AI 2015 Semester 1 Assignment 2
Christoph S. Herrmann
S3505919

Connect 4 Heuristics Function

This document discusses the basic idea of a board state evaluation algorithm for a connect-4 game. For this document, it is assumed that the winning number of connected tokens required to win is 4. This can, however, be easily extrapolated to any number N.

Basic Idea

Any given board state contains a number of theoretically possible lines that can be used to win a game. Each line could be horizontal, vertical, diagonal and contains 4 slots. The orientation information, however, is not important for the evaluation of the board state for this heuristic.

Every line can thus be described by the following equation:

$$4 = slotsUsedByPlayer1 + slotsUsedByPlayer2 + emptySlots$$

Furthermore, the following assumptions are the base of the heuristic:

1. A win for player 1 is a line where 4 = *slotsUsedByPlayer1*
2. A line where *slotsUsedByPlayer1>0* and *slotsUsedByPlayer2>0* can not be used to win for either player
3. A line with *slotsUsedByPlayer1 = 3* and *slotsUsedByPlayer2 = 0* is very dangerous to player 2
4. A line with *slotsUsedByPlayer1 = 3* and *slotsUsedByPlayer2 = 0* is very beneficial to player 1

The heuristic function

Basic algorithm

The algorithm iterates over all lines possible for the current board. For every line, a danger value and a good value is calculated independently. Both values can be on a scale between 0 and 100 (details on how to obtain those values are in the next chapters). After every iteration, these values are added to the overall danger and goodness of the board.

Once every line was evaluated this way, the larger value of the overall goodness vs. danger is used to determine whether or not the board is good or dangerous. The respective value is then divided by the linecount * 100 to create a value between 0 and 1. For dangerous values, the whole number is multiplied by -1.

Dangerous Factor Calculation

As described before, a line is potentially dangerous for a player A, if there are only tokens of player B in that particular line. The heuristic function uses a threshold to start counting a line

as dangerous. In its current settings, this threshold is 30% of all $\frac{Tokens\ by\ B}{Number\ to\ connect}$. This means, for a connect 4 game all lines with more than 2 of the opponent's tokens and none of the players tokens, a danger factor is calculated.

The danger factor itself is a the line occupation percentage multiplied by the highest danger factor (roughly 100). Since in a 4 in a row game, 4 out of 4 is a loose, the highest danger factor needs to be scored for 3 out of 4 tokens. The calculation is this:

maxAllowPerc = (NumberToConnect - 1)/NumberToConnect
percWon = otherPlayersToken/NumberToConnect
*dangerous = percWon * (DANGER_FACTOR/maxAllowablePerc)*

For example, 3 out of 4 tokens by the opponent gives a danger of *DANGER_FACTOR* as a danger score. This calculation had to be convoluted like that to serve games of the N in a row.

Goodness Factor Calculation

Goodness, similarly to danger is calculated on lines where the player possesses tokens. There are two kind of goodness

Blocking is good

Any line with >0 tokens for both players is considered good, because the opponent has been blocked from this winning opportunity. It is even better if the line is extremely balanced with both players filling up the together. This forces the game to be played in the same areas. In the implementation, the general blocking is rewarded 11 and the full line block is awarded 6 points out of the possible 100 good points per line. These values were discovered via trial and error.

Potential winning lines are great

Of course, in order to win, a line needs to be fully occupied by the player. Therefore, any line where there are only the player's tokens is considered good. Even more so, the more tokens there are, the better the line is. This way, a line can score up to 50 goodness points. 50 would be rewarded if it is 3 out of 4 for the current player.

Again, like with the danger factor, this is adjusted to cater to connect N in a row type games.

goodFac = GOOD_FACTOR/((NumberToConnect - 1)/(NumberToConnect - 1));
*goodness= myTokensInTheLine*myTokensInTheLine*goodFac;*

Danger vs. Goodness per line

As can be seen, the danger for every line can range from 0 to around 100 per line. The goodness, however, can reach around 50 as a maximum. This has evolved by trial and error and seems to work well because defense seems generally more important than offense.

Evaluation of the heuristic function

This section discusses the strengths and weaknesses of the heuristic function described in the previous chapter.

Strengths of the heuristic function

The heuristic function with the parameters described above works well and beats the provided agents reliably even in N-in a row type of games. (N >=4).

The overall idea is rather simple and therefore not as error prone and easily expandable.

Weaknesses of the heuristic function

The heuristic in its current implementation uses a rather defensive strategy. I believe a more offensive evaluation of the board (including additional variables however) could lead to even better results. In the same line of thought, the heuristic has a either good or bad approach, based on whether the goodness or the danger contains a higher value. There is no middle ground which again (based on the danger factors being a lot higher), leads to a more defensive evaluation of the board state.

Additionally, the line concept presented in this heuristic never takes into account if a line is currently an actual threat. For example, in a diagonal line ranging from the left lower corner [1,1] into the middle of the board, it is possible that there are three in a row, however, the pile ([4,1] to [4,3]) to stack the fourth at position [4, 4] has not been build. So at least three more rounds would have to be played for an immediate danger. This is not reflected in the algorithm.

Finally, there are many variables not considered by this heuristic because it didn’t fit in the overall simple concept. Such information includes things such as placement of tokens in the overall board (middle is better than left or right), trap detection and possibly a lot more sophisticated ideas.