Theory of Computer Games (Fall 2019) Homework #2

National Taiwan University

Due Date: 14:20 (UTC+8), December 19, 2019

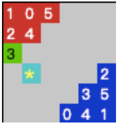
Homework Description

In this homework, you are required to

1. Implement an agent of modified-**Einstein Würfelt Nicht! (Kari)** using Monte-Carlo Tree Search.
2. Beat the conservative AI and the greedy AI.

Basics

- The game is played on a 6×6 board. Initially there are 6 red cubes and 6 blue cubes on the board.



1. Each cube has a number between 0 and 5, and no two cubes of the same color shares the same number.
2. Initial positions of both sides are randomized. 1st player's pieces start from the North-West, while 2nd player's pieces start from the South-East of the board.

- In each turn the 1st player chooses a red cube to move, and subsequently (if the game is not over) the 2nd player chooses a blue cube to move.

Moves

1. In a turn, a player can move any piece of its color.
2. The top-left player (red) can only move a cube to the **east**, **south**, or **southeast** adjacent square.
3. The bottom-right (blue) player can only move a cube to the **west**, **north**, or **northwest** adjacent square.
4. If there is another cube in the adjacent square, that cube is **captured**. A player is allowed to capture a cube of its own.
5. If there is no movable cube, a player should **pass** in that turn. A player is NOT allowed pass if there is at least one legal move.

Terminal Condition

The game is over when

1. A red cube reaches the SouthEast corner, and a blue cube reaches the NorthWest corner...
    - If the SouthEast red cube has numbers smaller than the NorthWest corner, the red player wins.
    - If the SouthEast red cube has numbers bigger than the NorthWest corner, the blue player wins.
    - If the SouthEast red cube has numbers equal to the NorthWest corner, then it is a draw.
2. If the last red cube is captured, blue player wins.
3. IF the last blue cube is captured, red player wins.

Execution Files

- Unzip, there will be 2 folders, game and baseline.
- Under game, make for the executable gaming environment - game
- The game game supports AI-AI mode, AI-human (1P) mode, and human-human (2P) mode.
- Under baseline, make for 3 given agents, random, greedy, and conservative.
- To begin with, use

  ```
  $ ./game -p0 ./greedy
  ```

  to start playing Human vs. AI with the agent greedy.

Protocol

- An agent receives the last move of the opponent from `game` and sends its move accordingly back.
- We've handled most parts of the communication. Receive messages by reading from `stdin` and send messages by writing to `stdout`.
- Read everything character-by-character; if you expect a message of length *k* to be received, read one character *k* times instead of directly reading a string.
- Remember to flush every time after writing a message to `stdout`.

Frame of an Agent

```
while true do
    receive $R_1, R_2$
    $B \leftarrow$ the initial board given $R_1$
    $YourTurn \leftarrow R_2 =$ “f”? $True : False$
    while true do
        if “game has reached terminal condition” then
            break
        end if
        if $YourTurn = False$ then
            receive $R_3$
            do the opponent’s move $R_3$ on $B$
        else
            choose a move $M$
            do the move $M$ on $B$
            send $M$
        end if
        $YourTurn \leftarrow !YourTurn$
    end while
end while
```

Formats of Received / Sent Messages

1. $R_1 := R_1[0:1][0:5]$: a permutation of "012345".
   - number of $(1,1) = R_2[0][0], (4,6) = R_2[1][0]$
   - number of $(1,2) = R_2[0][1], (5,5) = R_2[1][1]$
   - number of $(1,3) = R_2[0][2], (5,6) = R_2[1][2]$
   - number of $(2,1) = R_2[0][3], (6,4) = R_2[1][3]$
   - number of $(2,2) = R_2[0][4], (6,5) = R_2[1][4]$
   - number of $(3,1) = R_2[0][5], (6,6) = R_2[1][5]$
2. R_2: a single character.
   - 'f': you are the 1st player in this round
   - 's': you are the 2nd player in this round
3. R_3: can be "??" (pass), or *nd* (otherwise), where
   - $n =$ number of cube to be moved
   - $d =$ direction: 0 (vertical), 1 (horizontal), 2 (diagonal)
4. M: a 2-sized string, can be "??" (pass) or *nd* (otherwise) only.

Misc

- You can assume that every move your agent receives is valid.
- Your agent should send a valid move within 10 seconds. If `game` receives an invalid move, or doesn't receive a move within the time limit, your agent will be killed, and your opponent wins immediately.

Code

- You're required to implement the following algorithms:
  - UCB score and UCT
  - Progressive Pruning **or** RAVE
- Your execution file should be named with your student ID, with all alphabets in lower case, e.g., b08902000, not B08902000.
  - If your programming language is python3, add `#!/usr/bin/env python3` in the first line and remove `.py` from the filename.
- Your agent can use at most 1 thread.
- Your agent will be tested by

```
$ ./game -p0 [your_id] -p1 [our_agent] -r 5
```

Report

- Your report should include but not limit to the following:
  - How to compile your code into an agent (if your code must be compiled). Don't upload the compiled executable file!
  - What algorithms and heuristics you've implemented.
  - Experiment results and findings of your implementation.
- Your report should be named `report.pdf`.

Directory Hierarchy

- [your_id] // e.g. b08902000
  - src // the directory contains your code
  - makefile
  - report.pdf
- Compress your folder into a zip file.

Grading Policy

- Basics: 15%
  - Beat the agent `conservative`. 5%
  - Beat the agent `greedy`. 5%
  - `report.pdf` 5%
- Bonus:
  - Ranked high in class.
  - Beat Hidden Boss !?