evMOGA - User Guide

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

Description

ev-MOGA Multiobjective Evolutionary Algorithm has been developed by the Predictive Control and Heuristic optimization Group (CPOH) at Universitat Politècnica de València (Spain). ev-MOGA is an elitist multi-objective evolutionary algorithm based on the concept of epsilon dominance. ev-MOGA, tries to obtain a good approximation to the Pareto Front in a smart distributed manner with limited memory resources. It also adjusts the limits of the Pareto front dynamically.

Details about ev-MOGA are described in (please, cite this algorithm as):

[1] M. Martínez, J.M. Herrero, J. Sanchis, X. Blasco and S. García-Nieto. Applied Pareto multi-objective optimization by stochastic solvers. Engineering Applications of Artificial Intelligence. Vol. 22 pp. 455 - 465, 2009 (ISSN:0952-1976).

The algorithm is also described in:

[2] J.M. Herrero, M. Martínez, J. Sanchis and X. Blasco. Well-Distributed Pareto Front by Using the epsilon-MOGA Evolutionary Algorithm. Lecture Notes in Computer Science, 4507, pp. 292-299, 2007. Springer-Verlag. (ISSN: 0302-9743)

The basic functions are:

- `evMOGA.m` : Main program.
- `evMOGAiteration.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.
- `evMOGAresults.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 supplied by a variable:

```
%% Parameters that have to (or can) be tuned by user
%% Minimal parameters set: MOP description (have to be tuned by user)
eMOGA.objfun            % m-function name for objectives computation
eMOGA.objfun_dim        % Objective space dimension
eMOGA.searchspaceUB     % Search space upper bound
eMOGA.searchspaceLB     % Search space lower bound
%% Additional MOP parameters (can be tuned by user)
eMOGA.param             % Objectives additional parameter
                        %   empty if not used
                        %   default value: []
%% Algorithm parameters (can be tuned by user)
eMOGA.Nind_P            % Individuals for the P population
                        %   defaultvalue:: 100
eMOGA.Generations       % Number of generations
eMOGA.n_div             % Number of division for each dimension
                        % to build the objective space grid
                        %   default value: 50 divisions per dim
eMOGA.Nind_GA           % Individuals for aux population GA
                        %   have to be an even number
                        %   defaultvalue:: 8
eMOGA.subpobIni         % Initial subpopulation
                        %   empty if not used
                        %   default value: []
eMOGA.dd_ini            % Parameter used for crossover
                        %   defaultvalue:: 0.25
eMOGA.dd_fin            % Parameter used for crossover
                        %   defaultvalue:: 0.1
eMOGA.Pm                % Mutation Probability
                        %   defaultvalue:: 0.2
eMOGA.Sigma_Pm_ini      % Gaussian Mutation Sigma_Pm
                        %   defaultvalue:: 20
eMOGA.Sigma_Pm_fin      % Gaussian Mutation Sigma_Pm
                        %   defaultvalue:: 0.1
```

Other internal parameters exist but they have not to be set by user.

```
  %% Internal parameters
  eMOGA.searchSpace_dim   % Search space dimension
  eMOGA.epsilon           % box size
  eMOGA.max_f             % maximum objective function values
  eMOGA.min_f             % minimun objective functions values
  eMOGA.ele_P:            % individuals of population P
  eMOGA.coste_P           % objective function values of P
  eMOGA.mod               % dominance parameter
  eMOGA.ele_A             % individuals of population A
  eMOGA.coste_A           % objective function values of A
  eMOGA.box_A             % box for individuals of A
  eMOGA.Nind_A            % size of pareto front
  eMOGA.ele_GA            % individuals of population GA
  eMOGA.coste_GA          % objective function values of GA
  eMOGA.box_GA            % box for individuals of GA
  eMOGA.gen_counter       % generation counter
```

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

Examples of use

Three examples of use are supplied with the evMOGA algorithm. The first one illustrated a basic use. The problem is a test function `mop3.m`, the script for MOP3 problem is:

```
function J=mop3(theta)
%  J=mop3(theta)
%  J=[J1 J2]
%  theta=[theta_1 theta_2]
%  -pi≤theta_i<pi i=1,2

a1=0.5*sin(1)-2*cos(1)+sin(2)-1.5*cos(2);
a2=1.5*sin(1)-cos(1)+2*sin(2)-0.5*cos(2);
b1=0.5*sin(theta(1,1))-2*cos(theta(1,1))+sin(theta(1,2))-1.5*cos(theta(1,2));
b2=1.5*sin(theta(1,1))-cos(theta(1,1))+2*sin(theta(1,2))-0.5*cos(theta(1,2));

J1=(1+(a1-b1)^2+(a2-b2)^2);
J2=((theta(1,1)+3)^2+(theta(1,2)+1)^2);
J=[J1 J2];
```

`evMOGAiteration.m` and `evMOGAresults.m` are used to show partial and final information about algorithm progress.

```
function eMOGA=evMOGAiteration(eMOGA)
%   evMOGA=evMOGAiteration(evMOGA)
%     User tasks to be evaluated at the end of each iteration.
%     For instance to plot partial results, made a backup, etc.

disp(['######### iteration ' num2str(eMOGA.ite)])
disp(['Nind_A=' num2str(eMOGA.Nind_A)])
disp(['Objectives min values: ' num2str(eMOGA.min_f)])
```

```
function eMOGA=evMOGAresults(eMOGA)
%   eMOGA=evMOGAresult(eMOGA)
%     User tasks to be evaluated at the end of the algorithm.
%     For instance to plot final results, final analysis, etc.

disp('------------------------------------------------------')
disp('######   RESULT   #########')
disp(['Nind_A=' num2str(eMOGA.Nind_A)])
disp(['Objectives min values: ' num2str(eMOGA.min_f)])
disp('------------------------------------------------------')

% Plotting results
load mop3aux          % load true pareto front
ParetoFront=eMOGA.coste_A(1:eMOGA.Nind_A,:);
% Plot true pareto front
plot(pfront(:,1),pfront(:,2),'.r')
% Pareto front obtained by evMOGA
hold on,plot(ParetoFront(:,1),ParetoFront(:,2),'o')
```

`evMOGA_example1.m` is the script for a basic use, where only the dimensions of objective and parameters spaces with its upper and lower bounds. The other parameters are set to their default values.

```
%% evMOGA example 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Minimal algorithm parameters set (problem characteristics)
eMOGA.objfun='mop3';             % m-function name for objectives computation
eMOGA.objfun_dim=2;              % Objective space dimension
eMOGA.searchspaceUB=[pi pi];     % Search space upper bound
eMOGA.searchspaceLB=[-pi -pi];   % Search space lower bound

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Algorithm execution
[pfront,pset,eMOGA]=evMOGA(eMOGA);
```

The second example `evMOGA_example2.m` shows how to use other optional parameters, in this case the number of generation, individuals of the population and the size of the grid are selected by the user.

```
%% evMOGA example 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Minimal algorithm parameters set (problem characteristics)
eMOGA.objfun='mop3';            % m-function name for objectives computation
eMOGA.objfun_dim=2;             % Objective space dimension
eMOGA.searchspaceUB=[pi pi];    % Search space upper bound
eMOGA.searchspaceLB=[-pi -pi];  % Search space lower bound
eMOGA.Nind_P= 500;              % Individuals for the P population
eMOGA.Generations= 100;         % Number of generations
eMOGA.n_div= 200;               % Number of division for each dimension

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Algorithm execution
[pfront,pset,eMOGA]=evMOGA(eMOGA);
```

The last example evMOGA_example3.m shows how to pass additional parameters to the objective function. In this case the parameters are only used to waste time. The objective function is:

```
function J=mop3_p(theta,param)
%  J=mop3(theta,param)
%  J=[J1 J2]
%  theta=[theta_1 theta_2]
%  -pi≤theta_i<pi i=1,2

for i=1:param.n
    valor=param.p*param.p;  % wasting time ;)
end

a1=0.5*sin(1)-2*cos(1)+sin(2)-1.5*cos(2);
a2=1.5*sin(1)-cos(1)+2*sin(2)-0.5*cos(2);
b1=0.5*sin(theta(1,1))-2*cos(theta(1,1))+sin(theta(1,2))-1.5*cos(theta(1,2));
b2=1.5*sin(theta(1,1))-cos(theta(1,1))+2*sin(theta(1,2))-0.5*cos(theta(1,2));
J1=(1+(a1-b1)^2+(a2-b2)^2);
J2=((theta(1,1)+3).^2+(theta(1,2)+1).^2);
J=[J1 J2];
```

evMOGA_example3.m is:

```
%% evMOGA example 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Minimal algorithm parameters set (problem characteristics)
eMOGA.objfun='mop3_p';          % m-function name for objectives computation
eMOGA.searchspaceUB=[pi pi];    % Search space upper bound
eMOGA.searchspaceLB=[-pi -pi]; % Search space lower bound
eMOGA.objfun_dim=2;             % Objective space dimension
% additional parameter of the objective function
% in this case just to waist time
p.n=1000000;
p.p=50000;
eMOGA.param=p;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Algorithm execution
[pfront,pset,eMOGA]=evMOGA(eMOGA);
```