F21BC Biologically Inspired Computation for Master students only

Patricia A. Vargas & Marta Vallejo
Coursework 1 - Black-Box Optimization

September 22, 2016

1 Introduction

Quantifying and comparing the performance of numerical optimization algorithms is an important aspect of research in search and optimization. Black-Box Optimization software can be used to perform this task. In this regard, algorithms have to face a testbed of benchmark functions, generating homogeneous output data that can be easily post-processed allowing the presentation of the results in graphs and tables. This analysis provides a quantitative performance assessment over the algorithms.

To perform this task you will be asked to implement an evolutionary algorithm to evolve an artificial network and to test different evolutionary configurations within the COCO (COmparing Continuous Optimisers) platform [1]. COCO platform has been used in some of the most important conferences in evolutionary computation as the basis for the black box optimization benchmarking for example at the GECCO workshop (`https://numbbo.github.io/workshops/BBOB-2016/`) and in the IEEE Congress on Evolutionary Computation (`http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2015/CEC2015.htm`). So far, the framework has been successfully used to benchmark far over a hundred different algorithms by dozens of researchers.

2 Group List

For this assessment you can do it individually or in pairs. Irrespectively or your choice, you will be assessed using the same criteria. If you decide to do the assessment with another person, you should communicate your partner name to Dr. Patricia A. Vargas (p.a.vargas@hw.ac.uk) at most by the 30/09/2016 (Friday) via email, with the subject: 'F21BC 2016 CW PARTNER' including: - both names and surnames - user ID - student ID. **No changes will be allowed after that date.** Remember to copy your partner on the e-mail as well to confirm that the person has agreed to work with you.

The final group allocation will be circulated by 04/10/2016 (Tuesday) @ VISION. PLEASE CONTACT PATRICIA IF THERE ARE PROBLEMS IN YOUR GROUP.

3 Overview of the Assessment

Every individual or pair will develop two algorithms:

- Evolve a MLP (Multi-Layer Perceptron) artificial neural network using a Genetic Algorithm (GA) to minimise a set of continuous functions where no prior knowledge about the functions is available to the algorithm. These functions represent various types of difficulties observed in real-world problems like ill-conditioning, multimodality, global structure, and separability.

  You should choose the MLP parameters to be evolved: the weights, the activation functions or both. You could evolve the MLP topology if you want.

  For the chosen genetic algorithm, you are expected to try different population sizes, selection methods and so on. You should also include other advanced characteristics that you can find in the literature in order to improve its performance. The complexity of your algorithms will be taken into account during the assessment of your work.

- An algorithm to create an Evolutionary Programming (EP) approach only without the MLP (ANN) and run the algorithm using COCO. Remember that to use a GA as a EP you just need to remove the crossover operator.

  You should run your algorithm several times, each using a different configuration of parameters (population size, selection mechanism, mutation operator...). Create a table with a ranking of your attempts, the variations in the parameters and the performances over the functions tested in the blackbox. At least 5 improvements in your algorithm are required to be included in the table.

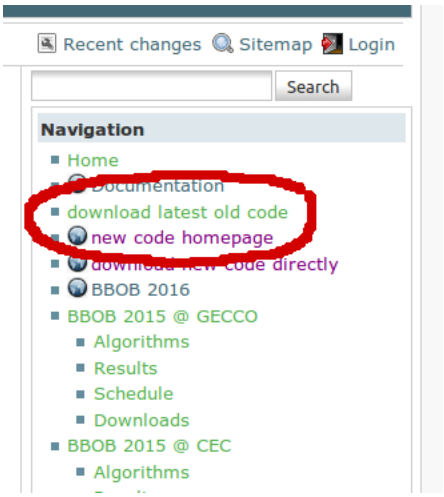
By means of the COCO blackbox you will be able to have a concrete measurement of the level of successfulness over a set of 24 noiseless continuous functions [2]. Once you generate a new stable version of your algorithm you can compare it with previous approaches to see how your algorithm behaves in relation to the previous ones. The performance of your most promising algorithms and these previous comparisons should be used within your report.

4 What you need to do

1. Go to the COCO web page:

   `http://coco.gforge.inria.fr/`

2. Go to the blackbox code by clicking in the following link:

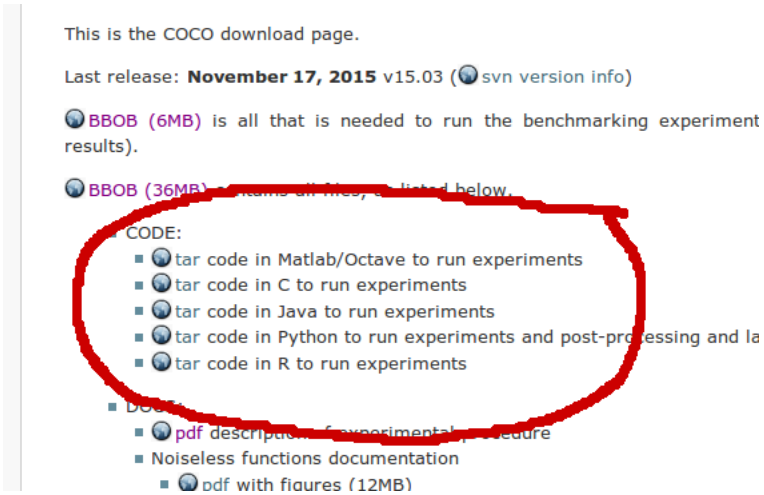


3. Decide the language code you want to implement your algorithms in. COCO blackbox offers an interface in five different languages:

   - Matlab/Octave
   - C/C++
   - Python
   - R
   - Java

   Post-processing tools provided by COCO, however are only implemented in Python. In this document a step-by-step example will be explained in Matlab. Limited support will be given in other languages. However, documentation regarding other approaches can be found in COCO web page.

4. After you know in which language you want to work, download the compressed *tar* file in the following links:



5. Uncompress the file. In Matlab the following list of files will be shown:

   - **benchmarkinfos.txt**
     Here a short description of the noiseless and noisy functions will be depicted. Each of them are grouped according to their different nature and level of complexity in different types: Separate, Low or moderate condition, High condition, Multi-modal and Multi-modal with weak global structure. Notice that this coursework will be **only** tested in noiseless functions (ranged from 1 to 24).
   - **benchmarks.m**
     Do not modify this file.
   - **benchmarksnoisy.m**
     You can ignore this file.
   - **exampleexperiment.m**
     You need to modify this file. This Matlab script launches a quick but complete experiment using the random MY_OPTIMIZER optimizer. You will use is to generate your results afterwards.
   - **exampletiming.m**
     This Matlab script launches the timing experiment. You can ignore this file.
   - **fgeneric.m**
     Do not modify this file.
   - **LICENSE.txt**
   - **MY_OPTIMIZER.m**
     You need to modify this file. This is a trivial algorithm that you can use to test the environment and see how you should configure your algorithms. Later on you need to include here your own code.
   - **README.txt**
     Instructions on how to run an experiment.

6. Create your own optimisers, i.e algorithms: Name your script MY_OPTIMIZER.m or change its call in **exampleexperiment.m**. This last function will set up everything for your optimizer. Your algorithm should have four arguments in the following order:

   - the evaluation function
   - the dimensionality of the search space of the problem D. All functions are scalable with respect to this dimension D.
   - the target value to reach
   - the maximum number of evaluations

   I suggest you to take a look to the example COCO provides within **MY_OPTIMIZER.m** script to see how to deal with these parameters.

7. Edit the script **exampleexperiment.m** and replace the strings "PUT_..." with the correct names of the folders where you are going to run the Matlab experiment.

8. Run the **exampleexperiment.m** script to collect your results.

9. After you run the experiment, the blackbox generates a set of folders, one for each evaluated function. In every folder, the function creates groups of files in 'dat' format storing the result of each instance of the problem tested *Ntrial* number of times. You do not need to understand these 'raw' results. The python scripts described later will create for you tables and figures that are easier to interpret and understand.

Another important point is that the blackbox code must not be **never** externally modified.

4.1 Running the Benchmark

The created algorithm will be run on a testbed of benchmark functions to be minimized. On each function and for each dimensionality a number of trials defined in the variable *Ntrial* trials are carried out. Different function instances will be used. This is specified in the variable *dimensions* which is defined by default with the values $dimensions = [2, 3, 5, 10, 20, 40]$. You can modify this array if you want to have a fast execution for testing, removing temporally the most demanding dimensions.

Once you have created your own algorithm, it will be run on all the continuous functions of the testbed under consideration.These functions are given as an argument to the algorithm. You can find information in each of the functions in the COCO web page. The evaluation functions provided by the black-box return a fitness as a single value. The common settings however must be identical for all benchmark functions and for all function instances in order to allow the final comparison.

Different parameter settings in your algorithm should be treated as 'different algorithms' on the entire testbed. In order to combine different parameter settings, one might use either multiple runs with different parameters.

4.2 Post Processing: how to generate your tables and figures

Python scripts are provided by COCO to produce tables and figures reporting the outcome of the benchmarking experiment. Scripts for post processing are located in the folder **python/bbob_pproc**.

Documentation about post processing in COCO can be found in:

`http://coco.lri.fr/COCOdoc/postprocessing.html`

and information about the provided API:

`http://coco.lri.fr/COCOdoc/bbob_pproc.html`

a guide on how to run the scripts to generate the outcome from the data generated by the blackbox, comparison among algorithms and how to use the LaTeX templates can be found in the following link:

`http://coco.lri.fr/COCOdoc/pp_commandline.html`

4.3 Where to store your code & documents: GibHub repository

The coding of your project should be uploaded to GibHub, the open source platform that allows version control and collaboration. All the information you use to code your algorithm should be allocated there.

`https://github.com/`

GibHub is very well documented software. Here is the link to an introductory tutorial:

`https://guides.github.com/activities/hello-world/`

4.4 Data to Provide

Each individual or pair will be asked to hand in at the MACS School Office on the 11th of November before 3pm the following information:

- A hard copy of a 2000 words report on your results and program development rationale (please check marking scheme below). This document should include:
  - Introduction to your algorithm and a general description.
  - The setup of the experiments and the algorithm settings that should be described thoroughly, with a separate section for the different configurations of the algorithms you are going to work on.
  - Results
  - You should provide a discussion of the pros and cons of the different configurations of the algorithms compared to each other.
  - Conclusions
- In the report it should be clearly specified the link to your GitHub project that it should include:
  1. the program code
  2. the results gathered from your experiments
  3. and an electronic copy in PDF format of your report

  Note: The code you will provide in GibHub should allow us to run your algorithm with no bugs and generate sensible results that should match the ones you reported in the submitted data.

4.5 Marking scheme

This assessment scores 30% of the overall course assessment.

A - REPORT MARKS					
MARKS	$A(100\%)$	$B(< 80\%)$	$C(< 60\%)$	$D(< 40\%)$	**Final**
Content					
40	Very clear; make use of relevant material. Strong evidence of use of the techniques learned during the course.	Clear; make use of some relevant material. Some evidence of use of the techniques learned during the course.	Not so clear; some relevant material is missing. No evidence of use of the techniques learned during the course.	Unclear; no relevant material. No evidence of use of the techniques learned during the course;	
Discussion					
40	Strong evidence of reading and thought.	Evidence of reading and thought.	Some evidence of reading and thought.	No evidence of reading and thought	
App/Style					
15	Report very well structured and divided into sections; use of font Arial, size 12.	Report well structured and divided into sections; use of font Arial, size 12.	Report structured and but not divided into sections; use of different fonts and sizes.	No structure and no division into sections; use of different fonts and sizes.	
References					
5	Perfect use of the Harvard Referencing Style	Use of the Harvard Referencing Style	Mix use of referencing styles.	No referencing style.	
Total A = 100					

B - CODE MARKS					
MARKS	$A(100\%)$	$B(< 80\%)$	$C(< 60\%)$	$D(< 40\%)$	**Final**
Development					
50	Very good implementation and use of the techniques learned during the course; creativity.	Good implementation and use of the techniques learned during the course; some creativity.	Fair implementation and use of the techniques learned during the course; some creativity.	No implementation or evidence of use of the techniques learned during the course; no creativity.	
Performance					
50	Excellent performance of the algorithms. Parameters were fully explored to achieve excellent results.	Good performance of the algorithms. Parameters were explored to achieve good results.	Fair performance of the algorithms. Parameters were fairly explored to achieve basic results.	Poor performance of the algorithm or no results were retrieved from the blackbox. Few of no parameters were explored.	
Total B = 100					
Total (A + B)/2 = 100					

References

[1] N. H. A. Finck and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical report, Technical Report 2009/21, Research Center PPE, 2010.

[2] N. Hansen, S. Finck, R. Ros, and A. Auger. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions.* PhD thesis, INRIA, 2009.