Homework 4

Particle Swarm Optimization

(Max useful score: 100 - Available points: 160)

15-382: Collective Intelligence (Spring 2019)

Instructions

Homework Policy

Homework is due on Autolab by the posted deadline. As a general rule, you have a total of 6 late days. For this homework you cannot use more than 1 late day. No credit will be given for homework submitted after the late day. After your 6 late days have been used you will receive 20% off for each additional day late.

If you find a solution in any source other than the material provided, you must mention the source.

Submission

Create a zipped archive including: a PDF file with the answers to the provided questions (they can be handwritten, but in this case you must have / use a "readable" handwriting), files that have been used for dealing with the questions that require programming, a README file that specifies how to use / run the programming files. The zipped archive should be submitted to Homework 4 on Autolab.

Contents

1 Continuous optimization with PSO (160 points)

The optimization of complex multi-modal functions is a difficult task. When the number of local optima is very large, gradient descent would need to be applied in some iterated form, possibly restarting many times from different initial points, in order to hope to hit the global optimum. PSO performs multi-agent black-box optimization, and might be a suitable alternative choice in these cases.

Given the intrinsic difficulty of the task, at a number of major international conferences it is common practice to set up black-box and/or white-box optimization competitions. A benchmark set of optimization problems is defined and the goal is to find the best (optimal) solutions given a specified maximum number of function evaluations (or of cpu time). For instance, you can peek at the black-box competition currently open at the GECCO conference, a major venue for nature-inspired algorithms, https://bbcomp.ini.rub.de/.

Below, some of the functions that are regularly used in these competitions are considered for you to optimize. For instance, you can check how these functions (and many other benchmark functions) look like at: http://al-roomi.org/benchmarks/unconstrained/n-dimensions.

☛ For all the tasks discussed below, you'll be scored on the soundness of your design choices, on the quality of your data reporting, and on the quality of the minima that you'll be able to find (poor minima will imply a low score, this is a *competition!*).

1.1 Basic PSO (50 points)

You are asked to implement an instance of the basic PSO (as shown in the lectures, with the constriction factor) for finding the minimum of the following set of d-dimensional scalar functions:

- Grienwank:
$$f_1(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$
where $d = 30,\ -600 \leq x_i \leq 600, \forall i = 1,\ldots,d$. The known optimum is found at $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

- Schwefel:
$$f_2(\boldsymbol{x}) = -\sum_{i=1}^{d}\left(x_i \sin\left(\sqrt{|x_i|}\right)\right),$$
where $d = 30,\ -500 \leq x_i \leq 500 \forall i = 1,\ldots,d$. The known optimum is found at $x_i^* = 420.968746,\ i = 1,\ldots,d$, with $f(\boldsymbol{x}^*) = -418.9828 \cdot d$

- Rastrigin:
$$f_3(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i) + 10,$$
with $d = 10,\ -5.12 \leq x_i \leq 5.12 \forall i = 1,\ldots,d$. The known optimum is found at $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

- Rosenbrock:
$$f_4(\boldsymbol{x}) = \sum_{i=1}^{d-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right],$$
with $d = 30,\ -30 \leq x_i \leq 30 \forall i = 1,\ldots,d$. The known optimum is found at $x_i^* = 1\ \forall i = 1,\ldots,d$, with $f(\boldsymbol{x}^*) = 0$.

You will have to make a number of design choices for your PSO algorithm (number or particles, neighborhoods, local vs. global best, etc.). The educational task is precisely to explore the possible alternatives by reasoning about their impact and mutual interaction (i.e, you can't really use a brute force approach trying out all possible alternatives, you must make reasoned choices and assumptions). You MUST describe and discuss your design choices in the report.

For each run of your PSO algorithm, you have to compute the *relative error*, defined as:

$$\varepsilon = \frac{|f^{pso} - f^{opt}|}{|f^{opt}|}.$$

The numerator is the absolute error, the magnitude of the difference between the optimal value (known in these cases) and the approximation provided by your PSO algorithm. The relative error is the absolute error divided by the magnitude of the optimal value. The *percent error* is the relative error expressed in terms of percentage.

Since PSO depends on stochastic aspects, including the initial deployment and velocities of the particles, for each function you must run your PSO algorithm (at least) $n = 5$ times starting with different initial deployment and velocities for the particles. The performance of your algorithm on each function $f_i,\ i = 1,\ldots,4$ is computed as the average performance over the n runs.

For the experimental evaluation, you have a maximum of *10 minutes of cpu time per function* (imagine you are using function optimization for a real life problem that requires to take optimized decisions every 10-15 minutes based on a changing environment). You are free to use this time as you want, but of course you must assume no knowledge about where the optimum of the function is!

For each function, you must report the following data in a table:

- Average relative error;
- Variance of the relative error;
- Average time when the best solution was found.

You are warmly encouraged to write a *script* program (python was originally design precisely for this!) that makes you performing all the experiments, collect all data, and prepare the result table in automatic (you have 4 functions, each run is 10 minutes, and you have 5 runs, therefore, each "experimental campaign" will take 200 minutes, about 3h. You *must* keep your computer doing the job for you while you're doing other things!

1.2 PSO variants (60 points)

Once you have gone through the previous question, you should consider possible improvements of your PSO algorithm. A number of variants have been proposed through the years. Some of these are discussed in the book extract included with the course website (lecture of March 12, 2019). Your task is to go through the pages of the extract, read about the different variants (described in pages 317- onward), and pick-up one of them as guideline to implement your PSO variant.

1. Discuss the reasons of your choice vs. other three different possible alternatives described in the book extract, at your choice (it's not enough to say: "Because I like it", you must bring substantial arguments!)
2. Describe your implementation and strategic choices.
3. Perform an experimental campaign analogous to the one you have performed with the basic PSO and report the data in a table as before.
4. Discuss the observed differences (if any) in behavior. At this aim you will likely have to collect some extra data apart from when the minimum is achieved (e.g., best solution so far improvement trend, spatial distribution of the particles over time, diversity in the particles, ...)

1.3 *Niching* optimization (50 points)

PSO has the tendency to "converge" to a single global optimum. However, dealing with multi-modal functions, it might be interesting to discover and maintain/converge on more than one optimum. This can be thought as the case when the agent population corresponds to multiple *species*, with each species being interested in finding an ecological *niche*, which is optimized and at the same time is far enough from the ecological niche of other sub-populations / species. The optimization function precisely defines where the different niches are in the "environment".

You can check the niching competition that is currently open at GECCO'19 http://www.epitropakis.co.uk/gecco2019/

Your task consists in modifying one of the two PSO algorithms that you have developed for the previous questions in order to address the issue of niching, aiming to two niches. The goal is to have, at the end of the 10 minutes run, the particle population having discovered and more or less clustered around two good minima. The performance J_{pso} of you algorithm is measured as follows:

$$J_{pso} = \frac{\varepsilon_1 + \varepsilon_2}{2d_{12}},$$

where ε_1 and ε_2 are, respectively the relative errors for niche 1 and niche 2 minima, d_{12} is the Euclidean distance between the locations where the minima of the two niches are, normalized to the maximal distance D between two points in the function domain (for all functions f_i, $i = 1, \dots, 4$, the domain is squared, centered in $\mathbf{0}$, such that the maximal distance D between two points is easily computed). More specifically, d_{12} is defined as:

$$d_{12} = \frac{||\boldsymbol{x}_1 - \boldsymbol{x}_2||}{D}.$$

In practice, the goal is to find two good minima that are sufficiently far apart in the function domain.

As in previous questions, you have the same time limits, must perform multiple runs, and report you result in a table. Moreover, you have to describe and discuss your design choices.