Swarm Intelligence Course
Exam Projects

Prof. Marco DORIGO
Dr. Yara KHALUF

1 Modalities

The exam is divided in two parts:
Project:
You have to pick a project among the ones proposed below and provide the required deliverables. In addition, you will present your project in a 7-minute talk, followed by questions.
Questions:
You will be asked a number of questions concerning the entire course material in addition to the submitted project.

1.1 Timeline:

The project submission deadline one week before the date of the exam. The date of the exam is still to be scheduled after contacting me at the Email address yara.khaluf@uni-paderborn.de.

1.2 Project Deliverables

For the project, you will have to provide:

- Your code in digital format (i.e., text files), so we can test it. Send it by e-mail to the Email address yara.khaluf@uni-paderborn.de.
- A short document (6-8 pages) written in English that describes your work. You have to explain both the idea and the implementation of your solution.
- A presentation of your project.

2 Projects

2.1 General Remarks

Apply what you learnt. It is mostly important that you stick to the principles of swarm intelligence: simple, local interactions, no global communication.
Avoid complexity. If your solution gets too complex, it is because it is the wrong one.
Honesty pays off. If you find the solution in a paper or in a website, cite it and say why you used that idea and how.
Cooperation is forbidden. Always remember that this is an individual work.

The basic precondition for you to succeed in the project is that it must work. If it does not, the project won't be considered sufficient. In addition, code must be understandable – comments are mandatory.
The document is very important too. We will evaluate the quality of your text, its clarity, its completeness and its soundness. References to existing methods are considered a plus – honesty does pay off!
More specifically, the document is good if it contains all the information needed to reproduce your work without having seen the code and a good and complete analysis of the results.
The oral presentation is also very important. In contrast to the document, a good talk deals with ideas and leaves the technical details out. Be sure that it fits in the 7-minute slot.

2.2 Optimization: *Path Planning using ACO*

2.2.1 Introduction

The path planning problem is a main issue in multi-robot systems for robot navigation and control. The key of this basic problem is to find a collision free path from a starting point to a destination point, in order to make the robot bypass the obstacles safely. The planning is performed according to some performance measures, e.g., the path length, the time, and/or the energy. In this project, we are considering the length of the obtained path as our performance measure.
In this project, we focus on a simplified version of the path planning problem using 2-D workspaces, in which a grid model is used to represent the robot's navigation area. The workspace is represented as a matrix of free and occupied cells. Free cells are available for the robots to move in, whereas the other cells are occupied by obstacles and are not feasible to use. An example is given in Figure 1, in which the black cells represent the cells occupied by obstacles and the white cells are the free cells to move in. The "S" cell is the start point and the "G" cell is the goal cell.

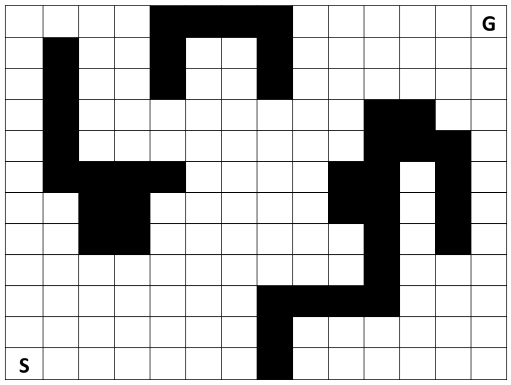

Figure 1: An example of a grid model of robot's navigation area.

The solution is free-collision path that has the following characteristics:

- The cells which build up the path are adjacent to each other.
- The cells sequence that represents the path starts at the start cell and ends at the goal cell.

The robot moves on the given grid randomly till it encounters the goal point. At each cell the robot can move with the following rules:

- The robot can move to any of its adjacent cells in case it is free, therefore, it can move potentially in 8 directions. In Figure 2, the possible movements of the robot are only in 6 directions.

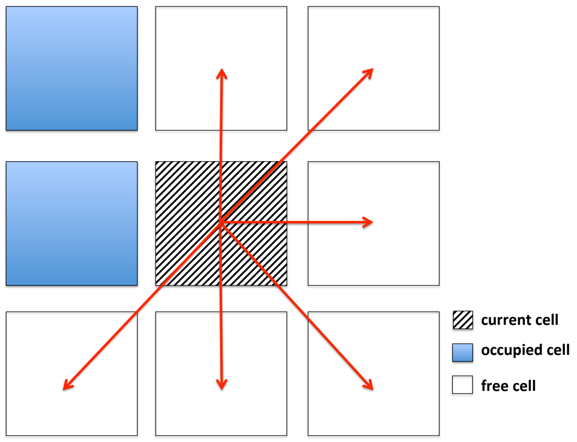

Figure 2: Rules of robot's movement.

- The robot stops moving in one of the following cases, either if the goal cell is reached or if all adjacent cells are occupied as we can see in Figure 3.

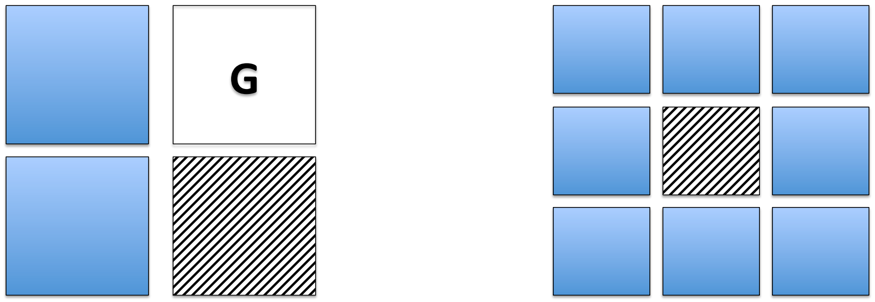

Figure 3: The two cases where the robot stops it movement behaviors.

2.2.2 Goals

The goal of the project is to design an ACO algorithm for solving the simplified path planning problem presented above. The implementation of the algorithm should result in the shortest path (the least number of cells) between the starting point ”S” and the goal point ”G”.

2.2.3 Problem Instances

We provide a set of instance where the algorithms implemented by you should be tested. Each instance file contains a $n \times n$ matrix with elements:

- 1 for the free cells.
- 0 for the obstacle cells.

The starting cell is always the cell at the bottom left corner and the goal cell is always the cell at the top right corner of the considered workspace. We provide 3 sizes of the defined matrices: 20×20, 30×30, 40×40.

2.2.4 Implementation

You are required to implement two ACO algorithms in order to solve the proposed problem.

- The first algorithm must be an Ant System (AS) algorithm following the description provided in the course.
- The second algorithm (MY_ACO) you are free to chose any of the ACO variants studied in the class or even other variants from the literature (should be well cited).

Please note that for both algorithms you will have to make modifications to the components of the studied algorithms in order to adapt them to the path planning problem. For example, the way you are going to define the pheromone or the heuristic information.
Bonus points are rewarded for implementing ideas (from the literature or your own ideas) which aim to improve the performance of the algorithm. This implies that the second algorithm is not required to follow strictly the structure of the studied variant you have selected (we encourage you to think out of the box!).

2.2.5 Deliverables

The final deliverables include the source codes and documents.
Report

a. Algorithms description

   Here you need to describe your algorithms in detail. Please include

   - Representation of the solution.
   - Definition of the pheromone and the heuristic information (in case it is needed).
   - The transition rule.
   - The pheromone update and evaporate rules.
   - The parameters of your algorithm and their influence in the algorithms.
   - Any other component or modification included in your algorithms.

b. Test the AS and MY_ACO for the provided instances using the values you think are adequate for the parameters and report the results[1] per problem size ($20 \times 20, 30 \times 30, 40 \times 40$). Please pay attention that reporting the results include strictly the analysis of the obtained results.

c. Find out the best parameter settings of one of the previous algorithms. In order to do so, test 3 values for each parameter of your algorithm for a particular instance size which you select freely (e.g. all the instances of the size 20×20).

   - Compare the algorithm performance under the influence of the 3 selected values of each of its parameters.
   - Select the best parameter settings found and execute them in all the instances provided. Compare the results with the ones of the non-tuned version.

d. Conclusions.

[1]Sample R scripts for generating the "Box plot" and the "Convergence" plots are available in the project subfolder "Scripts", in addition to a number of data files for testing. You can build your plotting code by help of the simple examples in the scripts "example-boxplot.R" and "example-convergence.R" for box plot and convergence plot, respectively.

Code

- The code should be properly commented and ordered.
- The code must be efficient and well-written.
- The code should include a README file with the main instructions and specifications of your code and parameters.
- Any detected copy will be panellized and the project will be rejected.

2.3 Swarm Robotics: *Collective Decision Making with Heterogeneous Agents*

Decision making is a cognitive capability which allows individuals to select the best option among a set of possible alternatives. When a decision is taken by a group of agents, the process is defined as collective decision making.
Collective decision making is a fundamental cognitive capability that can be found in several natural systems. For instance, each spring, honeybee colonies collectively tackle as a decision problem the nest site selection. A different example is the collective decision of bird flocks, which collectively move in one direction without need of any leader.
In this project, the student is asked to provide a robot swarm with the capability of discriminating between four options and collectively select the best one. The robot swarm is composed of heterogenous robots, that is, robots have different types of sensors which allow them to perceive only a subset of the quality features characterising an option. While each robot has partial knowledge of the available alternatives and can only partially estimate each option's quality, the swarm, as a whole, can perceive the entire set of features and is able to collectively decide for the best option. Therefore, robots must explicitly collaborate to agree on the selection of the best alternative.

2.3.1 Problem definition

The robot swarm operates in a scenario composed of 4 *target* rooms r_i (with $i \in [0, 3]$) connected by a central room. Each target room has a quality value v_i (with $i \in [0, 3]$), which is computed as the average of three metrics:

$$v = (v_G + v_L + v_O)/3 \tag{1}$$

The three metrics are computed as function of the following environmental characteristics:

- **The ground color (v_G)**: the ground color varies in the full gray scale from white to black and determines the value $v_G \in [0, 1]$. We assume that $v_G = 1$ for white grounds, $v_G = 0$ for black grounds and in the between v_G scales linearly. The robots can perceive the ground color through the ground sensor.

- **The light intensity (v_L)**: each room has a different illumination which determines $v_L \in [0,1]$. The room's light intensity is determined by the strength of a light source placed in the center of each room. We assume that $v_L = 1$ for light sources with maximum light intensity and $v_L = 0$ for light sources with minimum intensity (i.e., no light) and in the between v_L scales linearly. The robots can perceive the light intensity through the light sensor.
  Note: the light sensor returns a measure that varies with respect to the distance from the light source. Therefore, a robot may integrate several sensor readings over time while moving in the room for estimating an average light intensity of the room.

- **The number of objects (v_O)**: each room stores a number of objects $\mathcal{O}_i \in [2,12]$ which determines $v_O \in [0,1]$. We assume that v_O is linearly proportional to the number of objects $\mathcal{O}$, i.e., the more the better. Each object is marked with a green LED, which can be perceived by the robots through the camera.

Goal The robot swarm has to select the room r_i with best quality v_i:

$$r_i \in \underset{i \in [0,3]}{\arg\max}\, v_i \tag{2}$$

To select a room a robot must physically move into the selected room. We assume that the swarm has done a collective decision when the quorum $Q = 0.9$ of the total swarm moved into a room r. For instance, for a swarm of 100 robots, the decision is considered as taken when at least 90 robots have moved into the same target room.

Heterogeneity The swarm, in the given configuration, is composed of 20 heterogenous robots of two types:

- **Type G**: 10 robots of type G are equipped with a ground sensor that allows them to sense the color of the ground. However, robots of type G do not have sensors to perceive the light intensity value.

- **Type L**: 10 robots of type L are equipped with a light sensor that allows them to sense the light intensity. However, robots of type L are not equipped with the ground sensor, thus cannot perceive the color of the ground.

Robots of both types are equipped with camera and therefore are able to sense and count the objects (marked with green LED) stored in each room.

Additional Remarks

- The environment is designed with a central room connecting the 4 target rooms, so that the robots can encounter in the central room and interact with each other.

- Robots are able to differentiate the target rooms from each other by mean of LED lights of different colors that are placed on the door of each target room. The colors are assigned as follows: magenta, blu, orange, and red for rooms 0,1,2,3 respectively.
- The student can develop a single lua script that checks whether the robot is of type G or L. This can be done either by checking if a sensor is present or not (not null or null), or by simply checking the id of the robot. Robots of type G are assigned an id prefix "fbG", while the prefix of robots of type L is "fbL".
- The student can use the LEDs actuators to visualise the type of each robot, e.g. red LEDs for robots with ground sensors and blue LEDs for robots with light sensors.

Figure 4 shows a top view of the environment composed of four target rooms and a central room. The central room has no light and black ground. Each room has a different quality which may vary each experiment and is determined by the composition of the three metrics v_G, v_L and v_O (see Equation (1)).

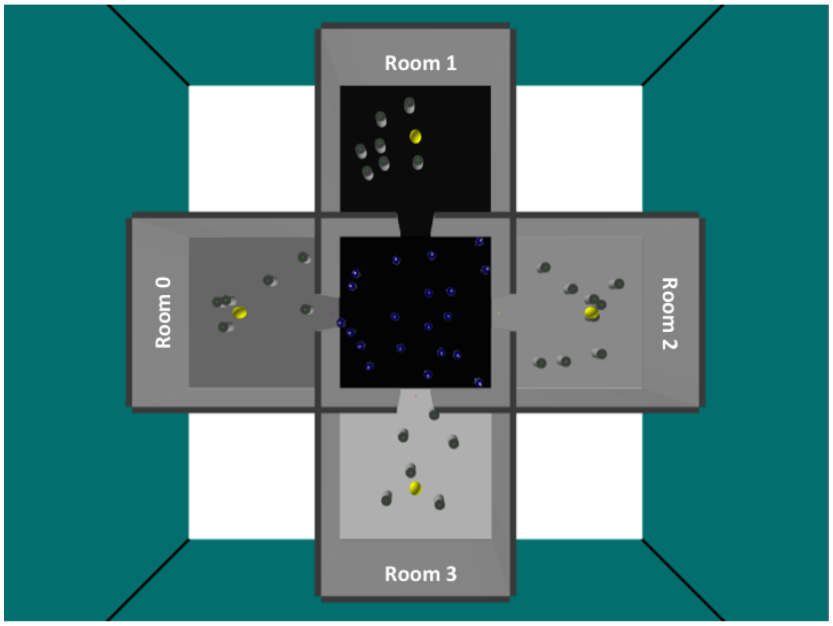

Figure 4: A top view of the environment.

The quality measures in addition to the room quality are given in the following table for the 4 rooms illustrated in Figure 4:

Room	v_G	v_L	v_O (O)	**Room quality**
Room 0	0.478243	0.579924	0.4 (6)	0.486056
Room 1	0.050487	0.72393	0.5 (7)	0.424806
Room 2	0.609765	0.285806	0.7 (9)	0.531857
Room 3	0.743975	0.386149	0.4 (6)	0.510041

The best is room is room number 2 with the evaluation: 0.531857.

2.3.2 Goals

The goal of this project is to design, implement and test a robot controller that allows the swarm to collectively select the best room r_i between 4 possible target rooms. The best room has the the highest quality v_i.
Different solutions are possible. Bonus points are awarded for simple and reactive controllers with limited (or no) memory. Remember to follow the principles of swarm robotics and apply what you learned during the course.
The student is required to analyze the solution at first with the provided configuration, and after by varying the following parameters:

- The swarm size N in the range of values $\{25, 30, 35, 40\}$ robots.
- The ratio $\rho = T_G/T_L$ where T_G is the size of the subpopulation of robots of Type G, and T_L is the size of the Type L robot subpopulation. In the initial configuration, $\rho = 1$ with the swarm splited in two equal halves of the two Types. The student is asked to analyse how varying ρ affect the system.

A complete analysis must be performed to evaluate the quality of the behavior. In particular, quantitative numerical measures of the performance of the system must be produced. ARGoS automatically dumps data on a file whose name must be set by you in the XML experiment configuration. This file is composed of several lines, each line containing five elements:

- The current step
- The number of robots in room 0
- The number of robots in room 1
- The number of robots in room 2
- The number of robots in room 3

To present statistically meaningful results we suggest that you repeat your experiments at least 30 times.
Be aware that, for the project evaluation, **the analysis is as important as the implementation**. A project presenting a behavior that is capable of selecting the optimal room wonderfully will be evaluated poorly if the analysis part is missed or limited.

Setting up the code

- Download the experiment files: SR_Project_WS_20142015.tar.bz2
- Unpack the archive and compile the code:
  - $ tar xvf SR_Project_WS_20142015.tar.bz2 # Unpacking
  - $ cd SR_Project_WS_20142015 # Enter the directory
  - $ mkdir build # Creating build dir
  - $ cd build # Entering build dir
  - $ cmake -DCMAKE_BUILD_TYPE=Release ../src # Configuring the build dir
  - $ make # Compiling the code
- Set the environment variable ARGOS_PLUGIN_PATH to the full path in which the build/ directory is located:

  $ export ARGOS_PLUGIN_PATH=/path/to/SR_Project_WS_20142015/build/
- You can also put this line into your $HOME/.bashrc file, so it will be automatically executed every time you open a console. Run the experiment to check that everything is OK:
  - $ cd /path/to/SR_Project_WS_20142015 # Make sure you are in the right directory
  - $ argos3 -c decision-making.xml # Run the experiment

If the usual ARGoS GUI appears, you're ready to go.

2.3.3 Deliverables

The final deliverables include the source codes and documents.
Report

A report of 6-8 pages structured as follows:

- Main idea of your approach.
- Structure of your solution (the state machine).
- Analysis of the results.

Code

The Lua scripts that you developed, well-commented and well-structured.