Bio-Inspired Artificial Intelligence

Prof. Giovanni Iacca
giovanni.iacca@unitn.it

Module 4–Lab Exercises

Introduction

Goal. The goal of this lab is to familiarize yourself with some advanced forms of evolutionary computation (EC). In particular, you will explore the use of multi-objective evolutionary algorithms to find the optimal trade-off solutions of problems with multiple objectives.

Getting started. Download the file `04.Exercises.zip` from Moodle and unzip it. This lab continues the use of the *inspyred* framework for the Python programming language seen in the previous labs. If you did not participate in the previous labs, you may want to look those over first and then start this lab's exercises.

Each exercise has a corresponding `.py` file. To solve the exercises, you will have to open, edit, and run these `.py` files.

Note once again that, unless otherwise specified, in this module's exercises we will use real-valued genotypes and that the aim of the algorithms will be to *minimize* all fitness functions $f_i(\mathbf{x})$, i.e. lower values correspond to a better fitness!

Exercise 1

In this first exercise, you will explore the optimization of multiple objectives. Before you begin using multi-objective evolutionary algorithms you will first experiment with turning multi-objective problems into single-objective problems so that you may employ a standard Genetic Algorithm (such as was used earlier in the course). One way that multiple objectives may be combined into a single-objective is by combining the objectives in a weighted sum.

To start the exercise, from a command prompt run[1]:

```
$>python exercise_1.py
```

by default, this will try to optimize the 2-objective Kursawe benchmark problem[2] with a GA by additively combining the two objectives with equal weights. Please note that the Kursawe benchmark problem is scalable in the number of variables (although it's normally tested with 3

[1] For all the exercises in this lab you may follow the name of the `.py` file with an integer value, which will serve as the seed for the pseudo-random number generator. This will allow you to reproduce your results. Also, please note that in this document `$>` represents your command prompt, do not re-type these symbols!

[2] `http://pythonhosted.org/inspyred/reference.html#inspyred.benchmarks.Kursawe`

variables, e.g. the 2-D Pareto Front you'll see on the `inpsyred` web-page refers to a 3-variables version of this problem), therefore you should set the number of variables to 2 or more (see the comments in the script `python exercise_1.py`).

You will first see this combined fitness function plotted over evolutionary time (best and mean), and then (if you set the number of variables equal to 2) you will see the initial and final populations (left and right, respectively) plotted on top of heatmaps depicting objective 1 (top) and objective 2 (bottom).

- What happens when you run the GA with this fitness function?
- Why do you obtain this result? (Hint: pay attention to the relative scaling of the two objectives!)

Try altering the fitness weights to give the first objective greater importance.

- What happens if you give the first (or second) objective all of the weight?
- Can you find a weighting able to find a solution that approaches the optimum on both objectives?

Try this out on a different problem (e.g. the DTLZ7 benchmark[3], you can change the problem by changing the parameter `problem` in the script `python exercise_1.py`) and/or try increasing the dimensionality of the Kursawe problem (change the variable `num_vars` in the script). Please note that the DTLZ7 benchmark problem is scalable in the number of variables and objectives (although it's normally tested with a number of variables equal to the number of objectives + 19, and a number of objectives equal to 3, e.g. the 3-D Pareto Front you'll see on the `inpsyred` web-page refers to a 22-variables and 3-objectives version of this problem), therefore you should set those parameters accordingly (see the comments in the script `python exercise_1.py`).

- Does your weighting still work on the new problem? (NOTE: The size of the array `args["fitness_weights"]` must be the same as the number of objectives.)
- Can you think of a method for combining the objectives that might work better than using a weighted sum? (NOTE: If you want to try your idea out, edit the `_init_` method of `CombinedObjectives` in `inspyred_utils.py`).

Exercise 2

In this exercise you will use the Non-dominated Sorting Genetic Algorithm-II (NSGA2)[4] to evolve not just a single optimum but to find the "Pareto-set" of non-dominated solutions.

Recall that one solution Pareto-dominates another if it is no-worse on all objectives and it is better on at least one-objective.

From a command prompt run:

```
$>python exercise_2.py
```

This will again try to optimize the 2-objective Kursawe benchmark problem, but now using the multi-objective algorithm.

[3] `http://pythonhosted.org/inspyred/reference.html#inspyred.benchmarks.DTLZ7`

[4] Deb, K., A. Pratap, and S. Agarwal. "A fast and elitist multi-objective genetic algorithm: NSGA2." *IEEE Transactions on Evolutionary Computation* 6.2 (2002): 149172.

- How do the solutions you find here compare to those found in exercise 1?
- Is there a single solution that is clearly the best?

Now, try the more difficult DTLZ7 problem (or, alternatively, one of the other benchmark problems available in `inspyred`[5]).

- Can you still find good solutions?
- What happens if you increase the population size (change the variable `args['pop_size']` in the script) or the number of generations (see the parameter `args['max_generations']`)?

Finally, it may aid your intuition to try building up new problems out of single-objective benchmark problems. This may be done by creating a `MyBenchmark` instance with a list of single-objective benchmarks[6]. You can add as many objectives as you like, but the Pareto-front won't be plotted with more than two objectives.

Exercise 3

While benchmark problems can be interesting to investigate, multi-objective genetic algorithms can be (and are) used for many real-world problems as well.

In this exercise you will investigate optimizing the parameters of a multiple-disk clutch brake[7] as shown in Figure 1.

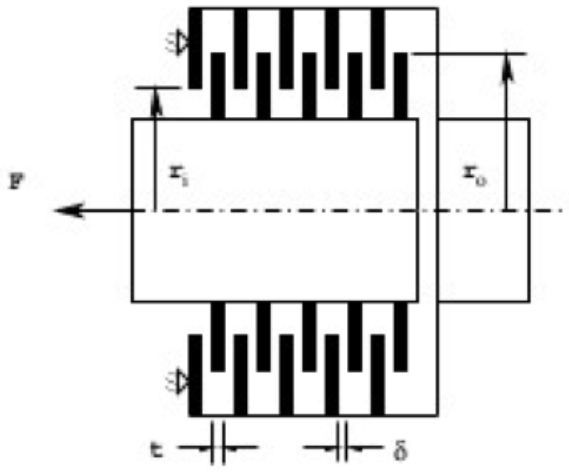

Figure 1: A graphical representation of a multiple-disk clutch brake

[5]See `https://pythonhosted.org/inspyred/reference.html#multi-objective-benchmarks` for a list of multi-objective benchmark problems.

[6]See `https://pythonhosted.org/inspyred/reference.html#single-objective-benchmarks` for a list of single-objective benchmark problems.

[7]This example comes from: Deb, Kalyanmoy, and Aravind Srinivasan."Innovization: Innovating design principles through optimization." *Proceedings of the 8th annual conference on Genetic and evolutionary computation.* ACM, 2006.

Two-conflicting objectives are considered: (i) minimization of mass (f_1 in kg) of the brake system and (ii) minimization of stopping time (f_2 in s).

There are five decision variables: $x = (r_i, r_o, t, F, Z)$, where r_i in [60, 80] (in steps of one) is the inner radius in mm, r_o in [90, 110] (in steps of one) is the outer radius in mm, t in [1.5, 3] (in steps of 0.5) is the thickness of discs in mm, F in [600, 1000] (in steps of 10) is the actuating force in N and Z in [2, 9] (in steps of one) is the number of friction surfaces (or discs).

This problem has been implemented for you in `disk_clutch_brake.py`. You may want to look through that code to see how it is possible to deal with a problem with fixed values for the decision variables such as this one.

From a command prompt run:

```
$>python exercise_3.py
```

This will attempt to optimize the five variables to minimize the two objectives.

- Is the algorithm able to find reasonable solutions to this problem? Use what you have learned about population sizes and number of generations to improve the quality of the found solutions. (NOTE: Do not make them too big or you will have to wait a long time to obtain results!)

Now that you are solving a "real" problem it is useful to examine the results. Inspect the final Pareto-front from a successful optimization run.

- Do you see any patterns in the Pareto-optimal solutions that may help you in designing a well-performing disk-brake in the future?

The final population and fitness values are saved in `exercise_3.csv` in the form {r_i, r_o, t, F, Z, $mass$, $time$}, one line for each solution in the Pareto front. You may want to try plotting these data in different ways to gain further insights. This process is what Deb has dubbed "innovization".

Instructions and questions

Concisely note down your observations from the previous exercises (follow the bullet points) and think about the following questions.

- When do you think it is appropriate to use a multi-objective evolutionary algorithm vs. combining multiple objectives into a single fitness function?
- What can the results of a multi-objective algorithm teach us about exploring the design spaces of engineering problems?
- In biological evolution it is possible to think of many phenotypic traits that contribute to the ultimate fitness of an organism (try to enumerate some of these). What (if any) relevance do multi-objective evolutionary algorithms have to biology?