Lab Verification Test 2

This lab verification test requires the following equipment:

a) Matlab

b) Webots

The duration of the test is 3 hours. It consists of two parts: (i) Sensor Networks, (ii) Particle Swarm Optimization. Each part consists of 60 points. The maximum score you can get is 120 points; you will be awarded the maximum grade if you obtain 100 or more points; your potential bonus of points above 100 will be integrated in the overall weighted sum for the course grade.

The test is open book, i.e., all printed/written material is allowed (e.g., books, lecture notes, exercises, solutions, personal notes). In addition, you will use a computer from the computer room running Ubuntu, as you did in the labs. You can use internet to look for the necessary information**, but never to communicate with other students or anyone outside the computer room. Note that we will log all the communications and connections that your computer will create. Before starting, please log out and close all e-mail and messaging services (Gmail, Google Docs, Yahoo, EPFL mail, Facebook, etc.). You are NOT allowed to use your personal computer, tablet, phone or any other digital device.**

Getting Started

To start with this test, you will need to download the material available on Moodle. Download `test.tar.gz` and extract it in your home directory (you can type `tar xvzf test.tar.gz`). The uncompressed folder *test* has the following contents:

- *DIS_17-18_test_answer_sheet.odt:* LibreOffice file to be filled with most of your answers.
- *part1:* Folder with files needed to answer Part I.
- *part2:* Folder with files needed to answer Part II.

Submitting your answers

Answers must be written on the provided answer-sheet file (*DIS_17-18_test_answer_sheet.odt*), and in a couple of different source files that you will need to complete along the test. In total, **6 files** must be uploaded to Moodle by the end of the test. Make sure that your codes compile before submitting them, otherwise they cannot be graded. You can upload each source file as soon as you are finished with it. Do not forget to submit the answer sheet *DIS_17-18_test_answer_sheet.odt* on Moodle once you finished the whole test.

You will have to upload to Moodle the following files:

1. *DIS_17-18_test_answer_sheet.odt*: with answers for all the parts.
2. `guided_con.c` : Webots controller for Part 1.1.
3. `guided_gradient.c` : Webots controller for Part 1.1.
4. `pso_coop_sup.tar.gz` : Webots supervisor for Part 2.1.
5. `S13_results_new_strategy.txt` : Logs of the simulation results for Part 2.1.
6. `S13_fitness_plot.png` : plot of the simulation results for Part 2.1.

Important Notes:

1. Do not answer on this sheet!

2. Please double-check your submitted solution files.

3. Make sure that your codes compile before submitting them, otherwise they cannot be graded.

Part 1: Sensor Networks (60 points)

As you saw in the part 4 of Lab 10, adding mobility to a sensor network offers the advantage of being able to sample at multiple locations with the same sensor node. Further, you used controlled mobility to adjust the behavior of the robots based on the environmental situation.

In the following exercise, you will use a group of robots to find the location where a scalar field is maximum.

1.1. Controlling a Mobile Sensor Network (33 pts)

Arena description: The arena contains four e-puck robots and a single light source at a fixed location (unknown to the robots). The robots are equipped with a light sensor, and can communicate with each other (global neighborhood). At every time step, each robot broadcasts its position and light sensor measurement.

Task: Design a controller that can guide the e-pucks in order to find the location where the light intensity is maximum. Obviously, this location coincides with the position of the light source.

Performance metric / error: Distance between location of the maximum light intensity found by the robots and the actual position of the light source. This value will be lower for a better controller.

Open the world file `controlled_mobile_net.wbt` (located in `test/part1/webots/worlds/`). Open the controller `guided_con.c` (located in `test/part1/webots/controllers/guided_con/`).

S_1 (4 pts): Compile the controller code for the robots (`guided_con.c`) and also the supervisor (`supervisor_controller.c`). Run the simulation. Note that the simulation stops if the robots find a good result, or after 150 seconds.

Write down the location of the maximum found light intensity, error in this result, and time taken to obtain this result. These are printed in the console by the supervisor controller at the end of the simulation.

Q_2 (2 pts): Explain what the robot controller is doing. Look at the code between the lines 280-285 and 290-295 in `guided_con.c`.

I_3 (8 pts): In the same controller file (`guided_con.c`) (located in the directory `test/part1/webots/controllers/guided_con/`), implement an improved robot controller where the velocity of the robot is computed as the weighted sum of vectors pointing to the neighborhood best, personal best and a random walk vector, similar to what you did in lab 10.

The part of the code to modify is marked with `TODO` (Around lines 285-290).

Mention the following in your answer sheet:

- Results (location of maximum, error, time taken, as printed in the console).
- Paste your additions / modifications to the code in the answer sheet.

Note: Also save and **submit** your solution code **`guided_con.c`**.

Q_4 (4 pts): Briefly explain what will happen if the random walk component is removed in the question above. Further, knowing that there is only one global optimum and no local optima, and assuming that the neighborhood is NOT global, what would you change in the question above to make your solution simpler?

I_5 (10 pts): Open the world `controlled_mobile_net_gradient.wbt`. It is same as the earlier world except that the robots use the controller `guided_gradient.c`. Here, we will compute the

gradient of the light field based on the spatial measurements we get from other robots, and use that to compute the direction of motion of the robot.

Let s_j be the light sensor measurement of robot j.

For a particular robot i, we will compute the gradient with respect to measurements of other robots as

$$\nabla f_j = \left(\frac{s_j - s_i}{x_j - x_i}, \frac{s_j - s_i}{y_j - y_i}\right), j \in \{0,1,2,3\}, j \neq i$$

Additionally, we will compute the gradient with respect to the personal best positon as

$$\nabla f_{sb} = \left(\frac{s_{selfbest} - s_i}{x_{selfbest} - x_i}, \frac{s_{selfbest} - s_i}{y_{selfbest} - y_i}\right)$$

Note that ∇f_j and ∇f_{sb} are 2D vectors.

Then, we will compute our desired direction of motion $\hat{d}$ (a normalized vector) as

$$\vec{d} = \nabla f_{sb} + \sum_{j \neq i} \nabla f_j, \qquad \hat{d} = \frac{\vec{d}}{\|\vec{d}\|}$$

You need to implement the following in the code. The part of the code to modify is marked with `TODO-1` and `TODO-2`.

- Implement the computation of the quantity ∇f_j (see `TODO-1` in code, lines 254-260)
- Implement the computation of the quantity ∇f_{sb} (see `TODO-2` in code, lines 280-286)

Mention the following in your answer sheet:

- Results (location of maximum, error, time taken, as printed in the console).
- Paste your additions / modifications to the code in the answer sheet.

Note: Also save and **submit** your solution code **`guided_gradient.c`**.

Q_6 (5 pts): Do you expect your controller in **I_5** to work better than the one in **I_3**? Why so? Which one *actually* works better? Why so?

1.2. Modeling a Mobile Sensor Network (27 pts)

Assume that in addition to having controlled mobility as described in Part 1.1., the robots vary their sampling frequency as well. By default, the controller samples at a high frequency of f_H . When a robot detects at least 2 robots nearby, it starts sampling at a lower frequency f_L until it has less than 2 robots nearby or a deterministic time out of duration T_d is elapsed. It then switches back to higher sampling frequency f_H.

Let n be the number of nearby robots detected by a particular robot. The following quantities are given to you.

$f\_H =$ default higher sampling frequency

$f\_L =$ lower sampling frequency when $n \geq 2$

$T\_d =$ timeout (number of time steps) after which the robot switches back to f_H

$p\_r =$ geometric detection probability, calculated as $\frac{\text{Area of detection}}{\text{Total area of arena}}$

p_2 = probability that 2 **or more** robots are detected by a particular robot, i.e., $n \geq 2$

$N\_L(k)$ = Number of robots in the low frequency state at time step k

$N\_H(k)$ = Number of robots in the high frequency state at time step k

N = Total number of robots

Note that state transition of a particular robot solely depends on detecting other robots, not on the states of the other robots.

Q7 (5 pts): Let us begin with a microscopic model. Given is a PFSM describing the behavior of an individual robot. Label the states and transition conditions.

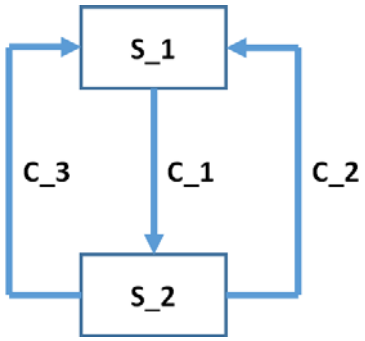

Figure 1: The PFSM for one robot.

a. (1 pt): S_1 = …

b. (1 pt): S_2 = …

c. (1 pt): C_1 = …

d. (1 pt): C_2 = …

e. (1 pt): C_3 = …

Q8 (6 pts): Let us now move on to a macroscopic model. Derive an expression for the following quantities (in terms of $N\_L(k)$, $N\_H(k)$ and other given quantities)

a. (2 pts) $D\_L(k)$, average number of robots switching to lower sampling frequency f_L at time step k on detecting 2 or more robots. *(Hint: p_2 is assumed to be known as mentioned above)*

b. (2 pts) $D\_H(k)$, average number of robots switching to the higher sampling frequency f_H at time step k on not finding 2 robots nearby.

c. (2 pts) $D\_T\_H(k)$, average number of robots switching to the higher sampling frequency f_H at time step k due to the timeout T_d.

Q9 (6 pts): Formulate a difference equation that describes the dynamics of the population of the robots sampling at high frequency. Use the terms D_L(k), D_H(k), D_T_H(k) from the previous question even if you have not answered them.

$$N\_H(k+1) = \cdots$$

Q10 (10 pts): Derive an expression for the following terms.

a. (5 pts) p_R, Probability that a particular robot detects exactly one other robot.

b. (5 pts) p_2, probability that 2 **or more** robots are detected by a particular robot.

Part II: Particle Swarm Optimization (60 points)

2.1 Learning cooperative tasks (40pts)

Consider the cooperative scenario studied in Lab 9. The task is a coordinated motion, where two e-puck robots must travel as far as possible while remaining in each other's sensor range. We use PSO to learn the optimal controller.

Open the world `test/part2/webots/worlds/pso_coop.wbt` in Webots, as well as the supervisor code `test/part2/webots/controllers/pso_coop_sup/pso_coop_sup.c`. Take a look at the supervisor code `pso_coop_sup.c`. As you saw in Lab 9, the solution sharing strategy implemented in this code is *group-public (heterogeneous)*. The drawback of this strategy is being non-scalable.

Q_{11} (10pts): We are looking for another solution sharing strategy to avoid this problem of non-scalability.

- **a.** (4pts): Which solution sharing strategy would be the best for this scenario? Justify your answer.
- **b.** (6pts): Compare your proposed strategy with the group-public (heterogeneous) strategy, in terms of implementation and performance.

I_{12} (20pts): Make all the necessary modifications in the files of the folder `pso_coop_sup` in order to implement your proposed strategy. Briefly mention and explain in your answer sheet all the modifications you make in the code, while mentioning the line number. Compress the folder `pso_coop_sup` to `.tar.gz` and **submit it on Moodle**.

S_{13} (10pts): Run the simulation in fast mode so it finishes as soon as possible. You can work on Part 2.2 while waiting for the simulation to finish. Once it is finished, revert the simulation first, then open the folder `test/part2/matlab/` where all the results are stored in the file `S13_results_new_strategy.txt`. Run the Matlab script `plot_fitness.m` in order to plot the value of the fitness over time.

In your answer sheet explain and justify the influence of your proposed solution sharing strategy on the fitness value compared to the public-group (heterogeneous) strategy.

The Matlab code saves the plot with the name `S13_fitness_plot.png` in the folder `test/part2/matlab/`. Submit both files `S13_fitness_plot.png` and `S13_results_new_strategy.txt` on Moodle. Do not copy-paste the picture in the answer sheet.

Hint: *if you need the simulation to finish faster, you can reduce the number of iterations in* `pso_coop_sup.c` *line 25, but the results that you submit should be done with 20 iterations.*

2.2 Robotic learning (20pts)

Q_{14} (10pts): In any iterative machine learning approach with a stochastic component, such as PSO, there is always a tradeoff between exploration and exploitation of the results already found. In PSO, what are the parameters that manage this tradeoff? Describe the effect of each parameter and include any equations that may be useful.

Q_{15} (10pts): Consider the task of learning obstacle avoidance in Webots using PSO. E-puck robots can either move forwards or backwards in the arena. However, robots are better able to avoid obstacles going forwards, as they have more proximity sensors to see what they are approaching. As a result, the controllers where the robot moves forwards give a slightly better fitness overall compared to the controllers where the robot moves backwards. In such scenario, considering the topology of the fitness function, would a global or local neighborhood be preferable for the particles? Justify your answer.