How to use the code of Imperialist Competitive Algorithm (ICA)

Control and Intelligent Processing Center of Excellence
University of Tehran, Iran.

Esmaeil Atashpaz Gargari

Email: e.atshpaz@ece.ut.ac.ir , atashpaz_e@yahoo.com
Home Page: http://atashpaz.com/

1) A Brief Description

ICA is a novel global search heuristic that uses imperialism and imperialistic competition process as a source of inspiration. We have developed this algorithm at Control and Intelligent Processing Center of Excellence, ECE School of University of Theran. More information about this algorithm can be found at my home page:

http://atashpaz.com/

2) How to define the Cost Function

Lets assume that we are using the algorithm to find the global minimum of the function
$y = f(x_1, x_2, x_3,\ldots, x_{Nvar})$
where *Nvar* is the number of variables to be optimized, that is the dimension of the optimization problem. In the prepared code, the cost (or inversely the

power) of all the countries are calculated all together. That is, to run the code for finding the global minimum of the function f, we need the function F in the following form:

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_{NC} \end{bmatrix} = F\left(\begin{bmatrix} Country_1 \\ Country_2 \\ . \\ . \\ . \\ Country_{NC} \end{bmatrix}\right) = \begin{bmatrix} f(Country_1) \\ f(Country_2) \\ . \\ . \\ . \\ f(Country_{NC}) \end{bmatrix}$$

For example if we want to find the global minimum of the function $f(x_1, x_2) = x_1.\sin(4x_1) + 1.1x_2 \sin(2x_2)$

The function written in MATLAB should be in the following form:

```
function F1=CostFunction(x)
F1=x(:,1).*sin(4*x(:,1))+1.1*x(:,2).*sin(2*x(:,2));
```

If you have a function written in MATLAB in the form that evaluates the cost function for all of the countries one by one, you can easily change it to the mentioned form by adding a `for` loop to your function. For Example, for the above function the cost function is written as following.

```
function f=CostFunction(x)
f=x(1,1)*sin(4*x(1,1))+1.1*x(1,2)*sin(2*x(1,2));
```

We can rewrite the above function in the following form

```
function F2=CostFunction(x)
for i = 1:size(x,1)
     F2(i)=x(i,1)*sin(4*x(i,1))+1.1*x(i,2)*sin(2*x(i,2));
end
```

Functions `F1` and `F2` are the same and the Colonial Competitive Algorithm can be easily applied to find their global minimum.

3) Initialization of the Parameters

There are three sets of parameters to be set. The most important parameters that should be initialized are listed in the below table along with a brief description of each parameter.

a) Problem Statement

```
%% Problem Statement
ProblemParams.CostFuncName = 'BenchmarkFunction';    % You should state
the name of your cost function here.
ProblemParams.CostFuncExtraParams = 6;
ProblemParams.NPar = 30;                             % Number of
optimization variables of your objective function. "NPar" is the
dimention of the optimization problem.
ProblemParams.VarMin = -6;                           % Lower limit of the
optimization parameters. You can state the limit in two ways. 1)   2)
ProblemParams.VarMax = 6;                          % Lower limit of the
optimization parameters. You can state the limit in two ways. 1)   2)
```

b) Algorithmic Parameters

```
%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 200;                % Number of initial
countries.
AlgorithmParams.NumOfInitialImperialists = 8;      % Number of Initial
Imperialists.
AlgorithmParams.NumOfDecades = 2000;
AlgorithmParams.RevolutionRate = 0.3;                % Revolution is the
process in which the socio-political characteristics of a country
change suddenly.
AlgorithmParams.AssimilationCoefficient = 2;         % In the original
paper assimilation coefficient is shown by "beta".
AlgorithmParams.AssimilationAngleCoefficient = .5;  % In the original
paper assimilation angle coefficient is shown by "gama".
AlgorithmParams.Zeta = 0.02;                         % Total Cost of
Empire = Cost of Imperialist + Zeta * mean(Cost of All Colonies);
AlgorithmParams.DampRatio = 0.99;

AlgorithmParams.StopIfJustOneEmpire = false;         % Use "true" to
stop the algorithm when just one empire is remaining. Use "false" to
continue the algorithm.
AlgorithmParams.UnitingThreshold = 0.02;             % The percent of
Search Space Size, which enables the uniting process of two Empires.
```

c) Display Setting

```
%% Display Setting
DisplayParams.PlotEmpires = false;    % "true" to plot. "false" to
cancel ploting.

DisplayParams.PlotCost = true;    % "true" to plot. "false"
```