Homework 3
Computer Science
B551 Spring 2018
Hasan Kurban

Jinju Jiang

March 16, 2018

Introduction

The aim of this homework is to get you well-acquainted with $\alpha\beta$. You will turn-in two files

- A *pdf with the written answers called `hw3_jiangjinju.pdf`
- A Python script called gobblet.py

I am providing this LaTeX document for you to freely use as well. Please enjoy this homework and ask yourself what interests you and then how can you add that interest to it! Finally, problems 1 and 2 are worth 20 each and problem 3 is worth 120 points. Include that statement, "All the work herein is mine."

Homework Questions

1. A general search strategy is to work from both the start and goal–think of navigating a maze. Is this a sound strategy for a game? Recall that soundness means $\vdash_{Robot}$ A $\rightarrow \models_{Human}$ A
   Answer to question 1:
   The search from both the start and goal is Bidirectional Search,simultaneously search forward from S and backwards from G,stop when both meet in the middle,need to keep track of the intersection of 2 open sets of nodes. Take "navigating a maze" as an example, bidirectional search is a sound strategy since it can find the solution if there is solution, which means if the system is true,$\vdash_{Robot}$ A, then $\rightarrow\models_{Human}$ A is true.
   But I have some additional questions about bidirectional search:which to take if there are multiple goal states?where to start if there is only a goal test, no explicit list?

2. Assume there is a game with three players A, B, C. You have an evaluation function h that returns 3 numbers: $h(\sigma) = \langle A_s, B_s, C_s\rangle$ where σ is a state of the game and the numbers reflect the goodness of the state for the respectively named players. In no more than two paragraphs, is there a way to modify minimax to work with 3 players? Assume that you cannot exhaustively search the game space and you're able to generate from a state all the next states.
   Answer to question 2:
   For 3 players, each player is to max his/her goodness, then for each player, we use maximum strategy, here it is the graphs to demonstrate how it works:

   - First player. A, choose the maximum A_s from the next states list.
   - Second player, then from Max(A_s) state's next states, B choose the maximum goodness Max(B_s) for himself.
   - The third player, choose the Max(C_s) from the next states of Max(B_s).

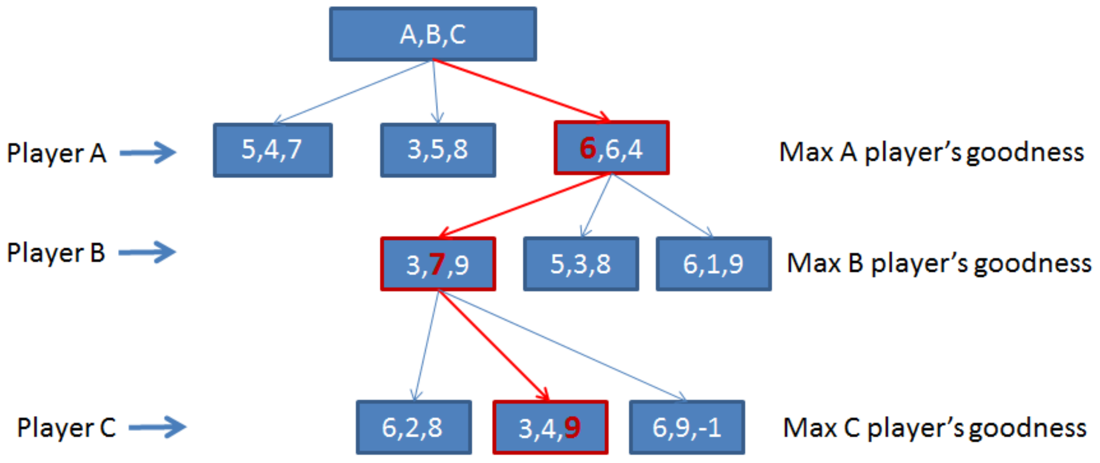


Figure 1: Illustrate 3 players for minimax strategy

3. Implement $\alpha\beta$ for the game of gobblet. Assume that if a state σ is repeated, then the game is a draw:

$$\forall(\sigma)\ [\mathsf{move}(\mathsf{move}(\sigma, \mathsf{player1}), \mathsf{player2}) = \sigma] \rightarrow (\mathsf{game}(\sigma) := \mathsf{draw}) \quad (1)$$

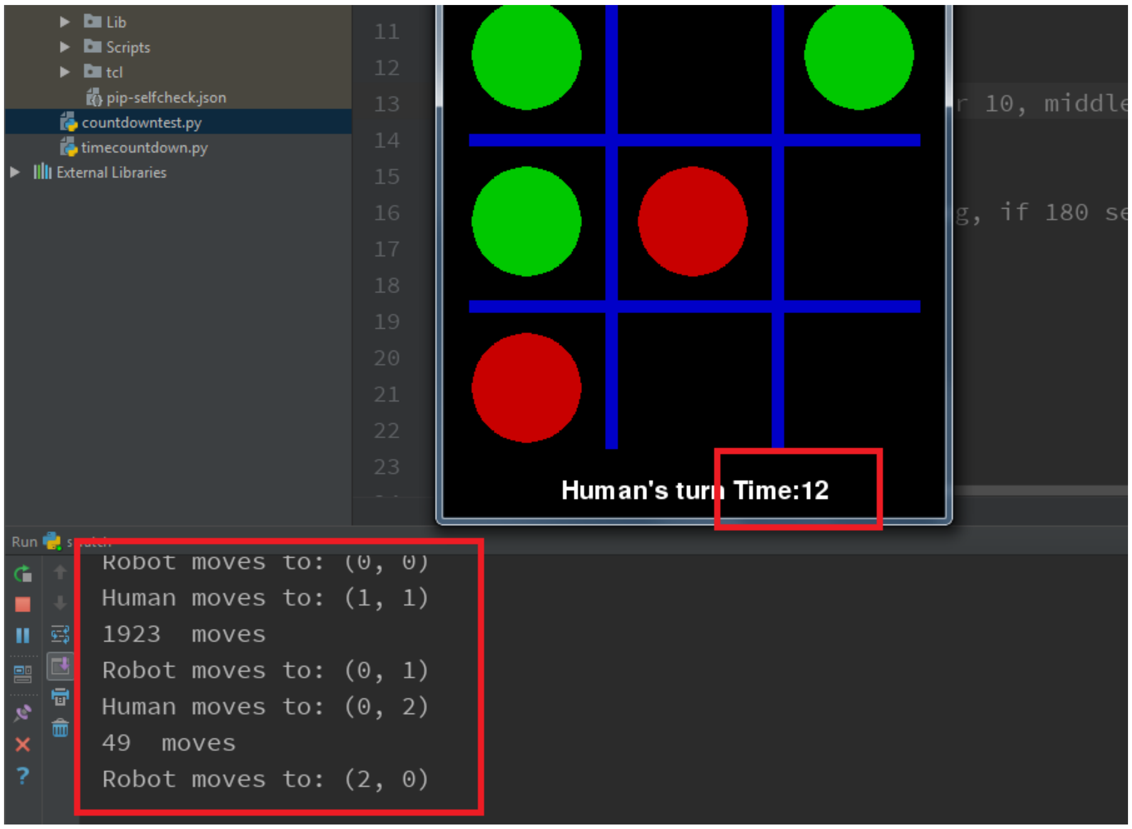
The game should be able to play: human vs. human = h2, human vs. robot = hr (human goes first), robot vs. human (robot goes first), robot vs. robot = r2. There are two sets of parameters: Level: beginer = 0, intermediate = 1, expert = 2. This places a limit on the depth of the search. You must decide experimentally how this is applied; Time: x min. This is the bound on the time you run $\alpha\beta$. The main function should be called gobby(players,level,time) where players $\in$ {h2,hr,rh,rr}. The output should be a sequence of moves with the final state when it's won or drawn.

gobby(r2,2,2.5)

means robot versus robot, expert, no longer than two minutes and 30 seconds should elapse after the opponent makes a move. Whatever is unspecified at this point, you must make decisions on.
<u>Answer to Problem 3:</u>

- First I replace gobblet game rule with tic tac toe game rule to make it easier
- In my implementation, there is human player and computer player, which we can choose h2,hr,or r2
- time elapsed is showing during the game to reminder the game player, if the player can not take action within the time controller(eg, 180 seconds), then game over
- the player level is implemented as clock.tick() which controls the running speed.
- Full code is available in *gobby1_jinjujiang.py*

Here it is some screen shot about game running:



```
Robot moves to: (0, 0)
Human moves to: (1, 1)
1923  moves
Robot moves to: (0, 1)
Human moves to: (0, 2)
49  moves
Robot moves to: (2, 0)
```

Figure 2: Illustrate human and robot game playing

Here came more about Gobblet implementation:

- Then further investigate Gobblet game rule
- The difference is how to calculate the heuristic function, and find min,max value, then using the same alpha beta pruning strategy
- since Gobblet game rule is more complicated, the player level and game control time impacts expansions of the game state during each turn's search. I use MAX_EXPAND which allows each player MAX_EXPAND expansions of the game state during each turn's search
- MAX_EXPAND=int(player level*time control)
- Implement Gobblet game rule into the first implementation, then got the final solution
- the code is available in $gobby2\_jinjujiang.py$