Game Theory Assignment 2

Parth Kalkar (BS19-DS-01)
Dept. of Computer Science
Innopolis University
Innopolis, Russia
p.kalkar@innopolis.university

***Abstract*—The ability to solve game theory problems is extremely useful in a variety of settings. The game with two Moose and three territories is one of the most popular. This study outlines eight possible ways for deciding which field Moose will graze on. The outcomes of tournaments with strategies in place are displayed in this context.**

I. Task Description

The aim is to play a three-field simultaneous game with two participants. Each player controls a moose that eats grass from one of the three fields. The grass in the fields grows in response to X.

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. How does the grass grows

Vegetation growth is defined by the following function:

$$f(x) = \frac{10 * \exp x}{1 + \exp x}.$$

The game begins with X equal to 1 and ends with X equal to 0. If there were one or two moose on the field after each round, the value of X is reduced by one, otherwise it is increased by one. Because X is never less than 0, the growth function is exponential and always bigger than 0, implying that X and f(X) are proportionally rising.

B. How much did moose eat

Moose are territorial, and when they meet, they fight. As a result, the amount of vegetation eaten by one moose is dependent on whether or not there are other moose around.

1) Moose are in the same region: They battle and come up empty-handed.

2) Moose are in the different regions: Based on the formula below, both of them receive a certain amount of vegetation.

$$f(X_k) - f(0).$$

where k is the corresponding field

C. Input

The last move of the opponent is given to the agents, as well as the Xs values for each field. The first field is A, the second is B, and the third is C, and the values of X are xA, xB, and xC, respectively.

D. Output

As output agents, you must offer a number between 1 and 3 indicating the zone where your moose will consume in the next round.

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) *ParthKalkarCode.java* - an implementation of the Player interface that participates in the tournament.
3) *ParthKalkarTesting.java* - a test class that contains implementation of all created agents and tournament functions.

A. Main

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

B. Tournaments

1) Big tournaments: Depending on the number of individual agents in the competition, each strategy occurs many times. Two such functions were created, one that included all strategies and the other that only included greedy strategies.

2) Tournaments: In addition to the standard tournament with one sample of strategies, two functions, one with all strategies and the other with only greedy strategies, were implemented.

3) Sparring: Six functions are used to compare all of the greedy strategies to the final strategy.

C. Agents

Nine classes that implement Player interface with different strategies

III. Strategies

To begin, various rudimentary strategies were constructed to provide a baseline against which smart strategies could be compared.

To create the *ParthKalkarCode.java* class, three types of greedy strategies were implemented, and the last two greedy techniques were integrated.

A. Random

Random agent always picks the next field by the following random formula:

$$random.nextInt(11)\%3 + 1.$$

After taking modulus of 3, the bound value is 11 to have equal probability of getting 0, 1, and 2, and after adding 1, the outcome is a random value from 1 to 3.

B. Static

The static agent starts at random using the random strategy's formula, but then stays in the same spot until the finish.

C. Sequential

The sequential agent begins at random using the random strategy's formula, and each number in the next picked field is increased by one. When the previous move is 3, the next move is 1.

D. CopyMoose

Because the CopyCat strategy was addressed in the lab sessions, it is included here as well. The approach is to use the previous opponent's move as the starting point for the next move.

E. Greedy

The greedy algorithm compares all X values and moves to the field with the highest X. The problem with this strategy is that when there are many maximums, the agent chooses the one with the lowest number; hence, greedy strategies can have a lot of collisions when they compete against one other.

F. Random Greedy

The problem of the ordinary greedy strategy is solved by the greedy random approach. As a result, this technique looks for maximums and, if there are multiple, chooses one at random.

G. Smart Greedy

The smart greedy approach is the first to analyse the opponent's plan and attempt to comprehend it. What matters to a smart greedy agent is determining whether or not the opponent is greedy. If the opponent is greedy, smart greedy picks between the first and second maximums at random. There is no need to not choose maximal if the opponent is not greedy, thus the agent chooses the field with the highest X.

The agent keeps two numbers in order to comprehend the opponent's greediness: the number of greedy moves made by the opponent and the number of rounds played; the ratio of these two values is the greediness coefficient. The greedy move is one in which the opponent choose the maximum number of options. When the opponent's greediness coefficient exceeds 0.5, the opponent is greedy.

H. Advanced Greedy

Smart greedy technique is improved in two ways by advanced greedy strategy. First way is to add one more category to the opponents variations, so now opponent can be greedy (if greedy coefficient is greater than 0.6), not greedy (if greedy coefficient is less than 0.4) and smart (if g.c $\geq$ 0.4 and g.c $\leq$ 0.6). If opponent is greedy then choose second maximum always, If your opponent is greedy, always choose the second maximum; if your opponent isn't greedy, always choose the first maximum; and if your opponent is smart, choose first or second maximum at random.

The second enhancement is to avoid selecting the second maximum if it is zero, even if random suggests it. Going to the field with X equal to 0 and allowing the opponent to score more points makes no sense. It is preferable to wait at least one round until both players have a score of 0.

I. ParthKalkarCode

After putting all of the tactics to the test, it was discovered that the Random Greedy strategy works practically every time, but Advanced Greedy performs worse than Smart Greedy. As a result, Smart Greedy and Advanced Greedy were merged to create a strategy that outperforms random. The procedure for categorising opponents is identical to that used in the Smart Greedy approach (opponent can be greedy or not greedy). The technique of selecting the second maximum is similar to that of the Advanced Greedy strategy (if the second maximum is zero, select the first maximum).

IV. Tests

Several tournaments were developed to compare and contrast different strategies.

A. Big tournament with all strategies

After running a tournament with all strategies appearing 10 times (9 * 10 = 90 agents in total), the following results were gotten:

RandomMoose	134781.43327041718
ParthKalkarCode	193593.93166207106
SequentialMoose	117419.33986996024
RandomGreedyMoose	186847.0764913932
GreedyMoose	149256.62269478381
StaticMoose	1349.3820991992295
CopyMoose	24437.462177882302
SmartGreedyMoose	187335.61112976598
AdvancedGreedyMoose	170922.0705247184

The surprising result is that Random Greedy performs better. It is worse than the combined form of Smart and Advanced Greedy. The first four methods were as expected.

As it was expected first four strategies works not so well as greedy ones and just random strategy is almost as good as usual greedy strategy.

B. Big tournament with greedy strategies

It's quite unlikely that techniques like random, static, copy moose, or sequential will be used in the ultimate tournament amongst all students' agents. As a result, another large tournament was constructed to see how greedy strategies interact with one another. The competition features 20 agents from each type of approach, for a total of 20 * 5 = 100 competitors.

ParthKalkarCode	353069.0355365165
RandomGreedyMoose	346963.2360964782
GreedyMoose	272424.51380382053
SmartGreedyMoose	335420.46229123994
AdvancedGreedyMoose	262530.7866702693

Because the combined strategy of smart and advanced greedy performs better in the competition than random greedy, the final submission file includes the code for this approach. Following that, six tournaments were established using a combination of greedy and *ParthKalkarCode* agents, and the final approach outperformed other greedy strategies.

V. CONCLUSION

Finally, greedy strategies perform well in the given task; nevertheless, the variation of greedy score can be smaller or bigger. Although random greedy has a high overall score in tournaments, the results vary from execution to execution. At the same time, the final submission strategy has a significantly higher score than the others. url same

VI. DEMO

Video demonstration link - www.shorturl.at/fsDIJ

REFERENCES

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