CSL 302 – Artificial Intelligence

Project 2

This project is divided into three phases to help you complete project. The final score for the project is divided as follows Phase 1- 20%, Phase 2- 40%, Phase 3- 40%. The deadlines for the three phases are February 28, 2014, March 7, 2014 and March 14, 2014 respectively. At the end of each phase of the project submit the updated code and a pdf document containing a sample run of the game and other observations according to the submission instructions. You are allowed to work in pairs for this project. Late submission is not allowed without the prior approval of instructor. You are expected to follow the honor code for the course while working on this homework.

Submission Instructions:

Email one zip file containing the following to ckn@iitrpr.ac.in.

1. Neatly formatted PDF document with your answers for each of the questions in the homework. You can use latex/MS Word or any other software to create the PDF.
2. For homework/projects that contain problems that require programming, include a separate folder named as "code" containing the scripts for the homework along with the necessary data files. Name the scripts using the problem number. Include a README file explaining how to execute the scripts.
3. Name the ZIP file using the following convention:
   rollnumber1_rollnumber2_p(number).zip.
4. Please **do not** submit a hardcopy of your solutions.

In this project you will be implementing a popular game called 'othello' or 'reversi'. Check the wiki article (http://en.wikipedia.org/wiki/Reversi) to learn rules and other details about the game. If you would like to practice and get familiar with the game, here is an online version (http://www.othelloonline.org/) of it.

You can implement the game in C, C++, Java or Python or any other language in which you are proficient. However, the program needs to be able to be compiled and run on Linux machines. **Do not download and plagiarize code from the Internet.**

Phase 1: Due on February 28, 2014

Implement a simple console based Othello game for a human to play against a computer. The implementation should have the following functionalities

- Display the board configuration

- The location of the disc to be placed during a human player's turn should be taken as input from the player. A simple way to do this would be to request the player the column and row number of the square in which the disc has to be placed at the command prompt.
- Check if a move is valid
- Update the board after making a move
- The current score for the human player and computer
- Make the computer play a random move among the set of valid moves.

Phase 2: Due on March 7, 2014

- Implement *minimax* algorithm to make the computer more intelligent.
- It is impractical to expect the optimal brute-force search till the terminal state to function in real-time. Hence implement the following evaluation functions for early search termination
    - Disc count – the difference in the number of each player's discs.
    - Weighted Disc count – weighted difference in the number of each player's discs. Discs placed in the corners and sides of the board have higher weights.
- Combine these two evaluation functions with different weights ($\in [0,1]$) for experimentation.
- Experiment with different ply levels (as deep as possible) to see the effect of searching deeper with the best combination of these evaluation functions on the final outcome (who wins and what is the difference in the score) of the game.

Phase 3: Due on March 14, 2014

- Implement the alpha-beta pruning algorithm.
- Add functionality to store the number of nodes generated, explored and pruned.
- Experiment alpha-beta pruning with different cutoff depths and tabulate the resulting changes on the statistics of the nodes.
- Since you are anyway checking if a move is valid, add another component to the evaluation function that computes the number of valid moves for the player.
- Experiment with different weights for the three components of the evaluation function and check its effect on the outcome of the game for varying ply levels (cut off depths).

Bonus

Add/modify any other evaluation function for improving the search. Experiment with the new function/modification and show how it affects the search process and the outcome of the game.