Programming Assignment

Game Theory

March 10, 2022

Notes

Teams

This is a team assignment. Odd numbered teams should attempt Question 1 while even numbered teams should attempt Question 2.

Grading

The code will be evaluated automatically on a number of test cases and marks will be awarded accordingly.

In order to help in verifying your code, you may submit your own custom test cases to the TA along with your output and the TA will check it against the model solution. (You can message Sanjay Chandlekar on Teams)

Programming Languages

You may attempt the assignment in C/C++ or Python.
You may not use any external libraries unless explicitly permitted by the TAs.

Plagiarism

Please do not copy code from your friends or online sources. Plagiarism cases may be penalized.

Submission Format

You may upload exactly one file containing the source code with the file name as `<Team_No>.c` for C Programs or `<Team_No>.cpp` for C++ Programs or `<Team_No>.py` for Python Programs. There should be only one submission per team.

Deadline

- Soft Deadline: 18$^{\text{th}}$ March 2022 **11:55 PM**
- Hard Deadline: 20$^{\text{th}}$ March 2022 **11:55 PM** (with 15% Penalty)

1 Find PSNE and Very Weakly Dominant Strategies for a n-Player Game

Given a n-Player Game list all Pure Strategy Nash Equilibria and list all Very Weakly Dominant Strategies for each player.

1.1 Input Format

The input is a n-Player Game with the payoffs listed in the NFG Format (as described in the Gambit Project).

- First line contains the number of players n.
- The second line contains n space-separated numbers, the i^{th} number corresponding to the number of strategies available to the i^{th} player. (S_i)
- The third line contains the list of payoffs in the NFG Format.

1.2 Output Format

- First line should contain the number of PSNE. (n_{psne})
- Followed by n_{psne} lines, the i^{th} line containing n space-separated numbers corresponding to the equilibrium strategies for each player respectively.
- Next, you should output n lines, with the i^{th} line listing the number of very weakly dominant strategies for the i^{th} player followed by the dominant strategies.

1.3 Constraints

- $n \times \prod_{\forall i} |S_i| \leq 10^6$ where $|S_i|$ is the number of strategies available to the i^{th} player.
- Time Limit: 60 seconds per test case
- Memory Limit: 1 GB

1.4 Sample Test Cases

Input 1

```
2
2 2
-2 -2 -2 -10 -10 -2 -5 -5
```

Output 1

```
2
1 1
2 2
1 2
1 2
```

Input 2

```
2
3 3
6 6 10 0 8 0 8 20 5 5 20 0 0 8 2 8 4 4
```

Output 2

```
1
3 3
0
0
```

Input 3

```
2
4 4
5 2 0 0 7 0 9 5 2 6 3 2 2 2 1 3 1 4 2 1 1 5 0 2 0 4 1 1 5 1 4 8
```

Output 3

```
1
2 2
0
0
```

2 Find Nash Equilibria of a Bimatrix Game

Given a Bimatrix Game, find Nash Equilibria of the Bimatrix Game. If there are multiple Nash Equilibria, output in any order.

2.1 Input Format

The input is a *non-degenerate* bimatrix game with the payoffs listed in the NFG Format (as described in the Gambit Project).

- First line contains the number of strategies for the row player. (R)
- Second line contains the number of strategies for the column player. (C)
- The third line contains the list of payoffs in the NFG Format.

2.2 Output Format

- The first line must contain the number of equilibria. (n). You are required to output only $R + C$ equilibria if they exist.
- The next $2n$ lines must contain the description of the n equilibria as follows:
  - $2i + 1^{\text{th}}$ line contains the R floating point numbers $\in [0, 1]$ denoting the probability that the row player should play the R strategies respectively.
  - $2i + 2^{\text{th}}$ line contains the C floating point numbers $\in [0, 1]$ denoting the probability that the row player should play the C strategies respectively.

2.3 Constraints

- $R \times C \leq 25$
- Time Limit: 60 seconds per test case
- Memory Limit: 1 GB

2.4 Sample Test Cases

Input 1

```
2
2
-2 -2 -1 -10 -10 -1 -5 -5
```

Output 1

```
1
0.0 1.0
0.0 1.0
```

Input 2

```
2
2
2 1 0 0 0 0 1 2
```

Output 2

```
3
1.0 0.0
1.0 0.0
0.0 1.0
0.0 1.0
0.6666666666666666 0.3333333333333333
0.3333333333333333 0.6666666666666666
```

2.5 A Note on the Lemke-Howson Algorithm

The Lemke-Howson Algorithm is a well-known algorithm to find the Nash Equilibrium of a two-player non-degenerate game. A high-level description of the algorithm on an input bimatrix game $\in \mathbb{R}^(m \times n)^2$ is as follows:

1. Let P and Q be two Polytopes representing the best response having dimensions m and n respectively. A polytope is nothing but a generalization of a polygon in n-dimensions. The faces of this polygon correspond to the inequalities characterizing the players' best response.

2. The Theorem due to Lemke and Howson states that if $u \in P$ and $v \in Q$ are vertices such that they are not at their respective origins, then $L(v, w) = \{1, \ldots m + n\}$ iff (v, w) represent a Nash Equilibrium.

3. You start at the origin (the Theorem is not trivially satisfied) so you drop a label from this set (make that inequality strict) and chose another vertex by moving from whatever vertex you are currently at.

4. If the Theorem is satisfied, then you have a Nash Equilibrium.

You can chose to output this Nash Equilibrium, or you can drop any label and move to another vertex to find the next Nash Equilibrium until you repeat a vertex. In this case you can restart the entire algorithm to drop a new label from your initial set and discover more Nash Equilibria.

You can find at most $m + n$ Nash Equilibria by doing this.

2.5.1 Interesting observations

After looking at the the Theorem due to Lemke and Howson, one might be mistaken to think that this procedure discovers all Nash Equilibria. However, this is not the case, there can in fact, be equilibria that are completely missed by our "traversal" through the polytopes. One example of this case would be,

Input 3

```
3
3
0 0 2 3 3 0 3 2 2 2 0 0 0 3 0 0 1 1
```

I leave it up to the interested reader to figure out how Nash Equilibria in this game could be missed by the aforementioned procedure :-)