Pontifical Catholic University of Parana
Polytechnic School
Industrial and Systems Engineering Graduate Program
Course: **Continuous Optimization based on Direct Methods**
Professors: Leandro dos Santos Coelho
Roberto Zanetti Freire

Phase I: Study of the Optimization Methods

At first, students should implement and try to understand the theory behind the proposed algorithms. The following optimization methods must be studied:

Classical Approaches:

- Nelder-Mead (*available in MATLAB*);
- Hooke-Jeeves;
- Implicit Filtering;
- Multidirectional Search;
- Pattern Search (*available in MATLAB*);

Additional Technique:

- *Optimization technique of your choice;

The classical methods can be found in the following reference:
KELLEY, C. T. "***Iterative Methods for Optimization".*** Frontiers in Applied Mathematics, 18, 1999. [http://www4.ncsu.edu/~ctk/matlab_darts.html]. For the additional technique*, students must select a distinct optimization technique when compared to the previous mentioned classical approaches. Additionally, this technique must different to the Particle Swarm Optimization Technique (PSO) that will be provide as an example. The additional method selection must be justified in the final report, and it does not need to be classified as a classical approach.

Phase II: Problems

In both engineering and computer science, optimization methods are applied to solve real world problems. However, to test these techniques, some mathematical functions can be adopted. Here, the following functions must be optimized:

- Sphere Function (F_1);
- Rotated High Conditioned Elliptic Function (F_2)
- Rotated Bent Cigar Function (F_3);
- Rotated Discus Function (F_4);
- Different Powers Function (F_5);

The detailed description of each problem (function) and implementation details are available in the "**Problems_CEC2013.pdf**" file at folder "Evaluation". These problems were adopted for the "***single objective real-parameter optimization***" competition of the ***Annual IEEE Congress on Evolutionary Computation (CEC 2013), held in Mexico***.

More information about the problems could be obtained in at [http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm]. The function number (F_x) is directly related to function number in the "**Problems_CEC2013.pdf**" file.

Example:

There is an example about how to use the implemented functions that were adopted in the CEC 2013 competition. This example is available at the subfolder "Example", inside folder "Evaluation".

Evaluation Criteria:

All the optimization methods (Nelder-Mead, Hooke-Jeeves, Implicit Filtering, Multidirectional Search, DIRECT, Pattern Search, and the additional technique of your choice should be evaluated according to the specification reported at "**Problems_CEC2013.pdf**". However, this course will limit the evaluation criteria to:

- Dimensions: D = 10 e D = 30;
- Runs per problem: 51;
- Initialization: Uniform random initialization within the search space. Random seed is based on time. In MATLAB users can perform the initialization using: ('state', sum(100*clock));
- Search space: x = [-100, 100]^D;
- Maximum number of function evaluations (MaxFES): 10000*D (MaxFES for 10D = 100000, for 30D = 300000);
- Termination criteria: terminate when reaching MaxFES or the error value is smaller than 10^{-8}.
- **Global Optimum**: All problems have the global optimum within the given bounds and there is no need to perform search outside of the given bounds for these problems, see F_i *= $F_i(x^*)$ in Table I available in "**Problems_CEC2013.pdf**" to verify the global optimum of each required function.

Results:

Results of the optimization for both 10 and 30 dimensions associated to each problems must follow the procedures presented in the sequence. These procedures are quite similar to CEC 2013 competition but are not the same. The idea is to include in the final report items 1 to 4 presented bellow.

1. Error and Convergence

Record function error value *(Fi(x) – Fi(x*))* after (0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)*MaxFES for each run. In this case, 14 error values must be recorded for each function for each run. Sort the error values achieved after MaxFES in 51 runs from the smallest (best) to the largest (worst) and present them in Table 1 format. Best, worst, mean, median and standard deviation values of function error values after 51 runs must be provided.

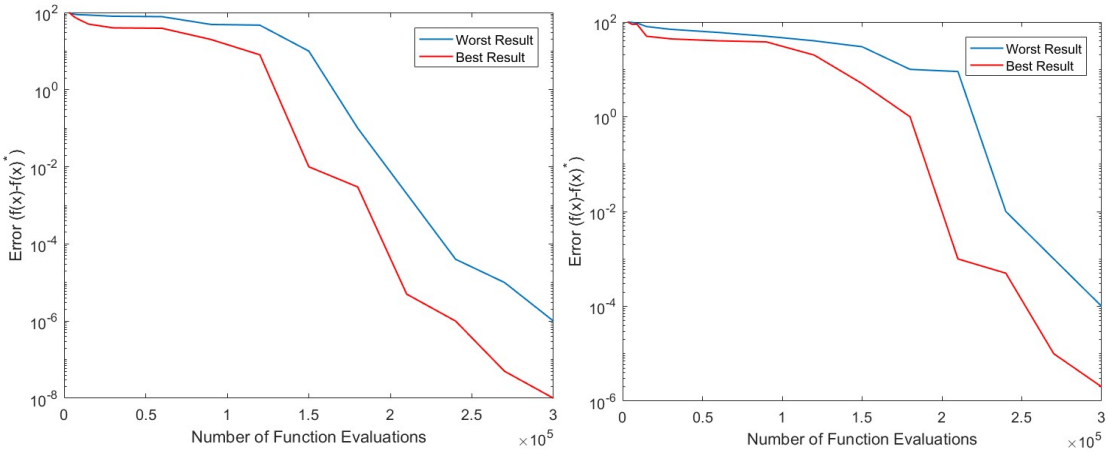
Notice that error values smaller than 10^{-8} should be substitute by zero, and that two tables should be provided, considering 10 and 30 dimensions.

Table 1. Error: method "X" – dimension "Y" after 51 runs.

Problem	Best	Mean	Worst	Standard Deviation
F_1				
F_2				
F_3				
F_4				
F_5				

For both best and worst solution, plot the convergence graphic of all 14 error values. The vertical axis has to be set as the error *Fi(x) – Fi(x*)*, and at the horizontal axis the number of function evaluations following the rule 0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)*MaxFES. The convergence should be evaluated for both 10 and 30 dimensions.

Figure 1 shows and example of convergence graphical analysis that must be included in the final report. Note that the vertical axis is presented in logarithm scale.



(a) (b)

Figure 01. Example of convergence analysis of method "X" applied to problem F_x after 51 runs, best and worst results: (a) dimension equal 10; (b) dimension equal 30.

2. Precision

Store the number of function evaluations in each one of the 51 runs in order to evaluate the technique precision. Results should be provided according to Table 2. The idea is that methods with high precision will reach almost the same number of function evaluations.

Table 2. Method precision: Method "X" after 51 runs.

Problem	Mean of the function evaluation number ± standard deviation	Mean of the function evaluation number ± standard deviation
F_1	*Example: 300230 ± 1200*	
F_2		
F_3		
F_4		
F_5		

3. Success Rate and Performance

Success rate can be defined as the number of times that one method find the global optimum prior to reach the maximum number of function evaluations (MaxFES), divided by the number of tests performed. Additionally, the performance of one specific method, when applied to a problem, can be evaluated according to the second equation presented in the sequence. Notice that if no successful tests were obtained, the performance will be zero.

$$\text{Success Rate} = \frac{\text{Number of Successful Tests}}{51};$$

$$\text{Performance} = \frac{Mean(\text{Number of Function Evaluations During Successful Tests}) * 51}{\text{Number of Successful Tests}}.$$

Both values described above, should be provided according to Table 3 for both 10 and 30 dimensions.

Table 3. Method success rate and performance: Method "X" – Dimension "Y".

Problem	Success Rate	Performance
F_1		
F_2		
F_3		
F_4		
F_5		

4. Algorithm Complexity

To perform the algorithm complexity analysis follow the steps presented in the sequence:

a. Run the test program described in Figure 2 and follow the procedures provided in the sequence;

```
x = 0.55;
for I = 1:1000000
    x = x + x;
    x=x/2;
    x=x*x;
    x=sqrt(x);
    x=log(x);
    x=exp(x);
    x=x/(x+2);
end
Computing time for the above = T0;
```

Figure 2. Computational code for complexity evaluation.

b. Evaluate the computing time just for function F_5. For 200000 evaluations of a certain dimension *D*, it gives $T1$;

c. The complete computing time for each optimization algorithm with 200000 evaluations of the same *D* function F_5 is $T2$;

d. Execute step **c** five times and get five $T2$ values $\hat{T}2 = mean(T2)$;

e. The complexity of the algorithm is reflected by: $T2$, $T1$, $T0$ and $(\hat{T}2 - T1)/T0$. The algorithm complexities are calculated on 10 and 30 dimensions, to show the algorithm complexity's relationship with dimension. This evaluation also provide sufficient details on the computing system and the programming language used. In step **c**, we execute the complete algorithm five times to accommodate variations in execution time due adaptive nature of some algorithms.
Note that similar programming styles should be used for all $T0$, $T1$ and $T2$. Complete Table 4 to conclude this step.

Table 4. Complexity analysis.

$T0$	Dimension	Method	$T1$	$T2$	$(\hat{T}2 - T1)/T0$
	10	Nelder-Mead			
		Hooke-Jeeves			
		Implicit Filtering			
		Multidirectional Search			
		Pattern Search			
		*Your Choice			
	30	Nelder-Mead			
		Hooke-Jeeves			
		Implicit Filtering			
		Multidirectional Search			
		Pattern Search			

		*Your Choice			

Phase III: Comparison and Analysis of the Optimization Methods

This is the final phase of this course. Results obtained from procedures 1 to 4 of phase II must be included in the final report in a "pdf" format. Codes must also be provided. The deliver will be through Dropbox. Each group must create a subfolder in "Reports and Codes" using the first names of the team. Example: "Leandro_Roberto".

Specification: This work can be delivered in pairs (maximum).

Deadline: December 16th, 2019

Report requirements:

The report must include:

- Reports can be written in English or Portuguese languages;
- An introduction (maximum 2 pages) and a conclusion (maximum 2 pages);
- A brief description about each optimization method (maximum 1 page dedicated to each one);
- Brief description about each problem (functions to be optimized);
- Results comparing all the optimization techniques associated to each problem according to Phase II, procedures 1 to 4;
- The report file must be in "pdf" format following one of these nomenclatures: "Student_Full_Name.pdf" or "Student_01_Full_Name_-_Student_02_Full_Name.pdf".
- Reports delivered out of these specifications will have reduced scores.

Computational codes requirements:

Computation codes must be provided in the following form:

- Compacted: "Student_Full_Name_-_Codes.zip" or "rar" file;
- Inside the compacted file, a "instructions.txt" file must be included to explain the procedures to run and to reproduce the results presented in the report. It is suggested to create a single file that run all the procedures. Of course additional functions (distinct files) should be included to organize your computational code. The MATLAB version adopted by the student should also be addressed in this file;
- Computational codes must be provided in MATLAB language.