

Adaptive Systems – Spring 2009

COURSEWORK – IN-LAB-TEST

17th March 2009

Instructions

- Create two new StarLogo projects CourseWork1 and CourseWork2, one for each of the problems defined below.
- The StarLogo home page is: <http://education.mit.edu/starlogo/>
- Submit your solution through Blackboard (Assignments), upload your two StarLogo projects: CourseWork1.slogo, CourseWork2.slogo

**There are two questions.
Questions are 50 marks each.
Answer all questions.**

This test starts at 18h30 and ends at 20h30.

Plagiarism Warning

Students are reminded that the School and the College apply a very strong policy in case of plagiarism. Helping other students or taking advantage of help from other students would be considered as acts of plagiarism.

Any similarity will be clearly identified and carefully checked.

The lighter penalty in case of plagiarism for those involved will be a mark of 0 for the **whole** test.

Students, who will be caught during the test while communicating with other students by any means, will be asked to leave the room immediately, and will be given a mark of 0.

Mobile phones are not allowed.

Questions

Problem 1: (Simple) Gossiping (50 Marks)

In this system, there are mobile nodes (e.g mobile phones) who exchange information. We represent mobile nodes as squares. When two mobile nodes meet, they update the information they keep by the **average** of their respective information. They then use this information to change their colour.

You should observe that the colour of the nodes converges towards the same value.

To do:

1. Create and place randomly mobile nodes with a square shape.
2. Define a Turtle variable `information`. Initialise `information` with a random value between 0 and 100 and use this value as the colour of the mobile node.
3. Define a slider defining the total number of mobile nodes in the system.
4. Create a histogram chart showing the value `information` for all mobile nodes. Do not forget to complete the chart X and Y axis labels.

As during the labs:

- Create an Observer procedure setup that initialises your system (points 1 and 2 above), and its corresponding button.
- Create a Turtle procedure `gossip` that defines the main behaviour of a mobile node: it moves around, when it encounters another mobile node (**at the same place on the grid**), it averages its own `information` and the one of the other mobile node, updates its own `information` with this value and changes its colour to that value.
- Create a Turtle procedure `go` and its corresponding button that starts the gossiping behaviour.

Useful StarLogo Commands (that you may use)

- **Turtle commands**
 - `setc colour` changes the colour of the turtle to *colour*
 - `random number` returns a number between 0 and *number*
 - `setinformation value` allows to set the `information` variable of the current turtle to *value*
 - `information-at xcor ycor` returns the value of variable `information` for the turtle that is *xcor* units away in the X direction and *ycor* units away in the Y direction from the current turtle.
 - `round value` returns the integer value closest to *value*
- **Observer commands**
 - `ask-turtles [list of commands]` allows the Observer to have turtles performing the specified `[list of commands]`

Problem 2: Cells (50 Marks)

In this system, there are three types of cells: Producers, Consumers and Cleaners. Producers cells regularly produce food for the Consumers cells. Consumers permanently look for food. As soon as they find food, they eat it. Once eaten, what remains of the eaten food is cleaned (removed) by the Cleaner cells.

In this system, we represent Producers, Consumers and Cleaners as three different families of turtles. Produced food is represented by green patches. Eaten food, not yet cleaned, is represented by orange patches.

To do:

1. Create and place randomly the Producers, Consumers and Cleaners (three families of turtles with shapes of bubbles, enzyme and h2o2 respectively). You can also use the shapes you want provided they are different for each family.
2. Define 3 sliders, one for each family. Use them to control the number of Producers, Consumers and Cleaners in your system.
3. Create a line chart plot showing the progression of the number of green and orange patches.
4. Create a line chart plot showing the progression of the Producers, Consumers and Cleaner cells. They should remain stable during the simulation.
5. Do not forget to complete the charts X and Y axis labels, and lines labels.

As during the labs:

- Create an Observer procedure setup that initialises your system (point 1) and its corresponding button.
- Create a Turtle procedure produce-food that defines the main behaviour of Producers: they move around and regularly (**not at each step**) produce food (green patches).
- Create a Turtle procedure consume-food that defines the main behaviour of Consumers: they move around looking for food (green patches), they eat them (turn them orange).
- Create a Turtle procedure clean-food that defines the main behaviour of the Cleaners: they move around looking for orange patches and remove them.
- Create an Observer procedure go-Producers (and its corresponding button) that starts the Producers, an Observer procedure go-Consumers (and its corresponding button) that starts the Consumers, and an Observer procedure go-Cleaners that starts the Cleaners.

Extra [10 marks]

- Food gives consumers increased energy, while roaming to find food takes away their energy. Use an extra variable energy for the turtles.
- Modify the Observer setup producer so that Consumers have an initial energy set up at random (between 0 and 20).
- Modify the Turtles procedures accordingly so that eaten food increases the energy (e.g by 10), while moving around for finding food decreases the energy (e.g. by 1 at each hop). If energy is too low (e.g below 3), the Consumer cell dies.

Useful StarLogo Commands (that you may use)

- **Patch commands**
 - `setpc colour` change the colour of the patch to *colour*
- **Turtle commands**
 - `setc colour` changes the colour of the turtle to *colour*
 - `stamp colour` change the colour of the patch (under the turtle) to *colour*
 - `repeat number [list of commands]`
performs [list of commands] as many times as specified by *number*
 - `die` to make a turtle die
- **Observer commands**
 - `ask-turtles [list of commands]` allows the Observer to have turtles performing the specified [list of commands]
 - `repeat number [list of commands]`
performs [list of commands] as many times as specified by *number*

**Do not forget to submit your Coursework through Blackboard
(Assignments)**