Middle board, (a, 1)-(b, 1)-(c, 1) are connected. Only (d, 1) is available. The red player is probably going to win at (d, 1) in the next move or be stopped by the opponent. Therefore, this feature will be given a lower score, 900,000. In the Right board, (d, 1)-(e, 1)-(f, 1) are connected. Both squares adjacent to these three men are not available. It has no promising future horizontally. Therefore, this feature will be given 0.

The unavailability of a square not only depends on if there is an opponent chessman in that square but also depends on if the square can be put a chessman immediately. In Figure 7, for (c, 2)-(d, 2)-(e, 2), (b, 2) is not available because there cannot be put a chessman immediately. The chessman put in the column will fall down to (b, 1).

One special situation in Feature 2 is that a same chessman is met one square away from the direction of two connected men. It is shown in Figure 8. This special situation is (b, 1)-(c, 2)-(e, 4). For this situation, the red player can win the game at (d, 3). It only needs to force the other player to play at (d, 2). But we

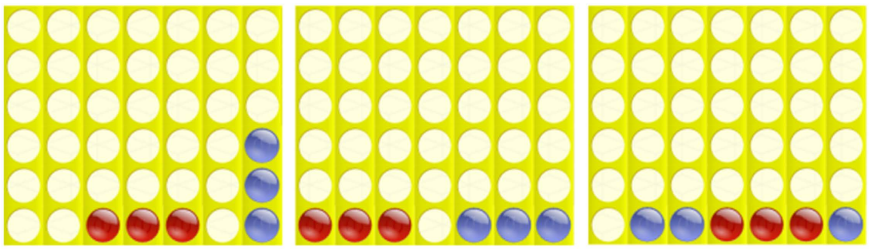

Figure 6. Three possibilities of the adjacent squares for feature 2.

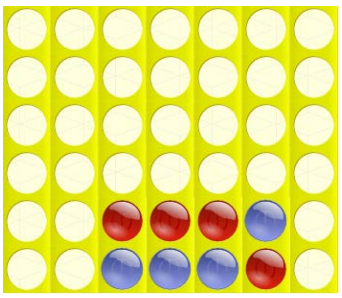

Figure 7. Unavailable squares (b, 2) and (f, 2) for the three connected men (c, 2)-(d, 2)-(e, 2), (b, 2).

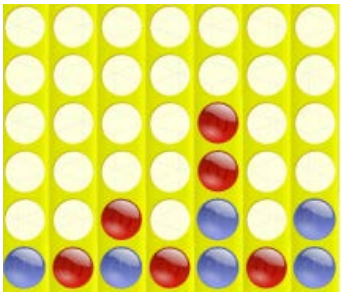

Figure 8. One special situation in Feature 2.

are not sure if red will play at (d, 3). Actually, red's win can be stopped, which is just like the situation in the Middle board of **Figure 6**. Therefore, we give it 900,000.

Feature 3 is the occasion when two men are connected. We discussed 3 situations of Feature 3 because they will have different influence on the game. The 3 situations are listed in **Figure 9**.

The most promising future for the Left situation in **Figure 8** is that it can form three connected men (b, 1)-(c, 1)-(d, 1) in the row. And a chessman can be put in either adjacent columns (a, 1) and (e, 1). For the Middle situation it will form three connected men (b, 2)-(c, 2)-(d, 2) with only one adjacent column (e, 1) in which there can be put a chessman. However, for the Right situation, there cannot be put a chessman in the adjacent columns (a, 2) and (d, 2) of the two connected men (b, 2)-(c, 2). Therefore, we give this feature in the Left situation a bigger value than the other two features. And the value for the feature in the Middle situation is also bigger than in the Right situation.

For the Middle situation, we consider the number of available squares to the two connected men's right till an unavailable square. If there are more available squares, the player has more choices to make a move, which guarantees a more promising future, hence a higher score.

However, the value for Feature 3 should be much lower than Feature 2. Feature 3 and Feature 2 are separately considered because Feature 2 provides more threats than Feature 3. Besides, for a certain search depth, Feature 3 in a terminal node is never going to become Feature 2 due to the limited depth.

Feature 4 is the occasion when a chessman is not adjacently connected to any other same chessman. It is shown in **Figure 10**. Feature 1 has 4 situations: (d, 1), (c, 1) and (e, 1), (b, 1) and (f, 1), (a, 1) and (g, 1). We will discuss how we give them values in the following.

The values for this feature are much lower than Feature 3 because Feature 4 is much less powerful than any combinations. If the chessman is put in the middle, it can form 4 connected men horizontally, vertically, diagonally. Therefore, we give it biggest value among the 4 situations. If the chessman is in column c or e, it cannot form 4 connected men in one of the diagonal line. Therefore, it is given a lower value. If the chessman is in column b or f, it has less expansion space in one of the directions in its row and diagonal lines. Therefore, it is given a lower value. The value for the last situation is lower for the same reason.

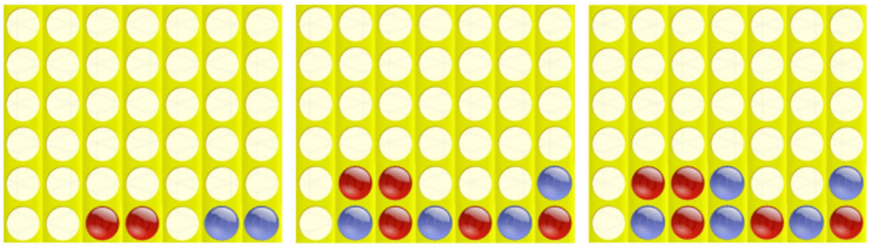

Figure 9. Three of the situations of Feature 3.

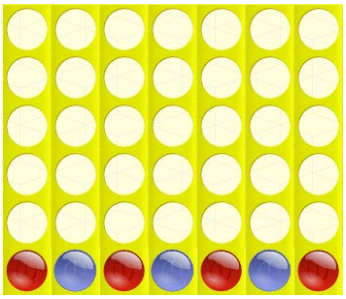

Figure 10. Single chessman.

Now values for all the features are given. In the heuristic function, it looks for all the features and calculates the summation of the values for the detected features. If the feature is in favor of us, it has a positive value. Otherwise, it has a negative value. The flowchart for the heuristic function is shown below (**Figure 11**).

3.3.2. Heuristics-2

Different to heuristic 1, heuristic 2 doesn't look for specific features on the board. Instead, it looks into every square on the board and gives them different evaluation values. If the square is more promising, it will get a higher value. The value for each square is shown in the following matrix.

$$\begin{bmatrix} 3 & 4 & 5 & 7 & 5 & 4 & 3 \\ 4 & 6 & 8 & 10 & 8 & 6 & 4 \\ 5 & 8 & 11 & 13 & 11 & 8 & 5 \\ 5 & 8 & 11 & 13 & 11 & 8 & 5 \\ 4 & 6 & 8 & 10 & 8 & 6 & 4 \\ 3 & 4 & 5 & 7 & 5 & 4 & 3 \end{bmatrix}$$

If the square is close to the middle column and row, it has a bigger value. For example, if a chessman is at (d, 3) or (d, 4), it has the biggest expansion space. It can form 4 connected men in its whole horizontal line, whole vertical line and whole diagonal line. However, if a chessman is put at (a, 1), it can only form 4 connected men in its half horizontal line, half vertical line and half diagonal line. This square has much less possibility in forming 4 connected men than middle squares. Therefore, the values are corresponding to the square's expansion space.

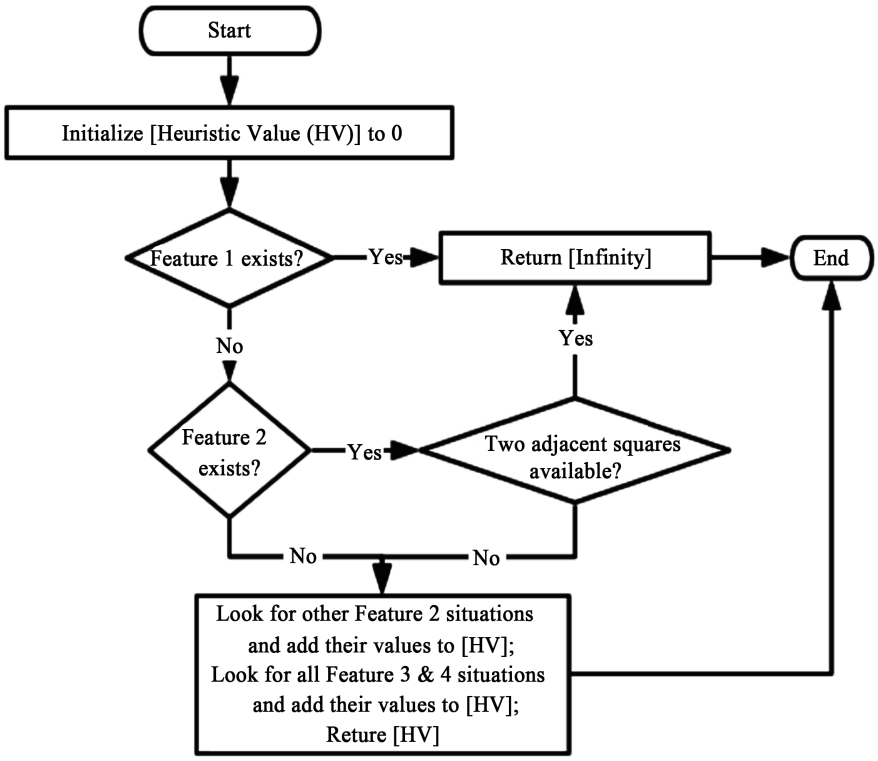

Figure 11. Flowchart for heuristic function.

If the expansion space is larger, it has a bigger value. Otherwise, it has a lower value. In other words, this gives a measurement of how useful each square is for winning the game, so it helps decide the strategy.

3.4. Assessment

In order to assess a heuristic function, we put forward 3 indexes to evaluate it. We discuss changes of the 3 indexes under different situations. In each situation, we fix other variables and let one variable vary to see how the indexes change. The 3 indexes and 4 variables are listed below **Table 3** and **Table 4**.

We will discuss changes of indexes under the following situations.

4. Experiment Design & Results & Analysis

Using visual studio 2015, we made a program and assessed the two heuristics under the situations in **Table 5**. The results are shown below.

In situation 1, in order to find out the influence of depth on its winning percentage, we let two H1s fight again each other using different depths. H11 plays the first move. **Table 6** is the result. In **Table 6**, the first row is the depth for the first H1 and the first column is the depth for the second H1. The left values in the bracket are the winning percentage of the first H1. And the right values are the winning percentage of the second H2.

From the table, we can observe that in the same column, if the depth is higher, it tends to have a higher winning percentage. It is mainly because that if the

Table 3. Indexes used to evaluate a heuristic function.

Time	Time that it takes to make a decision
The number of nodes	The number of nodes that Minimax creates to make a decision
Winning percentage	The number of winning rounds/the number of total rounds

Table 4. Four controlled variables.

Depth	Depth that Minimax searches to
Random move	In our research, we set Random move = 3. Then, Minimax chooses randomly from the first two best columns at 3^{th}, 6^{th}, 9^{th}, ... move.
The number of features	The 4 features can be included or excluded in the heuristic function.
Heuristic function	In our research, we will let two different heuristic functions fight against each other to see how the indexes change.

Table 5. Situations of which we discussed the influence on the indexes.

	Heuristic 1
Situation 1	Use all features. Let heuristic 1 with different even depth fight against each other for 100 rounds. The depths can be 2, 4, 6 and 8. We will discuss the influence of depth on the winning percentage, the number of nodes Minimax searched at each move, time Minimax consumed to make a decision.
Situation 2	Let Depth = 4. Discuss the winning percentage under the following situations: Use only one of the features, Use feature 4 and feature 3, Use feature 4 and feature 3 and feature 2, use all the features.
Situation 3	Let Depth = 2, 4, 6, 8 separately. Let heuristic 1 fight against heuristic 2 with a same depth for 100 rounds. Discuss the influence of different views of the features of the board on the winning percentage.

Table 6. Winning percentage of each depth pair when H1 fights against itself for 10 rounds.

Depth (h11) \ Depth (h12)	2	4	6	8
2	(0.5, 0.5)	(0.5, 0.3)	(0.9, 0.1)	(0.8, 0.2)
4	(0.2, 0.7)	(0.5, 0.4)	(0.6, 0.4)	(0.7, 0.3)
6	(0.1, 0.9)	(0.4, 0.6)	(0.5, 0.5)	(0.5, 0.3)
8	(0.0, 1.0)	(0.3, 0.7)	(0.2, 0.8)	(0.4, 0.4)

depth is higher, it can search more situations and thus make better decisions. However, the winning percentage of a heuristic with higher depth sometimes can be larger than the other heuristic. It is mainly because that the two heuristics will make a random move every three moves and the rounds of combat are small. If we increase the number of rounds, the winning percentage of a heuristic with larger depth is bigger than the one with lower depth.

We set depth = 4. And we let two heuristic 1 with the same depth fight against

each other. Figure 12 is the result of it. It shows the change of the node numbers and time at each move. From the figure, we can see that time and the node numbers decrease with the increase of moves. It is because that if there are many chessmen on the board, only a small number of squares are available. Besides, there is a positive correlation between the number of nodes and time.

We set depth = 6. Figure 13 is similar to Figure 12. It shows similar features as Figure 12 for similar reasons.

In situation 2, we discussed the winning percentage of H1 with different features over H2. The results are listed below.

Form Table 7, we can observe that only a single feature of H1 is not powerful enough to defeat H2. And Feature 1 has a similar winning percentage as Feature 3, which indicates the two features are approximately equally strong. And Feature 2 is also similar to Feature 4 in terms of their winning percentage.

From Table 8, we can observe that the winning percentage increases when more features are combined. And if we only use Feature 1 and Feature 2 for H1, the performance of H1 is close to H2.

In situation 3, we set Depth = 2, 4, 6 and 8, and discussed the winning percentage of H1 over H2 to compare the two heuristics to see which way of evaluating the board is better.

From Table 9, we can see that when Depth = 2, our heuristic H1 is so much weaker than H2 due to there are less features on the board. When the game just begins, there are not so many features on the board. And Minimax with depth 2 can only search the situations abundant in feature 4 or 3. It might make some wrong moves at the beginning, thus losing the game. And when the Depth = 4, our heuristic is a little stronger than H2. The overall tendency is that when the Depth is 4 or larger, Minimax can search more situations. Therefore Minimax can go for the right path, which guarantees a higher winning percentage.

Table 7. Winning percentage of using single feature of H1 over H2.

Feature1	Feature 2	Feature 3	Feature 4
0.48	0.40	0.48	0.40

Table 8. Winning percentage of using combined features of H1 over H2.

Feature 1 & 2	Feature 1 & 2 & 3	All features
0.54	0.60	0.80

Table 9. Winning percentage of H1 and H2 when let them fight against each other at same depths.

Depth = 2	0.24 for H1, 0.76 for H2
Depth = 4	0.60 for H1, 0.39 for H2
Depth = 6	0.76 for H1, 0.22 for H2
Depth = 8	0.81 for H1, 0.19 for H2

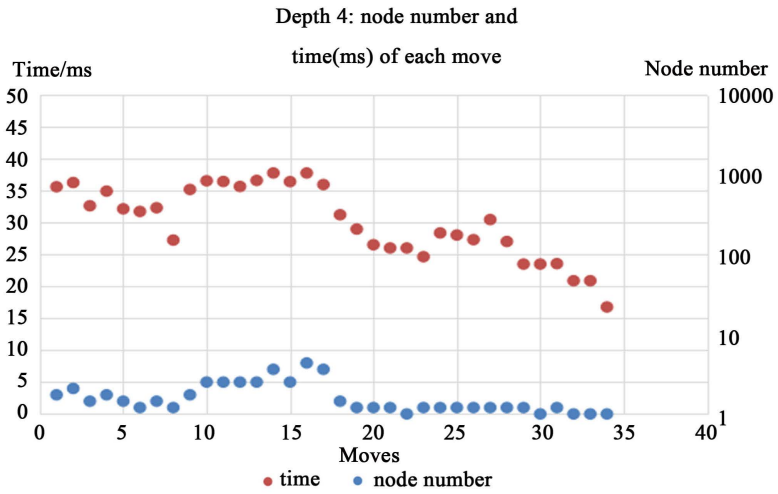

Figure 12. Node number and time of each move at depth 4

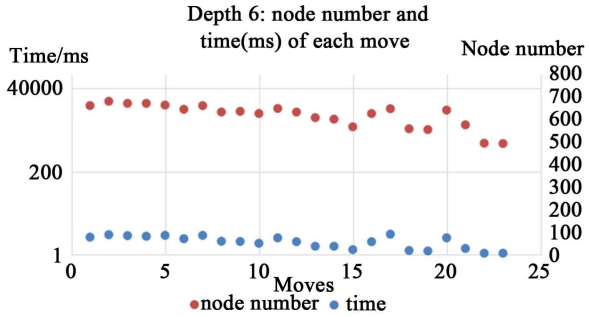

Figure 13. Node number and time of each move at depth 6.

5. Conclusions

In this paper, we have conducted three experiments on the relationship among functionality, depth of searching, and number of features. As most previous works focused on winning algorithms and knowledge based approaches, we complement these works with analysis of heuristics.

Furthermore, we find out that as depth of searching keeps increasing, a heuristic has better functionality. Moreover, we also find out that as number of features rises up, a heuristic becomes more optimal. Besides, if we increase the search depth of a relatively weaker heuristic with much less number of features, that "weaker" heuristic can beat its opponent with more features.

Finally, even though heuristics has been successfully applied to minimax in Connect-4, there is significant room for future improvement such as tuning method. For example, if one of the heuristics loses the game, we can adjust its heuristic value by 1% higher or lower to make it sounder. In the future, we can apply tuning method into minimax to make that algorithm more optimal.

Acknowledgements

Many people have offered us valuable help during the thesis writing, including our tutor, group mates and teachers from CIS project.

Firstly, we would like to give our sincere gratitude to Pro. Bart Selman, our tutor who, with extraordinary patience and consistent encouragement, gave us great help by providing us with necessary materials, advice of great value and inspiration of new ideas. It is his suggestions that make our group having consistent progress during the research. Without his strong support, this thesis could not been the present form.

Then, I pleased to acknowledge my classmates for their volunteered efforts during the testing part of the system developing. They graciously make considerable comments and sound suggestions to the function of the system.

Finally, I would like to express my gratitude to the teacher from CIS project. Without their help, our thesis could not reach this formal level. They have offered very useful suggestions and teach us how to fulfill the standard requirement of different periodical.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Krauthammer, C. (2018) "Be Afraid". https://www.weeklystandard.com/be-afraid/article/9802

[2] Allis, L.V. (1988) A Knowledge-Based Approach of Connect-Four. *ICGA Journal*, **11**, 165-165. https://doi.org/10.3233/ICG-1988-11410

[3] Shock, M. and Shaout, A. (2007) AI for a Connect 4 Game.

[4] Russell, S. J. and Norvig, P. (2016) Artificial Intelligence: A Modern Approach. Pearson Education Limited, Malaysia.

[5] Pearl, J. (1984) Heuristics: Intelligent Search Strategies for computer Problem Solving.

[6] Neummann, J. and Morgenstern, O. (1944) Theory of games and Economic Behaviour.

[7] Sion, M. (1958) On General Minimax Theorems. *Pacific Journal of Mathematics*, **8**, 171-176. https://doi.org/10.2140/pjm.1958.8.171

[8] Parthasarathy, T. (1970) On Games over the Unit Square. *SIAM Journal on Applied Mathematics*, **19**, 473-476. https://doi.org/10.1137/0119047

[9] Shannon, C.E. (1988) Programming a Computer for Playing Chess. In: Levy, D., Ed., *Computer Chess Compendium*, Springer, New York, NY, 2-13. https://doi.org/10.1007/978-1-4757-1968-0_1

[10] Roughgarden, T. (2018) Complexity Theory, Game Theory, and Economics. Electronic Colloquium on Computational Complexity, Trier.

[11] Madsen, K., Schjær-Jacobsen, H. and Voldby, J.B.R.G.E.N. (1975) Automated Minimax Design of Networks. *IEEE Transactions on Circuits and Systems*, **22**, 791-796. https://doi.org/10.1109/TCS.1975.1083973

[12] Neto, G. and Lima, P. (2005) Minimax Value Iteration Applied to Robotic Soccer. *Proceedings of the IEEE ICRA* 2005 *Workshop on Cooperative Robotics*, Barcelona, 18-22 April 2005.

http://pdfs.semanticscholar.org/8bec/440961868d889d7e2011569dc9239fe3535a.pdf

[13] Brams, S.J., Kilgour, D.M. and Sanver, M.R. (2007) A Minimax Procedure for Electing Committees. *Public Choice*, **132**, 401-420. https://doi.org/10.1007/s11127-007-9165-x

[14] https://github.com/Qtrain/Java/blob/master/src/unfinishedProjects/connectfour/Board.java