Project 1: Evolutionary Optimization Algorithms

ME498: Computational modeling and optimization

Mattia Gazzola

Issue Date : February 2, 2019
Teaching Assistant : Tejaswin Parthasarathy, `tp5@illinois.edu`
Submission Date/Time: Wednesday 11:59 PM, 27 February, 2019
Submission Instructions:

- You need to submit both your presentation (as `.pdf`, `.ppt[x]` or `.key`) and code (as `.py` or `.ipynb`) on Compass, in separate links
- Submit your code file(s) packaged as a `.zip` or `.tar.gz` files (please do not use `.7z` or other formats) and name the compressed file as `<net_id>_code_01`
- Submit your presentation file(s) packaged as a `.zip` or `.tar.gz` files (please do not use `.7z` or other formats) and name the compressed file as `<net_id>_slides_01`

1 Guidelines

General advice that pertains to all the problems include:

1. Pick sensible representation, fitness function, selection and mutation operators and parameters wherever applicable.
2. Remember to present (during the project evaluation) the motivation for the algorithmic design choices you have chosen…
3. Explore as much as possible—change parameters, selection/mutation operators etc… We want you to learn how the optimizer "thinks" and works.
4. We will evaluate your work based on your understanding and findings, and not on the code quality. However since you are learning `Python`,

it is always a good idea to code efficiently. Use canned algorithms in `numpy` or `scipy` wherever possible.

5. You are free to consult any material you want (including but not limited to section 6). However you are expected to follow the Student Code on academic integrity—plagiarism will not be tolerated!

2 The Knapsack problem

2.1 Definition

Use genetic algorithm and rank-μ, weighted recombination version of the CMA evolutionary strategy discussed in class to find the solution to the 0/1 variant of the Knapsack problem.

2.2 Details

More details about the 0/1 Knapsack problem are contained in the lecture slides. Alternatively, the Wikipedia page also has a lot of information about this problem. Quoting the introductory text,

> " The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. "

Your task is to find an optimal distribution of items to carry, that maximizes the overall value while adhering to the weight limit, using both

- GA
- CMA

and compare the performance of these algorithms on this function. Details about the dataset for this problem is given below in section 2.3. Note that this is a constrained optimization problem and your design choices in the algorithm need to reflect this. As this seems like a problem with integer solutions, consider how you can change CMA to reflect this property. As

we are interested in black-box optimization algorithms, we assume we do not know anything about the problem structure. This means that using a randomly generated population around $\mathbf{x}_0 = \mathbf{0}$—the vector of zeros—to initialize the algorithm is a good idea.

Finally, after answering all the questions above and implementing your algorithm, explore the design choices (be it parameters or operators) on the performance of the algorithms.

2.3 Data

There are two data sets–A, B. Each data set will be posted on Compass as simple comma delimited text files with the knapsack capacity, number of different items (which are all assumed to be distinct so you do not need to worry about *choosing* same items), and their values + weights. For convenience you can read the information using `numpy`'s `loadtxt` function, shown in the snippet below:

```
import numpy as np

# Load data from txt file
my_data = np.loadtxt('file_name.txt', delimiter=',')

# Data stored as a dictionary
# use data however you want
bag_capacity = my_data['capacity']
'''
other possible keys include
capacity : Bag capacity, float
n_items : Number of items, integer value
item_values : Value of each item, a (n_items, ) numpy array
item_weights : Weight of each item, a (n_items, ) numpy array
'''
```

3 Minima of the parabola

3.1 Definition

Use genetic algorithm and rank-μ, weighted recombination version of the CMA evolutionary strategy discussed in class to find the minima of a simple parabola.

3.2 Details

The one-dimensional parabola is a continuous, convex, unimodal function. We pick the parabola given by the formula

$$f(x) = 10 \cdot x^2 \tag{1}$$

Your task is to find the optimum of this function using

- GA
- CMA

and compare the performance of these algorithms on this function. Furthermore, explore the effect of the parameters (particularly in GA) on the performance of the algorithms. Start your search using a randomly initialized population around $\mathbf{x}_0 = \mathbf{0}$—the vector of zeros.

4 Minima of the Rotated Hyper-Ellipsoid

4.1 Definition

Use genetic algorithm and rank-μ, weighted recombination version of the CMA evolutionary strategy discussed in class to find the minima of the two-dimensional rotated ellispoid.

4.2 Details

The two-dimensional Rotated Hyper-Ellipsoid is a continuous, convex, unimodal and non-separable function. We pick the variant that is rotated $\frac{\pi}{6}$ rad clockwise from the x_1-axis and shifted along both axes, given by the formula below:

$$f(\mathbf{x}) = \left(\frac{\sqrt{3}}{2}(x_1 - 3) + \frac{1}{2}(x_2 - 5)\right)^2 + 5 \cdot \left(\frac{\sqrt{3}}{2}(x_2 - 5) - \frac{1}{2}(x_1 - 3)\right)^2 \tag{2}$$

Graphically, it's contour plot is depicted in fig. 1 for several values of c. Your task is to find the optimum of this function using

- GA
- CMA

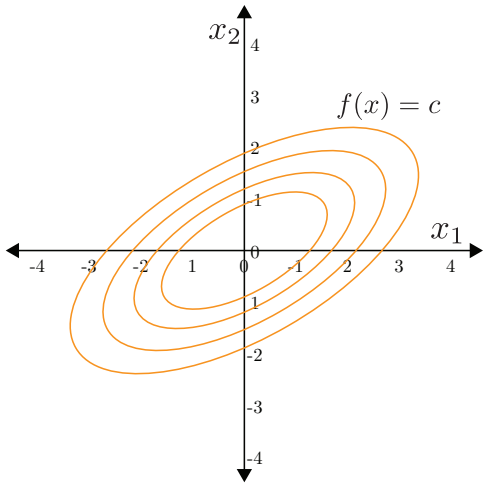

Figure 1: The rotated hyper-ellipsoid in two dimensions, the horizonal axis corresponds to x_1 and the vertical to x_2

and compare the performance of these algorithms on this function. Furthermore, explore the effect of the parameters on the performance of the algorithms. Start your search using a randomly initialized population around $\mathbf{x}_0 = \mathbf{0}$—the vector of zeros.

5 Minima of the Rastrigin function

5.1 Definition

Use the rank-μ, weighted recombination version of the CMA evolutionary strategy discussed in class to find the minima of the shifted *Rastrigin* function in two and five dimensions.

5.2 Details

The (unshifted) Rastrigin function is shown in fig. 2 for the case of two-dimensions.

It is a multi-modal function with several regularly distributed local minima, and can be generalized to arbitrary dimensions using the analytical formula shown below, for the shifted variant (which you should use as the objective function):

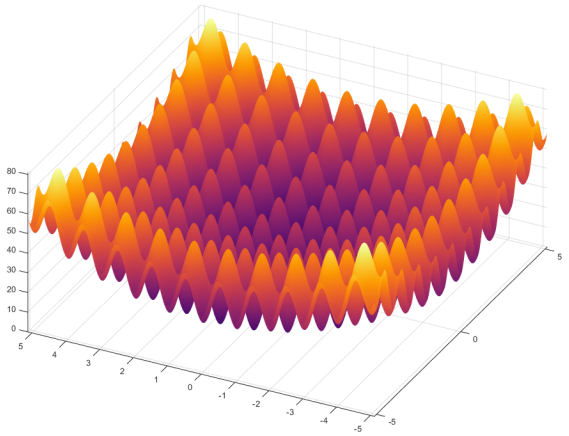

Figure 2: The (unshifted) Rastrigin function in two dimensions, taken from this SO thread

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[(x_i - 2)^2 - 10\cos\left(2\pi(x_i - 2)\right) \right] \tag{3}$$

where$d$ is the number of dimensions.

You need to find the local minima for this function in

- two dimensions ($d = 2$)
- five dimensions ($d = 5$)

using CMA. Start your search using a randomly initialized population around $\mathbf{x}_0 = \mathbf{0}$, the vector of zeros. Choose sensible/appropriate values for the other CMA parameters (default ones also suffice).

In both cases, consider what role does the population size play in the *performance* of the algorithm. Do you notice considerable differences at lower (2) and higher (5) dimensions? Explain.

6 The following resouces may prove useful:

- A short tutorial on the genetic algorithm found here
- The CMA-ES tutorial @ Arxiv, found here

- The CMA site maintained by Niko Hansen, found here
- Tutorial on CMA-ES, 2013 by Auger, Anne and Hansen, Nikolaus published in the Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion - GECCO '13 Companion found at `http://dx.doi.org/10.1145/2464576.2483910`