Connect 4 Theory

By Cal Henderson

NOTE: Connect 4 as a mathematical problem was solved in Victor Allis's Masters Thesis, October 1988.

A while ago I wrote a playable online version of Connect 4, here: http://www.pixelflo.com/009/

It was written in javascript. i'm a bit of a Connect 4 obsessive, and almost did my degree on game theory software.

The gameplay engine can be found here: http://www.pixelflo.com/009/engine4.js

A simple explanation of how my algorithm works:
Start with a set of all possible winning lines - each as an object with four co-ordiate pairs For each column, take the set of winning lines and: Knock out any winning lines that already have any pieces of the opposite color in Knock out any winning lines that don't include the current move in the current column The number of winning lines that remain is the base 'score' for that column (if one of the winning lines would be completed by this move, it gets a score of 999999 to ensure that move is taken) For each move, calculate the 'scores' the opponent would have on their next move, and take a fraction of this score away from the column's score (again, a winning move gets a score of 999999, which is then taken away, in fraction, from the row score to avoid making that move) Continue reading ahead for as long as you have the computing power: in my case that's just two turns because javascript is pretty slow. in the c and pascal versions it reads ahead 4-5 moves. you can edit line 18 of the javascript file to tell it to think more. This algorithm works on the princial that it should maximise the number of possible winning lines that each move could possibly contribute to.

An algorithm i used to use alot was as follows: Repeat for each possible move: Examine how many lines of 2,3 or 4 the move would contribute to Each of these is given a weight which is then added to the 'score' for that move Next, examine how many opponent lines of 2,3 or 4 would be blocked by each move The algorithm can either be offensive or defensive. for offensive play, your own lines score higher than blocking the opponents lines causing the algorithm to choose making lines over blocking lines. This only applies to lines of 2 and 3: lines of 4 are special cases: A line of 4 that can be made always has priority If a line of 4 can be blocked and no line of 4 can be made, the block is always made This whole process can then, again, read several moves ahead to ensure sensible play. the number of read aheads is based on processor power.

A third approach which i've discussed but never implemented work on the finite principals of the game. since this isn't chess, the possible number of games (once you cut out symetry and first turns) isn't too huge. so you could (probably) calculate the right move to make in any situation. it isn't as many as you think either, since many moves end the game, and many board situations aren't possible - we're talking about alot less than 3^(6*7) (which is 26,588,814,358,957,503,287,787).

Even if you only caclulated a slice of this data, it mighthelp to establish some unbeatable heuristics.

Finally, a method which i implemented as an a-level project is to make the game learn as it plays. this is very simple to do, and requires almost no thought: Take a move at random Record game sequences in which we lost Record game sequences in which we won If we are found to be following a path which has led to a loss before, decrease the chance of making that move. the amount to which we decrease the probability is dependant on how close we are to the final move in the losing sequence. If we are following a path which has led to a win before, positiverly weight probabilities as above. The downside to this approach is that it takes ages to get it to be any good

at all. there are two things i've tried to fix this: Combine it with one of the algorithm above. this also helps fill in any weaknesses of the other algorithms, but stopping them from always falling into the same traps. Force it to play against a random computer player. since neither player does much calculation, this can help make the 'intelligent' player very good, in a very short space of time.

This is just a taster - I have loads of ideas about programming for Connect 4, so if you have any specific questions then just drop me an email.

(end of article)