CS 4649 RBMC Project

CS 4649/7649 Robot Intelligence: Planning
Instructor: Matthew Gombolay

Notes:

- You may work with one or two classmates on this assignment (groups of 2-3). Every student in the group should participate equally. Your final report should state who contributed to each facet of the project. If you have trouble finding partners, please reach out to us and we will help!

Introduction:

For this project, you are asked to develop an AI that plays Recon Blind Multi-Chess – take Chess, remove the ability to see the opponent, but give yourself the ability to reveal a 3x3 grid on the board to "sense" before moving. After developing an AI, we will have a tournament where groups play against each other. Each player will have five minutes on their clock to play an entire game of RBMC –totaling 10 minutes per game. For a complete comprehensive rule guide, see this link: https://reconchess.readthedocs.io/en/latest/rules.html.

Grades will be based upon **1)** how well you have incorporated multiple concepts of the course into your approach, **2)** how well you document/present your approach in a final project report and presentations, and **3)** bonus points for performing well in the competition. The winner of the tournament will receive 15 bonus points. 2nd place will receive 10 points. 3rd place will receive 5 bonus points.

There are **three** important submission deadlines.

1) **Test Code [Sunday, April 24th @ 11:59 PM Eastern] –** We will playtest your code prior to the tournament to make sure it runs and provide feedback if it does not. You do not have to take advantage of this option. You will not receive points off for not submitting. However, if your code doesn't run for the tournament, that is that.
2) **Final Code [Sunday, May 1st @ 11:59 PM Eastern] –** Final code submission for tournament. Please include a README.md with your code. **If this code does not run, you will receive 0 credit.**
3) **Final Report [Monday, May 2nd @ 8:00 AM Eastern]** – Report documenting AI (see further details below)

What We Provide:

You will receive a package containing several files (short descriptions listed below). These files simulate the environment we will be running during the main competition. Download the assignment zip file from Canvas and unzip it in your workspace.

You will also need to install the python-chess library. You can install python-chess by using the pip command line:

```
pip install python-chess
```

Note: you might need to use ‘pip3’ instead of ‘pip’ to make sure it installs in your Python3 environment.

Once installed, to import the python-chess library to any additional file, write:

```
import chess
```

Files

game.py The game object class; methods included help with interacting with the game board. Do not modify.

human_agent.py Agent that allows for user input in the console to play RBMC. Do not modify.

my_agent.py Your assignment file containing methods that must be implemented. This is the file that you are submitting for this project.

- NOTE: Before submitting, you must rename your my_agent.py file to your team members’ Georgia Tech usernames, separated by an underscore, followed by ‘.py’ as follows:
    - For example, if Erin and Lakshita were teammates, my_agent.py → becomes → ehedlund6_ldodeja3.py
    - Fun Fact: Rename the class “MyAgent” to the name you wish your bot to appear as in the tournament. Please no profanity.

play_game.py Plays a game of RBMC between two agents passed in as arguments. This also saves a .TXT file to the “GameHistory” folder with a complete history of the match. Do not modify.

player.py Helper class for implementing player agents. Do not modify.

random_agent.py Agent that randomly selects moves; used as a dummy agent to play against. Do not modify.

Your Responsibilities:

You must implement the methods in my_agent.py to play RBMC with an artificial intelligence agent of your own design. Of the provided files, this is the only file that may be updated. Any alteration to any other file will not be beneficial to you as the originals will be used during the competition to evaluate your agent.

Additionally, the method headers and return statement of the methods that my_agent.py come with also should not be altered as this will cause them to not be called properly nor return properly.

In order to run the game:
python play_game.py [path to WHITE player] [path to BLACK player]

NOTE: If you want to the option to play as any color as human player, make sure to pass in the path to the human player first.
If you would like to provide any other methods or python files to aid in your agent, you may do the following:

- Add additional methods or classes to my_agent.py that are called within my_agent.py.
- Add additional python files containing functions or classes that are imported and used in my_agent.py. See additional notes on this below.
- NOTE: This last stipulation must be approved: Import additional helpful libraries not already on the approved library list (a pinned post on Piazza) to my_agent.py for use. To do this, you must email TAs detailing the provide the libraries you wish to use, how you plan to implement them, and why it is necessary you accomplish this with this library. You must get approval to do this before the competition deadline and should be prepared for rejection of approval should that occur. We reserve the right to approve, reject, and limit the use of any library.

For potential training purposes, you will have access to several functions and attributes within game.py, including the truth board. This will not be the case during competition so please take care that your agent not attempt to call these variables or methods directly from game.py, else it will fail the match.
We are passing the information directly to you through the handle methods, but if you find that there is other useful information for the decision making process, please email the TAs with what additional information you would like provided and why. We reserve the right to approve, reject, or limit the information requested.

Naming Conventions for Additional Files:
For each additional python file you wish to use in your game, please do the following:

- Make sure your my_agent.py file has been renamed to your group members Georgia Tech usernames, separated by an underscore.
- Using the same order of names, name each additional file as
  member1_member2_filename.py
  where "filename" is whatever identifying characteristic you wish the file to have. Ex. ehedlund6_ldodeja3_neural_net.py

Undergraduate Students Minimum Requirement: Must utilize at least MCTS and one other concept from class (ex. particle filter, machine learning, etc.) within their approach.

Reasoning Under Uncertainty

There are many approaches to integrate partial observability into the reasoning of your system. We present two baseline approaches here, but you are free to implement any approach you find effective. Remember the state space size is very large for Chess and you may want to encode the state you are in.

1) Given some observations, find the most likely state your RBMC agent is located in at the current timestep. Using this state as the ground-truth, perform NN-guided MCTS to choose an action (vanilla AlphaGo Zero).
2) Given some observations, maintain a distribution over possible states your RBMC agent may be located in. From each of the top *n* most probable states, perform NN-guided MCTS and determine high-value actions in each possible state. Pick the action that has the highest (probability_of_state* value_of_action).

Deliverables

Python files:

- **my_agent.py** – this python file should be called by the graders and run everything
- Additional files, such as model weights (if you use neural networks)
- README file explaining what your additional files include

PDF files:

- Report (group members can submit same report)
  - The report should be less than 800 words and include:
    - Details on the method used to develop AI
    - Details on how each member contributed to the team

Please submit your agent to Gradescope to test it against the random agent and a hidden agent. To submit to Gradescope, please submit your renamed my_agent.py file and a text file called filename.txt with the name of your renamed file (ex. ehedlund6_ldodeja3.py).

Rubric (100 pts)

- Report (30 pts)
- Beats random agent 100% of time (30 pts)
- Beats hidden agent 100% of time (20 pts)
- Agent can compete in tournament (20 pts)
- Tournament 1st place (+15 bonus points), 2nd place (+10 bonus points), 3rd place (+5 bonus points)