Basic Genetic Algorithm

Xavier Blasco
Grupo de Control Predictivo y Optimización
Instituto Universitario de Automática e Informática Industrial
Universitat Politècnica de València
xblasco@isa.upv.es
`http://cpoh.upv.es`

Description

These scritps implement the version of a Genetic Algorithm decribed in http://hdl.handle.net/10251/15995 *Control predictivo basado en modelos mediante técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables. F. Xavier Blasco Ferragud. PhD Tesis 1999 (in Spanish). Editorial UPV. ISBN 84-699-5429-6*

The basic functions are:

- `ga.m` : Main program.
- `gaiteration.m` : Optional user task executed at the end of each iteration. It can be modified by the user, usually to show some information during algorithm execution, or to make backups during algorithm execution when computational cost is high.
- `garesults.m` : Optional user task executed when the algorithm ends. It can be modified by the user to present results of the algorithm.

The algorithm needs a particular data structure supplyied by a variable:

```
% Parameter to set for a specific problem.
gaDat.Objfun      % Name of the objective function.
gaDat.FieldD      % Upper and lower bounds of the search space.
% Optional parameters. If the user hasn't set it there is a default value.
gaDat.MAXGEN      % Number of iterations (generations).
gaDat.NIND        % Number of individuals in the population.
gaDat.alfa        % Crossover parameter.
gaDat.Pc          % Crossover probability.
gaDat.Pm          % Mutation probability.
gaDat.ObjfunPar   % Additional parameters of the objective function.
gaDat.indini      % Individuals assigned by the user in the first iteration.
% Internal parameters. Only the algorithm can change it. The user can see their value ...
    but never change.
gaDat.NVAR        % Number of variables (dimension of search space)
gaDat.Chrom       % Population
gaDat.ObjV        % Value of the objective function for each individual
gaDat.rf          % Ranking vector
gaDat.gen         % Generation counter
gaDat.xmin        % Solution that produce the minimal value of the objective function
gaDat.fxmin       % Minimal value of the objective function
gaDat.xmingen     % Array with the individuals that produce the minimal value at each ...
    iteration
gaDat.fxmingen    % Array with the minimal objective function vualue
```

Of course, it is necessary to define the objective function in a separated script.

Examples of use

Three examples of use are supplied with the genetic algorithm. The first one illustrated a basic use of the functions. The problem is the 2D Schwefel function 1. The script for the Schwefel function is:

```
function value=objfun_schwefel(x)
%
% Schwefel function
%
% Function : f(X1,X2)=-x1*sin(sqrt(abs(x1)))-x2*sin(sqrt(abs(x2)))
% Search space constraints : -500≤xi≤500
%
% Solution: min(f)=f6(420.9687,420.9687)=-837.9658
%

value=-x(1).*sin(abs(x(1)).^0.5)-x(2).*sin(abs(x(2)).^0.5);
```

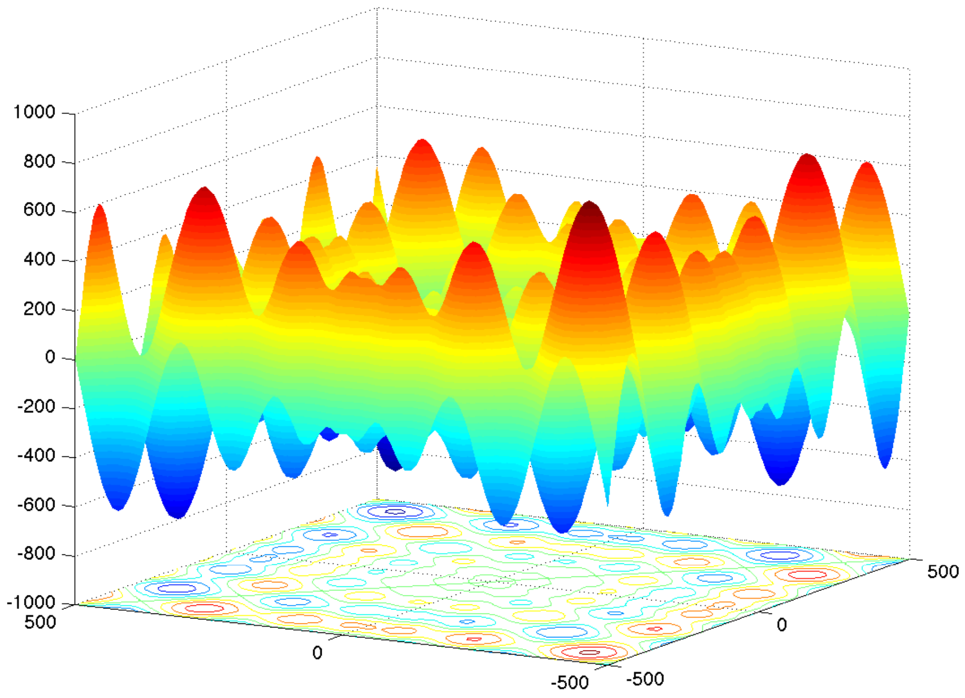

Figura 1: Schwefel function.

`gaiteration.m` and mcodegaresults.m are used to show partial and final information about algorithm progress.

```
function gaiteracion(gaDat)
%  Optional user task executed at the end of each iteration
%

% For instance, results of the iteration
 disp('------------------------------------------------')
 disp(['Iteration: ' num2str(gaDat.gen)])
 disp(['   xmin: ' mat2str(gaDat.xmin) ' -- f(xmin): ',num2str(gaDat.fxmin)])
%% --------------------------------------------------------
```

```
function garesults(gaDat)
% Optional user task executed when the algorithm ends

% For instance, final result
disp('------------------------------------------------')
disp('######   RESULT   #########')
disp(['   Objective function for xmin: ' num2str(gaDat.fxmin)])
disp(['   xmin: ' mat2str(gaDat.xmin)])
disp('------------------------------------------------')
```

`gaExample.m` is the script for a basic use, where only the upper and lower bounds are defined.

```
%% Basic GA parameters
gaDat.Objfun='objfun_schwefel';
lb=[-500 -500];
ub=[500 500];
gaDat.FieldD=[lb; ub];
% Execute GA
gaDat=ga(gaDat);
```

The results are stored in the variables `gaDat.xmin` and `gaDat.fxmin`.

The second example `gaExample2.m` shows how to use other optional parameters, in this case the number of generation and individual of the population are selected by the user.

```
%% Defining other parameters MAXGEN  and  NIND

gaDat2.Objfun='objfun_schwefel';
lb=[-500 -500];
ub=[500 500];
gaDat2.FieldD=[lb; ub];
gaDat2.MAXGEN=50;
gaDat2.NIND=400;  % increasing number individuals produce better solution
                  % but higher computational cost
% Execute GA
gaDat2=ga(gaDat2);
% Result are in
gaDat2.xmin
gaDat2.fxmin
```

The last example `gaExample3.m` shows how to pass additional parameters to the objective function. In this case the parameters are use only to waste time.

The objective function is:

```
function valor=objfun_schwefel_p(x,param)
%
% Schwefel function
%
% Function : f(X1,X2)=-x1*sin(sqrt(abs(x1)))-x2*sin(sqrt(abs(x2)))
% Search space constraints : -500<=xi<=500
%
% Solution: min(f)=f6(420.9687,420.9687)=-837.9658
%
% param: useless parameter only to show how to pass parameters
% using parameters for wasting time ;)

for i=1:param.n
    valor=param.p*param.p;
end
valor=-x(1).*sin(abs(x(1)).^0.5)-x(2).*sin(abs(x(2)).^0.5);
```

`gaExample3.m` is:

```
%% Passing parameters to the objective function

gaDat3.Objfun='objfun_schwefel_p';
lb=[-500 -500];
ub=[500 500];
gaDat3.FieldD=[lb; ub];
% objective function parameters (in this example only used for wasting
% time), see objfun_schwefel_p.m for details.
p.n=1000000;
p.p=50000;
gaDat3.ObjfunPar=p;
% Execute GA
gaDat3=ga(gaDat3);
```