Moose game assignment report

[Spring 2020] Game Theory course

Ruslan Sabirov
Computer Science program
Innopolis University
Innopolis, Russia
r.sabirov@innopolis.ru

Munir Makhmutov
AI in Game Development Lab
Innopolis University
Innopolis, Russia
m.makhmutov@innopolis.ru

Joseph Alexander Brown
AI in Game Development Lab
Innopolis University
Innopolis, Russia
j.brown@innopolis.ru

***Abstract*—This document is a report for the second assignment in Game Theory course, which is being taught in Spring 2020 semester at Innopolis University. The report defines problem statement, goals, solutions, observations and conclusions.**

***Index Terms*—game theory, iterated games**

I. Introduction

Iterated games are most of the important part of Game Theory since iterative playing is more natural (we usually take some sequence of decisions related to the problem, not one decision). In this assignment, we were supposed to implement and analyze several agents for a Moose game and select the best one by tournament.

The rest of the report is organized as following: in the section II we define problem and the task for the assignment. Section III explains code organization of the project. Section IV explains logic of each agent. Section V is devoted for tournament results. Finally, in the section VI we conclude the work.

II. Problem statement

Moose are large quadrupedal herbivores which range in Northern longitudes from the Northern United States, Canada, Scandinavia, and Russia. In one particular game there are two Mooses fighting for survival.

There are three different fields: A, B, C, each has x values, representing amount of grass in the field (xA, xB, xC). These x values start with $x = 1$. When there no Mooses in the field, its x value is increased by 1. When there are at least one Moose in the field, its x value is decreased by 1 (minimum value of x is 0).

Mooses are playing iteratively. At each of the K game moves two Mooses simultaiously decide on each field to move (xK). If Moose is the only one in the field, it gets payoff given by equation $f(xK) - f(0)$, where $f(x) = \frac{10 \cdot e^x}{1+e^x}$ and xK is the amount of grass in the particular field. If there are two Mooses in one field, they will attack each other and none of them will get any payoff.

III. Code structure

A project is logically divided into two parts: implementation of agents and tournament (testing).

A. Agents

Each agent implements provided interface $Player$ with three methods:

- $reset()$ — resets state of the agent before each game;
- $move()$ — given field's state and opponent's last move returns current move of the agent;
- $getEmail()$ — returns E-mail of the student (used in grading).

Logic of each agent is implemented in each agent's class and will be described below.

B. Tournament

The tournament class consist of:

- $main()$ method. In this method we create a list of all implemented agents and corresponding scores (both with length N). After that agents play against each other ($\frac{N \cdot (N-1)}{2}$ games in total). Each game starts with reseting state of each agent, calling method $play()$ and updating total scores for each player. After all games mean score for each agent is calculated.
- $play()$ method. Given two players it makes K moves and computes scores for each player.
- $f()$ method. It computes the value of $f(x) = \frac{10 \cdot e^x}{1+e^x}$.
- $payoff()$ method. It computes the value of $f(xK) - f(0)$.
- $Field$ class. This class is auxiliary, used for dealing with fields: updating and getting values.

IV. Agents

Before introducing each agents structure let us define some properties and functions that will be used below:

- $getLow()$ — returns number of field with lowest value of x.
- $getHigh()$ — returns number of field with highest value of x.
- $getMid()$ — returns number of field which is neither Low nor High.
- $isNotEmpty()$ — returns True if given field is not empty, zero otherwise.

A. Random Agent

Return some random field.

B. Random Non-empty Agent

Return some random field which is not empty.

C. Random High-Mid Agent

Return some random field out of $getHigh()$ and $getMid()$.

D. Greedy Agent

Returns field $getHigh()$

E. Copycat Agent

If opponents last field is not empty, move there. Otherwise, move randomly.

F. Probabilistic X Agent

Returns move depending on the probability distribution of X values. The more X value of the field is, the more chance that agent will move there.

G. Probabilistic Payoff Agent

Returns move depending on the probability distribution of payoffs ($f(xK) - f(0)$). The more $f(xK) - f(0)$ value of the field is, the more chance that agent will move there.

TABLE I
TABLE OF AGENTS CHARACTERISTICS

No.	Agent	*Average Score*
1	Random Agent	587.72
2	Random Non-empty Agent	806.75
3	Random High-Mid Agent	691.05
4	Greedy Agent	703.37
5	Copycat Agent	775.50
6	Probabilistic X Agent	804.62
7	Probabilistic Payoff Agent	413.14

V. TOURNAMENT OBSERVATIONS

Average scores per game are presented in the table 1. Actually, since several agents depends on random, these values could be different when running one more time. This scores also depends on the opponents strategy and if we will compare them in another tournament with other opponents, results could change significantly.

As we can see, the leaders are *Random Non-empty Agent* and *Probabilistic X Agent*. There is a intuition behind that: randomness is vague and it is hard to determine random strategy.

The losers are *Probabilistic Payoff Agent* and *Saimple Random Agent*.

VI. CONCLUSION

While performing this assignment:

- I have practiced with creating agents for games.
- I have practiced programming in Java.
- I have practiced writing reports in LaTeX.

ACKNOWLEDGMENT

Thanks to Innopolis University and course instructors for interesting and challenging tasks in education process.

REFERENCES

[1] Game Theory, Assignment II description (Spring 2020).
[2] Brown, Joseph & Ashlock, Daniel. (2011). Domination in Iterated Prisoner's Dilemma. Canadian Conference on Electrical and Computer Engineering. 1125-1128. 10.1109/CCECE.2011.6030637.
[3] Code in VCS https://github.com/russab0/MooseGame