Game Theory Assignment 2

Jayveersinh Raj (BS20 DS-01)
j.raj@innopolis.university

***Abstract*— The ability to solve game theory problems is extremely useful in a variety of settings. Snowball fight is a physical game primarily played during winter when there is sufficient snowfall. Usually, players are dividing into teams and attack opponents or their fields. The environment where players play the snowball game is three territorial regions which we will helpfully entitled as A, B, and C fields. Fields A and B contain one player on each side. The field C does NOT contain any players and should be considered as a hot field where snowballs automatically melt and disappear. In this simulation, we will create a model of the snowball game environment in order to understand the nature of this game. The outcomes of tournaments with strategies in place are displayed in this context.**

***Keywords*— agents, strategies, tournaments.**

I. Task Description

The aim is to play a three-field simultaneous game with two participants. Both players use Snowball Cannons (SC) that can shoot snowballs. Assume that shooting does not cause snowballs to melt or burn. Cannons can be used once per minute on the opponent's field and once per minute on the hot field (total at most twice, assume it as sequential shots).

First, to the opponent's field). Each shot may contain one or more snowballs. Not shooting is also permitted, and a proper method must return 0. If no shooting occurred on the opponent's or hot fields, it is assumed that no shooting occurred during this minute. If SC was used once or twice per minute, it is assumed that shooting occurred during this minute.

The idea is to put in place a strategy that works in competitions. In other words, each approach competes against the others in numerous rounds, with the outcomes of each competition being summed together.

A. *Maximum possible snowballs that can be shot*

Maximum possible number of snowballs that can be shot is defined by the following function:

$$f(x) = \left\lfloor \frac{15 \times e^x}{15 + e^x} \right\rfloor$$

where x is the number of minutes passed after the previous shot (presence of shooting). This equation is not affected by the number of snowballs shoot during the last shots. Cannon is also limited by the number of snowballs present in the field.

B. *Inputs*

The last shot of the opponent is given to the agents, as well as the snow balls that they have, and minutes passed since their last shot. Two functions, each for each field to shoot, not shooting is also allowed.

C. *Output*

After 60 minutes (also called rounds), the number of snowballs left in your field has to be minimized for maximizing your payoff.

II. Project Structure

The assignment was implemented in **Java** programming language. Project contains three files:

1) *Player.java* - an interface given in the task
2) *JayveersinhRajCode.java* – Best agent code
3) *JayveersinhRajTesting.java* - a test class that contains implementation of all created agents and tournament function to have tournament between them all.

A. *Main*

Main function runs the tournaments. It is sufficient to run the main function of the JayveersinhRajTesting.java file to obtain results.

B. *Tournaments*

All agents compete with each other, and at the end most frequent winner is considered to be the champion of that tournament. 10 such tournaments are arranged to find the best one. Draw was considered a win to eliminate biases.

C. *Agents*

Five classes that implement Player interface with different strategies.

D. *Goal of these agents*

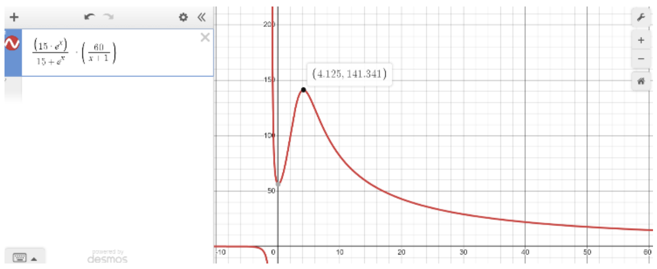
The main goal of these agents is to minimize the number of balls in their field to maximized their payoffs. They can shoot sequentially to opponent and hot field. However, it to be noted that shooting to either is considered as shooting while making minutes passed since last shot 0. Hence, players have to adapt some strategies. Not shooting is also allowed.

III. Strategies

There are five agents with five different strategies.

A. *Greedy*

It is the smartest, and the greediest for increasing its payoff, and reducing its own balls by increasing opponent's balls, hence pressuring the opponent to come up with some smart strategy to deal with it. Shown below is how it works:


$$\frac{(15 \cdot e^x)}{15+e^x} \cdot \left(\frac{60}{x+1}\right)$$

The picture above shows that maximum balls can be thrown when value of x is 4, i.e., minutes passed since last shot is 4. Thus, shooting on every 5th minute yield the maximum.
It utilizes the function provided for maximum possible shots, to come up with the best minute at which it would shoot.

A. *Random Agent*

This is a random strategy where in each round the agent generates a random number, and decides if he wants to shoot on this turn or now. It generates random numbers for both hot field and opponent field.

B. *Retaliatory*

This is a strategy where this agent will retaliate with full force, meaning that it does not matter how many balls were shot to this agent, if any ball is hit to it, it will retaliate with all the maximum possible balls that it can hit.

C. *SmartCopyCat*

This is a strategy where this agent will mimic the opponent's moves, however, it can detect if the opponent shoot every possible balls of his or her or not, and if not all possible balls were shot at this agent, it would throw the same amount to the opponent, and remaining would be shot at the hot field to reduce the number of balls on its side.

D. *SmartHotShooter*

This is same as *Greedy*, except that it is less offensive, and peace loving. It always shots at the hot field to reduce its balls, but without hurting the other. Other than this, it works exactly the same as greedy.

IV. Tests

One versus all tournament was performed 10 times, and the following was achieved:

The following table is the Agents versus agents' single matches results after one versus tournament performed 10 times. However, it is to be noted that draw was considered a win.

Agents	Wins
Greedy	4 wins
RandomAgent	3 wins
Retaliatory	3 wins
SmartHotShooter	3 wins
SmartCopyCat	3 wins

V. CONCLUSION

Finally, *Greedy* found to be either winning or drawing.
The rest either lost against *Greedy* or just made it draw
Even considering draw as a win, Greedy secured 4.

Hence, JayveersinhRajCode.java contains the Greedy agent code, i.e., the best agent.

REFERENCES

[1] McEachern, A. (2017). Game Theory: A Classical Introduction, Mathematical Games, and the Tournament (Synthesis Computational Intelligence). Morgan Claypool Publisher