CS4700 Fall 2011: Foundations of Artificial Intelligence

Homework #3

Due Date: Due Wednesday Oct 19 on CMS (PDF) and in class (hardcopy)
Thereafter 10% off each 24H period (except slack days) until homework is graded on CMS
Your graded homework can be picked up from Upson 360.

Submission: Submit paper copies at the beginning of class or to TA directly. Please include your **name**, **NetID**, the **CMS submission date and time**, **slack days** used and remaining, and a statement that the **hardcopy matches exactly the PDF** uploaded to CMS. Your submission should include the answers and an appendix with code printout. Code in `9pt courier` single spacing with any blank lines removed and function names highlighted. Your assignments should reflect your individual work – ok to discuss strategies but do not compare your answers with anyone else.

Question 1: Adversarial search (50%)

"Og" is a two-player game that starts with an empty board of squares (Fig. 1a). Players take turns adding a single marble of their color on an empty square (Fig. 1b, 1c). Each player's objective is to place their marbles in such a way that at the end of the game they control the most squares. A player controls a square when they place a marble on it or when all neighboring squares (left, right, above and below) are controlled by that player. When a player places a marble that completes the surrounding of a square, a marble of that player is also placed in the surrounded square (1d), and then the player gets another free turn.

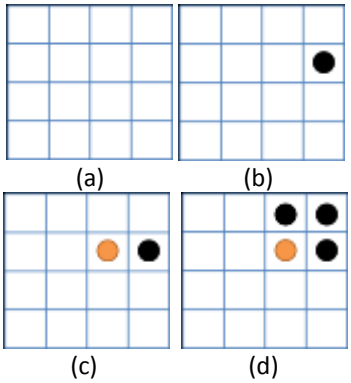


Fig 1: Example of the beginning of a game of Og on the 4×4 board. Just before (d), the black player placed a marble in the top row third column. That completed the surrounding of the top right square and filled it automatically.

Consider Og played on a 4 × 4 board (grid):

a. (10 POINTS) On a blank grid, how many initial moves are there really? Consider symmetries and other regularities.
b. (10 POINTS) What is an upper bound on how many possible different states (i.e., board configurations) exist? It is OK to overestimate the value slightly. Please briefly explain your answer. The most naïve possible answer is that the number of states is bounded by $3^{16} \cdot 2 = 86093442$, i.e., 3^{16} possible configurations of a square being empty, used by player 1 or used by player 2, and two possible configurations for who has the next turn. But you can do considerably better than this.
c. (10 POINTS) The game has the interesting property that a player that surrounds a square plays another disk, i.e., if your move involves surrounding a square, you place a marble inside the

surrounded square and you get another move (i.e. in Fig. 1d it is still the black player's turn). This appears to be at odds with the concept of minimax search (which uses alternating moves), but of course there are several ways to imagine this still in the context of minimax search. Which way do you favor? What are strengths and weaknesses of your way relative to other possible ways?

d. (50 POINTS) You are to write a computer program that plays Og, i.e., write a function in the language of your choice that given a board configuration, returns the optimal next move. This problem is small enough that straightforward complete depth minimax search is possible. Do the following:
    a. Implement the search with minimax search.
    b. Implement the search with alpha-beta pruning.
    c. Run your program on the blank grid. For this, output the number of nodes explored in minimax versus alpha-beta search
    d. Attach your source. The same rules and suggestions apply as they did in homework 2 – keep it simple, readable, and well documented. Write up anything else you think we'd find interest as regards your implementation, e.g., little tricks you used that help speed up your search. Some discussion of your experiences with this problem would be nice to hear.

   Helpful notes: Our minimax searcher worked only if it used a transposition table. This made the difference between finishing in a fraction of a second and not finishing after ten minutes. Our alpha-beta pruning was also greatly helped by memoization (avoiding repeating the calculation of results for previously-processed cases). While it finished without memoization, memorization made the difference between millions of nodes explored versus a few thousand. So, you probably want to use a language where hashing these structures or function memoization is trivial. However, even if you are in a language with no hashing/poor hashing (e.g., C or MATLAB), considering how small the board is, you can very easily map each possible board configuration to a unique number of relatively small magnitude, and just use a straightforward array/vector. Use your imagination. Just to give you an idea of the sort of output you probably want your program to produce, what follows is suggested output format (### inserted where "sensitive" output occurs).

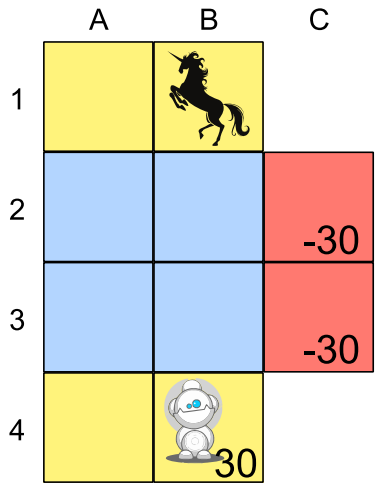
```
minimax first player has value ###
minimax examined ### nodes
alphabeta first player has value ###
alphabeta examined ### nodes
```

e. (10 POINTS) Based on the output you get, does either the first or second player have a winning strategy (i.e., can always force their win) or, failing that, a non-losing strategy (i.e., can always force at least a draw)?

f. (10 POINTS) Imagine that you had arbitrary sized boards, so that doing exhaustive search via minimax is not an option. What is a plausible static evaluation function for an Og board? Brief answers are preferred, and you certainly do not have to implement it.

Question 2: Tea Time (30%)

You are a fierce — but lovable — unicorn (at B1) on the north beach of a frozen lake. You promised your robotic friend (at B4) that you would have some tea with her, but you're a bit scared about crossing the lake. There are three terminal states (indicated on the map by their final reward). Though you are a generally sure-footed beast, the frozen lake is treacherous. Whenever you are on it and try to move to a neighboring square (even onto the beach), there is only a 70% chance that your intended move will succeed, and there is a 30% chance that you will slip and slide in one of the other available directions (with equal probability of sliding in each direction). Sliding about normally wouldn't phase you, but this time the stakes are a bit higher, as immediately to the east of the lake is a huge pit of lava, which will kill you immediately on entry (and unicorn heaven has recently run out of Earl Grey!). Note that moving on the beach (even onto the lake) is always reliable.



Let R indicate the reward for non-terminal states, that is, R(A1) = R(A2) = R(A3) = R(A4) = R(B1) = R(B2) = R(B3) = R. By decreasing R we decrease your patience for remaining in the world without your robotic sidekick. Actions available to you in each state may include North, South, East, and West (corresponding with an attempt to move in that direction). Movements that attempt to move off the board are excluded (e.g., attempting to move north is not a valid action in B1).

For each of the problems, you are to provide 1) a utility graph analogous to the book's Figure 17.3, 2) an optimal policy graph analogous to the book's Figure 17.2, and 3) a brief interpretation (1 sentence) of the obtained policy (To compute utility, we suggest using the Bellman equations solved with the Value-Iteration algorithm of Figure 17.4. Assume your utility is based on simple additive rewards, with a discount factor equal to 1.)

a. (10 POINTS) Suppose you are only slightly thirsty, so your R = −1.

b. (10 POINTS) Suppose you are really quite thirsty, so your R = −20.

c. (10 POINTS) Suppose you are totally desperate for tea, so your R = −50.

Question 3: Critique (20%)

Read the following paper and write a short critique concerning its main issues. Contrast this article with the article you critiqued for HW1.

Rodney Brooks (1991), "Intelligence without representation", Artificial Intelligence 47 (1-3): 139–159

Guidelines for Paper Critiques

The purpose of a critique is to discuss the merit of two or three points from the paper; question the results, the assumptions, the applications, etc. There are helpful guidelines for critiques in HW 1. And remember, your critique should not just be a summary of the paper!

Each critique should be no more than one page long. Less than a page is OK. The purpose of a critique is not to summarize the paper; rather you should choose two or three points about the work that you found interesting.

Your critique should be typed (single space) and should list the title of the paper and its authors at the top, along with your name. Avoid unsupported value judgments, like ”I liked...” or ”I disagreed with...” If you make judgments of this sort, explain why you liked or disagreed with the point you describe.

You may wish to look at other articles that followed after this paper and cited it.