# Determining the Best Move in Chess using Greedy Algorithm

Muhammad Cisco Zulfikar - 13518073
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jalan Ganesha 10 Bandung*
*13518073@std.stei.itb.ac.id*

***Abstract*—Board games have been the entertainment source of many people from old days. There is an abundance of board games to choose from. One of the most popular board games is chess. Chess is also one of the simplest and yet complex game to play. There are many combinations, play styles, and strategies on how to play it. This paper will be using the Greedy algorithm as a strategy for the game of chess.**

***Keywords—chess; Greedy; strategy.***

## I. Introduction

Games have been the source of entertainment for people for a long time. Nowadays, there are many types of games to choose from, for example, board games.

Board games have a range of genres to choose from. There are puzzle-based board games, strategy board games, and many more. Chess is one of those game people consider as a strategy game.

Chess is a turn-based strategy board game between two players. This game can be played by casual and professional players. Each player has a set of chess pieces. Those pieces are a piece of a King, a piece of a Queen, two pieces of Rooks, two pieces of Knights, two pieces of Bishops, and eight pieces of Pawns. The game revolves around capturing opponent's pieces with the goal of capturing the opponent's King.

*Image 1.1. Chess [1]*

## II. Greedy Algorithm

Greedy algorithm is one of the most basic and widely used algorithm strategy. The fundamental of this algorithm is, as its name suggests, *greed*. It has the principle "Take what you can get now!". That principle is then realized in every step of its solution.

The algorithm will choose the minimum or maximum value in each step. This decision guarantees that in each step, the most optimal solution is determined. For every step it chooses, there will be a solution candidate. The solution should lead to an optimal local solution. Then, it will be tested for the feasibility in contributing to the solution. If the candidate is deemed feasible, it will be inserted as a part of the global solution. The selection is based on the mindset that choosing the best solution will guarantee the optimum local solution and in turn hopefully guarantee the optimum global solution.

A note to remember is that unlike exhaustive search, an optimum local solution does not always bring us to the optimum global solution in some cases. In general cases, however, greedy can be guaranteed to give global optimum solution. Due to its simplicity, Greedy Algorithm is popular to choose as a viable algorithm to be used in optimization problems (problem concerning the need of maximum or minimum of a certain value).

In general, there are five elements in a greedy algorithm.

1. Candidate Set

   The candidate set, or C, is the amount of possible options to choose. One or none of those options will be the local optimum solution. In other words, the candidate set comprises of all the different possible choice of local solution which can be chosen and will be evaluated furthermore by the other functions.

2. Solution Set

   The solution set, or *S*, is the global solution given as result of inserting local optimum solution one by one after each iteration. When the algorithm is signalled to stop, the set will then be given as the solution, hoping to give the optimum global solution.

3. Selection Function

The selection function defines how the local optimum solution will be selected from the candidate set. It evaluates each candidate. If the local optimum solution satisfies the function requirement, it will be chosen for further evaluation. This function is arguably the core part of the algorithm. It is where the programmer defines what is really needed to solve the problem. If the programmer is concerned for a minimum solution, the programmer would consider what it really means by minimum, which value is taken as consideration, and so on.

4. Feasibility Function

Feasibility function is a function defined to aid the selection function in determining the local optimum solution. The function is used to determine whether the candidate chosen from the selection function as a result is feasible in contributing to the global optimum. In other words, "can it be considered a part of solution?".

5. Objective Function

The objective function is connected to the problem itself. It determines what kind of solution that the programmer wanted.

For example, we take the "money exchange" problem where the goal is to change the current value of money into a set of available coins. For this problem, we have a candidate set of coins valued at 1, 5, 10, and 25, respectively. We also have a solution set where the total value of coins is the exact same as the exchanged value. The selection function for this problem is to select the highest-valued coin first. The feasibility function is to check whether the chosen solution locally is no more than the previous value. The objective for this problem is to have the least amount of coins which can be exchanged for the money we have currently.

Therefore, if we wanted to exchange a money with a value of 7, the solution a greedy algorithm would provide would be a 5-value coin and two 1-value coins.

In another example, such as the "knapsack" problem, there are a few heuristics we could choose to obtain the optimum solution. Greedy by profit is where we choose purely based on choosing an object with the maximum value first. Greedy by weight is where we choose purely based on an object with the lightest weight to put in as much objects into the knapsack. Greedy by weight is a strategy where we choose an object based on the value by weight.

There is also a variation of the "knapsack" problem, called as "fractional knapsack" problem. This problem cannot be solved using exhaustive search but can be solved using greedy algorithm.

In comparison, greedy algorithm provides a strategy that is viable. It also generally does better performance than brute force or exhaustive search. [2]

## III. CHESS

Chess is a turn-based strategy board game between two players. It has been around sometimes before the 7th century, although it was a different kind of chess. The chess we know now is based on the standardized modern rules in the 19th century.

Each player has a set of chess pieces. Those pieces are a piece of a King, a piece of a Queen, two pieces of Rooks, two pieces of Knights, two pieces of Bishops, and eight pieces of Pawns. Each piece type moves differently. The most powerful being the Queen and the least powerful being the Pawn.

The objective of this game is to capture or *checkmate* the opponent's King by placing it under inescapable threat of capture, and when a checkmate happens, the player that did the checkmate will be considered as the winner. The player also wins if the opponent resigns or runs out of time in a timed game. Draw can be concluded if the two players moves repetitively, when there is a stalemate, or if an agreement is made between the two players.

### A. Board

The game is played on a square board of eight rows (called *ranks*, denoted 1 to 8 from bottom to top according to White's perspective) and eight columns (called *files*, denoted *a* to *h* from left to right according to White's perspective). The 64 squares alternate in color and are referred to as *light* and *dark* squares.

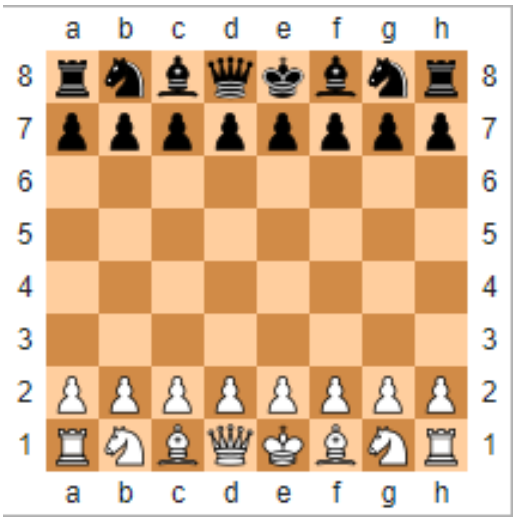


*Image 2.1. Initial setup with pieces and board denotation on the chess board*

### B. Movement

Moving is necessary and it is illegal to skip a turn, even when moving a piece would put the player in a disadvantage. A piece is moved to either an unoccupied square or one occupied by an opponent's piece. This means the opponent's piece is captured and then removed from the board.

*C. Pieces*

Each piece can move in its own way. A piece cannot leap other piece intervening its way, except for the Knight.

1. King

The King moves one square in every direction.

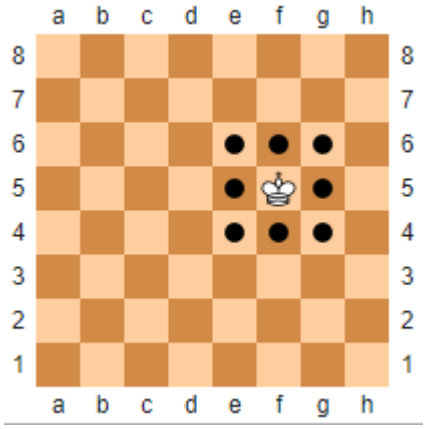

*Image 2.2. Moves of a king*

2. Queen

The Queen combines the power of a Rook and a Bishop and can move indefinitely along the row, column, or diagonals it currently sits unless a piece intervenes its movement.

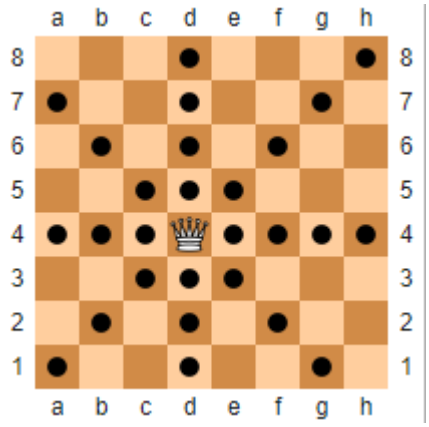

*Image 2.3. Moves of a queen*

3. Rook

The Rook can move indefinitely along the row or column it currently sits unless a piece intervenes its movement.

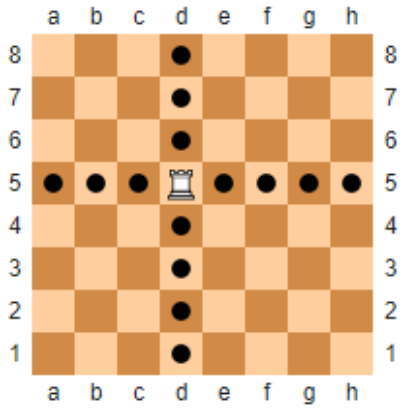

*Image 2.4. Moves of a rook*

4. Bishop

The Bishop can move indefinitely along the diagonals unless a piece intevenes its movement.

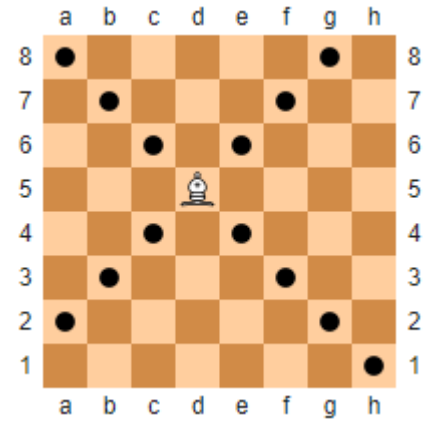

*Image 2.5. Moves of a bishop*

5. Knight

The Knight can move to any closest square that is not along its row, column, or diagonal, thus forming an L-shape. It can also leap other pieces.

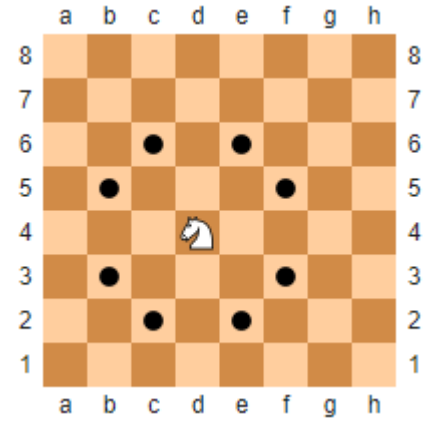

*Image 2.6. Moves of a knight*

6. Pawn

The Pawn can move forward two squares on its first move, then moves one square unless it is intervened by another piece. It can also capture opponent's piece on a square diagonally in front of it (marked by X).

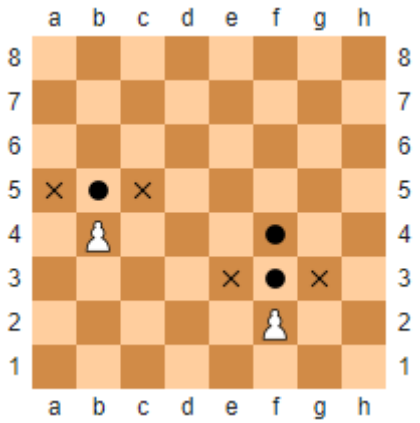


*Image 2.7. Moves of a pawn*

### *D. Chess Notation*

Chess games and positions are recorded in a system of notation, most used, the algebraic chess notation. It generally records moves in the format *initials of the piece – file where it moved – rank where it moved.*

The pieces initial is *K* (king), *R* (rook), *B* (bishop), *N* (knight), and none for pawns. An additional letter is added to indicate the file or rank from which the piece moved. If a piece makes a capture, "x" is added before the destination square. If a piece does a check, "+" is added after the initial formats. If a piece does a checkmate, "#" is added after the initial formats. These added letters and symbols are to remove confusions. [3] For example:

a. **Ngf3** means "knight from the *g*-file moves to the square *f3*"

b. **R1e2** means "rook from the first rank moves to the square *e2*"

c. **Bcxf3** means "bishop from the *c*-file captures on the square *f3*"

d. **Bxc3+** means "bishop moves to the square *c3* and checks the king"

## IV. Determining the Moves

In this section, we will try to determine the best move possible using the greedy algorithm in chess. Before determining, we need to define the selection function for this problem.

For this problem, we need to define two play styles.

1. Cautious

   This play style will use their weaker pieces (the pawns) to move forward first. This hopefully will get the attention of the opponent by using their stronger pieces to capture the pawns, which in turn free the player's stronger pieces movement.

2. Aggressive

   This play style will use the stronger pieces available to move. This hopefully will result in the opponent opening precious square and free up the movement for the player's pieces to get to the king piece.

After defining those two play styles, we will determine the selection function for both play styles. We use these values to easily determine whether selecting one of those option is the optimum solution. [4]

a. Pawns are valued 1.

b. Knights and Bishops are valued 3.

c. Rooks are valued 5.

d. Queens are valued 9.

e. Kings are valued 100.

With those pieces got their valuation, these are the following selection function for both playstyles.

1. Cautious

   Selection functions for this play style are by the following consideration.

   a. Use the lowest-valued piece type (the pawn), start moving forward from the most left and continue by moving forward the next pawn to the closest right column. A pattern of 2-1-2-1 (then mirrored) if possible, will be prioritized.

   b. Move the next higher-valued piece type in a possible direction that make the pieces closer to the opponent's king, starting from the most left with each piece moves one turn. Choose the farthest piece from the opponent's king to move if two types of the same value have possible moves. Capturing to get closer is permissible. Do this except for the King.

   c. Repeat step a and b, after no higher possible piece type can move, except the King

   d. If possible, capture the opponent's piece to defend only if it goes through more than halfway the ranks

   e. If the player's king piece is checked, move to the furthest possible square from the piece it got checked from. If not possible, capture or use any piece to block.

2. Aggressive

   Selection function for this play style are by the following consideration.

a. Start by moving the highest-valued piece type possible, except the King, to move from the most left in the direction that leads the piece closer to the king. Each piece moves one turn. Choose the farthest piece from the opponent's king to move if two types of the same value have possible moves.

b. Move the two most left and right pawn pieces in a 2-1 (then mirrored) pattern.

c. Repeat step a and b if possible.

d. Always try to capture opponent's piece regardless of the result. If possible, capture the highest-value piece that is capture-able.

e. If an opponent's piece is more than halfway through the ranks (row), capture or move forward a pawn on the same row as the opponent's piece if possible.

f. If the player's king piece is checked, capture or move to the closest possible square the piece it got checked from. If not possible, capture or use any piece to block.

## V. Tests and Results

The test will be executed against a level 1 Computer. This test will be conducted using the website, *chess.com*. The player (in this case, the writer) will always conduct this test using the White pieces from the White's perspective. The result in this paper and the result from the YouTube video might be different due to different testing.

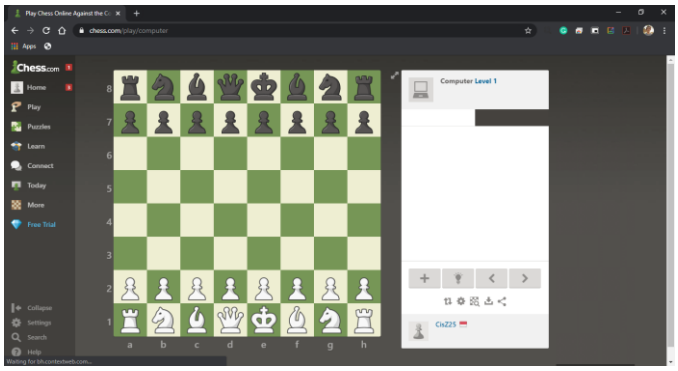

*Image 3.1. Initial setup of a digital chess board against a Level 1 Computer.*

### A. *Using the Cautious Playstyle*

This playstyle managed to form a line of defense in a form of a row of pawns. Thus, it managed to deter a bit of offense for the first few moves played. However, this playstyle quickly fails after the row of pawns try to forward its defense. The player loses from a checkmate.

*Image 3.2. A row of pawns acting as a first line defense using Cautious playstyle*

History of the gameplay:

**(1) a4, Nf6 -- (2) b3, Nc6 – (3) c4, Nb4 -- (4)d3h5 -- (5) e3, c5 -- (6) f4, Ng4 -- (7) g3, g6 -- (8)h4e6 -- (9) Nc3, Qc7 -- (10) Nf3, b6 – (11) Bd2, Qb7 – (12) Be2, e5 – (13) Rc1, e4 -- (14) Rf1, f5 – (15) a5, Bg7 – (16) d4, d6 – (17) Bd3d5 – (18) Nxd5, Nxd3+ -- (19) Ke2, cxd4 – (20) Ne5, Be6 – (21) Bb4, Nxc1+ -- (22 Qxc1, bxa5 – (23) c5, axb4 – (24) Ne7, Qxe7 – (25) Nd7, d3+ -- (26) Kd1, Bxb3+ -- (27) Ke1, Bc3+ --(28) Qd2, Bxd2+ -- (29) Kxd2, Qg7 – (30) c6, Qc3# -- White loses**

### B. *Using the Aggressive Playstyle*

This playstyle managed to put an offense using the high-value pieces. The player also managed to capture more pieces and checked the opponent's king. But consequently, the player lost the high-value piece very quickly. Thus, it breaks the strategy of this playstyle. The player loses from a checkmate.

*Image 3.3. The player managed to do a check on the opponent's King using Aggressive playstyle*

History of the gameplay:

**(1) Nc3, d6 – (2) Nf3, h5 – (3) a4, c5 – (4) b3, Nd7 – (5) g3, a6 – (6) h4, e6 – (7) Nd5, Ngf6 – (8) Ne5, Ne4 – (9) Bb2, Nxg3 – (10) fxg3, exd5 – (11) Bg2, Nxe5 – (12) Ra3, g5 – (13) Rf1, f5 – (14) a5, Bd7 – (15) b4, Bc8 – (16) g4, fxg4 – (17) hxg5, Bg7 – (18) Bxe5, Bxe5 – (19) Bxd5, Qxg5 – (20) Re3, Qh4+ -- (21) Rf2, g3 – (22) b5, gxf2+ -- (23) Kf1, Qh2 – (24) Bf7+, Kd8 – (25) Rxe5, axb5 – (26) a6, Qg1# -- White loses.**

## VI. Room for Improvements

The selection function could be improved more by expanding the possibility of choices it has. This solution using greedy algorithm is just as a proof of concept that this can be done. However, this solution to the problem can also be used as a proof that it is almost impossible to beat someone in chess using the greedy algorithm.

## VII. Conclusion

Greedy algorithm as one of the most widely used algorithm has proven itself to be useful in certain cases. However, making a strategy to win and determining the best moves available on chess are one of the hardest things to do using greedy algorithm. A lot of factors are needed into consideration. Nevertheless, greedy algorithm still provides a solution, even though it is not a globally optimum solution.

## Video Link at Youtube

## Acknowledgment

First, I wanted to thank God for giving me strength and passion in writing this paper. I also wanted to express my gratitude to my Algorithm Strategy lecturer, Ms. Masayu Leylia Khodra. Her passionate and imaginative way of teaching and exploring induces the author to have more creativity and passion in learning. I would also like to express my happiness to my parents and friends for supporting my idea on this paper. I wanted to apologize for any kind of mistake during the making of this paper and learning process. Finally, I hoped that this paper can read publicly and induces curiosity to its reader.

## References

[1] Staxringold. "Photo of a Staunton chess set taken by me from the vantage point of the white player opening.". 11 July 2006. Wikimedia. Taken from https://commons.wikimedia.org/wiki/File:Chess_board_opening_staunton.jpg. Accessed on 3 May 2020.

[2] Munir, Rinaldi. Algoritma *Greedy*. Taken from http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf. Accessed on 3 May 2020.

[3] Erik. Chess Notation – the Language of the Game. 25 March 2019. Taken from https://www.chess.com/article/view/chess-notation. Accessed on 3 May 2020.

[4] Capablanca, Jose; de Firmian, Nick (2006). Chess Fundamentals (Completely Revised and Updated for the 21st century). Random House. ISBN 0-8129-3681-7.

## VIII. Statement

With this statement, I declare that this paper is a product of my work, not an adaptation, a translation of other person's work, nor formed by a result of plagiarizing.

Bandung, 3 May 2020

Muhammad Cisco Zulfikar
13518073