Connect Four AI Game Player

Ming Wu

ID: 992474666
CDF Login: c9wuming

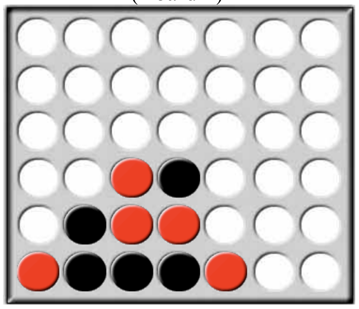
Abstract

A Java applet program, extended from Sean Bridges' original work, is written. It's to demo and evaluate the ideas discussed in this paper, and to verify the approaches are correct.
Using the applet program, it can host a game played by 2 AI players, with same or different AI algorithms. It also allows a human player to play against AI player.
In combination of game tree search, minimax algorithm, alpha beta pruning and other play rules defined, the AI player can play competitive games against above average skilled human players.

Introduction

Connect Four is a two player board game. The regular game board is 7x6 grid. There are total 42 slots on the board, and each player has 21 colored disks to use. The applet game assumes one player has black color disks, and another player has red color disks. Player with black color disks always start first. Player will drop a disk from the top into the column and occupying the next available slot. The goal of the game is to have four disks of your color to connect each other, vertically, horizontally or diagonally before your opponent does. If all disks have been played and neither player has archived the goal, then the game is a draw.

(Board 1)



For each row, from bottom to up, I name them 1 to 6.
For each column, from left to right, I name them a ~ g.
For example, the bottom left slot is (a, 1), the top right slot is (g, 6)

Approaches

My approaches are to deploy a set of algorithms to implement an intelligent player. Algorithms and techniques used include, game tree search, minimax algorithm, alpha beta pruning and some strategic play rules specific to connect four.

Game Search Tree

There are total 42 slots on the game board, and there are 69 possible four-in-a-row winning combinations includes 24 combinations horizontally, 21 combinations vertically, 12 combinations diagonally with slope 1 and 12 combinations diagonally with slope -1.

It is impossible to search the game tree from initial state to a winning state due to the massive search space. As a result, we have to decide on a cut off depth. From my research, with out alpha pruning the maximum depth can search with a moderate computer are only 3 to 4 levels.

Heuristic Evaluation Functions

At any moment of the game, we evaluate the current player's strength by looking at how close is it to archive a four-in-a-row at any of the 69 possible combinations.

(Formula 1)

*Strength = Number of 3 In a Row * 32 +*
*Number of 2 In a Row * 4 +*
Number of 1 In a Row

Above formula is a naïve version of the final strength formular. "3 in a row" means there are 3 same color disks in a row. It is also called a threat. There must be one or two empty slots at the end of the current row. For example the board image on the left, there is no “3 in a row’ black threat. There is a “2 in a row” for black. There is a “3 in a row” red threat.

For board 1, the strength of black is 11

Strength (black) = 0 * 32 + 1 * 4 + 7 = 11

For board 1, the strength of red is

Strength (red) = 1 * 32 + 2 * 4 + 4 = 44

Clearly, the current board state is in favor of red player.

Minimax Algorithm

The AI game player will make move according to minimax search result. At AI player’s turn, every possible move will be evaluated and subsequent opponent’s move will be evaluated as well. The current move score is determined by following formula.

(Formula 2)

Score = Strength (Player A) – Strength (Player B)

Depends on the search tree level, min nodes or max node, both player can either be red or black.

Alpha Beta Pruning

One effective solution to save search time and improve the game tree search is pruning. Alpha beta pruning is a basic pruning technique, but it is very effective and minimax values are kept accurate. With alpha beta pruning, in my applet game, it can search as deep as 8 levels on a moderate computer. Keep in mind that this demo program is in Java applet program running in browser, which is slow in nature. If the program is implemented in C, it possibly can search the tree even 1 or 2 levels deeper.

Other Pruning Algorithms

I have also looked at killer heuristic, which is a popular pruning algorithm used in chess game programs to improve alpha beta pruning. In this project, I did not have chance to evaluate its effectiveness. Because killer heuristic will cut off more nodes while search the game tree, I estimate the search depth can go as deep as 10 or 11 with killer heuristic used.

Other Strategies

In 1988, Victor Allis solved connect 4, and he wrote a program to play the game perfectly based on 9 rules. My interest here is not really to re-implement those 9 rules to make a perfect game player. But I can borrow some ideas that he presented in his master thesis paper. Some of those ideas are also commonly used among many other connect 4 expert players.

Start Game

Start game in connect 4 is very important, some threats can be created at the very beginning of the game, and give the player control of the game until the end. A good threat can even guarantee a win at very early stage of the game.

End Game

End game in connect 4 is easy to play, although it’s very difficult to establish a good threat near the end of the game. For player already have a good threat, the end game play involves keep in control of the game; for player does not have a good threat, the end game play will tries to make a tie.

Even / Odd Threats

At the same time, not all threats are the same. Some threats are useless because they can be defeated easily. Some threats are undefeatable, thus it guarantees a win.

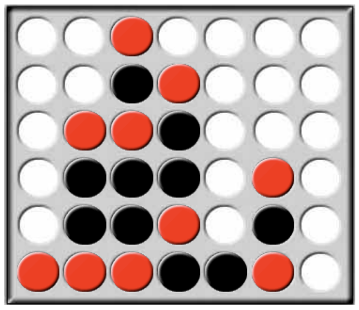
Based on analysis of the steps. It’s proven that first player (black) should try to construct odd threats. And second player (red) should try to construct even threats. As a result, following is the revised strength formula.

(Formula 3)

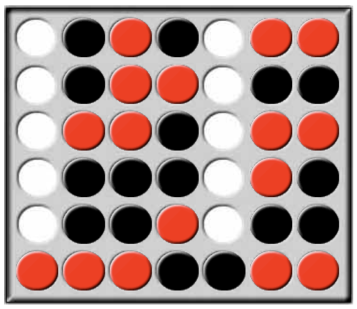
*Strength (black) = Number of 3 In a Row Odd * 32 +*
*Number of 3 In a Row Even * 16 +*
*Number of 2 In a Row * 4 +*
Number of 1 In a Row

*Strength (red) = Number of 3 In a Row Even * 32 +*
*Number of 3 In a Row Odd * 16 +*
*Number of 2 In a Row * 4 +*
Number of 1 In a Row

Examples of Black Player Odd Threat

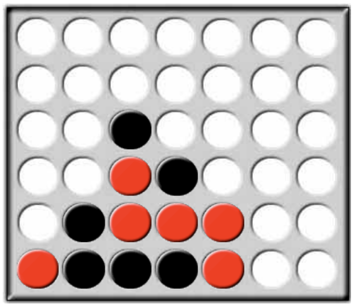


Above black has a 3 in a row threat at (a, 3) and (e, 3). This threat is an even thread, as the threat is at the second empty slot from the same column count from bottom. This Threat can be easily defeated by red. Black player has another 3 in a row threat at (a, 4), which is an odd threat. This threat is created at the middle stage of the game, but it gives black an advantage to win. As you can see, black player will win from the follow board at a later time of the same game.

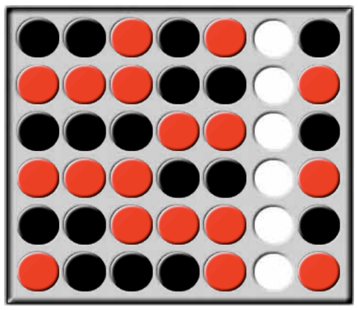


Above board is black to move, (a, 3) and (e, 3) will be refuted by red easily. Red will have to play (a, 3) and black will win at (a, 4)

Example of Red Player Even Threat



On above board, red has multiple 3 in a row threats at (b, 4) and (f, 2). Both threads are even threads. AI player (red) constructs those 2 threads at early stage of the game, and will have control of the game until win if no mistake made. Following is the board after a few more moves.



Above board is black to move. Black has no choice to play (f, 1), and red will win at (f2).

Other Special Case

If one player's threat is above another player's threat, there are 2 possibilities. One possibility, player will refute each other's threat. This kind of threats is useless threat. The other possibility is the lower threat will win.
Another strategy is play follow up at near end of the game. After a player establishes a good threat or a play want to have a tie game, the player can play follow up to keep the game situation. Follow up means to place the disk on top of another player's disk just played. In other words, play the same column that another player just played.
Those special cases are not implemented in the applet demo program due to time limitation.

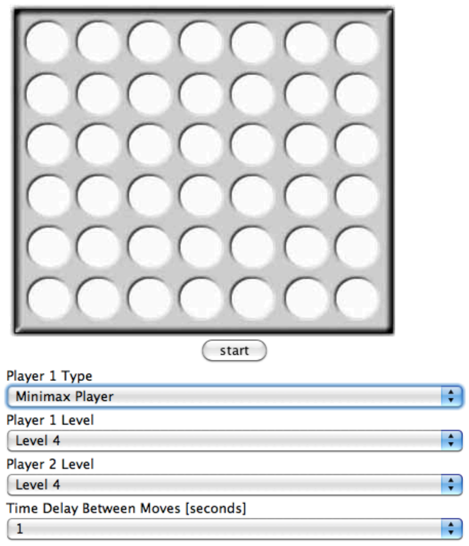
Evaluation

The java applet game program is implemented to evaluate the AI player. There are a few different modes for the game. Basic AI Player is based on minimax algorithm only.

• AI Player vs. AI Player

• AI Player (odd/even check) vs. AI Player

• Human vs. AI Player



Player 1 Type
Minimax Player
Player 1 Level
Level 4
Player 2 Level
Level 4
Time Delay Between Moves [seconds]
1

AI Player vs. AI Player

During evaluation plays, player with deeper search depth will defeat player with less deep search depth.

AI Player vs. AI Player Odd/Even Checking

Most the game comes out with draw. Maybe a result of playing follow ups.

Human vs. AI Player

AI player is hard to defeat. AI player has above average player skill level for sure. It is very difficult to find some one who plays connect four at expert level. As a result most human vs. AI player testing is done by myself. On average, AI player win 70% of the time.

Conclusion

This project gives me an opportunity to experiment and apply the knowledge learn in classroom to a real project. Along with the project, I also explored some other advanced algorithms that were not covered in class, which are very interesting and helped me to further enhance my understanding with artificial intelligence in general. There are things in our daily life that can be improved by taking advantage of the AI technologies.

References

Allis, Victor 1988. *A Knowledge Based Approach of Connect-Four*.Master Thesis,Amsterdam: Vrije University.

Bridges, Sean *Java Applet Program*
http://www.geocities.com/ResearchTriangle/System/3517/C4/C4Conv.html

James D. Allen *Expert play in Connect-Four*
http://homepages.cwi.nl/~tromp/c4.html