Lab2 Report

Shang Da

1. Class:

CheckerApp: GUI for game;
CheckersMove: Data structure for actions;
CheckerState: Game State;
CheckerGame: Game class, implement Game<S, A, P>;
AlphaBetaSearch: Search Algorithm;
Game, Metrix, AdversarialSearch: virtual class from AIMA repository.

2. Algorithm:

The algorithm is alpha-beta search. Implementation is from AIMA repository. But it is modified:

1) Add a depth limit. Depth limit is set to 8 in class AlphaBetaSearch

2) Modified algorithm to deal with the problem that, when a player can jump a piece continuously the algorithm should keep maximize/minimize the utility until the jump is over.

3.Heuristic:

We take simple heuristic: <number of men> + 2*<number of kings>. This already perform very well. It defeats me overwhelmingly it also defeated another checker AI from this webpage: http://www.247checkers.com/.

4. Question 6:

1.As search depth increases, the algorithm become more “rational” at playing checkers. Below depth 4 I can sometimes defeat the AI. But after depth 8 it defeats me overwhelmingly. But as search depth increases the time for each move for AI also increase significantly.

2.With trivial heuristic, for example uniform utility, the AI does not behave rational. By adding a number proper heuristic, the performance improves. For example, AI behaves better when consider both number of men and number of kings compared with only consider number of men. But with bad heuristic the performance it not good. For example, if we assigned too much weight to number of kings, then AI will “focus” on creating kings and can possibly make bad moves.