Swarm Intelligence Course (INFO-H-414)
Exams

July 1, 2014

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 8-minute talk, followed by 5 minutes of questions. This will account for up to 10 points of your final grade.

Questions You will be asked a number of questions concerning *the entire course material*. This will account for up to 10 points of your final grade.

To pass the exam one must collect at least 5 points for each part of the exam.

1.1 Timeline

- The date of the exam is still to be decided. At present, we know that it will probably be on September 4th or 5th, 2014.
- The project submission deadline is 5 working days before the date of the exam (August 27th or 28th at 23.59).
- Each day of delay on the submission will entail a penalty of 1 point on the final evaluation of the project.

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 people responsible for your project (see contacts at the end of the document);
- 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.

- On the day of the oral presentation, you are expected to show slides and come with your own laptop. If you happen to not have a laptop, send us a PDF version of your slides **before** the day of the exam.

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 project counts for 50% of your final grade. 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 8-minute slot.

2.2 Ant Colony Optimization

2.2.1 Introduction

This project is based on the GECCO 2014 *Competition on Permutation-based Combinatorial Optimization Problems*. At the competition, participants need to propose an algorithm for solving five combinatorial optimization problems. In this project, you will implement three of the ACO algorithms available in ACOTSP for the four classical problems considered in the competition, namely:

- Traveling Salesman Problem (TSP)
- Permutation Flowshop Scheduling Problem (PFSP)
- Linear Ordering Problem (LOP)

- Quadratic Assignment Problem (QAP)

A detailed description of each problem can be found in the webpage of the competition: http://www.sc.ehu.es/ccwbayes/Competition_GECCO2014/. Furthermore, students are to propose an ACO variant of their own and, depending on its results, we encourage you to take part in the actual competition!

2.2.2 Goal

The goal of this project is to implement and analyze ACO algorithms for different combinatorial optimization problems. For doing so, your task is to extend three ACO algorithms that you have studied during the ACO part of the course:

- Ant System (AS)
- $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS)
- Ant Colony System (ACS)

Notice that, although these are all permutation-based problems, modeling them to be solved by an ACO algorithm may not be trivial. Moreover, you are expected to implement problem-specific local search methods, such as the ones already provided in ACOTSP for the TSP. Check the corresponding literature for existing methods.

After implementing the algorithms above for the given problems, you will need to analyze their performance on each problem. As explained during the course, the numerical parameters can improve or reduce the performance of an ACO algorithm. To have a fair comparison, you will first tune the algorithms using irace (http://iridia.ulb.ac.be/irace/) on a set of tuning instances provided in the competition's webpage. We recommend a reasonable/feasible tuning budget (200 experiments per algorithm per problem). Once you have tuned all algorithms, run them 10 times on each of the following testing problem instances:

- TSP – TSPLIB (symetrical): berlin52, brazil58, eil76, rat99, kroA100, kroB100, lin105, lin318, rd400, att532;
- PFSP – Éric Taillard's instances (use the first instance of each size): 20 (jobs) x 10 (machines), 20x20, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20, 500x20;
- LOP – xLOLIB: N-be75np_{150,200}, N-stabu1{150,200}, N-t65d11xx_{150,200}, N-t70l11xx_{150,200}, N-tiw56r58_{150,250};
- QAP – Éric Taillard's instances (symetrical and structured): tai27e01 → tai27e02, tai45e01 → tai45e03, tai75e01 → tai75e03, tai125e01 → tai125e02.

To ensure that the ACO part of the algorithm is indeed effective, you will also run a random start local search algorithm. The stopping criterion for all algorithms will be either the maximum number of evaluations ($1000 \cdot n^2$, where n is the problem size), or the runtime of 1 minute – whichever is reached first. When you have all the experimental results ready, compare the average runtime and the average relative percentage deviation (RPD) from the known optimal or best known solution for those instances. These optimal/best known values can be found on the webpages where the instances are made available. The average RPDs and runtimes should be presented as both tables and boxplots.

To conclude, discuss the results and the performance of all algorithms. Are the ACO algorithms indeed effective, or are their results similar to the random start local search? Explain eventual patterns if they appear, or why results are different across the different problems if they don't. Finally, evaluate your own algorithm: how does it perform in comparison to the others? Would you recommend that others researchers explore the research path you attempted, or would you say it is better to try other approaches?

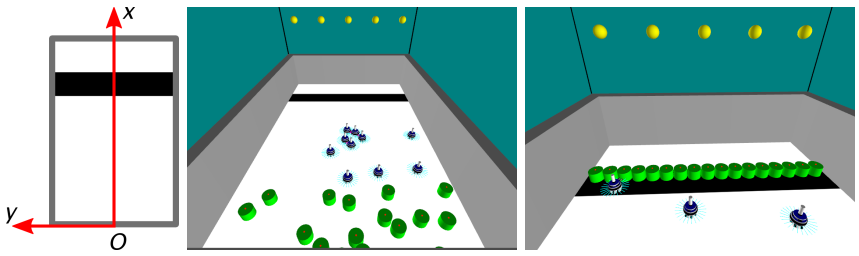
Remember that a quality work can be submitted to the actual competition, which is a great experience for someone learning to use ant colony optimization. However, to attain that you will have to meet the competition's deadline.

2.2.3 Deliverables

The final deliverables include the source codes and documents.

Report

1. Implement AS, ACS, and MMAS for solving TSP, PFSP, QAP, and LOP.
   (a) Describe the details of your implementations (i.e. pheromone and heuristic information definition, local search, etc.).
   (b) Describe your ACO variant.
2. Tune the algorithms using irace.
   (a) Describe the tuning setup.
   (b) List the parameter settings selected for each algorithm/problem.
3. Compare the algorithms.
   (a) Perform 10 runs of each of your algorithms on each instance provided in the test set;
   (b) Report for each instance/algorithm best (B), worst (W), mean (M), standard deviation (SD), relative percentage deviation (RPD), and average runtime (in seconds);
   (c) Discussion of the results for all algorithms;
   (d) Discussion of the results for your algorithm.



4. Conclusions.

Code

- The code should run in the same way as the ACOTSP code, that is, by a given command line. (Check ACOTSP help)
- 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.

2.3 Swarm Robotics

2.3.1 Introduction

The goal of the project is to develop a *collective building* behavior. Collective building is a common activity in nature. Many social insects (e.g., ants, bees, termites) are capable of building amazing structures in a completely distributed, robust and flexible way. Such collective building behaviors are very interesting for swarm robotics, as they can be used as an inspiration for simple distributed behaviors.

2.3.2 The Scenario

In this project, the environment is an area in which several objects scattered. Blocks can be seen by the robot using their camera. A robot can physically pick up a block using its gripper. The objects are located in the bottom part of the environment; robots are located north of the initial location of the objects; the exact positions of the robots and of the blocks are chosen at random each time you start an experiment. The area of the environment, characterized by a black ground color, is where the wall must be built. Differently from the assignment for the first exam session, in this setting the robots cannot use the light to navigate the environment.

2.3.3 Goals

Your goal is to develop a behavior that lets the robots build a wall. Ideally this wall should divide the corridor in two areas, that is, once the wall is built, no robot can move from one area to the other.

Different solutions are possible. Bonus points are awarded for simplicity and for behaviors that use a minimal set of sensors. Remember to follow the principles of swarm robotics and apply what you learned during the course.

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 N of objects that are part of the wall (that is, that are in the black area), the *thinness* of the wall, the *compactness* of the wall, and the *coverage*.

Thinness is defined as

$$\begin{cases} 0 & \text{if } N = 0, \\ 1 - \dfrac{A}{B} & \text{otherwise,} \end{cases}$$

where A is the standard deviation of the x coordinates of the objects within the construction area, B is the size along the x axis of the construction area ($B = 0.7\,\text{m}$). Thinness takes values in $[0, 1]$. You must maximize this measure.

Compactness is defined as

$$\begin{cases} 0 & \text{if } N < 2, \\ \dfrac{G_y}{D} & \text{otherwise,} \end{cases}$$

where G_y is the maximum distance along the y axis among the objects within the construction area, and D is the diameter of a construction object ($D = 0.2\,\text{m}$). Compactness is best when it is close to 1. A value lower than 1 means that the maximum distance (projected onto the y axis) between the center of the objects is smaller than D. A value larger than 1 means that the objects are too spread out. For the sake of this project, values slightly lower than 1 are preferable to values slightly larger than 1.

Coverage is defined as

$$\begin{cases} 0 & \text{if } N = 0, \\ \dfrac{C}{S} & \text{otherwise,} \end{cases}$$

where C is the projection of all the objects in the construction area on the Y axis, and S is the size of the construction area along the y axis ($S = 3.6\,\text{m}$). Coverage takes values in $[0, 1]$. You must maximize this measure.

To present statistically meaningful results we suggest that you repeat your experiments at least 30 times.

Test your behavior also in different experimental setups. You can change the initial distribution and the number of both the objects and the robots in the two groups in the XML file. Show how these parameters influence the performance of the system.

To evaluate your project we will test it not only in the default experimental setup (i.e., the one given by us), but also in different environments (i.e., with a different number of robots/objects). Solutions working only in the default experimental setup will be considered worse than flexible solutions.

If you developed multiple behaviors, you can compare them in your report, showing, for instance, how different design choices affect the performance of the system.

Be aware that, for the project evaluation, **the analysis is as important as the implementation**. A project presenting a behavior that is capable of constructing wonderful walls will be evaluated poorly if the analysis part is missing or limited.

Technical information on ARGoS, Lua, and the experiment configuration file is reported in the swarm robotics page of the course: http://iridia.ulb.ac.be/~cpinciroli/extra/h-414/.

2.3.4 Implementation Notes

- The robots can perceive the objects using their omnidirectional camera.
- The robots can grip the objects using their gripper.
- The robots do not possess a sensor telling them where 'north' is. Solutions in which the robots pick an object up and then perform random walk until they luckily find the construction area will be considered poor. Instead, concentrate on swarm-based solutions. HINT: have a look at social odometry,[1] or make it so that some robots do not actually build the wall, but act as beacons using the R&B system.
- The construction area is marked by a black ground color, which can be detected using the ground sensors.
- Other sensors are available. Note that, as said, it is not necessary to use all the available sensors. Solutions that exploit a minimal set of sensors, are usually considered more robust and better than those which are not minimal. However, there is usually a trade-off between the performance of the system and the number of sensors used. A good analysis could justify the use of different sensors by showing how the performance of the system changes when some sensors are not used.

[1] http://www.robolabo.etsit.upm.es/aguti/publications/GutCamMon-etal12IntModRob.pdf

- The wall must be constructed in the black area. Only the blocks that are in the black area will be considered to compute the statistics.

2.3.5 Deliverables

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

3 Contacts

Marco Dorigo	mdorigo@ulb.ac.be	for general questions
Mauro Birattari	mbiro@ulb.ac.be	for general questions
Leslie Pérez Cáceres	leslie.perez.caceres@ulb.ac.be	for ACO
Leonardo Bezerra	leonardo@iridia.ulb.ac.be	for ACO
Carlo Pinciroli	cpinciro@ulb.ac.be	for swarm robotics
Manuele Brambilla	mbrambil@ulb.ac.be	for swarm robotics

References