A Reversi Playing Artificial Intelligence Implementation

Ali Can ARIK
Email: acarik@gmail.com

Görkem KANDEMİR
Email: grkemkandemir@gmail.com

Hasan ATLI
h_atli@yahoo.com

***Abstract*—Reversi is one of the most popular board games. It is invented in 1883 and modern version of the game is known as Othello and is patented to Goro Hasegawa in 1970. This study represent an artificial intelligence (AI) design to compete human being players. This paper is composed of parts containing problem definition, implementation approach including AI design method, chosen framework, and work distribution between group members.**

Keywords*—*Reversi, artifical intelligence, MiniMax Algorithm, Alpha-Beta Pruning.

I. Introduction

In this study, it is aimed to develop an artificial intelligence to constitute a competitive adversary against a human player in Reversi. This report represents preliminary studies including AI design approach and implementation framework.

II. The Game

The game Reversi is believed to be invented in 1883. Othello is the name of the modern version of the game with slight changes of rules. Since Othello is a trademark, Reversi name can be used interchangeably.

Reversi is a two-player strategy game played on a 8x8 board. There are 64 identical game pieces called disks which are light on one side and dark in the other. Each of the disks' two sides corresponds to one player; they are referred to here as light and dark after the sides of Reversi pieces, but any counters with distinctive faces are suitable. Players take turns placing the disks on the board with with their assigned color facing up. During a play, disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color. The objective is to have the majority of the disks to turned to display your color when the last playable empty square is filled.

For the modern version of Reversi, the rules state that the game begins with four disks placed in a square in the middle of the grid, two facing white side up, two pieces with the dark side up, with same-colored disks on a diagonal with each other. Convention has initial board position such that the disks with dark side up are to the north-east and south-west from both players' perspectives. The initial state is depicted in Fig. 1.

Dark must place a piece with the dark side up on the board, in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another dark piece, with one or more contiguous

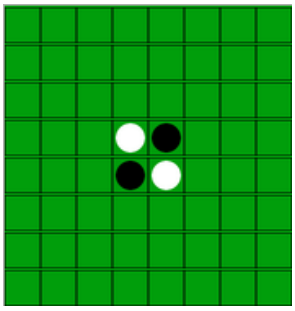

Fig. 1. Starting state of the game

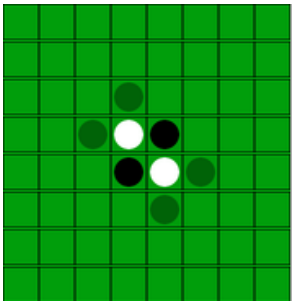

Fig. 2. Where dark may play

light pieces between them. At the initial state, dark has the options indicated by translucent pieces in Fig. 2.

After placing the piece, dark captures all light pieces lying on a straight line between the new piece and any anchoring dark pieces. All reversed pieces now show the dark side, and dark can use them in later movesunless light has reversed them back in the meantime. In other words, a valid move is one where at least one piece is reversed.

If dark decided to put a piece in the topmost location (all choices are strategically equivalent at this time), one piece gets turned over, so that the board appears as seen in Fig. 3.

Now light plays. And the games continues as such with alternating turns until neither player can move, e.g. the board is full. The player with the most pieces on the board at the end of the game wins. [1]

In summary, it can be said that the game has the following properties:

- Two-player
- Zero-sum: In any outcome of the game one person's gains equal the other one's losses.

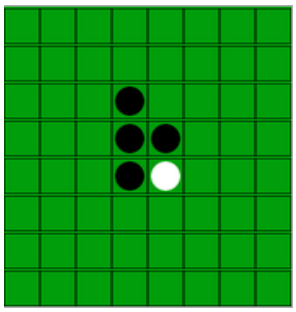

Fig. 3. After dark play

- Discrete: All game states and decisions are discrete.
- Finite: Number of states and decisions are finite.
- Deterministic: Nothing depends of randomness.
- Perfect information: Both players are fully informed about the current state at any time.

III. Previous Implementations

The Reversi computer programs have been developed since 1980s. The fact that the game requires in depth analysis of the game state after several moves, the computers have always had the upper hand against human opponents even at the first implementations. Even though the game is not solved for every move, most implementations are able to defeat best humans. In 1997, Logistello, a computer program that plays Reversi, defeated the human champion by a score of 6-0.

Early implementations of Reversi computer programs were heavily dependent on 3 strategies which are stated by [2] namely:

- The maximum disc strategy
- The weighted square strategy
- The minimum disc strategy

In maximum disc strategy, evaluation function tries to maximize the difference in the number of discs for each side. This is a very greedy algorithm and the most important drawback of this strategy is that it loses mobility very early in the mid-game phase. Mobility refers to number of moves available to player and loss of mobility force player to make the moves its opponent desires. Therefore, this is not a very successful strategy.

When Reversi is investigated, it can be seen that not every square on the board are equally strategically important. For example, corner squares are considered as stable square which cannot be claimed by the opponent once they are occupied. Therefore, these corner squares are more valuable than any other square. The weighted square strategy takes this fact into account. However, in the late-game this strategy has an important drawback, as well. At the late-game, players need to maximize number of their discs in order to win the game. However, the weighted square strategy suffers at late-game phase because it has no such concern.

In minimum disc strategy, players try to minimize their number of discs in order to maximize their mobility. Although having high mobility in mid-game is heavily desired in Reversi, it could not let the player to win the game.

Later implementations of Reversi computer programs combined these strategies together. One of the successful implementations of this approach is computer program called IAGO. It is developed by Paul Resenbloom in 1981.

IAGO uses mini-max approach with iterative deepening and employed Alpha-Beta pruning in order to reduce the number of expanded nodes. Furthermore, it solves the game tree completely at the late-game. Since number of legal moves are very limited, solving game tree completely is relatively easy at the late game phase.

Most crucial part of the IAGO is its evaluation function and as mentioned above, IAGO's evaluation function combines the 3 strategies discussed before. It consists of 4 components which are added up with different weights in order to provide a single value. Its equation can be seen below:

$$\begin{aligned} Eval(pos) = ESAC(MoveNumber) x EdgeStability \\ +36 x InternalStability \\ +CMAC(MoveNumber) x CurrentMobility \\ +99 x PotentialMobility \end{aligned} \tag{1}$$

where

$$ESAC(MoveNumber) = 312 + 6.24 x MoveNumber \tag{2}$$

and

$$CMAC(MoveNumber) = \begin{cases} 50 + 2 x MoveNumber, 1 <= MoveNumber <= 25 \\ 75 + MoveNumber, 25 <= MoveNumber \end{cases} \tag{3}$$

In this equation, EdgeStability is a table based evaluation that is precomputed by an iterative algorithm. This iterative algorithm assigns predefined heuristic values to filled edges and fills the remaining table.

InternalStability refers to whether internal squares are stable or not. IAGO uses iterative algorithm to calculate this value.

CurrentMobility refers to number legal moves player has for current position of the board. To compute this value IAGO measures the relative mobility by counting legal moves for both sides.

PotentialMobility term combines three future mobility measures: the number of frontier discs, the number of empty squares adjacent to opponent's discs, and the sum of the number of empty squares adjacent to each of the opponent's discs. IAGO calculates these 3 measures and add them up to get single value.

These 4 elements successfully covers the 3 basic strategies discussed before and efficient implementation of this evaluation function helps IAGO to became one of the most successful Reversi computer program in history.

BILL is an another successful reversi computer program developed by Kai-Fu Lee and Sanjoy Mahajan in 1986 [3].

It improves IAGO's performance further by applying some improvements in to its implementations. Moreover, it replaces InternalStability in the evaluation function of IAGO with a term called SequancePenalty. This new term recognizes long disc sequences of equal color along lines and assigns a negative value to them depending on the board location. Also to form a single value evaluation function, BILL uses a quadratic discriminant function. Coefficients of this function are calculated manually by its creator and updated with every game played. This makes BILL stronger than the IAGO and BILL wins all games against IAGO.

Finally, Logistello is the most successful Reversi computer program in the history [4]. It is developed by Michael Buro in 1997. It uses a similar evaluation function of BILL's. However, it constantly learns and updates itself by the help of thousands of opening books and closing books.

IV. Implementation Plan

In this study, the game will be expressed as a game tree, where the nodes are game states and the edges are the moves. Each leaf node corresponds a terminal state, which can either indicate an end game or a state bounded by the maximum depth of the search tree.

Play alternates between Max and Min. Max player aims to find a move that ends up with a game state whose the utility value is maximum from its point of view whereas Min aims to minimize it from the opponent's point of view.

In order to make optimal decisions, the MiniMax algorithm will be used in which it is principally assumed that every player will make struggle to maximize own gain. This algorithm relies on the assumption of optimal play the players.

The problem with MiniMax search is that the number of game states it has to examine is exponential in the depth of the tree. This can be halved using a trick by computing the correct decisions without looking at every node in the game tree. Therefore, Alpha-Beta pruning will be employed to reduce the number of nodes to evaluate. The algorithm stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further.

The utility function plays a key role in the performance of the AI in our problem. It estimates how good a state is using a numeric value. This can be done using several strategic features of the game. We are planning to use explore and develop our own utility functions once we have sufficient experience with the game. In this study, it is planned to implement the utility function similar to that of the computer program IAGO.

Our evaluation function will consist of 3 parts, as given in Eq 4.

$$\begin{aligned} eval(pos) = \\ coeffMD(moveNum) * maxDiscs(pos) \\ +const * squareWeight(pos) \\ +coeffMM(moveNum) * maxMobility(pos) \end{aligned} \tag{4}$$

- maxDiscs(pos) function tries to maximize difference in number of discs player and its opponent has in favor of player.
- coeffMD(moveNum) changes with move number. It ensures that at early-game and mid-game effect of maxDiscs(pos) is minimized and at late-game effect of maxDiscs(pos) is maximized.
- squareWeight(pos) function evaluates the importance of the square.
- const is a constant integer which ensures that effect of squareWeight is same during all stages of game.
- maxMobility(pos) function tries to maximize difference in number of legal moves player and its opponent has in favor of player.
- coeffMM(moveNum) changes with move number. It ensures that effect of maxMobility(pos) is maximized during early-game and mid-game and minimized during late-game.

V. Chosen Framework

Designed AI will be implemented in object oriented programming language C++ and Borland Developer Studio 2006 will be used as the integrated development environment (IDE). .NET VCL framework is used to accelerate the implementation process.

Preliminary studies have been done for implementation of the project. The current GUI using the draft implementation is given in Fig 4.

The main class to be used is named Game_State, whose public class variables and methods are

- Game_State, to initialize an instance of the class Game_State.
- Number_of_Possible_Moves, which is calculated once the constructor is called.
- Back_Pointer, to keep track of the last node.
- Data, to keep current game state in an array.
- Cost, to return the result of the utility function for the given game state.

It is planned to develop the implementation based on the current structure.

VI. Work Distribution

The work distributed among the team members as such:

- Ali Can ARIK: Development and implementation of the search problem as a game tree and the MiniMax algorithm.
- Görkem KANDEMİR: Development and implementation of the search problem as a game tree and the MiniMax algorithm; GUI design and implementation.
- Hasan ATLI: Development and implementation of the utility function; data structure and class abstraction.

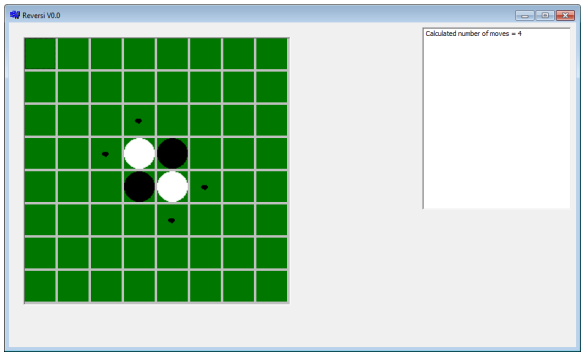

Fig. 4. The GUI

VII. Conclusion

In this study, it is aimed to implement an artificial intelligence for the game Reversi. This paper constitutes a preliminary report for the project. A general description of the game, previous efforts and our current approach to the problem along with the theorotical background are presented in this paper. Also, preliminary efforts in the selected programming framework are given.

Slight changes with the implementation plan presented are expected as the implementation progresses.

References

[1] Wikipedia contributors. Reversi. Wikipedia, The Free Encyclopedia. December 3, 2015, 16:36 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Reversi&oldid=693589941. Accessed December 3, 2015.

[2] Rosenbloom, Paul S. ”A world-championship-level Othello program.” Artificial Intelligence 19.3 (1982): 279-320.

[3] Lee, Kai-Fu, and Sanjoy Mahajan. ”BILL: a table-based, knowledge-intensive othello program.” (1986).

[4] Buro, Michael. ”Logistello: A strong learning othello program.” 19th Annual Conference Gesellschaft fr Klassifikation eV. 1995.