Last modification: February 26, 2023

# RBE595/CS525: Swarm Intelligence
# Spring-Term 2022/2023
# Homework 7

## Exercise 1: Group Size Detection [70 points]

In this homework, you are going to implement a simple algorithm for calculating a *very rough estimate* of the size of a group of agents in a completely decentralized manner. This algorithm is called *quorum sensing* and it is used by certain bacteria to know when to launch an attack on the body of the host. The algorithm works as follows:

- The agents are the cells of a $W \times H$ grid. Consider the cases $5 \times 5$, $10 \times 10$, $20 \times 20$.
- The agents do not know how many they are (that's what they need to estimate!), nor their position in the grid.
- The simulation proceeds in steps in the same fashion as the synchronization algorithm in HW6.
- At any time, the agents can be either *susceptible* or *refractory*.
- At any time, a susceptible agent has a probability $P$ of initiating a "signal wave" by emitting a signal. An agent can initiate a signal only once throughout the duration of an experiment.
- Upon receiving a signal, any susceptible neighbor emits a signal too, thus continuing the "signal wave".
- Any time it emits a signal (initiated or forwarded), the agent enters the *refractory* state. In this state, the agent ignores any received signal and cannot initiate a signal. This state lasts for $R$ steps, after which the agent swicthes back to *susceptible*.
- Any time it emits a signal (initiated or forwarded), the agent increases by 1 its estimate of the group size. The group size is initialized to 0 for all agents.
- In principle, the simulation finishes when all the agents have signalled. However, the agents have no way to know this; therefore, any agent considers itself "done" when it has received no signal for $1/P$ continuous steps. When all agents consider themselves "done", the simulation ends.

The pseudocode of the algorithm can be formalized as follows:

```
input:
  W = grid width [int]
  H = grid height [int]
  P = initiation probability [float]
  R = refractorytimer [int]
```

```
init:
  initiated = false [boolean]
  size = 0 [int]
  state = Susceptible

step:
  if (state == Susceptible)
    if (neighbor signalled)
      emit signal
      state = Refractory
      refractorytimer = R
      size = size + 1
    elif (not initiated) and (random() < P)
      emit signal
      state = Refractory
      refractorytimer = R
      initiated = true
      size = size + 1
    endif
  else
    refractorytimer = refractorytimer - 1
    if (refractorytimer <= 0)
      state = Susceptible
    endif
  endif
```

1. Implement the above algorithm in a language of your choice, as long as it's Python.
2. Look for the value of $P$ and $R$ that produces the best estimate across different grid sizes.
3. Does it matter if we use 4-distance or 8-distance?
4. This algorithm is obviously very inaccurate. How would you make this algorithm better? You can propose a modification of this algorithm, or a completely different algorithm.

The latter 3 points should be discussed in a short 2-page report.

## Exercise 2: Voting [30 points]

Voting systems based on majority rule are susceptible to strategic agenda-setting. Let's explore how one might do this on some basic examples.

1. Suppose there are four alternatives, named $A$, $B$, $C$, and $D$. There are three voters who have

the following individual rankings:

$$B \succ_1 C \succ_1 D \succ_1 A$$
$$C \succ_2 D \succ_2 A \succ_2 B$$
$$D \succ_3 A \succ_3 B \succ_3 C$$

You're in charge of designing an agenda for considering the alternatives in pairs and eliminating them using majority vote, via an elimination tournament in the style of the examples shown in Slide 14 (at PDF page 19) of Lecture 11.

(a) You would like alternative $A$ to win. Can you design an agenda (i.e., an elimination tournament) in which $A$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

(b) You would like alternative $B$ to win. Can you design an agenda (i.e., an elimination tournament) in which $B$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

(c) You would like alternative $C$ to win. Can you design an agenda (i.e., an elimination tournament) in which $C$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

(d) You would like alternative $D$ to win. Can you design an agenda (i.e., an elimination tournament) in which $D$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

2. Now, consider the same question, but for a slightly different set of individual rankings in which the last two positions in voter 3's ranking have been swapped. That is, we have:

$$B \succ_1 C \succ_1 D \succ_1 A$$
$$C \succ_2 D \succ_2 A \succ_2 B$$
$$D \succ_3 A \succ_3 C \succ_3 B$$

(a) Can you design an agenda in which $A$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

(b) Can you design an agenda in which $B$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

(c) Can you design an agenda in which $C$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

(d) Can you design an agenda in which $D$ wins? If so, describe how you would structure it; if not, explain why it is not possible.

## Deliverables

The usual deliverable instructions are in order:

Submit an archive called `LastnameFirstname.zip` with the following structure:

```
LastnameFirstname/
  ex1.pdf           (MUST be a PDF!)
  ex2.pdf           (MUST be a PDF!)
  README.txt        (a plain-text file that describes how to run your code,
                     along with the dependencies)
  <your code files>
```