The Game of Cannon (Phase I)

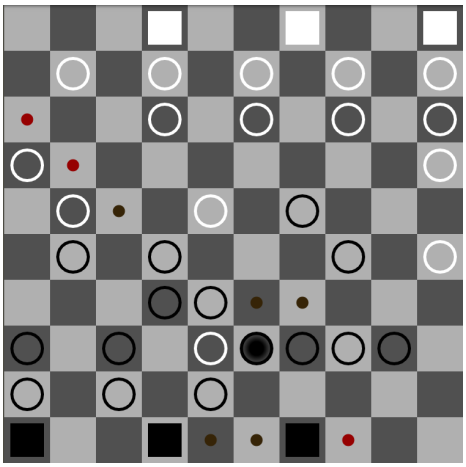
COL333

August 30, 2019

1 Goal

The goal of this assignment is to learn the adversarial search algorithms (minimax and alpha beta pruning), which arise in sequential deterministic adversarial situations.

2 The Game of Cannon (`N`)



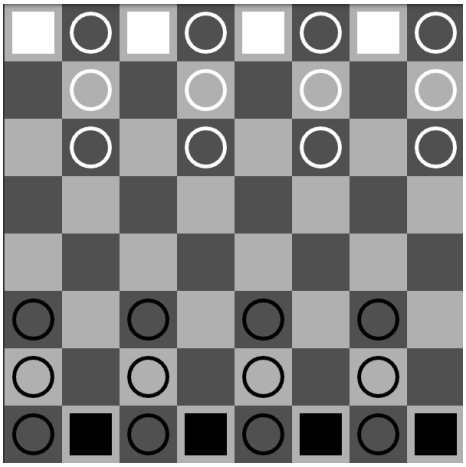
Cannon is a perfect information, 2-player abstract strategy war game. The game is based on protecting one's own Townhalls and destroying the opponent's Townhalls. For purpose of the assignment, each instance of the game shall be timed, i.e. there shall be an upper bound to the total time used by the bot for making the next move, totalled over all the moves played by the bot in the game.

2.1 Objective

The objective of the game is to attack the Townhalls of the Opponent player, keeping your own Townhalls safe. The player who eliminates two of the Opponent's Townhalls is declared the winner.

2.2 Start of Game

The game is played on a **8X8 board**. At the start of the game, each player has a fleet of **12** soldiers (represented by **rings** in the UI included in the assignment packet), placed in groups of three and a set of **four** Townhalls (represented by **solid squares** in the UI included in the assignment packet), placed in alternative blocks at the last row. The Townhalls shall remain stationary. A line



(orthogonal or diagnol) of three consecutive soldiers is termed a **cannon**, which is capable of "shooting".
Note:- Although for this assignment, we are using an 8X8 board. For the future assignment, the board size may not necessarily be square. We encourage students to make their code generic enough to allow an NXM board size without much changes, so as to save time in the future.

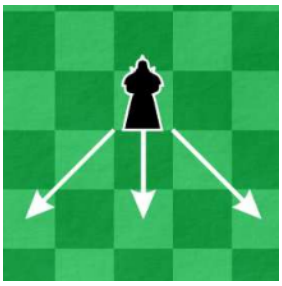
2.3 Movement

The two elements of the game of Cannon are:

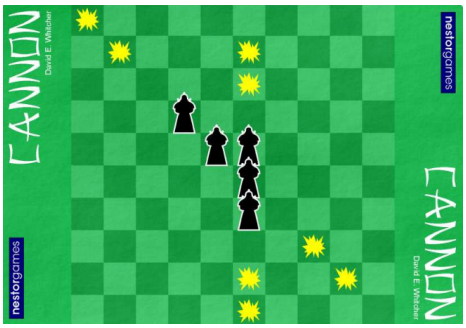
- Soldier :- A soldier is capable of the following moves.
  - May move to an adjacent (orthogonal or diagonal) forward *empty point*.

- Can capture an adjacent (orthogonal or diagonal) forward or sideways opponent piece. A captured soldier is **removed** from the game board.

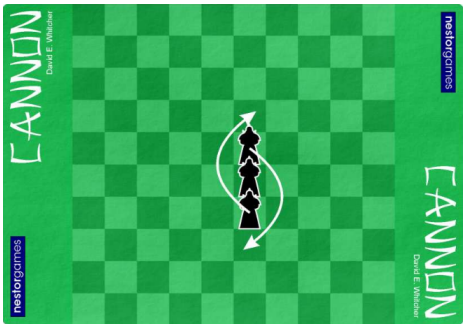
- Retreat backwards (orthogonal or diagonal) two blocks if it is adjacent to an enemy piece.



- Cannon :- A cannon is capable of the following moves.
  - Cannon Shot - May make a non-move capture two or three positions in line with the group *as long as the position immediately in front of the cannon is not occupied*. A cannon shot can eliminate a soldier as well as a Townhall from the game.



– Cannon Movement - It may also shift along its length in either direction without capturing.



At each turn, the player must either move one of it's soldiers or cannons. Passing is not allowed.

3 Scoring

The game is considered to be ended in either of the following scenarios:-

1. A player has *eliminated two* of the Opponent's Townhalls. This player is declared the winner.

2. A *stalemate* condition has been imposed, i.e. there is no legitimate move available to play for a player. The player with more remaining Townhalls is declared the winner. If both players have same number of remaining Townhalls, a *draw* is declared.

3. There shall be an upper bound to the total time taken by the player for making the next move, for the complete game. If a player runs *out of time*, the player is considered defeated.

3.1 Performance Evaluation

At the end of the game, each player shall be given a score, such that the following order is maintained:
Win > Stalemate Win > Draw > Stalemate Loss > Loss
The scoring shall be done in the following way.

3.1.1 Town Hall Margin Score

This is the score based on the extent of victory. Here is the scoring pattern.

Your Town Halls Remaining	Opponent Town Halls Remaining	Score
4	2	10
3	2	8
4	3	7
4	4	5
3	3	5
3	4	3
2	3	2
2	4	0

The above table gives the **Town Hall Margin Score** for the given game instance.

3.1.2 Army Margin Score

Since, soldiers are an important element for the game of Cannon, this score directly depends on the number of soldiers you have left at the end of the game. It is calculated as follows:

$$\textbf{Army Margin Score} = \#(\text{Soldiers Remaining})/100$$

3.1.3 Total Score

The total score of a player, at the end of a game, can be given by:-

$$\textbf{Total Score} = (\text{Town Hall Margin Score}) + (\text{Army Margin Score})$$

4 What is being provided?

Your assignment packet contains code for the game server in python and starter code for your player in Python that: (1) interacts with the game server, (2) allows you to manually send moves to the server. At the client end, you are provided a GUI which allows you to visualize the proceedings of the game.

This entire codebase can be found at: https://github.com/goelShashank007/Cannon-AI/. Please find the **instructions** to run, use and interact with the game server, in the README of the aforementioned repository. Please note that this game server shall be used for evaluations as well.

5 Interaction with Game Server

The game server has been developed using Python 2.x. The server won't work with Python 3.x.

5.1 Dependencies

1. Python 2.x dependencies: `pip install -r requirements.txt`
2. Chromedriver: Chromedriver is needed for GUI functionality. Download from http://chromedriver.chromium.org/downloads Unzip the downloaded file, and include the path to the directory in your `PATH` variable.

5.2 Running the Server

The code for server can be run using the following command:

```
python server.py <port> <arguments>
```

The arguments for the above command are:

`port` (mandatory) - The Server Port.

`ip` (optional) - The Server IP. Default: 0.0.0.0

`n` (optional) - The Board Size. Default: 8

`NC` (optional) - Number of Clients. Default: 2

`TL` (optional) - Time Limit. Default: 150

`LOG` (optional) - The Log File.

5.3 Running the Client

The code for the client can be run using the following command:

```
python client.py <server-ip> <port> <executable> <argument(s)>
```

The arguments for the above command are:

`ip` (mandatory) - The Server IP.

`port` (mandatory) - The Server Port.

`executable` (mandatory) - Executable. (C++ or Java Executable or Python file)

`GUI` (optional) - The View Mode ('GUI' / 'CUI'). Default: 'CUI'

5.4 Gameplay

The gameplay for Cannon, includes the movements of the soldier and the cannon. The following outputs from your executable file are desired on the standard output, which shall be read by the Client Wrapper and sent to the Server. Please note that an illegal/invalid move shall lead to **immediate termination**t of the game.
To locate a soldier, the coordinates convention is: The top-left corner block is the origin. The horizontal direction towards the right is the positive x-axis. The vertical direction downwards is the positive y-axis. Given this, the following moves shall be accepted by the Server:

1. **Moving a Soldier**
   To move a soldier from (1, 2) to (2, 4):
   `S 1 2 M 2 4`

2. **Cannon Shot**
   To throw a bomb, select any of the soldiers of a cannon, and throw it at any viable target of the cannon(s) formed by that soldier.
   `S 2 4 B 6 4`
   For the above command to be valid, one of the three soldiers in the cannon should be at (2, 4) **and** one of the valid shots by this cannon should be at (6, 4).

6 What to Submit?

6.1 Submission Files

We shall float a Google Sheet, where each of the teams shall mention the Entry Numbers of their team member(s) and the name of their bot. Submit your code for your game player. The code should be contained in zip file named in the format `<BotName>.zip`. Please ensure this **exactly** matches your bot name in the google sheet originally floated. This will be case-sensitive. Make sure that when we unzip the following files are produced:

1. `compile.sh`
2. `run.sh`
3. `writeup.txt`

You will be penalized for any submissions that do not conform to this requirement. Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 12.04. You are already provided information on the compilers on those VMs. These configurations are similar to GCL machines like 'todi' (have a RAM of 16 GB).
The `writeup.txt` should have two lines as follows

1. First line should be just a number between 1 and 3, denoting the following.

- Number 1 means C++
- Number 2 means Java
- Number 3 means Python

2. Second line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say "None".

After these first two lines you are welcome to write something about your code, though this is not necessary.

6.2 Code Verification

Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur a significant penalty. You will be able to check if your submissions comply with the proper format and runs as expected in the test environment by submitting to Moodle. The auto-grading in Moodle will run your bot against the RandomPlayer.py in the same environment as the final evaluation and display the game output.

7 Phases

The game of Cannon, shall be used twice for Assignment in this course. The two phases are:

7.1 Phase 1 - A2

1. This phase will be worth **5 Points** in the Course Total.
2. The submitted bot will be tested and scored, against 3 bots made by the TAs. Against each bot, once as Black Player and once, as White Player, making it a total of 6 games.
3. Each game will be given a score (Look at the scoring section above). The final score will be the sum of scores from the 6 games.
4. A **seed** for each bot will be created based on this final score for the next phase. In case of a tie between the final scores of two bots, the tiebreak will be determined by the sum of the scores of the TA bots.

7.2 Phase 2 - A5

1. This phase will be worth **15 Points** in the Course Total.

2. The submitted bots will be placed in a tournament tree based on their **seeds** from Phase 1 (Inspiration taken from a Wimbledon Tournament Tree). The tree will initially consist of a round robin stage (4-6 bots in each stage) followed by a knock-out tournament.

3. The scores for each bot will be based on how far it progresses in the tournament. This score distribution will be revealed in the future.

4. Each tournament will be worth 5 marks and a total of 3 tournaments will be played. The winner of each tournament will be removed from the subsequent tournaments and will receive full 15 marks.

5. More details on this shall be notified at the time of assignment 5.

8 What is Allowed / Not Allowed?

1. You may work in teams of two or by yourself. If you work in a team of two then make sure you mention the team details in the write-up. Our recommendation – this will lead to a much more open ended final assignment (phase 2); hence best to work with a partner with whom you communicate well. You **must** team up with the same partner for the final assignment. Also, you cannot use a partner from the previous assignment (except if you are a COL671 student, in which case you can team up with a partner up to twice).
2. While you are allowed one of three languages we recommend that you use C++ since it produces the most efficient code. This assignment requires you to play the best game within a given time constraint.
3. Your code must be your own. **You are not to take guidance from any general purpose AI code or problem specific code meant to solve this or related problem.**
4. It is preferable to develop your algorithm using your own efforts. However, we will not stop you from google searching.
5. You must not discuss this assignment with anyone outside the class. Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team. Please read academic integrity guidelines on the course home page and follow them carefully.
6. You get a zero if your player does not match with the interaction guidelines in this document.
7. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.