

Greedy Algorithm in Reversi

Bagus Rahman Aryabima - 13509044

Informatics Engineering

School of Electrical Engineering and Informatics

Bandung Institute of Technology, Jl. Ganesha 10 Bandung 40132, Indonesia

Email: bagus.axel@gmail.com

Abstract—Greedy algorithm is an algorithm that makes the locally optimum choice at each stage with the hope of finding the globally optimum solution. However, because of its mechanism, greedy algorithm may not provide the globally optimum solution for some problems. Certain conditions may also cause greedy algorithm to fail in providing the globally optimum solution.

Reversi is a board game played by two players on a board with 8 rows and 8 columns with a set of distinct pieces for each side. Pieces are coins with a light and a dark face, each face belonging to one player. The players' goal is to have a majority of their colored pieces showing at the end of the game, turning over as many of their opponents' pieces as possible. The game ends when all cells are filled with pieces from both players or when one of the players can't move anymore.

Index Terms—Greedy, Othello, Reversi, Strategy

I. INTRODUCTION

Reversi was invented in 1883 by Lewis Waterman and John W. Mollett and gained considerable popularity in England at the end of the 19th century. Back then, Reversi used a different rule set. The modern rule set used on the international tournament stage originated in Mito, Ibaraki, Japan.

The game begins with two light pieces and two dark pieces placed in the middle of the board. Usually, the dark player makes the first move.

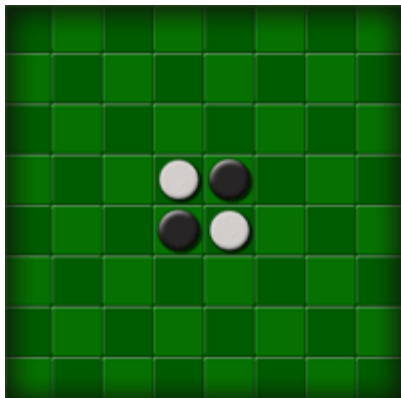


Figure 1 Reversi starting position

The dark player must place a dark piece in a position such that there exists at least one straight line (horizontal, vertical, or diagonal) between the new piece and another

dark piece already on the board, with one or more contiguous light pieces between them.

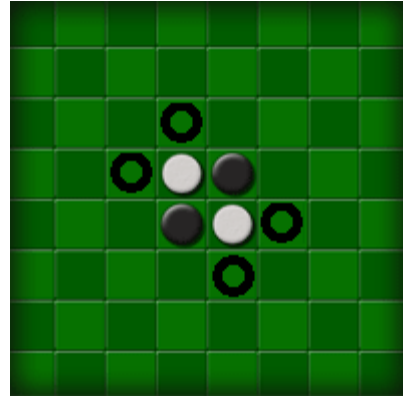


Figure 2 Black rings represent possible places to put a new black piece

After placing the piece, dark captures all light pieces lying on a straight line between the new piece and any other dark piece on the board. A valid move is one where at least one piece is reversed.

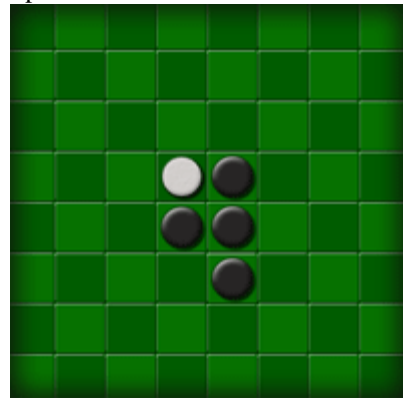


Figure 3 State of the game after a new black piece is placed on the board

Players take alternate turns. If one player can't make a valid move, then the other player gets to move instead. However, if both players can't make a valid move, then the game ends. In other words, the game may end before the board is full of pieces. At the end of the game, the player with most pieces on the board wins.

That is the usual rule set used in Reversi. The increasing popularity of Reversi actually gave birth to new and modified rule set. Nevertheless, the rule set that is explained above is the most used one.

II. ALGORITHMS

Greedy algorithm builds solution in a step-by-step basis. Any choice that we take on a step cannot be changed once we already move on to the next step. Greedy algorithm allows us to take the best choice possible at each step. In other words, greedy algorithm will always provide us with the locally optimum solution. By doing so, hopefully we will find the global optimum solution by the end of the last step.

The following elements build the greedy algorithm:

1. A candidate set, C.
A candidate set consists of elements that build the solution. In our case, the candidate set consists of pieces that we will place on the board. At each step, a candidate will be taken from this set.
2. A solution set, S.
A solution set is subset of a candidate set, because it consists of chosen candidates. Chosen candidates are members of candidate set that satisfies the feasibility function.
3. A selection function.
At each step, a selection function will choose the optimum candidate. Once we already move on to the next step, any chosen candidate will not be considered anymore. Therefore, in greedy algorithm context, we can't look back to any step that we took.
4. A feasibility function.
A feasibility function checks whether a chosen candidate, together with the entire solution set, conforms to the constraint of the problem. A valid candidate will be passed to the solution set. An invalid candidate will be disposed of, and it will be considered no more.
5. An objective function.
An objective function is responsible for optimizing the value of solution that we take at each step.

III. STRATEGIC ELEMENTS

Despite its simplicity, Reversi is known far and wide as a strategy game. Reversi players need to apply some strategies in order to have the upper hand against their opponent. There are so many ways to win in Reversi. Therefore, this paper will only elaborate some general strategic concepts in Reversi.

A. Corners

Many Reversi players, from casual players to professional players, aim to place their piece in a corner of the board. Obviously, the opponent can't subdue the pieces placed in the corners of the board. As stated above, a piece can only be flipped if it is placed in a straight line between one of the opponent's pieces and the opponent's new piece. Therefore, pieces placed in the corners of the board will be impossible to be flipped by the opponent. Another benefit of conquering the corners of the board is that we could use those positions to anchor groups of pieces permanently.

However, players must remember that grabbing a corner with bad timing may be a mistake. Players should make sure that they didn't leave any 'holes' along the edge as they are approaching the corners of the board. By simply putting their pieces in those 'holes', opponents could easily capture players' pieces along that edge. When this happens, conquering the corners of the board may not be so beneficial anymore.

B. Mobility

Formidable opponents won't let players conquer corners of the board easily, nor will they let players do any other good moves. Therefore, these opponents will force players to move further from corners, focus on defending existing pieces, and basically doing any trick that will limit players' mobility. Consequently, players will eventually do some undesirable moves, while they are doing those good moves themselves. Players must make sure that they can see the bigger picture of the game. By doing so, they can respond to these pressures well and hopefully be able to do some good moves.

C. Edges

Just like the corners of the board, conquering the edges of the board is essential in winning Reversi. When edge pieces successfully form a line with any existing piece far away from itself, it can deliver a powerful blow for the enemy. Watch your opponent staring at the board, speechless, as you flip most of his/her pieces. In other words, flips originating from edge play will usually influence moves to all regions of the board.

However, just like how it was with corners, players should not be too hurried in acquisitioning these edges. Most of the time, players are too focused in flipping as many of the opponent's piece as possible. This is what motivates them to take hold of the edges and corners as soon as possible. Few players realize that their opponent can always do the same technique that they used. Right after players succeed in securing an edge and enjoying the view of flipping most of the opponent's piece, the opponent could easily place a piece in another unguarded edge, flipping even more of the players' own piece instead. Players should play on edges only when the opponent cannot easily respond to the intended move, or when the intended move will drastically reduce the opponents' mobility.

D. Parity

Parity is all about getting the last move in every empty region in the end-game, increasing the number of stable pieces in the process. A piece is stable when it is fully guarded. That is, when along horizontal, vertical, and each diagonal axes, that piece is on a boundary (edges or corners), in a filled row, or next to another stable piece of the same color.

Players applying this strategy will take the defensive role on the early and middle stages of the game. They will

stay calm as they watch their piece subdued by the enemy, while they keep watching promising positions to subdue later in the game.

This is quite a risky strategy itself, for if this strategy played poorly, players will most likely watch most of their piece being flipped all game long. Moreover, the opponent will keep on securing the good positions, failing the carefully-planned counterattack to be played. A good parity player will start the game being defensive, while slowly being more offensive as the game progresses. At the later stages of the game, the player will have to simply fill the stable positions with his/her piece.

E. Endgame

At the later stages of the game, players will most likely change their approach to the game. The player that is dominating the game will use any means possible to limit the other party's mobility. On the other hand, the dominated player will start his/her own tricky counterattack. Consequently, counting pieces in the most final stages of the game is highly essential.

IV. IMPLEMENTATION

There are many algorithms other than greedy algorithm that we can use to solve Reversi. Unfortunately, in this paper, we will only consider the greedy algorithm approach in solving Reversi. Greedy algorithm explained in this paper will be combined with real-life strategies explained in the last chapter.

A. Greedy by Largest Number of Flips

The main objective of Reversi is to have more pieces placed on the board compared to the opponent's. Therefore, it's only normal to aim to flip as many opposing pieces as possible at every turn. This particular greedy algorithm will hopefully satisfy that desire.

Theoretically, our greedy algorithm is built based on these elements:

1. Candidate Set

The candidate set in this algorithm is all sets of same-colored pieces. Specifically, our candidate set consists of all same-colored pieces that haven't been placed on the board.

2. Solution Set

The solution set in this algorithm is all sets of pieces which placement fills as many of the opponent's piece as possible.

3. Selection Function

In Reversi, every piece is the same as any other piece in the game. If there is any difference in the pieces, it will only be the color. Therefore, there is no 'right' way to give any unique value to a piece. In this paper, every piece will be simply given a number. This number is the only thing we have regarding the selection function.

4. Feasibility Function

The feasibility function is actually quite simple.

Placement of a new piece on the board must flip as many opposing pieces as possible.

5. Objective Function

The objective function is to maximize the number of opposing pieces flipped as we place new pieces on the board.

Basically, the algorithm will do these things:

1. Check valid positions to put a new piece.
2. For every valid position, consider the location of fellow pieces that form a line with that position.
3. Measure the length of such lines. If the intended position forms two or more lines, add up the length into a single value.
4. Place the new piece where it forms the longest line possible with other fellow pieces on the board.

The algorithm will look like this, more or less, in pseudo code:

```
procedure MostFlips (input b: board)
{This procedure is used to find the
best place possible to put a new
piece. That is, the placement of that
new piece will allow us to flip as
many of the opponent's piece as
possible.}
```

Declaration

```
i : integer
NFlip : array [1..n] of integer
```

Algorithm

```
for each position in board do
IsValidPosition();
{check whether this position is a
valid position to place a new piece}
endfor
```

```
i ← 1;
for each valid positions do
MeasureCurrentLength();
{measure the length of line formed
by this position and other fellow
pieces on the board}
```

```
NFlip[i] ← CurrentLength();
i ← i + 1;
endfor
```

```
PlacePiece(Max(NFlip));
{Finally, places the piece where it
forms the longest line possible with
fellow pieces}
```

B. Greedy by Securing Key Positions

One of the best practices of Reversi that most players will apply is to secure key positions of the board. In other words, players will try to get hold of corners and edges of the board as soon as possible. Hopefully, the following

greedy algorithm can do just that.

Just like how it was with the last algorithm, there are certain elements that built this one, such as:

1. Candidate Set

The candidate set consists of all the pieces belonging to one of the players. Specifically speaking, this set consists of such pieces that haven't been placed on board.

2. Solution Set

In this algorithm, the solution set is all sets of pieces placed on key positions. As stated above, the key positions are the edges and corners of the board.

3. Selection Function

In Reversi, every piece is the same. The only unique element of those pieces, if there are any, is their color. Therefore, there is no 'absolute' way to give any unique value to a piece. In this paper, every piece will be simply given a number. This number is the only thing we have regarding the selection function.

4. Feasibility Function

The feasibility function is also quite simple. New pieces must be placed as near as possible to the key positions.

5. Objective Function

The objective function is to maximize the number of piece placed on the edges and corners of the board.

The general scheme of our algorithm is as follows:

1. Check valid positions to put a new piece.
2. For every position, consider the locations of the intended position and key positions of the board.
3. Measure the distance between the intended position and the nearest key position.
4. Place the new piece where it will be located as near as possible to edges or corners of the board.

In pseudo code notation, the algorithm will look like this, more or less:

```
procedure KeyPosition (input b: board)
{This procedure is used to place the
new piece at the best place possible.
In the context of this algorithm, the
best place is to be as near as
possible with the edges and corners of
the board.}
```

Declaration

```
i : integer
KeyP : array [1..N] of integer
```

Algorithm

```
for each position in board do
  IsValidPosition();
  {This is a procedure to check
  whether a position is valid to put a
  new piece.}
endfor

for each valid positions do
  if IsKeyPosition() then
```

```
    PutPiece();
    {If the intended position is an
    edge or corner of the board,
    immediately put a piece there.}
  endif

  break();
  {Get out of this loop.}
endfor

{If there are no valid position
located at key positions}
i ← 1;
for each valid positions do
  for all axes do
    MinDistanceToKeyP();
    {This function will measure the
    nearest key position from the
    intended position to put a new
    piece.}

    KeyP[i] ← CurrentDistanceToKeyP();
    i ← i + 1;
  endfor
endfor

if no piece placed yet this turn then
  {Because the intended position is
  not the edges nor corners of the
  board}
  PlacePiece (Min (KeyP));
endif
```

V. CONCLUSION

There are two greedy algorithms elaborated in this paper, greedy by most number of flips and greedy by securing key position early. Basically, the two algorithms approach the problem quite similarly at the start. They scan for valid positions, and then they apply their own feasibility function. After the application of feasibility function, things are getting different for these two algorithms. For the first algorithm, greedy by most number of flips, the best position is where it can flip as many of the opposing piece as possible. On the other hand, the greedy by securing key positions early, consider the best place is where it can be in the key positions of the board. If, by some circumstances, a new piece can't be placed in the key position, this algorithm will place a piece as near as possible to edges or corners of the board.

The first algorithm can be considered an offensive one. It will place a piece where it can deliver the most powerful blow possible to the enemy. When played right, this algorithm can be considered a formidable Reversi player. It will try to maintain the upper hand over the enemy, while possibly limiting their movement. For the early stages of the game, this algorithm may not be so effective. This is because there aren't that many pieces on the board yet. Benefits of using this algorithm will surface

at the mid stages of the game. There are already quite a number of pieces on the board, and the board itself hasn't been too crowded. At the later stages of the game, hopefully this algorithm will succeed in dominating the board. The only thing left to do at this stage is to maintain the upper hand over the opponent. The downside of this algorithm is that it can't consider future consequences of its move. Therefore, it can play right to the hands of a tricky enemy.

The second algorithm can be considered a tactical one. It will play normally at first, while keeping constant watch over the edges and corners of the board. When it's possible to put a new in those key positions, this algorithm will immediately do just that. As stated above, the key positions are essential in anchoring many pieces at once. Therefore, this algorithm will have that taste of counterattacking at playing Reversi. It will suddenly throw a surprising blow at enemy at will. The downside of this algorithm is that it may leave 'holes' while trying to grab those key positions. As stated above, those holes can be played to the opponent's benefit. A clever opponent will initiate an attack through those holes, flipping many of the players' pieces in the process.

VI. ACKNOWLEDGEMENT

I would like to thank God for allowing me to complete this paper. I would definitely not be able to achieve that without His help.

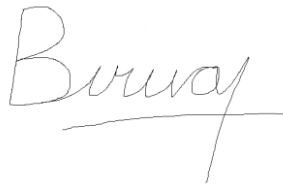
REFERENCES

- [1] http://en.wikipedia.org/wiki/Greedy_algorithm accessed on 2 December 2011
- [2] <http://en.wikipedia.org/wiki/Reversi> accessed on 2 December 2011
- [3] Munir, Rinaldi. *Strategi Algoritma*. Bandung: ITB, 2009, pp. 26-28.

STATEMENT

I hereby declare that this paper is my own writing, not an adaptation, a translation, nor a plagiarism of an existing paper.

Bandung, December 9th 2010



BAGUS RAHMAN ARYABIMA
13509044