The Application of Greedy Algorithm to Four in a Line Game

Adhiguna Surya 13509077
Informatics Engineering Study Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509077@std.stei.itb.ac.id

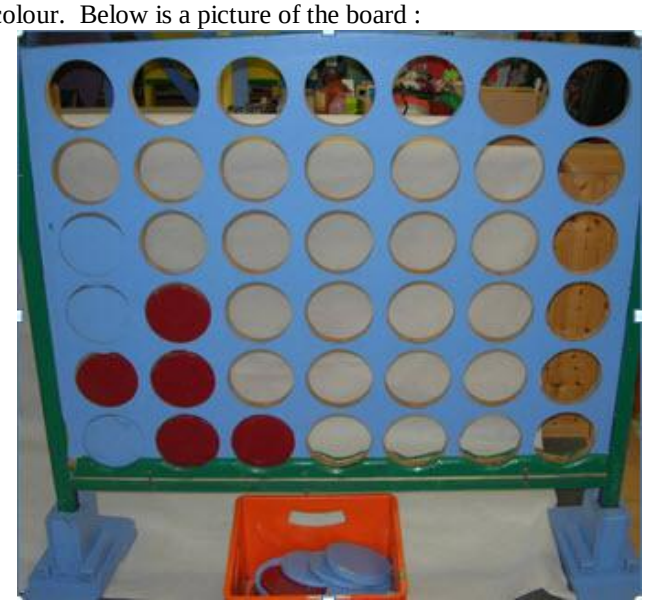
***Abstract*—This paper explains the use of the greedy algorithm in the game of 4 in a line, also known as connect four. The game has been in the market for decades, yet it has remained an exciting and intellectually stimulating game, enjoyed by a wide range of people, from children to adults alike. There are two alternatives of greedy algorithm covered in this paper, the first one is greedy by value, and the second one is greedy by opponent's value. The greedy algorithm makes locally optimal decisions with the hope of finding the globally optimal solution. The former focuses on maximising gain, while the latter focuses on minimising the opponent's gain. Since those algorithms are still in its elementary, unoptimised version, the algorithms still have a lot of weaknesses. In most cases, the algorithm fails to yield the optimal solution. The algorithm's main weakness is its short-sightedness, as all greedy algorithms do not search exhaustively, therefore rarely resulting in the best solution.**

***Index Terms*—ConnectFour, Greedy by value, Greedy by opponent's value, locally optimal.**

I. INTRODUCTION

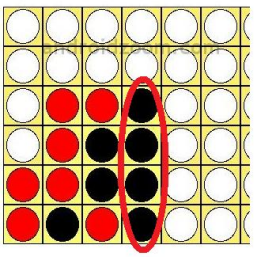
1.1 The Game Play of 4 in a Line

At the first glance, the game 4 in a line might seem overly simplistic. All that the game requires are two players, the board, and 42 coloured disks, 21 for each colour. Below is a picture of the board :



.**Picture 1 : The Board**

As seen above, the board consists of 7 columns and 6 rows. The gameplay is equally simple. Each player must pick a colour (in the picture above the two colours are red and woodish brown), and then the players take turns dropping their coloured disks, one disc for every player's turn. The coloured disks will then drop and occupy the next empty spot within the column. The first player who is able to connect four disks of the colour they picked, whether horizontally, vertically, or diagonally, becomes the winner of the game.



Picture 2 : winning condition

The picture above illustrates a winning condition for a game of 4 in a line. There are four vertically connected black-coloured disks, and therefore the black player becomes the winner of the game. The match might also yield a draw, when there are no unoccupied spaces left, yet no player is able to connect four disks of the colour he picked.

Despite its simplistic nature, over the years the game has gained much popularity. The intellectually stimulating nature of the game is one of the reasons why the it remains consistently popular since February 1974 when it was first sold under the brand of ConnectFour.

The game was mathematically solved by James Allen in 1981 and subsequently by Victor Allis in 1988. The first player can do a perfect play by dropping his first disk in the middle column. Despite that, the game remains wildly popular to this day.

1.2 The Greedy Algorithm

The greedy algorithm is a heuristics algorithm that makes locally optimal decision at each stage with the hope of finding the globally optimal solution. The greedy algorithm is mostly used for optimisation problems, yet it

can be used to solve other types of problems as well. However, contrary to other, more complex algorithms such as brute force and dynamic programming, the greedy algorithm does not guarantee a globally optimal solution.

Greedy algorithm is most often used in situations an approximation suffices. In such situations, finding the exact optimal solution usually requires significantly more steps. The greedy algorithm for the Travelling Salesman Problem (TSP) illustrates how greedy algorithm works.

Using the greedy algorithm, in every city visited, the salesman will visit the nearest unvisited city. The previous step is repeated until the salesman has come across all the cities. On average, the greedy algorithm yields a 25% longer route than the optimal one. Despite its seemingly poor result, the greedy algorithm has polynomial worst-case complexity, which is a hefty improvement from brute force and dynamic programming algorithm which have exponential complexity in terms of big-O notation. In short, the greedy algorithm is a preferable solution when only close enough approximation is required.

The greedy algorithm, however, yields optimal solutions for a particular subset of problems, namely the matroids. Some examples are the activity selection problem.

Formally, the algorithm is has five pillars ::

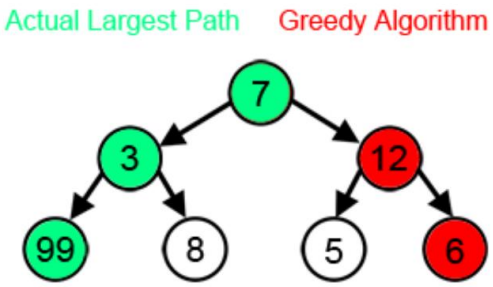
1. A candidate set, from which a solution is created

2. A selection function, which chooses the best candidate to be added to the solution

3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution

4. An objective function, which assigns a value to a solution, or a partial solution, and

5. A solution function, which will indicate when we have discovered a complete solution

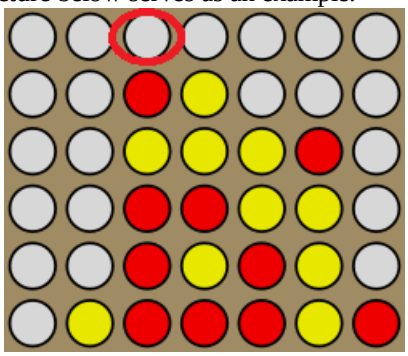


Picture 3 : Greedy algorithm illustration

The picture above further illustrates how the greedy algorithm works. The objective is to find the path that has the biggest cost, with each node visited will contribute to the total. The greedy path chooses the red-coloured nodes, namely 7, 12, and 6, with the total value of 25. Meanwhile, the optimal largest path consists of the green-coloured nodes, which are 7, 3, and 99, with the total value of 109. In this case, the greedy algorithm fails to yield the optimal solution because of the short-sighted nature of the algorithm.

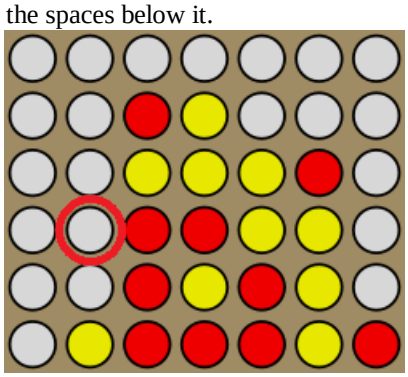
II. GREEDY BY VALUE

The first greedy algorithm for the game covered in this paper is greedy by value. The algorithm assigns a value for every unoccupied space within the board. The value is determined by the number of their disks that will be connected if the disk is to be dropped there. The disk is then dropped to the space with the biggest value. The picture below serves as an example.



Picture 4 : Application in the game

Assume that it is the red player's turn. The space that is marked in red circle has a value of two, since if another red disk is dropped there, two red disks can be connected. Before assigning a value to a space, the algorithm first checks whether a disk can be dropped there, since some spaces cannot be reachable without dropping some disks to the spaces below it.



Picture 5 : Application in the game

The space in the red circle has a value of three, since another red disk dropped there will cause three red disks to be horizontally connected. However, the space is actually inaccessible, because a disk dropped into the particular column (the second column from the left) will drop to the space right below the intended space. Therefore, the space marked in red circle is assigned a dummy value of -1 to mark the space as inaccessible and therefore not included in the set of plausible solutions.

The step-by-step algorithm is provided below :

1. For every unoccupied space, check whether or not a disk can be dropped there. If a disk dropped on the space's column will land precisely on the space,

then go to step 2. If not, assign a dummy value of -1 to the space, and move to step three.

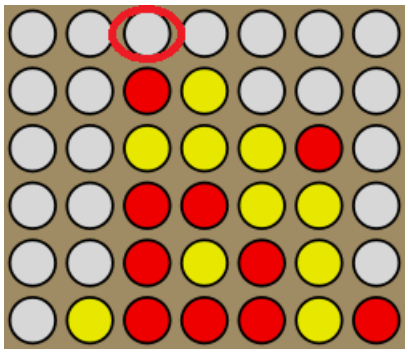
2. Assign a value to the space. The value is determined by how many disks of the same colour can be connected whether horizontally, vertically, or diagonally.
3. Repeat step 1 for the next available space until all the spaces have been assigned a value
4. Drop the disk on the space with the biggest value. If the largest value is shared by more than 1 space, pick any one of those spaces.

The algorithm above is expressed in the formal notation below.

1. The candidate set consists of all the unoccupied spaces.
2. The selection function chooses the available space with the largest assigned value.
3. The feasibility function checks whether a disk dropped on the space's column will drop exactly to the space, not any of the spaces below it.
4. The objective function assigns a value to each available space, with the value determined by the number of disks of the same colour that will be connected whether horizontally, vertically, or diagonally by dropping a disk on the space's column.
5. The solution is reached once every available space has been visited in the iteration.

Analysis of the greedy by value algorithm

Unsurprisingly, the algorithm fails to yield the optimal value for the game. The perfect play cannot be reached by using the greedy algorithm. In addition, the algorithm has several flaws that have to be taken into consideration. Consider the case below.



Picture 6 : Application in the game

The space marked by the red circle is assigned a value of two, since the space passes the feasibility function and a red disk dropped there will result in two vertically connected red disks. In the case above, the largest value is two.Consequently, the algorithm might choose to drop the red disk on the space marked by red circle.

However, the algorithm fails to consider other factors. If a red disk is dropped on the space, then the third column from the left will be full and no further disks can be dropped on the column. As a result, two vertically connected disks is the maximum number of connected disks. This falls short of the four connected disks required to win the game. The algorithm fails to take this factor into account and might still drop the red disk into the marked space.

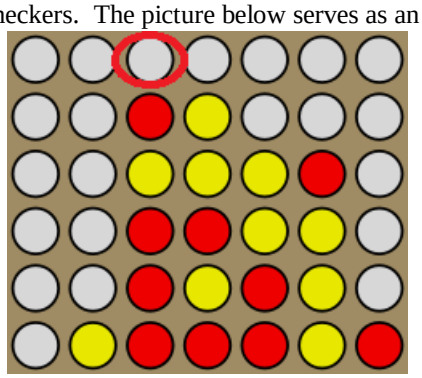
Another weakness of the greedy by value algorithm is that it does not stop the enemy from winning the game. When the enemy already has three connected disks of their colour and can put the fourth one in their next turn, the algorithm has no mechanisms to block the enemy in order to prevent them from winning the game.

Some of these weaknesses can be prevented by focusing on blocking the enemy. However, that alternative is also not flawless. This paper will cover this alternative on chapter three. Another alternative is adding some mechanism to the greedy by value algorithm that handles special case where the enemy has three connected disks. In such case, the algorithm will focus on blocking the enemy; once the enemy is prevented from winning, the algorithm will resume its normal mechanism.

The second alternative indeed has its own flaws. There are some cases where the opponent has more than one way of connecting four disks of the colour they picked on the next turn. In such case, regardless of what blocking movement the player makes, the opponent wins the game on his next turn.

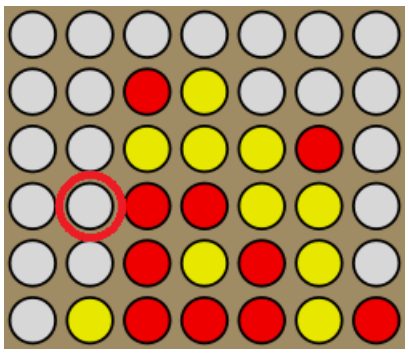
III. GREEDY BY OPPONENT'S VALUE

The second greedy algorithm for the game covered in this paper is greedy by opponent's value. The algorithm assigns a value for every unoccupied space within the board. The first algorithm, greedy by value, makes the moves that maximize gain. On the contrary, the greedy by opponent's value algorithm makes moves that minimize the enemy's gain. This is quite comparable with defensive approach in other board games such as chess and checkers. The picture below serves as an example.



Picture 7 : Application in the game

Assume that it is the red player's turn. The space that is marked in red circle has a value of four, since if the opponent drops another yellow disk there, the opponent will have four diagonally-connected yellow disks. The focus of this algorithm is to prevent the enemy from the maximum gain, leaving them with moves that only have little gains.



Picture 8 : Application in the game

The space in the red circle has a value of three, since if the space remains available until the next turn, the opponent can connect three yellow disks on that space. However, the space is actually inaccessible, because a disk dropped into the particular column (the second column from the left) will drop to the space right below the intended space. Therefore, the space marked in red circle is assigned a dummy value of -1 to mark the space as inaccessible and therefore not included in the set of plausible solutions.

The step-by-step algorithm is provided below :

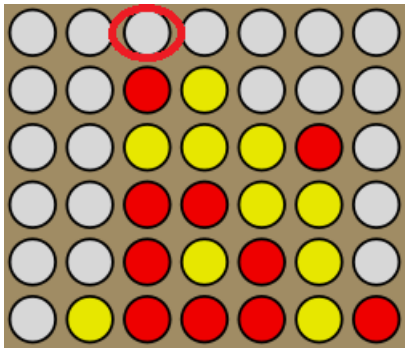
1. For every unoccupied space, check whether or not a disk can be dropped there. If a disk dropped on the space's column will land precisely on the space, then go to step 2. If not, assign a dummy value of -1 to the space, and move to step three.
2. Assign a value to the space. The value is determined by how many disks the enemy can connect if he is to put the disk of the colour he picked on the space.
3. Repeat step 1 for the next available space until all the spaces have been assigned a value
4. Drop the disk on the space with the biggest value. If the largest value is shared by more than 1 space, pick any one of those spaces.

The algorithm above is expressed in the formal notation below.

1. The candidate set consists of all the unoccupied spaces.
2. The selection function chooses the available space with the largest assigned value.
3. The feasibility function checks whether a disk dropped on the space's column will drop exactly to the space, not any of the spaces below it.
4. The objective function assigns a value to each available space, with the value determined by the number of opponent's disks connected if the opponent is to drop their coloured disk on that particular space.
5. The solution is reached once every available space has been visited in the iteration.

Analysis of the greedy by opponent's value algorithm

Unsurprisingly, the algorithm fails to yield the optimal value for the game. The perfect play cannot be reached by using the greedy algorithm. In addition, the algorithm has several flaws that have to be taken into consideration. Consider the case below.



Picture 9 : Application in the game

The space marked by the red circle is assigned a value of four, since the space passes the feasibility function and a red disk dropped there will result in two vertically connected red disks. In the case above, the largest value is two.Consequently, the algorithm might choose to drop the red disk on the space marked by red circle.

However, the algorithm fails to consider other factors. If a red disk is dropped on the space, then the third column from the left will be full and no further disks can be dropped on the column. As a result, two vertically connected disks is the maximum number of connected disks. This falls short of the four connected disks required to win the game. The algorithm fails to take this factor into account and might still drop the red disk into the marked space.

Another weakness of the greedy by opponent's value algorithm is that the algorithm merely blocks the opponent's movement, without considering its own chance of winning the game. Therefore, the player putting the greedy algorithm into practice is unlikely to win the game. In most cases, the best that can be achieved is a draw, since focusing on blocking the opponent's movement usually results in both players unable to secure any substantial advantage.

Some of these weaknesses can be prevented by providing some mechanisms for the algorithm to maximise its own gain when the opponent's state is not threatening. However, that alternative is also not flawless. Another alternative is adding some mechanism to the greedy by value algorithm that handles special case where the player has three connected disks. In such case, the algorithm will focus on securing victory, because the player already has three connected disks, he only needs one more connected disk to secure victory. When the opponent is not in any position to make threatening moves, focus should be shifted on maximising gain instead of merely minimising the opponent's gain.

IV. CONCLUSION

The game four in a line is an intellectually stimulating game which involves a lot of planning ahead and tactical strategies. The greedy algorithm can be applied to the

game. There are two alternatives of greedy, which are greedy by value and greedy by opponent's value. The first one focuses on maximising the player's own gain, while the latter focuses on minimising the enemy's gain. Both approaches have their own strengths and weaknesses. The first algorithm has a higher chance of securing victory. However, it is also more prone to defeat than the latter since the algorithm does not block the opponent's movement at all. The second algorithm, on the other hand, has a lower chance of defeat since the player focuses on blocking the enemy's movement. However, it is also very unlikely to yield victory, since the algorithm does not include any strategies on connecting four of their own disks in a line. Major improvements and optimisations are needed for the two algorithms to yield substantially better results.

VII. ACKNOWLEDGMENT

The author would like to thank Mr. Rinaldi Munir and Ms. Ayu Purwarianti as the lecturers of IF3051 Strategi Algoritma, as this paper would not be finished had it not been for their relentless teaching efforts.

REFERENCES

[1] *Cormen, Thomas H., et al. Algorithms, Cambridge : Mc-Graw Hill Book Company. 2001.*
[2] http://en.wikipedia.org/wiki/Connect_Four
[3] *Vazirani, www.cs.berkeley.edu/~vazirani/**algorithms**/chap5.pdf*.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

ttd

Adhiguna Surya
13509077