OTHELLO GAME

Ifrat Jahan

Your ID: 2014–2–60–051

Aiswarja Saha Joye

Your ID: 2014–2–60–114

Md.Lizur Rahman

Your ID: 2014–3–60–014

Course Name: Artificial Intelligence

Course Code: CSE365, Section: 1

Course Instructor: Amit Kumar Das

Lecturer, Department of CSE, East West University

Department of Computer Science and Engineering
East West University
Dhaka-1212, Bangladesh

Spring, 2017

Abstract

Othello utilizes the principles of game theory and implements them in a declarative programming language. Its main strength is its performance against other Othello-playing applications and it is can be distinguished from typical such applications in the following aspects. The package is built to be used as a powerful tool that can produce and analyze occurrences of interest in the games played or simulated. It can play a series of games independently and summaries the results of these in files. Amongst some of the more interesting features is the inclusion of opening libraries, customized move computation and different difficulty levels, each of which corresponds to a different approach of generating a valid move in the game. This report presents the development life-cycle of the project and derives further conclusions and contributions with respect to the initial plans.

Table of Contents

Chapter 1

Introduction

Othello is an implementation of the game Othello (also known as Reversi). The application will rely on principles of Game Theory, namely alpha-beta search and the utilities algorithm which will be explained later. Using these principles, the application is able to play the game at a very high level of competence, as well as provide a generic platform for games to be played on. Game-playing programs have existed from the early years of computer science and have improved in their ability very rapidly. Game Theory was developed and established in the 1940's, making a contribution to the world of mathematics. Some of the more prominent impacts that the topic has had is reflected in the field of economy where prediction of trends and human behavior is fundamental. Programs that play well, on the other hand, served people's curiosity and set new challenges in the scene of world-class Chess. Artificial Intelligence is now said to be tightly associated with such programs that can "think". Computers, however, use brute-force to simulate human thinking, whereas real neural functions use associations, visual memory, etc. It would be safe to say that computers have not yet reached this goal of being able to think. Brute-force may work for a game of Chess, but see Go for the very contrary. Othello can be considered an instance of the games for which brute-force is of real use, however, as explained later, there is a snag to this. Othello will make use of the classical approaches to carry out its process of playing. It will traverse the game tree (defined later) and inspect various properties systematically. There is no real thinking involved and the only rational decisions made are in the mind of the programmer.

Chapter 2

Related Work

The birth of artificial intelligence had spawned research to enable computers to adopt a methodology of thinking that simulates the human brain. But success was brought about through various other techniques such as alphabeta search, which do not mimic the human thought process. Such techniques, coupled with the massive computing power of a commodity system, have proven to be far superior to the human brain in certain situations. The use of Othello to demonstrate the game playing capabilities of machines is not new. Othellos rigorous constraints and rules severely constrain the next set of possible moves, hence implying a very low branching factor, making it practically feasible for a computer search. The most common approach to playing Othello and other related games, is to perform searches on game trees using the alpha-beta search.Computer Othello players, have comprehensively beaten human world champions, leading to a greater interest in this game to stay ahead of the human community of Othello players. Improved game play in Othello can result from (i) improvements in the search strategy used, or (ii) from better heuristic functions, or (iii) by using learning methods to enable the computer to learn for itself from massive amounts of data, or (iv) by adding extra hardware support. Coin-flipping and other Othello related tasks are extremely fast on such machines. Learning has proved to be an efficient component in game playing systems. Our work concentrates on identifying fruitful heuristic components and analyzing their behavior. To aid us in this process of analysis, various experiments were conducted.

Chapter 3

Background Theory

3.1 Othello

The game Othello is considered to be a game that requires comprehensive experience to be mastered. It is also said that Othello "takes a minute to learn and a lifetime to master". What makes it particularly interesting is the difficulty in telling which player is in a point of advantage until the late stages of the game. This is, in principle, where lack of skill and experience take their toll. The following is a brief summary of the rules: At the beginning of the game, four stones are already placed at the centre of an 8x8 standard game board. Two stones of each one of the players are placed diagonally and, by convention, the player to make the first move is of white color, whereas the other is of black color. The stones are all two-sided and flipping them changes their color to the opponent's color. A legal move is such that it reverses one or more stones of the opponent's color. To reverse a stone, a player places one of his/her stones in such a way so that it surrounds a sequence of one or more of the opponent's stones. Such a sequence of opponent's stones must be ending in a board slot that is occupied by the reversing player's stone. All straight lines are applicable in such a reversal: horizontal, vertical or diagonal. If no reversal is possible, the turn is passed to the opponent. The game is finished when neither player has any legal moves left. Usually, by this point the board is completely full. Whoever has the most stones placed on the board at that point wins the game.

3.2 Game Trees

A game where decisions are made by a set of players can be described as a tree. At the start, there is one single state which is the opening state; no choice has yet been made by this point. That state corresponds to the initial board layout and in the case of Othello, the state is one where 4 stones are placed in the centre of the board. As the game progresses, a set of players can pull the state of the board in different directions, meaning that the state of the board at any point will depend on which paths in the tree have been picked up by the players in the game. In most board games, any state of the board is changed upon a placement or displacement of a game piece. A node in the tree, depicted below as a circle, presents one distinct board state that can be reached in one distinct way from the root. In other words, only one path can lead to a given node, hence forming a tree and not a graph. The nodes are the decision point in the game and combined nodes, connected by the relevant edges, form a path. The game is played until a certain depth is reached. In the game of Othello that depth is not a fixed number. As explained before, most large games such as Othello are too complex to be fully explored and hence solved. With respect to game trees, what this means is that the game tree of the game Othello is too large to be fully traversed.

3.3 Strategies

A strategy defines a choice of a path in the game tree. It can be seen as the tendency adopted by one player, e.g. choosing to stand when having a sum of 20 in Blackjack. The focus of game-playing programs development is the discovery of good strategies.

3.4 Evaluation Function

For the time being, let us just think of this as any function that takes a state of a game as an input and returns some value that has been assigned to it. In order to look ahead in the game, one needs to choose an algorithm that will account for the usefulness of a move from the perspective of each player in the game. Another way of expressing this notion of usefulness is to talk about the state evaluation of a move. A simple example would be to describe a winning move as one that carries a high value. In order to find out what the usefulness is, we define an evaluation function which, given a state in the game, will return some information regarding that state. See the section named Game Engine to get familiar with the evaluation function that is used in Othello. The ideal would be to use a very powerful computer. In such a case evaluation functions are not necessary. Instead, searching the trees to leaf level reveals who can force a win and by which positions (or strategies) this can be achieved. In some games, no such strategies exist, e.g. Paper-Stone-Scissors. Since in most games that we are concerned with searching to this level is infeasible, limiting the search depth is a reasonable solution. The evaluation function then provides a value for a position instead.

3.5 The Utilities (minimax) algorithm

The following explanation is phrased in simple and general terms that are intended for readers with a fair knowledge in programming, but none in the field of Game Theory. To gain better knowledge that is expressed in mathematical and technical terms, please consult the corresponding references. The principles described here and onwards are strictly concerned with games where 2 players are involved and the information is complete . This means that the game position is always clearly known to both players. Also, due to the existence of only 2 players, from one player's point of view, any move made by his/her opponent has a direct influence on that one player. In a game tree, we can

incorporate the results obtained from the evaluation function. In other words, we can assign to each node of the tree the evaluation's returned value once it has been calculated. Since these game trees are usually enormous in size, the traversal to is limited to a certain depth. Knowing that each of the players wishes to maximize his/her pay-off at any given opportunity, we can predict the path that will be followed down the tree. Given these concepts, we finally see why the path picked is sensible. We choose the maximum, minimum, again maximum and so forth at each level of the tree. The name minimax has been extracted from this very intriguing behavior. For a step-by-step illustration of the minimax algorithms, see Appendix H. A substantial step towards our goal has been taken once the minimax algorithm has been realised. Various intricate extensions make it even more efficient and sophisticated, but the only one worth mentioning is alpha-beta pruning which helps minimax save computational effort.

3.6 Alpha-beta pruning

Alpha-beta pruning relies on the algorithm presented above. It is an extension that allows the process carried out by the minimax algorithms to be carried more wisely and explore more important paths in the game tree. It is based on the idea that branches in the game tree should no longer be explored if they offer a solution that is no better than what has already been found. The pruning of the tree means that we can leave out parts of the tree without needing to explore them. They simply do not offer any better choices than the ones already discovered. In order to allow for pruning to take place, the values of choices already explored need to be recorded as a range between two numbers. Alpha and beta were the original names for the variables holding these values and defining that range. The range indicates which values are still sought and it is modified as the tree is traversed. Making good traversal order allows for more pruning, but is a very difficult task. See the references for more details on the issues and implementation of alpha-beta

pruning.

3.7 Game Value

The game value indicates what the pay-off for each player is when the game is orchestrated by optimal strategies. It can indicate if one player can ensure at most a win, a draw or a loss. Within the population of large games, we usually do not know which of these categories a game falls into because the game tree is extraordinarily wide. Othello is one of these games.

3.8 Pseudo-code

The following is an excerpt from the program's pseudo-code. Many such algorithmic pieces could be included, but it is the principle that counts and not the quantity. Let us look at a pseudo-code example that is meant to generate a valid random move in the game. The following assumes some other function have been or will be provided 1. Reset placement status

2. While the placement is unset do the following:
1. Scan each row in the game board
1. Scan each board slot in the row
1. If the slot is occupied, do nothing
2. Otherwise, check validity of placement

If the placement is valid

Perform it

Set placement status

Otherwise, do nothing

3. Do the appropriate flips on the board
4. Check if a deadlock has occurred

Chapter 4

Conclusion

Game playing has always been one of the most attractive fields of artificial intelligence research, and it will continue to be so. It is one of the parts of artificial intelligence that the common man observes and interacts with. We evaluated the importance of a few heuristics in enhancing Othello game play. This paper tried to explain the interaction amongst the various heuristics that are utilized to evaluate the state of an Othello game. We also analyzed their importance. We found that increasing the accuracy of the stability heuristic would enhance game play greatly. It was also interesting to note that though corners was the most powerful standalone heuristic, stability played a major role in the component-wise heuristic function. This study would allow one to identify the most important aspects, and enable more processor power to be thrown into it in order to increase the accuracy of the heuristic. Future work would include the incorporation of learning strategies into the system. Such strategies tend to be very powerful, since they could make use of vast amounts of already existing data and can avoid pitfalls that deterministic algorithms can suffer from. We can also build more upon our framework to give rise to an extremely good Othello player, that can then go on to participate in tournaments. That would require careful weight modification, optimized look up tables, and powerful learning strategies.

4.1 References

http://www.danielsorogon.com/Webmaster/Projects/OM.html

http://home.nc.rr.com/othello/rules/

http://www.mattelothello.com/purpleindex.html.

http://www.dcc.unicamp.br/stolfi/EXPORT/images/textures/ppm-400x400/

http://crow.cs.und.ac.za/angus/GameProgramming/OglTextures.html

http://www.nada.kth.se/gunnar/howto.html

http://home.tiscalinet.ch/twolf/tw/misc/reversi/html/index.html

http://vcg.iei.pi.cnr.it/swform.html

http://www.codesampler.com/oglsrc.htm

http://www1.ics.uci.edu/eppstein/180a/w99.html

http://www.maths.nott.ac.uk/personal/anw/G13GAM/