Lab Verification Test 1

This lab verification test requires the following equipment:

a) C compiler

b) Matlab

c) Webots

The duration of the test is 3 hours. It consists of two parts: (i) Ant Colony Optimization, (ii) Single and Multi-Robot Navigation. 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_15-16_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 Webots files needed to answer Part II.

Submitting your answers

Answers must be written on the provided answer-sheet file (*DIS_15-16_test_answer_sheet.odt*), and in a couple of different source files that you will need to complete along the test. In total, **eight files** must be uploaded to Moodle by the end of the test. You can upload each source file as soon as you are finished with it, but you should wait until you finish the whole test to upload *DIS_15-16_test_answer_sheet.odt.*

You will have to upload to Moodle the following 8 files:

1. *DIS_15-16_test_answer_sheet.odt*: with answers for all the parts.
2. tsp_i8.m: Matlab file for Part 1.1.
3. tsp_i9.m: Matlab file for Part 1.1.
4. tsp_i10.m: Matlab file for Part 1.1.
5. tsp_i12.m: Matlab file for Part 1.1.
6. reynolds_escape.tar.gz: Compressed Webots world and controllers for Part II.
7. laplacian_formation2.m: Matlab file for Part II.
8. laplacian_formation3.m: Matlab file for Part II.

Important Note:

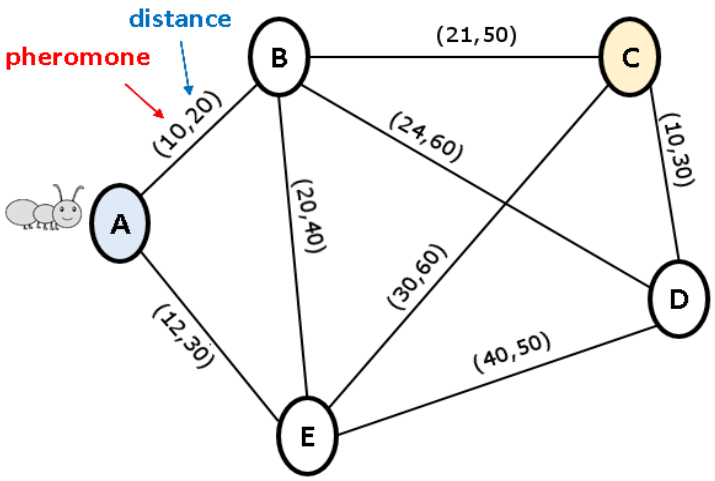
1. Do not answer on this sheet!

2. Please double-check your submitted solution files.

Part 1: Ant Colony Optimization (60 points)

1.1. Ant Colony Optimization and virtual ants (25 pts)

The following graph shows 5 cities (nodes) connected with a few links (edges). The 'initial pheromone' distribution and the 'edge lengths' (distance between cities) are marked on the links. Ants start their journey from city 'A' and want to get to the food source is in city 'C'.



AS with limited ant memory (12 pts)

Consider a variation of the basic Ant System (AS) algorithm explained in Lab01-part2 with modified ants, called the AlzheimerAnt, **it is capable of memorizing only the last visited city** and thus at each step will not return to the city that it has just come from. In every node, it chooses its next link based on the amount of pheromones already deposited and the distance to the next node. The influence parameter of pheromones (α) on the decision making formula of AlzheimerAnt is 2, and the parameter of heuristic (i.e. inverse of distance) information (β) is 1 (in other words α= 2 and β= 1).

Q1) What is the most probable path that one AlzheimerAnt starting at node A would take, assuming the initial distribution of pheromones as depicted in the figure above? (7pts)

Q2) Assume that there are many AlzheimerAnts (let's say 1000) and they all start from node 'A' sequentially one after the other. The ants that find the source release pheromone on their taken edges, proportional to the inverse of the total path length. What is the path with the highest pheromone concentration at the end? (5pts)

Virtual ants (13 pts)

Q3) In the setup explained above, imagine we experiment with one Justant (that you learned in lab01-part1). What is the most probable path that one Justant would take, assuming the initial distribution of pheromones as depicted in the figure? (5pts)

Q4) Assume that there are many Justants (let's say 1000s) and they all start from node 'A' sequentially one after the other. Similar to Lab 1, they all release pheromone while traveling from node to node. We assume that the journey of each ant ends when it gets to the food source (so no return to the nest). What is the path with the highest pheromone at the end? (4pts)

Q5) Answer the above question for the case of 1000 **Greedy** ants (defined in Lab01-part1). (4pts)

1.2. Ant Colony Optimization and TSP (35 pts)

In this question you will have to apply a few modifications to the EAS algorithm in solving the Travelling Salesman Problem (TSP), which you worked on in Lab 1. In the test/part1/matlab folder you will find a solution code for Lab 1. Use the Matlab command "load eil51" to load a map of 51 cities. This map is a fully connected graph with the distance between the cities ranging between 2.2 and 85.7. You can use the command "tsp('ant', cities)" to run the algorithm.

Open 'tsp.m' and answer the following question. Note that for some of the questions you need to modify this file and save it with another name. **Preserve the original tsp.m file for answering questions Q6 to I12.**

Q6) Explain briefly the purpose of line 229. (4pts)

Q7) As you know in the EAS algorithm, in each iteration extra pheromone is deposited on the best tour found in that iteration. Which lines in the code implement this concept? (4pts)

I8) Following the previous question, we now want to increase the amount of extra pheromone deposited on the best tour in each iteration by a factor of 2. Modify the code according to this request. At which line(s) did you insert/modify your solution? What would be the impact of this change on the performance of EAS in terms of exploration/exploitation trade-off? (6pts)

Note: Save and **submit** your solution code with name **tsp_i8.m**.

I9) **Start from the original 'tsp.m' code.** The TSP problem in this code considers fully connected graphs, i.e. every city can be directly reached from every other city by traversing one link. Now we want to modify the code in order to take into account cities that may not be directly connected. Make proper changes in the code in order to disconnect the link between city1 and city2. At which line(s) did you insert/modify your solution? (4 pts)

Note: Save and **submit** your solution code with name **tsp_i9.m**.

I10) **Start from the original 'tsp.m' code.** In the current code the pheromone does not evaporate. Implement evaporation with a rate of 30%, i.e. at each iteration 30% of the previously laid pheromone evaporates. At which line(s) did you insert/modify your solution? Explain what would be the impact of evaporation on the performance of the EAS algorithm. (6pts)

Note: Save and **submit** your solution code with name **tsp_i10.m**.

Q11) **Start from the original 'tsp.m' code.** Uncomment the following lines in your code: 160, 161, 170, and 171. Explain (in 2, 3 sentences) the possible effect of these changes on the behavior of the ants. (6pts)

I12) **Start from the original 'tsp.m' code.** As you know, in this course we have considered three different types of pheromone update for Ant Colony Optimization algorithms: i) AS, ii) EAS, and iii) ACS. The current code is based on EAS. Explain how you would change the code in order to implement ACS with global pheromone update (i.e., no local update) as explain in the course lecture (week 2). Modify the code in order to implement ACS with global pheromone update. At which line(s) did you insert/modify your solution? (5pts)

Note: Save and **submit** your solution code with name **tsp_i12.m**.

Part 2: Single and Multi-Robot Navigation (60 points)

1.3. **Localization (18 pts)**

I13): A differential drive robot has a wheel radius of 22.5 mm and the two wheels are 53 mm apart. It is equipped with a wheel encoder with 100 ticks per revolution. You can assume that there is no wheel slip. The robot starts at the origin, with orientation aligned with the x-axis. What is the pose of the robot after moving at constant wheel speed for some time T, assuming that the constant wheel speeds for the left and right wheel are 390 and 480 respectively? You can provide your answer by writing a piece of code in Matlab or by writing the proper equations in your answer sheet. (In case you write a code, copy and paste your Matlab code into your answer sheet). (18pts)

1.4. Reynolds Flocking (20 pts)

In Lab 4 you learned how to implement a flocking behavior using the three Reynolds rules: *Cohesion, Separation* and *Alignment*. A complete code similar to the Lab 4 is provided to you, in the exam materials "test/part2/webots". The robots begin in random locations and start the flocking behavior. Similarly as in the lab, they are able to avoid obstacles using Braitenberg behavior. Compile the controllers for the robots and the supervisor.

I14) On top of the Reynolds rules, the robots migrate towards a target position (using a migration urge). Change the vector of the migration urge such that the robots move towards the blue cylinder located in the arena. Hint: you have to change the world file. Save any changes you made. (4pts)

Q15) Imagine that your flock has to avoid light. Assuming that the positions of the light sources are known to the robots, write down in the answer sheet how you would implement the new *Escape* rule. You need to provide an equation, with a short explanation. (6pts)

Q16) Which of the three Reynolds rules is your *Escape* rule most similar to? (2pts)

I17) The supervisor code provides the robots with the position of the light sources. Implement your *Escape* rule in the e-puck controller. Your new controller should make the robots avoid light while migrating towards the target position in a flock. To do so, you need to carefully choose the weight of the *Escape* rule. (8pts)

Notes:

1. Save and submit your complete code (whatever exists in your "test/part2/webots" folder including the controllers and the world file) with name "reynolds_escape.tar.gz". To generate and submit the .tar.gz file you need to do the following: (i) right-click on the "webots" folder (in "test/part2", which contains your solution), (ii) click on "Compress", (iii) type the correct name ("reynolds_escape"), (iv) click on "Create", (v) click on "Close", (vi) submit the file on Moodle.
2. In your answer sheet write the major changes you made in the code.

1.5. Graph-Based Formations (22pts)

Consider six agents, shown in Figure 2. The goal is to design a graph-based control algorithm which leads them to the desired topological formation.

I18) Write a controller in Matlab that brings the robots from their initial positions (the white circles in Figure 2) to the desired formation (the grey circles). Assume that the robots are holonomic, i.e. they can instantly move in any direction. You can use the code provided to you "test/part2/matlab/laplacian_formation.m" (which is similar to the solution of Lab 4).

Save and submit your solution script with the name "laplacian_formation1.m". Hint: It is not important in which absolute positions the desired formation converges to, only the relative topology of the agents should be the one shown in Figure 2. (8pts)

Q19) List the consequences of assuming holonomicity of the robots? (4pts)

I20) The simple graph-based controller you implemented in I18 makes robots converge into a formation, but not move afterwards. How can you control migration of the formation? Make the formation migrate at a constant velocity towards the y-axis. You can keep the previous changes or start from the original code. Save and submit your solution script with name "laplacian_formation2.m". (2pts)

I21) Another way to move the formation is to establish a leader-follower relationship between the robots. Modify your code so that one of the robots is a leader, which moves towards the y-axis. The rest of the robots (i.e. the followers) maintain the formation on the same manner as in part I18. You can keep the changes from part I18 or start from the original code. The number of robots does not matter in this question. Save and submit your solution script with name "laplacian_formation3.m". In your answer sheet write how the leader-follower formation controller affects performance of the formation, particularly its shape. How could you solve this problem? (8pts)

a)

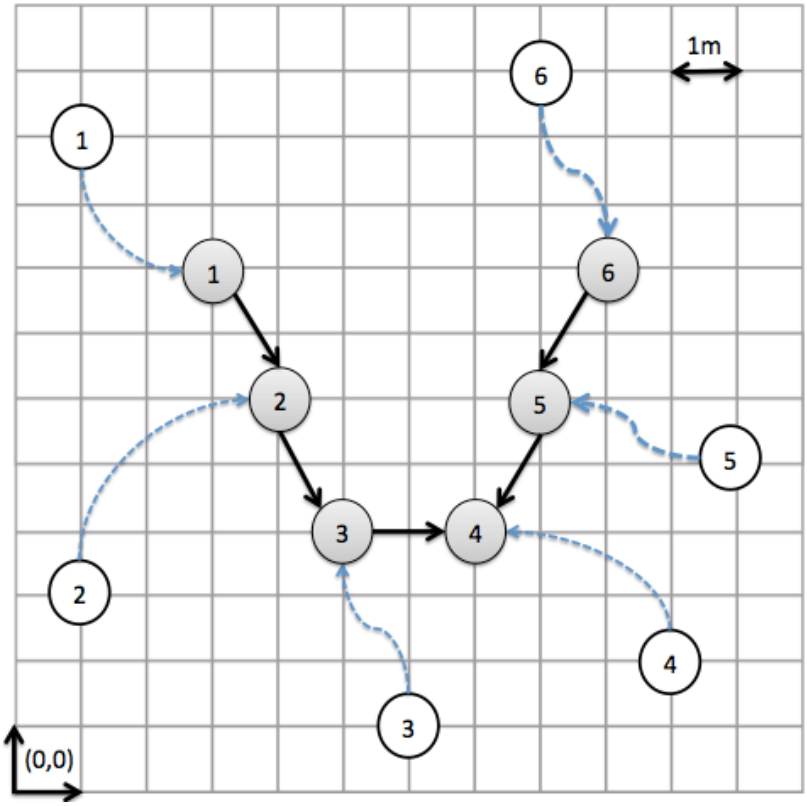


Figure 2: The initial (white) and the desired (grey) formation of six robots.