Connect-K Project Specification

GENERAL:
You are employed by a company that makes interactive video games. You have been provided with a “dumb” GUI shell and assigned to code the “smarts.” Your product should (1) make each move in no more than five seconds (which should be an adjustable parameter in your code --- never hard-code anything!); and (2) beat you (and the Reader) at Connect-K. You may code the project by yourself, or form a project team of two people maximum (if so, follow the “Pair Programming Paradigm,” and switch roles often; //en.wikipedia.org/wiki/Pair_programming).

- Where S is a game state, you must code a heuristic evaluation function, Eval(S), which estimates how good is the current board configuration S. Eval(S) is usually the dot product of a vector of weights with a vector of feature values, as in the lecture slides.
- Implement Alpha-Beta pruning in your search of the game tree (put it on a switch so that you can turn it on and off, and thereby evaluate how much it helps you).
- Implement Iterative Deepening Search. When the clock runs out you return the solution found at the previous depth limit. (If coding in Java, allow for garbage collection.)
- By remembering the values associated with each node in the game tree at the previous depth limit, you can sort the children at each node of the current iteration so that the best values for each player are (usually) found first, and so alpha-beta pruning will become (almost) optimally efficient.
- It certainly would be foolish to cut off search in a game state where the opponent could win the game in the very next move. The quiescence test checks for this condition. If true, the search is extended one more move, and the quiescence test is applied again.

CODING:
Your code must run on the ICS lab machines. You may develop it on any platform you like. Please start early. All coding takes longer and is more complicated than initially anticipated. It is far better to have code that is simple and working than code that is complicated and unfinished.

Eval(S).

The Possible Win Paths heuristic is discussed in the lecture slides (Game Search; Adversarial Search). A possible win path is any possible extension of the current board state that results in a win for the player. The possible win path evaluation counts the number of possible win paths for each player, then subtracts one from the other.

A better solution is the dot product of a vector of weights with a vector of features (MFEF, Multiple Features Evaluation Function), as discussed in the lecture slides. The Possible Win Paths above would be one feature, and your insight and creativity would provide others. For example, multiple intersecting win paths, win paths near the center of activity, ability to block possible win paths, etc., as limited only by your own creativity in thinking about the problem. Set the weight vector empirically to maximize performance, either by playing you (slow) or by playing itself with different weight vectors (faster).

Quiescence(S)

The quiescence test evaluates whether or not a quiescent position has been reached, and if not, extends the search one more level (recursively). This strategy guards against the “Horizon Effect” (see section 5.4.2). Your quiescence test should at least check to see if the game can be won in the next move. Your creativity may suggest other cases as well.

GRADING:
Your grade will be based partly on your code, partly on its performance, and partly on a written report. A detailed grading rubric appears below:

- 20 Points: Code --- Clean, neat, well-structured, well-commented. The kind of code you would like to see if you ever were assigned to maintain your own legacy code.
- 20 Points: Playing --- Score of zero points if your code does not run on the ICS lab machines. Otherwise 20 points total possible playing the Reader. 10 possible points your program making the first move, 10 possible points the Reader making the first move. For each match, zero points if the Reader beats your program, 5 points if the Reader and your program tie, and 10 points if the Reader loses to your program. (The Reader will play quickly and without deep reflective thought; but not stupidly.)
- 20 Points: Report --- Follow the Report Template. Do not exceed two pages.
- 10 Points: Timing --- Score of zero points if your code does not run on the ICS lab machines, otherwise 10 points if your program always makes a move in 5 seconds or less, otherwise lose 2 points for each move that takes over 5 seconds, but not negative.
- 5 points: Implement Eval(S) as described above.
- 5 points: Implement Eval (S) as Multiple Features Evaluation Function (MFEF) and implement at least five new features (not Win-Path; must do MFEF to qualify).
- 5 points: Implement Iterated Deepening Search (IDS) and terminate within time limit.
- 5 points: Implement a Quiescence Test (must do IDS above).
- 5 points: Implement Alpha-Beta Pruning (ABP).
- 5 points: Sort children for ABP using values found in prior iteration of IDS.

Please put on a switch, so that you can turn them on and off:
(1) Quiescence Test; (2) Alpha-Beta Pruning; and (3) sort for ABP using prior IDS values.
Please discuss in your Project Report whether or not these features helped, and how much?

The following are requirements on your submission. Points may be deducted if not followed. You must use either Java or C++. Your project must run on the ICS lab machines.

- Submit a single zip file. There are two sub-folders and a report in the zip file.
- Folder one named “Source” including Source code
- Folder two named “CK” including Executable files
- Each team hand in only ONE copy of project file, please do not upload more than one
- Make sure team members' names are on the report and the file meets requirements.

Tournament.
I hope to be able to offer a tournament. If so, the top 10% teams will get 10 Bonus Points, the second top 10% teams will get 9 Bonus Points, the third 8, the fourth 7, and so on. Entry in the tournament, if offered, is automatic. If offered, it will be a round-robin, each team playing each other team twice, once moving first and once moving second. Chess tournament scoring will be used: win = 1, tie = ½, loss = 0. Gravity will be on. Board size and winning length (M, N, K) TBD.