Assignment 2: Solving simple games on networks

Multi agents search

The 2nd stage towards creating your final project, at the end of the semester, uses a simple synchronous search to solve two types of very simple games on networks (GoNs). You are required to write a program that solves two types of simple games on randomly generated networks. You can modify the code from the first assignment to generate those games.

As in all assignments, the agents are implemented by threads, and the Mailer object that sends and receives messages between Agents is the same. In other words, your program solves a GoN by a distributed algorithm. There are two games – the Prisoner's Dilemma (PD) and the Battle of the Sexes (BoS) – both games have 2x2 matrices that are given below and that define the games *among all pairs of neighbors/agents*. The user's input defines the games to be solved (one of two) and the density of the network to be generated, which is the same as in the first assignment, the probability for a pair of agents to be connected p_1.

The input to the program includes the number of agents ***n***; the parameter defining the density probability for random networks p_1; and the type of game among agents. When the game selected is BoS, an additional parameter is needed, the fraction of the players of type Wife of all the ***n*** players.

Please remember that on a network each agent gains the sum of its gains in the games it plays against its neighboring agents.

The two games:

The prisoner's dilemma game is defined by the following matrix. Each player of the two has two strategies it can choose from: Defect (D) and Cooperate C. The gain on the left of each square is for the left hand side (LHS) player and the gain on the right of each pair in a square is the gain of the right hand side (RHS) player.

	Defect	Cooperate
Defect	5, 5	10, 0
Cooperate	0, 10	8, 8

In the game of Battle of the Sexes (BoS) each player has two options of selecting its choice of going out, creating two strategies: Theatre (T) and Soccer (S). One of the players represents the Wife and she prefers the Theatre (T) and the other player represents the Husband, who prefers Soccer (S). However, both of them prefer to go together, creating a situation in which choosing differently results in no gain for both. But, the gains for the each agent/player from choosing either T or S are different, due to their different nature.

		Husband	
		Theatre	Soccer
Wife	Theatre	3, 1	0, 0
	Soccer	0, 0	1, 3

Comment: The specific number values of the gains in the specific matrices above can be different in versions of the same two games, but the general structure is the same.

The solving algorithm: Best Response

Best response is perhaps the most well known (and simple) algorithm for solving standard Games on Networks (GoS). The agents play in turns in a fixed order on the network and each agent in its turn selects a strategy that "best responds" to the selection of the player(s) it plays against.

Take for example the PD game. If one player selected the strategy Cooperate (C), then for the other player its best response is Defect (D). Incidentally, for the PD game, the best response is D also if the other agent selected D. ***Remember, an agent/player computes its best response strategy to all the agents it plays against (its neighbors).*** If an agent has already selected its

best response, it does not change its selection and if it selected another strategy before, it changes its selection to the best-response.

Each agent in its turn selects a strategy and sends a message that carries its selected strategy to all its neighboring agents. The agents are ordered in a fixed order, by their IDs, as in the former assignment. Agents run in this fixed order. The first agent sends a message to all its neighbors with its initial strategy selection, which is selected by it randomly. From now on, the solving algorithm works in turns (rounds) of all n agents. ***In each turn (round) every agent selects its best response to the selections of the last round, of all its neighbors.***

The algorithm terminates with a solution when all agents choose as their best response ***not to change their strategy for a complete round.***

Performance measures:
Since this is a solving algorithm that can return its solution, its performance measures are global:
1. the total gain of all agents (the global Social Welfare – SW).
2. the total number of rounds.

Agents' implementation:
These are already the second search agents that you are implementing, and very few additional data structures are needed for their implementation.

1. strategy/value – the current strategy of the agent.

2. AgenView – the assignments of agents it is playing against (its neighbors).

3. Agent's Gain – the total gain of the agent with its current strategy.

4. The list of neighboring agents.

Test-runs:
Your program is required to perform test-runs of the search as follows:
1. Extract command-line parameters **n, PD/BoS, p_1.**
2. Generate ***100*** random problems with these parameters.
3. Run the ***Best-Response*** algorithm to solve each of the generated problems, accumulate the values of the parameters (Num_Iterations, SW) over ***100*** runs and report the average of each parameter.

Submission:

Your code is required to generate random problems, run the ***Best-Response*** algorithm on the generated problems and report the running performance (i.e., the measured parameters) averaged over ***100*** runs.

Due date – Sunday, ***September 10****, 2023*

Enjoy !!!