Swarm Intelligence Course (INFO-H-414)
Exams

May 3, 2013

1 Modalities

The exam is divided in two parts:

Project You have to pick a project among the three proposed below and provide the required deliverables. In addition, you will present your project in a 10-minute talk, followed by 5 minutes of questions. This will account for 50% of your final grade.

Questions You will be asked a number of questions concerning *the entire course material*. This will account for 50% of your final grade.

1.1 Timeline

- The project submission deadline is 5 working days before the date of the exam.
- The date of the exam is still to be decided. At present, we know that it will be in the period June 17 – June 21, 2013.

1.2 Project Deliverables

For the project, you will have to provide:

- the code you implemented 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 (4 to 6 pages) written in English that describes your work. You have to explain both the idea and the implementation of your solution.

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.

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 10-minute slot.

2.2 Ant Colony Optimization

2.2.1 Introduction

This project comprises the design, implementation and analysis of ACO algorithms for solving the General Assignment Problem (GAP).
The general assignment problem is a classical NP-hard combinatorial optimization problem, using its formulation is possible to model several real world problems in order to solve them automatically. The problem can be defined as follow:

There are N jobs to be completed by M agents:

- Agent n has a cost c_{nm} when assigned to job m
- An amount of resource a_{nm} is consumed (i.e. time, maintenance hours, etc.) when agent n is assigned to job m
- Each agent m has maximum amount of resource b_m

The objective is to find an assignment that has minimum cost and:

- Each job has only one agent assigned.
- The maximum amount of resource per agent is not exceeded

A detailed description of the problem can be found in [3].

2.2.2 Goal

The goal of this project is to design, implement and analyse ACO algorithms for solving GAP. For doing so, your task is to adapt and extend four of the ACO algorithms that you have studied during the exercise sessions:

- Ant System (AS)
- Elitist Ant System (EAS)
- Rank-based Ant System (RAS)
- Ant Colony System (ACS)

Note that since the algorithms will be designed for a different problem (GAP), modifications will be needed. You are free to define the components of the algorithms as you think best, as long as they follow the general definition of each algorithm. If you think they can improve your algorithms you are allowed to implement ideas proposed in the literature, as long a citation is included.

Once the algorithms are implemented experiments should be carried out in order to compare and analyse the performance of the different solvers. Additionally, the addition of localsearch to an ACO algorithm will be studied. You are free to propose any kind of localsearch for the problem, you can find examples of localsearch procedures for GAP and results of other approaches in [4], [5], [2], [6] and [1]. Finally, a complete survey of GAP can be found in [3].

The test instances for the experiments will be 10 instances available in OR-LIBRARY `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html`. A description of the instance files can be found in the instance folder, more information about the format of the files can be found in the OR-LIBRARY website. When reporting results show clearly the instance that corresponds to.

2.2.3 Deliverables

The final deliverables include the source codes and documents.

Report:

1. Implement AS, EAS, RAS and ACS for solving GAP.
   (a) Describe shortly your design decisions (i.e. pheromone and heuristic information definition, update mechanism, transition rule, etc.). List the parameter settings that you used for each algorithm and explain why.

(b) Perform 25 runs for each of your algorithms on each instance of the test set.

(c) Report for each instance/algorithm experiment: best (B), worst (W), mean (M), standard deviation (SD), average number of assignments generated (NA), average number of assignments generated to find the best result (NAB).

(d) Perform a pairwise comparison of the algorithms using the Wilcoxon rank-sum test.

(e) Based on c) and d), which of the implemented algorithms would you recommend for solving this problem? Comment.

(f) Compare the convergence* of the algorithms. Comment. * The goal is to compare the computation effort vs solution quality. One idea could be to make a graph of the Mean (M) vs Average assignments to best (NAB).

2. Choose an algorithm and add localseach search to it.

   (a) Describe shortly the localsearch procedure implemented.

   (b) Perform 25 runs per instance for the new algorithm and compare it with its version without local search. Report as above.

   (c) Are there any improvements? Why? Comment.

3. Optional 1: Tune the parameters of the algorithm chosen in 2). Analyse your results, is the algorithm more sensitive to some parameters?

4. Optional 2: Compare your results with the ones you can find in the literature!.

Code:

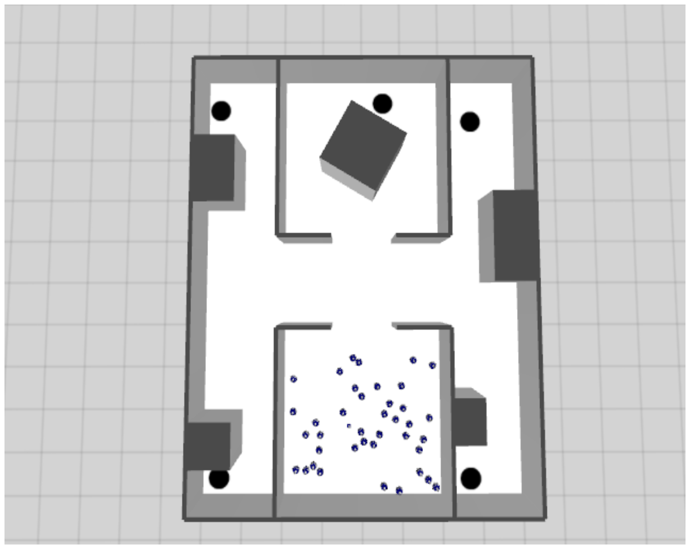
- The code should run in the same way as the ACOTSP code, that is, by a given command line. (Check ACOTSP help)
- The code should be properly commented and ordered (indent!).
- The code should include a README file with the main instructions and specifications of your algorithms and parameters.

2.3 Swarm Robotics

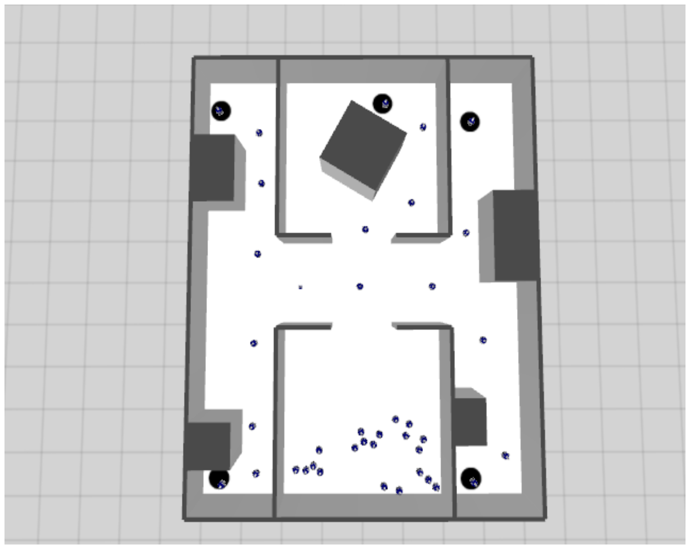
2.3.1 Introduction

In real-world swarm robotics applications, exploration is an important behavior. Often, the environment in which the robots are deployed is unknown and potentially dangerous for humans, so the robots must be capable to spread and discover its structure and the important locations.

In the course, we learnt the simplest exploration strategy possible—*diffusion*. Diffusion can be obtained with a simple obstacle avoidance strategy. This strategy is very efficient in those cases in which the robots are in large number



(a)



(b)

Figure 1: The environment for the swarm robotics project. (a) State at the beginning. (b) Successful state at the end.

(high density of robots per area unit), and the structure of the environment is relatively simple.

When the environment is very large and its structure is complex, obstacle avoidance is not sufficient anymore. The robots must engage in some sort of exploration activity that retains memory of interesting locations and avoid exploring multiple times the same places, potentially neglecting important portions of the environment.

Several methods exist to obtain this goal. Since we are working with robot swarms, we are going to consider a solution that imposes minimal requirements on the robots—*chaining*.

In a nutshell, chaining consists in connecting two places (i.e., the nest and an important location) with a set of robots positioned along the path that leads from one place to the other. Once the path is formed, additional robots can communicate with the robots along the path to move from a place to the other without the need for exploration. In other words, chaining is a form of collective memory.

2.3.2 Goals

You must implement a chaining strategy for the environment in Fig. 1(a). The environment loosely mimicks an indoor structure, in which five target locations are marked with black dots. The task of the robots, initially placed in an area that we will call the *nest*, is to form five chains that connect the nest with the target locations. The experiment is automatically stopped when there is a robot in each location.

Structure your solution as a *state machine*. This is the most important coding aspect you must respect. A state machine allows you to decouple a complex problem into simpler instances, whose solution is easier to develop.

Use the distance scanner. It is a powerful sensor, that provides long-range information on the structure of the environment around the robots.

As you know from the course, the robots can communicate wirelessly through the range-and-bearing sensor. The communication range impacts both the number of robots necessary to complete the chain, and the structure of the chain. Pay attention to the communication range! You will get extra points if you perform a study of the role of the communication range on your behavior.

Characterize your solutions *at least* with respect to these two metrics:

1. Average time necessary to reach all the locations and standard deviation;
2. Average number of robots involved in the chains and standard deviation.

ARGoS automatically dumps data on a file whose name must be set by you in the XML experiment configuration. This file is composed by several lines, each line containing two elements: the current step and the number of robots that are part of a chain at that step. Thus, to calculate the two metrics, you just need to consider the last line of the output file. To calculate the metrics run *at least* 30 repetitions of each experimental setup.

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/`.

2.3.3 Deliverables

- The Lua scripts that you developed, well commented and well structured.
- A report of 4-6 pages structured as follows:
  - Main idea of your approach [1 page]
  - Structure of your solution (the state machine) [2 pages]
  - Analysis of the results [1-3 pages]

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

[1] P.C. Chu and J.E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers and Operations Research*, 24(1):17 – 23, 1997.

[2] Juan A. Daz and Elena Fernndez. A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research*, 132(1):22 – 38, 2001.

[3] O.Erhun Kundakcioglu and Saed Alizamir. Generalized assignment problem. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1153–1162. Springer US, 2009.

[4] Helena R. Loureno and Daniel Serra. Adaptive search heuristics for the generalized assignment problem. *Mathware and Soft Computing*, 9:209–234, 2002. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.19`.

[5] M. Yagiura and T. Ibaraki. Recent metaheuristic algorithms for the generalized assignment problem. In *Informatics Research for Development of Knowledge Society Infrastructure, 2004. ICKS 2004. International Conference on*, pages 229–237, 2004. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.19`.

[6] Mutsunori Yagiura, Toshihide Ibaraki, and Fred Glover. An ejection chain approach for the generalized assignment problem. *INFORMS J. on Computing*, 16(2):133–151, March 2004.