Final Project for the Swarm Intelligence course

Second Session

July 1, 2019

1 Modalities

The exam is divided in two parts:

Project + presentation You have to select one of the two projects proposed below and provide the required deliverables. Please send an email with your choice by July 7, 2019 to the responsible of the topic (contacts at the end of the document).

In addition, you will present your project in a 5-minute talk, followed by 5 minutes of questions. We strongly encourage you to focus **only** on presenting your ideas (how you solved the problem), your results and findings. The project and the presentation will account for up to 10 points of your final grade.

Oral examination 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, both parts must be above the threshold.

1.1 Timeline

- You have until **July 7** to send the email specifying the project of your choice.
- The project submission deadline is **August 9** at 23.59.
- Delay on the submission will entail a penalty of 1 point every 12 hours delay on the final evaluation of the project. Max delay is **August 11**, at 23:59. After this deadline you fail the exam.

1.2 Project Deliverables

For the project, you will have to provide:

- Your code in digital format, so we can test it (more details on the format are given on the description of each project). It has to be sent by e-mail to the responsible(s) for your project (see contacts at the end of the document).
- A short document (max 6 pages not counting the references[1], single spaced, font 10) written in English that describes your work. You have to explain both the idea and the implementation of your solution.

  The document must be typeset using the template of Lecture Notes in Computer Sciences[2] (LNCS – Springer), available on `http://iridia.ulb.ac.be/~ccamacho/INFO-H-414/project/templates`, in the format of a scientific article, including abstract, introduction, short description of the problem, description of the solution, results, conclusions and references (see `http://iridia.ulb.ac.be/~ccamacho/INFO-H-414/project/examples` for a few examples on how to structure your project document).
- On the day of the oral examination, you are expected to show slides of your work. Please bring your own portable computer and make sure it is already switched on and ready when you are called in for your oral exam. The rest of the exam will consist of a question answering session on all the subjects covered by the course. 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 learned — It is very important that you stick to the principles of swarm intelligence: simple, local interactions, no global communication, no global information.

Avoid complexity — Good solutions are elegantly simple in most cases. So, if your solution is becoming too complex you are very likely in the wrong direction.

Honesty pays off — If you find the solution in a paper or in a website, cite the source and say why and how you used the idea.

[1] You can add as many pages of bibliographical references as necessary to support your work.

[2] `https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines`

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 will not 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 5-minute slot. Because you only have 5 minutes for your presentation, you should focus on presenting what you have done and the results you obtained, rather than on the presentation of the problem.

2.2 Particle Swarm Optimization

2.2.1 Introduction

Consider the problem of learning how to drive a car. One has to learn when to change gear, or when to brake depending on the features of the road ahead. Additionally, one has to pay attention to the surroundings, the speedometer, the tachometer, the fuel-level meter, etc. In general, one can say that learning how to drive a car is a difficult task.

In this project, you will crate an agent that will "learn" how to drive a simulated racing car. To do so, you will use The Open Racing Car Simulator (TORCS) [1] to design and implement a car driving agent, and then, using particle swarm optimization (PSO), you will optimize its parameters.

2.2.2 Goal

The goals of this project are the following:

1. Problem definition. Many industrial problems can be cast as optimization problems. The definition of the optimization problem is normally a non-trivial task. In this project, you will cast the problem of designing a car driving agent as a continuous optimization problem suitable to be solved with a PSO algorithm.

2. Algorithm selection and parameterization. Once the optimization problem is defined, there is still the problem of choosing an optimization algorithm capable of solving it. In this project, you will experience

the problems associated with algorithm selection and parameter setting.

3. Integration of systems. In many real scenarios, an optimization algorithm resides in a separate piece of software that needs to be integrated with the application software. It is not rare that this application software is a simulator of some sort. In this project, you will integrate your PSO code with a car racing simulator.

2.2.3 Materials and Methods

You have to design, implement, and optimize a car controller for the TORCS simulator that will compete with other cars. TORCS is an open source car racing simulator that provides a full 3D visualization, a sophisticated physics engine, and accurate car dynamics taking into account traction, aerodynamics, fuel consumption, etc. The simulator comes with a number of tracks and cars with different features each. You will test your car controller in three tracks: “G-Speedway”, which belongs to the oval tracks category, “Wheel 2”, which belongs to the road tracks category, and “Dirt 3”, which belongs to the dirt tracks category. The car you are going to use is “car1-ow1”, the only Formula One that comes with TORCS.

Using a PSO algorithm, you have to optimize at least 8 parameters of your car driving agent, which must complete a lap of each track as fast as possible. To assess the actual quality of the optimized controller, you must compare it, through a race, with other car controllers included in TORCS that use the same kind of car.

You are free to choose the type of controller. For example, you can use a rule-base controller and optimize threshold values, or you can use an artificial neural network and optimize its connection weights. However, as a suggestion, you should start with a simple controller, and use more complex controllers once the initial one works. **The focus is on optimization, not on the technology you use for your agent**.

The software you have to use is available for download at `http://iridia.ulb.ac.be/~ccamacho/INFO-H-414/project/torcs`. You will use the stand-alone application of TORCS that runs in GNU/Linux and Windows operating systems (Mac OS is not supported by the TORCS developers) in which the controllers are compiled as separate modules that are loaded into main memory when a race takes place. Please follow the instructions given in the file “README_installation_and_template” to learn how to fix a few problems related to the installation of TORCS in Ubuntu 18.04, and also to know how to install a template for your controller. The controller has to be codified in C/C++ and loaded as a module of the main TORCS application which is written in C++. You have to modify the code of the controller so that you can interface it with the PSO code. Feel free to use the PSO algorithm that you implemented during the last practical session of the course. However, note that a number of modifications are needed in order to use it for this project.

You can find a lot of documentation about TORCS in the next websites:

- `http://torcs.sourceforge.net/`
- `http://www.berniw.org/trb/index.php`
- `http://xed.ch/help/torcs.html`

2.2.4 Deliverables

The final deliverable will be a zip file containing the next two items (please create a folder for each one):

- Report
- Code

Please make sure that both items comply with the guidelines indicated in Sec. 1.2

Report (up to 6 pages)

1. Implementation of your car driving agent:
   (a) Describe the design process of your car driving agent.
   (b) Describe the optimization process setup, which includes the objective function, the PSO algorithm, parameter settings, etc. Please include an explanation of the relevant design decisions (representation of solutions, perturbation mechanism, topology, etc.)
   (c) Set the parameters of PSO running simulations on TORCS using the mode `Practice` and report the best parameter settings. Describe the process you used to obtain the parameter settings. If you use automatic tuning, please report: parameters tuned, set of values (domain) for each parameter and number of evaluations allowed for the tuner (some freely available tuners are [3] and [2]).
   (d) Perform 20 simulations of 3 laps on each track (that is, "G-Speedway", "Wheel 2" and "Dirt 3") using TORCS on mode `Quick Race` and using different random seeds for PSO.
   Note: you must compare your controller against `olethros 2`, `inferno 1`, `lliaw 1`, and `tita 1`, which use the same car as you.

(e) Include in the report a table showing, for each track, the best (B-rank) and worst (W-rank) rank obtained in the race; and the best (B), worst (W), mean (M) and standard deviation (SD) of the total time that your agent needed to finish the race, where each row of the table is a track.

(f) Analyze the performance of the PSO algorithm:

- Use plots and/or tables to show the results.
- Compare the convergence of the algorithm in the different tracks.
  (Note: The goal is to compare the computational effort (e.g. number of evaluations of the objective function) vs. solution quality).

Code

- The code of your controller should be able to be integrated with TORCS 1.3.7, and to be compiled and executed on a computer running Ubuntu 18.04.
- The source code should be properly commented and indented.
- You must include a README file with the instructions and specifications of how to compile, install and execute your controller using TORCS 1.3.7. If you use libraries and/or packages not natively included in the version 1.3.7 of the simulator or Ubuntu 18.04, please send them together with your code.
- You must implement your own code. **Plagiarism will be strongly penalized**.

2.3 Swarm Robotics: *Foraging with Forbidden Areas*

The activity of food search and retrieval is commonly referred to as *foraging*. In swarm robotics, foraging is a commonly used task to compare different algorithms for exploration (what is the best way to discover interesting places in the environment?), division of labor (who should explore? for how long?), etc. In the most general setting, food items are scattered in an environment at locations unknown to the robots and the robots need to explore the environment, find the food, and take it to the nest. The foraging environments often contain cues, such as light sources, that help the robots in navigating through the environment.

In this project, we consider a foraging environment with *forbidden areas*: robots than enter these areas automatically lose the item that they carry, if any. The swarm is distributed in the arena and the location of the food source is originally unknown to the robots. A light is placed above the food source to indicate its location to the swarm. The students are asked to develop and provide control software for the robot swarm, and to evaluate its performance.

2.3.1 Problem definition

The robot swarm operates in a diamond-shaped arena that includes a food source, a nest, and two forbidden areas. The food source is represented by a black circle, the nest is represented by a white area in the bottom the diamond, and the forbidden areas are gray rectangles—see Figure 1. A light is placed on top of the food source to indicate its position to the robots. When a robot enters one of the forbidden area while carrying an item, the item is automatically lost.

Goal The goal of the robot swarm is to retrieve and transport items from the food source to the nest. The overall performance of the swarm is measured by the number of items it is able to collect during a fixed experiment time. Each experiment is automatically terminated after 1000 seconds (10000 time-steps). More precisely, the performance of the swarm is described by

$$\max N_d \tag{1}$$

with N_d being the number of items successfully delivered to the nest (that is, items carried by robots from the food source to the nest without entering the forbidden areas).

Swarm composition The swarm comprises 30 homogeneous robots. The robots are equipped with the following sensors and actuators:

- `colored_blob_omnidirectional_camera` to detect LEDs;
- `range_and_bearing` to communicate between robots;
- `light` to perceive the light above the food source;
- `motor_ground` to sense the color of the ground;
- `proximity` to detect obstacles;
- `wheels` to explore the environment;
- `LEDs` to display information.

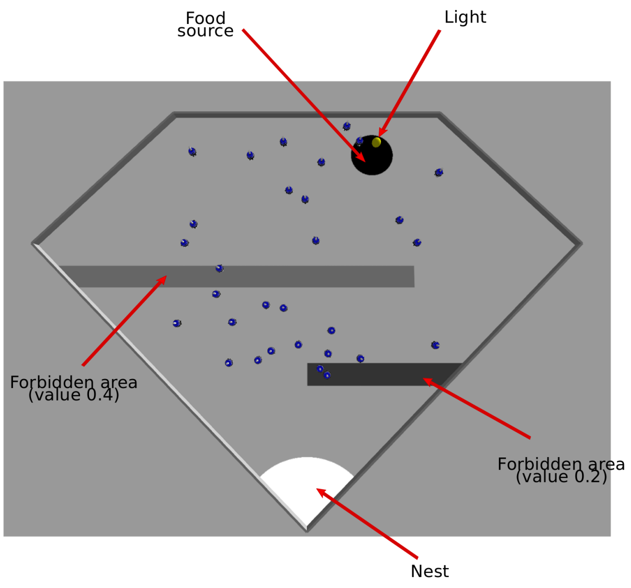

Figure 1: Top view of the arena. The forbidden areas are of different colors so that the robots can distinguish between the two areas, but their effect is the same: any carried item is automatically lost.

Additional Remarks

- The maximal wheel velocity should not exceed 15 cm/s.
- ARGoS has been configured so that the robots collect items on the food source and drop them at the nest and at the forbidden areas automatically. A robot can only carry one item at a time.
- The positions of the nest and the forbidden areas are fixed, but the position of the food source can change from one experiment to another. See Figure 1.

2.3.2 Requirements

The goal of this project is to design, implement and evaluate control software that aims to maximize the number of items delivered at the nest. The control software has to demonstrate a *cooperative* behavior: one that takes advantage of the swarm's principles.

ARGoS is configured to automatically dump data on a file whose name can be changed in the .argos experiment configuration (see Section 2.4). This file contains a table with two columns:

- CLOCK: Column indicating the current step
- ITEMS: Column indicating the number of items collected so far

A complete analysis must be performed to evaluate the quality of the controller implemented. In particular, to present statistically meaningful results, we suggest you to execute and collect the results over at least 30 runs of the control software. The 30 runs should differ from each other for the random seed specified in the experiment file (.argos). In your report, you will include a table containing the following columns:

- Random seed
- Final number of items collected from source

Beside this table, quantitative numerical measures of the performance of the two solutions must be produced. Specifically, you should produce i) a plot that shows the trend over time of the items collected by the swarm; ii) a plot that shows the variance of the number of items collected by the swarm in different runs. On the basis of these measures, you should discuss the results and draw the appropriate conclusions.

Be aware that, for the project evaluation, **the analysis is as important as the implementation**. Make sure that the information provided in the report is meaningful, clearly written and complete. Any meaningful additional content will be rewarded (e.g., comparison of different control software).

Setting up the code

- Download the experiment files: SR_Project_H414_2nd.tar from `http://iridia.ulb.ac.be/courses/2019/h-414/SR_Project_H414_2nd.tar.gz`.
- Unpack the archive into your $HOME directory and compile the code

```
$ tar -xzf SR_Project_H414_2nd.tar.gz # Unpacking
$ cd SR_Project_H414                  # Enter the directory
$ mkdir build                         # Creating build dir
$ cd build                            # Entering build dir
$ cmake ../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=$HOME/SR_Project_H414_2nd/build/
```

  You can also put this line into your $HOME/.bashrc file, so it will be automatically executed every time you open a terminal.

- Run the experiment to check that everything is OK:

```
$ cd $HOME/SR_Project_H414_2nd        # Enter the directory
$ argos3 -c foraging.argos            # Run the experiment
```

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

2.4 Setting up the experiment

Switching the visualization on and off. The experiment configuration file allows you to launch ARGoS both with and without visualization. When you launch ARGoS with the visualization, you can program the robots interactively exactly like you did during the course. Launching ARGoS without the visualization allows you to run multiple repetitions of an experiment automatically, e.g., through a script. By default, the script launches ARGoS in interactive mode. To switch the visualization off, just substitute the visualization section with: <visualization />, or, equivalently, comment out the entire qt-opengl section.

Loading a script at init time. When you launch ARGoS without visualization, you cannot use the GUI to set the running script. However, you can modify the XML configuration file to load automatically a script for you. At line 50 of foraging.argos you'll see that the Lua controller has an empty section <params />. An example of how to set the script is at line 53 of the same file. Just comment line 50, uncomment line 53 and set the script attribute to the file name of your script.

Changing the random seed. When you want to run multiple repetitions of an experiment, it is necessary to change the random seed every time. To change the random seed, set the value at line 11 of foraging.argos, attribute random_seed.

Changing the output file name. As explained above, ARGoS automatically dumps data to a file as the experiment goes. To set the name of this file, set a new value for the attribute output at line 17 of foraging.argos.

Making ARGoS run faster. Sometimes ARGoS is a little slow, especially when many robots and many sensors are being simulated. You can make ARGoS go faster by setting the attribute threads at line 9 of foraging.argos. Experiment with the values, because the best setting depends on your computer.

2.4.1 Deliverables

The final deliverables must include source code and documentation:

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

Documentation: A report of up to 6 pages structured as a scientific article and containing:

– Main idea and swarm robotics principles in your approach.

– Structure of your solution (the state machine).

– Analysis and discussion of the results.

See Section 1.2 for more details about the deliverables.

2.5 Evaluation

The evaluation criteria are the following: a) ingenuity of the solution, b) performance of the control software, and c) relevance and clarity of the report.

3 Contacts

Marco Dorigo	mdorigo@ulb.ac.be	for general questions
Mauro Birattari	mbiro@ulb.ac.be	for general questions
Christian Camacho Villalón	ccamacho@ulb.ac.be	for PSO
Federico Pagnozzi	federico.pagnozzi@ulb.ac.be	for PSO
Antoine Ligot	aligot@ulb.ac.be	for swarm robotics
David Garzon Ramos	dgarzonr@ulb.ac.be	for swarm robotics

4 Bibliography

References

[1] Christophe Guionneau Christos Dimitrakakis Rémi Coulom Andrew Sumner Bernhard Wymann, Eric Espié. TORCS, The Open Racing Car Simulator. http://www.torcs.org, 2014.

[2] Frank Hutter, Holger Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: An automatic algorithm configurationframework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.

[3] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.