**CS570 Artificial Intelligence**

**Spring 2012**

**Project 2**

**March 15, 2012**

**Sanqing Yuan**

**Connect4 designed by using MiniMAX algorithm**

**Abstract**

In this project, a Connect4 game program was designed by using minimax algorithm with Alpha-Beta pruning. The minimax algorithm works by a recursion function with a limited recursive level (3-5). An evaluation heuristic function based on the opportunities for the computer and for the player was used to avoid long time searching. The C++ programming language was used. It was found that this algorithm is able to block the win of the player and make a winning move if one is available. It actually can make very reasonable move especially when increasing recursive level to 5 in minimax algorithm. The response is fast when the recursive level is 3 and the response time increases when the recursive level increases. The evaluation heuristic function is critical to make smart moves.

**Introduction**

Game is a special search problem 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 [1]. Therefore in artificial intelligence, games are known as adversarial search. Game problem is difficult to solve because of the extremely high time and space complexity in most games. In reality, the game search algorithms have to make some decision based on evaluation heuristic function 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. Connect4 is a relatively simple search problem since it has relatively smaller branching factor (7) compared to other games. Even so, the search tree is still very large and searching the entire tree is not feasible. Accordingly, instead of using an original minimax algorithm, which moves all the way down to terminal states, I used an evaluation heuristic function and Alpha-Beta pruning for minimax algorithm in this project.

**Algorithms**

(1) Data structures

In this project, all algorithms were implemented in C++. Since Connect4 has a fix board size (6x7), static 2-dementional arrays with a size of 6x7 would be good enough to store states. Two 2-dementional arrays with 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. Because the program displays the board after each move, we don’t need the data structure like linked list used in project 1 to store “move path”.

(2) Search algorithm

In this project, 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 [1]. The details were shown in figure 1. The evaluation heuristic function was introduced in 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** Recursive-level = *depth* **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*

**Figure 1** Pseudocode of the Alpha-Beta pruning search algorithm with an evaluation heuristic function

How to control recursive levels is important and difficult in this algorithm. The recursive level in the algorithm of figure 1 decides when to check for the utility of a state (for computer). In this project, I controlled the recursive levels by counting the pieces (including pieces dropped by play and computer) on the board. For example, if we want recursive level to be 3, we count the number of pieces currently on the board and let computer and player play until we have 3 more pieces on the board. Then the utility values are calculated and the backed-up values are determined accordingly. The more we give the algorithm more recursion levels, 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. For example, we can make the easy level has 3 levels of recursion and the hard level has 5 levels of recursion.

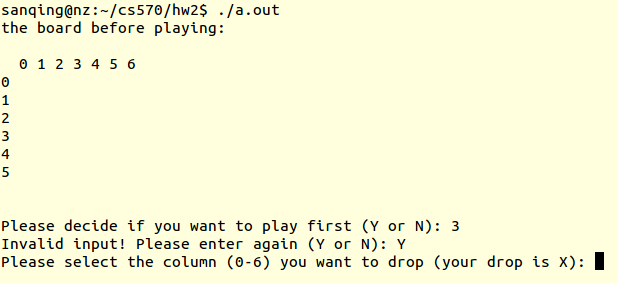
(3) Evaluation function

The evaluation function is critical in this game because it not only avoids long time computing but also has a significant impact on if the computer makes smart moves. 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 arr[6][7]) in my program (see attached codes for more details). An opportunity is an empty square (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. Each position on the board was examined by scanning the all 2-dimensional array. At the end, the board position (state) score is the sum of all the scores of the computer minus the scores of the player. This evaluation function was developed by Bishoy Labib [2].

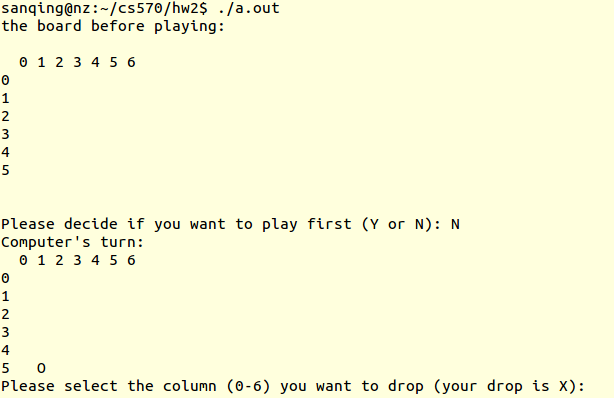
**Results**

As shown in Figure 2, once the program was executed, we can see the a board with the size of 6x7. More importantly, we can decide who play first. If you don’t want to play first, the computer will make the first move (Figure 3).

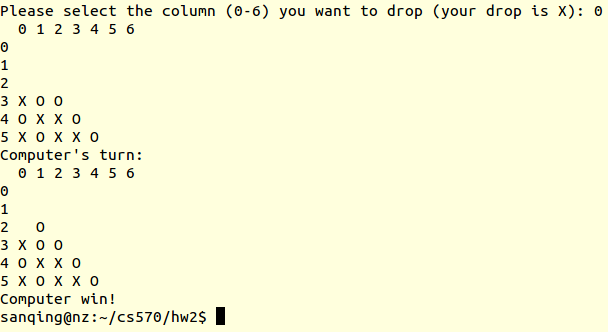
Because of the excellent evaluation function I used in the program, I don’t need to write additional statements to take care of blocking a win and taking a win. The evaluation function has already taken all those into account. Based on the utility values given by the evaluation function, the computer 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 (Figure 4 – Figure 5).



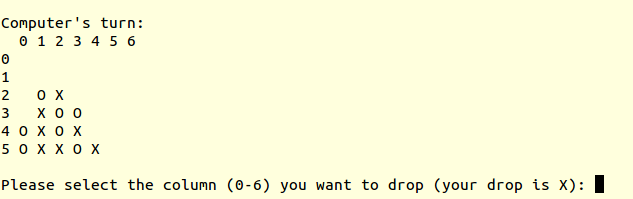
**Figure 2** Initialization of Connect4 game.



**Figure 3** Computer makes the first move



**Figure 4** Computer takes a win



**Figure 5** Computer blocks a win

**Discussion**

The original minimax algorithm, which searches the entire game tree, is not practical although it can provide optimal decisions in Connect4 game. Alpha-Beta pruning used in this project can eliminate large parts of the tree from consideration without making wrong decisions. However, Alpha-Beta pruning still has to search all the way to terminal states for at least part of the search tree. The time and space complexities are still quite high. 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 [2]. 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 Connect4 game in this project. The algorithm works efficiently and the response time is short with a recursion level of 3-5. 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://bishoylabib.blogspot.com/).