HW2

Deterministic games

Index

Chinese Dark Chess

- All assignments of this course will be based on the popular park-bench pastime.
- Chinese Chess with extra quirks
  - **Stochastic**: pieces are randomly distributed face down
  - Simpler movement rules
  - Played on half a board: 4x8 squares
- For HW2, we will play a slightly different version of the game

Chinese Dark Chess Rules

- Movements
  - All pieces move one square in any of 4 directions
  - The cannon must *capture* by jumping over another piece
- Ranks
  - There is a hierarchy of pieces, only certain pieces can capture others
  - General (將) > Advisors (士) > Elephant (象) > Chariot (車) > Horse (馬) > Cannon (包) > Soldier (卒)
    - Cannons can capture anyone
    - Soldiers can capture Generals, and *not* vice versa
  - You don’t really need to remember this, the code provided will take care of it for you

Fullbright Chinese Dark Chess

You will be playing Chinese Chess but:

- Fullbright: the board starts with all pieces revealed
- (that's it)

Forget about HW1, no more ducks, cannons are back, and chariots now move like any other piece.

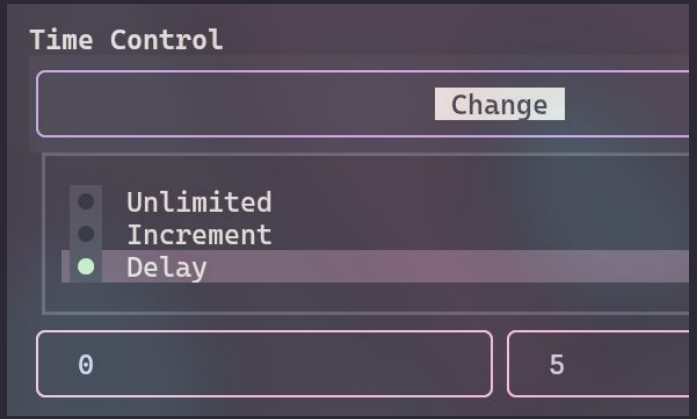
Your goal

- Do not lose.
- Do not not win.
    - The game will end in a draw if no captures are made by either side in 30 consecutive plies.

Timing

“The clock is the 33rd chess piece.”

- The time limit is **5 seconds for every ply**
  - You should choose Delay 0|5 in the time control setting
- **Running out of time loses the game.**
- **Don’t forget the network latency!**



Programming - Baselines (60%)

Your agent will play against three of our bots for grades.

- Trivial Baseline - 25%
- Easy Baseline - 25%
- Medium Baseline - 10%

1. For this part, you get 1 point for winning and 0.4 points for a draw.
2. The difficulties may be adjusted in the future.

Programming - Baselines (60%)

- The grades scale linearly below the threshold for maximum grade.

  (8 wins + 5 draws against Easy gets (8+5*0.4)/14 * 25% = 17.9%)

Baseline	Total games	Points for maximum grade	Grade	How do I get this?
Trivial	20	17	25%	Basic MCTS
Easy	20	14	25%	Enhancements
Medium	20	10	10%	Good enhancements

Programming - Code (20%)

Your code will be audited to make sure it actually implements MCTS.

- MCTS - 8%
  - Correct (bug-free) implementation of Monte-Carlo Tree Search
- Enhancements - 7%
  - Implement RAVE
  - One other advanced MCTS techniques, introduced in class or related papers
  - You receive marks based on the effectiveness of your implementation.
- Readability - 5%

Programming - Enhancements

- You must implement RAVE,

  and *at least one other* advanced techniques
  - You may use any other technique related to MCTS, except machine learning!
- You may edit the makefile
  - We will run the plain `make` for your baseline run
  - You can enable your extra features under `make full`
  - This is useful if some enhancements does not enhance your agent

Programming - Note

- Your agent may not be restarted between games.
- However, you can detect a new game by count the number of pieces.
- Output format:
  - A move
  - `info << move` will do the trick, you do not need any additional newlines.
- For PieceType, only ‘>’ is overloaded
  - Other operators will not follow proper rankings for Generals and Soldiers
  - A > A is true

Programming - Wakasagi

- The engine has been updated for this assignment
- NEW - Position::time_left()
  - For your time management (more so in HW3)
- NEW - Position::undo_move()
  - Can be called any number of times
- NEW - Position::simulate(Move (*strategy))
  - You may write your own strategy

Documentation here.

Program rules

- Your code should run on the CSIE workstations.
- **You get one (1) thread.** No parallelism, forking, threading.
- No pragmas or any other similar gcc witchery.
  - We reserve the right to witch hunt.
- Memory limit: 400 MiB (virtual address space)
- We will not compile your code if there is any warnings.
- **Do not edit:**
  - lib/*

The game platform

- We have set up a platform for your agents to play on.
- The client program is provided in the files.
- You can log in with your account and password
  - Will be provided later
  - It is NOT your credentials for submitting homeworks
  - You can change the password if you wish

The game platform

- pip install -r requirements.txt
- Written for Python 3.12.8
- Replay files are saved to
  - $CONFIG_HOME$/tcg_wasabi/replays/
  - For most Linux users, $CONFIG_HOME$ is ~/.config

The game platform

- Known issues:
  - Matches lasting over an hour or so may crash.
  - Leaving the room while playing the game may crash the client.
  - If your agent crashes, the client will crash.
  - The clock displayed is all kinds of broken, especially when using delay time controls.

The client may be updated soon to address (some of) these issue.

Written part - Report (20%)

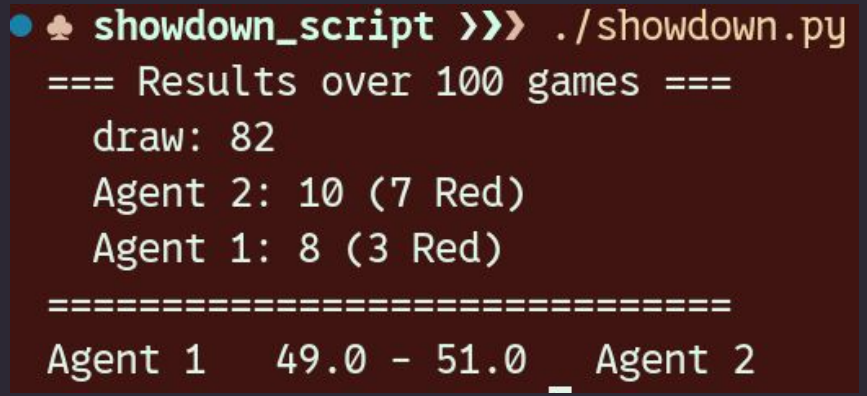
Your report should contain the following:

- Explanation of your implementation (10%)
- Experiment results (10%)
  - You can show the different versions of your agents as you fix bugs and add enhancements
  - Showing the head-to-head results between each of them is recommended

Showdown script

showdown.py is a script that will help you compare two agents.

- Before using, you should compile the referee with `make`
- `./showdown.py --help`
- This script does not timeout your agents. You should test the time usage on the server.



Submission

- For this assignment, the code and the report have separate pages
- **Code**
  - **Due in three weeks (Thu. 14:20)**
  - Simply zip all your files, no top level directory needed
  - Do not include lib/
- **Report**
  - Due 3.5 days after the code (Sun. 23:59)
  - You can submit the pdf directly

Late policy

- Your submission time is server-sided, **do not submit at the last second**
- Each late day incurs a 0.9x penalty
  - Rounded up to the nearest day
  - 1 second of delay counts as a full day
- Maximum of 7 days of delay accepted