// CS570 Project 2

// deepwaterooo Mar 5, 2013

Connect 4 Human-AI Game Searching using MiniMAX Alpha-Beta pruning algorithm

Abstract

In this project, a Connect 4 game program was designed by using minimax algorithm with Alpha-Beta pruning. The minimax algorithm works by two recursion functions mutually iteratively calling the other returning utility values, and the root function returns the best move. An evaluation heuristic function based on the board win opportunities and win board configurations for the computer and for the player was used to find the best move for artifical intelligence and to avoid long time searching. The C++ programming language was used. It was found that this algorithm is able to make a winning move if one is available, and able to block the win of the opponent. It can make very reasonable move especially when increasing recursive level in the minimax algorithm. The response is fast when the recursive level is low and the response time increases when the recursive level increases. Apha-beta pruning technique significantly decreased the searching time.And the evaluation heuristic function is critical to make smart moves.

Introduction

Game is a special search problem compared with path-finding because there are more than one agents in the environment and agents are in a competitive relationship. The goals of the agents are in conflict, and agents have to consider the actions of other agents. Therefore in artificial intelligence, games are known as adversarial search. Game problem is difficult to solve because of the extraordinarily high time and space complexity for most games. In reality, the game search algorithms have to make some decision based on evaluation heuristic function and levels of depth to search in order to avoid long time computing.

Connect4 is a two-player game using a seven columns by six rows rack. The goal of the game is to connect four tokens with the same color horizontally, vertically or diagonally. Compared with other games, connect 4 is a relatively simple search problem since it has a small branching factor 7. While even with a branching factor of 7, the full search tree is still huge and searching the entire tree is especially time/space consuming and not feasible. Accordingly, instead of using an original minimax algorithm searching full tree, which searches all the way down to terminal states, I used an evaluation heuristic function and Alpha-Beta pruning for minimax algorithm to implement the game design for this project.

Algorithms

(1) Data structures

In this project, all algorithms were implemented using C++ programming language. Since Connect 4 has a fix board size (6 rows by 7 columns), static 2-dementional arrays with a size of 6x7 would be good enough to present board/states. Two static 2-dementional arrays with fixed size of 6x7 were created. One was used to store the board information (states) and the other was to store information which was used for evaluation function.

(2) Search algorithm

A alpha-Beta pruning minimax algorithm with an evaluation heuristic function was used. The Alpha-Beta pruning minimax algorithm is the same as the one in the text book. The searching details pseudo-codes were shown followed. The evaluation heuristic function was introduced in the next section.

function Alpha\_Beta\_Search (state) returns an action (a column number)

v = Max\_Value(state, -65535, 65535)

/\* there are 7 possible actions, which are dropping at one of the 7 columns \*/

return the action (a column number) in ACTIONS(state) with value v

function Max\_Value(state, α, β) returns a utility value

if terminal-test(state) then return Heuristic(state)

v = -65535

for each a in ACTIONS(state) do

v = MAX(v, Min\_Value(RESULT(s, a), α, β))

if v>= β then return v

α = MAX(α, v)

return v

function Min\_Value(state, α, β) returns a utility value

if Recursive-level = depth then return Heuristic(state)

v = 65535

for each a in ACTIONS(state) do

v = MIN(v, Max\_Value(RESULT(s, a), α, β))

if v<= α then return v

β = Min(α, v)

return v

The recursive levels is one important deterministic factor for the computer player to make intelligence moves. The recursive level in the algorithm decides when to check for the utility of a state (for the computer player). Though in game parlance, one move deep consists of two half-moves, in this project, I controlled the recursive levels by counting the half-moves (counting the moves including pieces dropped both by the player and the computer) on the board. For example, if we want recursive level to be 4, we count the number of pieces currently on the board and let computer and player play until we have 4 more pieces on the board. Then the utility values are calculated and the backed-up values are determined accordingly. The more recursion levels we give the algorithm, the more it can predict the best play in the future. Actually we can make the difficult level of this Connect4 game by selecting different recursion level.

(3) Evaluation function

The evaluation function is the most critical factor in this game for the computer to make smart moves, and affecting time complexity and space complexity as well. In this project, the utility value was calculated as a score for the board position based on opportunities for both the computer and the player. The function is called int Heuristic(char A[6][7]) in the program (see attached codes for more details). An opportunity is an empty place (a hole) that has around it, a row of 3 pieces of the same color (horizontal, vertical and diagonal). If around it 2 pieces, then it takes less weight (3-in-a-row weights 40, and 2-in-a-row weights 10 only). Each position on the board was examined by scanning the 2-dimensional array. At the end, the board position score (state) is the sum of all the scores of the player minus the scores of the opponent player. This evaluation function was originally developed by Bishoy Labib.

Results

Once the program was executed, we can see the a board array with the size of 6x7. And human player can decide if he wants to play first. If the human player doesn’t want to play first, the computer will make the first move.

Because of the fairely good evaluation function I used in the program, I don’t need to write additional statements to take care of blocking a win and taking a win. The evaluation function has already taken all those into consideration. Based on the utility values given by the evaluation function, the computer makes moves that can take a win or block a win of the player. If these two moves are available at the same time, the computer will take the win.

Discussion

The original minimax algorithm, which searches the entire game tree, is not practical although it does provide optimal decisions in Connect 4 game. Alpha-Beta pruning used in this project can eliminate large parts of the tree from consideration without making losing optimal solutions. However, Alpha-Beta pruning still has to search all the way to terminal states for certain parts of the search tree. The time and space complexities are still high for this algorithm. Consequently, an evaluation heuristic function was used to make the decisions when it is appropriate. The quality of the evaluation heuristic function is very important. At first I used a simple evaluation function that counts the number of piece chains in which the pieces have the same color. The number of pieces in chain could be 1, 2 and 3, and they have the different weights. A chain that has 3 pieces in the same color has higher weight than the chain that has 1 or 2 pieces in the same color. It turned out this evaluation function worked but not well. It blocked or took a win sometimes but not all the time. Then I used a complex evaluation function developed by Labib, B. The algorithm that uses this evaluation function made reasonable moves all the time, and actually I can hardly beat the computer.

Conclusions

Alpha-Beta pruning minimax algorithm with an evaluation function was used to design a Connect 4 game in this project. The algorithm works efficiently and the response time is short with a recursion level of 3-7. The evaluation function is critical and decides if the computer makes reasonable moves.

References

1. Russell, S.J. and P. Norvig, Artificial intelligence: a modern approach. 2010: Prentice Hall.

2. Labib, B. [cited; Available from: http://bishoylabib.blogspot.com., http://www.codeproject.com/articles/35375/connect4.