**PROS Source Code Instructions v1.0**by Vagelis Plevris, Nikos Bakas and German Solorzano

v1.00, April 10, 2021

**Pure Random Orthogonal Search (PROS):  
A Plain and Elegant Parameterless Algorithm for Global Optimization Instructions**

The version of MATLAB used is R2020b.

The package has the following structure:

* File: **plot\_function.m**
* File: **PROS Instructions.docx**
* File: **test\_functions\_time.m**
* Folder: **mainprog**, containing the following:
  + File: **gaoutputfcn.m**
  + File: **main.m**
  + File: **pswplotranges.m**
  + Folder: **functions**, containing the following:
    1. File: ackley\_func.m
    2. File: alpine1\_func.m
    3. File: drop\_wave\_func.m
    4. File: ellipsoid\_func.m
    5. File: griewank\_func.m
    6. File: happycat\_func.m
    7. File: hgbat\_func.m
    8. File: quintic\_func.m
    9. File: rastrigin\_func.m
    10. File: sphere\_func.m
    11. File: sumpow\_func.m
    12. File: weierstrass\_func.m

File: **PROS Instructions.docx**

This MS Word file with the instructions.

Folder: **mainprog\functions**

The folder contains the following 12 functions (shown here in alphabetical order based on file name):

|  |  |  |
| --- | --- | --- |
| **No** | **Function** | **File name** |
| 1 | F08 Ackley's function | ackley\_func.m |
| 2 | F07 Alpine 1 function | alpine1\_func |
| 3 | F05 Drop-Wave function | drop\_wave\_func |
| 4 | F02 Ellipsoid function | ellipsoid\_func |
| 5 | F09 Griewank's function | griewank\_func |
| 6 | F11 HappyCat function | happycat\_func |
| 7 | F12 HGBat function | hgbat\_func |
| 8 | F04 Quintic function | quintic\_func |
| 9 | F10 Rastrigin's function | rastrigin\_func |
| 10 | F01 Sphere function | sphere\_func |
| 11 | F03 Sum of Different Powers function | sumpow\_func |
| 12 | F06 Weierstrass function | weierstrass\_func |

These functions are called from other programs, when the calculation of the value of an objective function is needed. The functions are used in the PROS code. The functions are optimized for use in MATLAB, avoiding for loops and doing the calculations based on vector operations.

File: **test\_functions\_time.m**

**Instructions**: Simply run the file with MATLAB and you will get the figure with the results.

**Prerequisites**: The file needs the folder **functions** (containing the 12 functions) to be in the **mainprog** directory.

It calculates the values of the 12 objective functions **NumRuns** times for D = 5, 10, 30 and 50, records the time needed (in sec) and plots the results in a figure (y-axis is in logarithmic scale).

In the PROS paper, we use NumRuns=100,000 but in the code provided here we have set this number to 1000 for faster results. You can adjust this using the following line of code:

NumRuns=1e3; %Times to calculate each function

Note that different runs of the **test\_functions\_time.m** program will give slightly different results each time, while the results depend also on the hardware (computer) used. The results presented in the paper were taken running the program on a computer with Windows 10, equipped with an Intel i9-8950HK CPU @ 2.90GHz and 32 GB RAM.

The figure in the paper is the following (for NumRuns=100,000):



File: **mainprog\gaoutputfcn.m**

**Instructions**: No need to run this program. This program is called from the **main.m** PROS file. It is the **gaoutputfcn** GA function of MATLAB with some modifications needed to record the results of GA.

File: **mainprog\pswplotranges.m**

**Instructions**: No need to run this program. This program is called from the **main.m** PROS file. It is the **pswplotranges** PSO function of MATLAB with some modifications needed to record the results of PSO.

File: **plot\_function.m**

**Instructions**: Run the file with MATLAB and you will get the figure with the plot of a function in 2D. You need to adjust the following lines of code (highlighted in red):

fun=@**sphere\_func**; % \*\*\* SELECT THE FUNCTION TO PLOT HERE \*\*\*

OptiLocation=[**0 0**] % \*\*\* SELECT THE LOCATION OF THE OPTIMUM HERE \*\*\*

% Location of the optimum, to highlight in the figure

% For the case of F04 it is OptiLocation=[-1 -1 ; 2 2] as this function

% has two optima. Both points are plotted this way

Optimum=**0** % Value of the optimum, specified manually

% It is zero for all 12 function considered

Bounds=**10**; % \*\*\* SELECT THE BOUND OF THE PLOT HERE FOR X1, X2 \*\*\*

**Prerequisites**: The file needs the folder **functions** (containing the 12 functions) to be in the **mainprog** directory.

The program plots the **fun** function (in the above example, it is the **sphere\_func**) from **-Bound** to **Bound** in both directions (*x*1, *x*2) (in the above example **[-10, 10**]). It also plots the location of the optimum which is defined in the **OptiLocation** variable (in the above example **(0, 0)**) with a value of **Optimum** (in the above example, **0**).

The code gives the following result:



File: **mainprog\main.m**

**Instructions**: Run the file with MATLAB and you will get the 4 figures with the convergence histories of all five algorithms (GA, PSO, DE, PRS, PROS) for *D*=5, 10, 30 and 50.

**Prerequisites**: The file needs the following to be in the same directory:

* Folder **functions** (containing the 12 functions) to be in the same directory.
* File **gaoutputfcn.m**
* File **pswplotranges.m**

This is the main file of the PROS package. It includes the PROS optimizer together with four other optimizers for comparison purposes.

Note that in the PROS paper we use 10 runs (**NumRuns**=10), i.e. we run each optimization problem 10 times and we take the average of the results in the end. This takes too much time (it can take a few hours to run all the problems so many times). In this version of the code, we use **NumRuns**=1 so that each optimization problem is run 1 time only, for faster results. You can adjust this using the following line:

AllProblems{iDim}=Problems;

AllTimes{iDim}=Times;

Running the program will give you 4 figures with the convergence histories. In the end, all the results of the optimization problems are stored in the following two variables:

**AllProblems**: Holds the optimum locations and the optimum (best) values of the objective functions, for each algorithm and each problem

**AllTimes**: Holds the time needed for each algorithm and each problem

* AllProblems{1} holds the results for D=5
* AllProblems{2} holds the results for D=10
* AllProblems{3} holds the results for D=30
* AllProblems{4} holds the results for D=50

And similarly for the variable AllTimes

For example, see the following commands:

mean(AllProblems{2}(6).fvalGA,1)

Will give you the average (of the NumRuns times) of the best objective function for the 6th problem for the GA algorithm, for *D*=10.

mean(AllProblems{3}(8).fvalPSO,1)

Will give you the average (of the NumRuns times) of the best objective function for the 8th problem for the PSO algorithm, for *D*=30.

mean(AllProblems{1}(3).fvalDE,1)

Will give you the average (of the NumRuns times) of the best objective function for the 3rd problem for the DE algorithm, for *D*=5.

mean(AllProblems{4}(11).fvalPRS,1)

Will give you the average (of the NumRuns times) of the best objective function for the 11th problem for the PRS algorithm, for *D*=50.

mean(AllProblems{1}(5).fvalPROS,1)

Will give you the average (of the NumRuns times) of the best objective function for the 5th problem for the PROS algorithm, for *D*=5.

AllProblems{1}(5).xPROS

Will give you NumRuns vectors (1x5 each) with the location of the optima for the 5th problem for the PROS algorithm, for *D*=5.

AllProblems{3}(7).xGA

Will give you NumRuns vectors (1x30 each) with the location of the optima for the 7th problem for the GA algorithm, for *D*=30.