TP : Tooltbox des Algorithmes Génétiques

Le logiciel 'Tooltbox' des Algorithmes Génétiques est intégré dans l'environnement MATLAB permet de résoudre des problèmes d'optimisation tout en déterminant le minimum de la fonction objective.
Pour cela, deux possibilités sont offertes :

a) L'utilisation de l'interface 'gatool' ;
b) L'appel de la 'soubroutine' 'ga'.

1- Interface 'gatool'

Cette interface est un outil très simple pour la recherche des minimums des fonctions.
Il suffit de définir la fonction objective du problème à résoudre avec le nombre de variables et les contraintes de cette fonction telles que le poule (l'intervalle) de recherche.
Il offre la possibilité de visualiser les résultats d'optimisation.

Son mode d'utilisation suit les étapes suivantes :

- Taper **gatool** dans la fenêtre de l'espace de travail
- Introduire fonction objective dans le champ '**Fitness function : *@filename*'**
- Introduire le nombre de variables dans le champ '**Number of variables : *n*'**
- Définir l'intervalle de recherche dans le champ '**Contraints : Bonds (Lower=*XMIN*, Upper= *XMAX*) '**
- Sélectionner les options de l'algorithme génétique telles le nombre de population '**Population : Population size**', le type de sélection '**Selection** : **Roulette**', le type de mutation '**Mutation** : **Uniform**', le type de croisement '**Crossover** : **Single point**', le critère d'arrêt '**Stopping criteria : Generations**'…..
- Lancer l'exécution en appuyant sur le boutton '**Start**' pour que les résultats s'affichent dans le champ '**Status and results'**.
  N.B. : les résultats, qu'on obtient, différent légèrement d'une exécution à une autre car l'algorithme est utilisent des générateurs aléatoires.

2- Commande (Instruction) 'ga' ; Taper 'help ga' dans l'espace de travail

Cette instruction fait appel au sous programme de l'algorithme génétique ga à travers un fichier M-File pour rechercher les minimums des fonctions.
Elle est donnée par :

[x fval] = ga(@rastriginsfcn, Nv)

Où *rastriginsfcn* : nom du fichier contenant la fonction objective.
Nv : Nombre de variable de la fonction

3- Exemples d'optimisation (recherche du minimum des fonctions)

- Ecrire la fonction objective dans un fichier 'M-File' sous environnement MATLAB
- Sauvegarder ce fichier
- Vérification de l'évaluation de la fonction

a) La fonction 'z = $(x-1)^2 - 10$. Elle admet un minimum f(1)=-10 pour x=1.
'Name : *myfun1 ;* nombre de variables : *1*

* Contenu du fichier Mfile:

function z = myfun1(x)

z = (x(1)-1)^2 - 10;

* Vérification : enter dans l'espace de travail 'Workspace'

myfun1(2) ;

le résultat est *: ans=-9*

i) Réponse de gatool est *:*

GA terminated.

Fitness function value: ***-9.999993425073953***

Optimization terminated: ……..

Final point : ***0.99744*** *qui représente la solution* $x_.$

ii) Réponse de l'appel de la sous programme 'ga' est *:*

L'instruction : *[x fval] = ga(@myfun1, 1)*

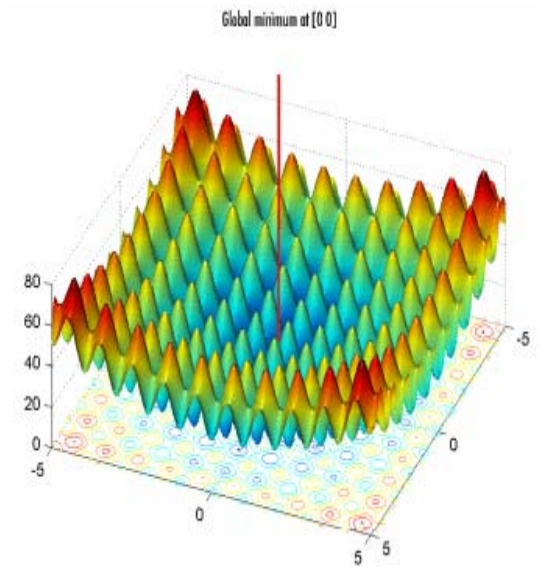
Optimization terminated: average change in the fitness value less than options.TolFun.

x = 0.9979

fval = -10.0000

b) Fonction objective *Rastrigin* de MATLAB :

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$



Elle admet un minimum f(0,0)=0 pour x_1=0 et x_2=0.

'Name : *rastriginsfcn' ;* nombre de variables : *2*

i) Réponse de gatool est *:*

GA terminated.

Fitness function value: ***0.00850545349074***

Optimization terminated: average change in the fitness value less than options. TolFun.

Final point 1 2 :

-0.00637 0.00151 *qui représentent les solutions* x_1 et x_2.

ii) Réponse de l'appel de la sous programme 'ga' est *:*

L'instruction : [x fval exitflag] = ga(@rastriginsfcn, 2)

Optimization terminated:

average change in the fitness value less than options.TolFun.

x = 0.0229 0.0106

fval = 0.1258

c) Chercher le min de la fonction suivante :

$z = (x_1-1)^2 + (x_2-2)^2 + (x_3-3)^2 + (x_4-4)^2 - 10$ à 4 variables

d) Chercher le min de la fonction suivante :

$z = 10\sin(\pi x_1) + 20\sin(2\pi x_2) + 30\sin(3\pi x_3)$ à 3 variables

e) Proposer deux autres fonctions à optimiser.

f) Introduire l'espace de recherche

N.B. : la fonction objective peut être écrite comme suit :

*x = ga(@(x) 3*sin(x(1))+exp(x(2)),2)*

4- Exemples d'optimisation par algorithmes génétiques sous Matlab

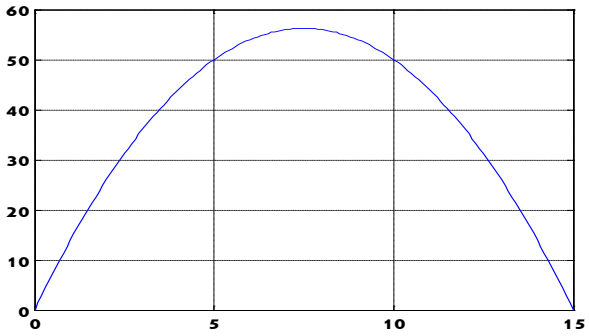
4.1- Exemple 1

Déterminer le maximum de la fonction f(x)=(15x - x^2) où $x \in$ [0 15].

D'après son graphe tracé sous Matlab, elle présente un maximum de F_{max}=56.250 pour x= 7.50:

```
x=0:0.1:15;    y=15*x-x.^2;
figure(2), plot(x,y), grid
 [Ymax, Indices] = max(y);
[x(Indices),  Ymax],

 ans =  7.50   56.250
```



Application des AGs à la fonction f(x)=(15x - x^2) sous Matlab du fichier ''*Prog*1'':

```
Fitness_F=@(x)    (- (15*x(1)-x(1)^2));
options=gaoptimset('populationsize', 30, 'generation', 60, 'plotfcns ', @gaplotbestindiv)
[x0 fmin0]=ga(Fitness_F, 1, options);  fmax0=-fmin0
ans = 7.5010   56.2500;
```

La solution global par AGs est F_{max}=56.25 at x* = 7.5. Elle est identique à celle obtenue graphiquement.

4.2- Exemple 2

Déterminer par AGs le minimum de la fonction *f(x,y)= (x-1)^2+(y-2)^2-20*.

Programme Matlab du fichier ''*Prog*2 :

```
options=gaoptimset('populationsize', 30, 'generation', 60, 'plotfcns ', @gaplotbestindiv)
[x0 fmin0]=ga(@(x) (x(1)-1)^2+(x(2)-2)^2-20, 2, options)
```

Solution:

Les résultats de deux exécutions:

1ère exécution: x0=[0.9976 2.0015]; 2ème exécution: x0=[1.0006 1.9866]

fmin0 = -20.0000 fmin0 = -19.9998

On remarque que les solutions obtenus par les AGs sont voisines à la solution théorique F_{min}= -20 pour x_1* = 1 & x_2* = y*=2.

4.3- Exemple 3

Déterminer par AGs le minimum de la fonction suivante

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3) e^{-x^2-y^2}$$

Programme Matlab du fichier ''*Prog*3 :

```
clc, clear all
options=gaoptimset('populationsize', 30, 'generation', 50, 'plotfcns ', @gaplotbestindiv)
[x0 fmin0]=ga(@Obj_3,2, [],[],[], [], [],[], [], options),
```

Fonction Matlab du fichier ''***Obj_3***'':

```
function z = Obj_3(x)
```

```
z=(1-x(1)).^2 .* (exp( -x(1).^2-(x(2)+1).^2))-( x(1) - x(1).^3 - x(2).^3 ).* (exp( -x(1).^2 - x(2).^2));
```

Les résultats des exécutions:

1ère exécution**: x0 = 0.6745 -1.0983 fmin0 = -0.2548**

2ème exécution**: x0 = 0.5497 0.1179 fmin0 = -0.2355**

3me exécution**: x0 = -1.7022 0.4290 fmin0 = -0.0923 (Best)**

4.4- Exemple 4

Déterminer par AGs le minimum de la fonction : $f(x)= 100*(y-x^2)^2+(1-x)^2$

La solution théorique est: 0 pour x1* = 1 & x2* = 1

Programme Matlab du fichier ‘‘*Prog4*’’ :

```
FitnessFcn = @dejong2fcn;
numberOfVariables = 2;
options = gaoptimset('PlotFcns',{@gaplotbestf,@gaplotstopping});
rng('default')
[x,fmin] = ga(FitnessFcn,numberOfVariables,[],[],[],[],[],[],[],options)
```

La solution par AG est 0.0017 pour x1* = 0.9652 & x2* = 0.9340

4.5- Exemple 5 avec hybridation de GA avec fminunc

La fonction **fminunc** cherche le minimum des fonctions multivariables.

La solution théorique est: 0 pour x1* = 1 & x2* = 1

Déterminer par AGs le minimum de la fonction : $f(x)= 100*(y-x^2)^2+(1-x)^2$

Programme Matlab du fichier ‘‘*Prog5*’’ :

```
clear all
FitnessFcn = @dejong2fcn;
numberOfVariables = 2;
 options = gaoptimset('PlotFcns',{@gaplotbestf,@gaplotstopping});
 rng('default')
fminuncOptions = optimoptions(@fminunc,'Display','iter','Algorithm','quasi-newton');
options = gaoptimset(options,'HybridFcn',{@fminunc, fminuncOptions});
%Run GA solver again with fminunc as the hybrid function.
[x,fmin] = ga(FitnessFcn,numberOfVariables,[],[],[],[],[],[],[],options)
```

La solution par AG est: 2.0084e-11 pour x1* = 1.0000 & x2* = 1.0000

Selon ce résultat, la fonction hybride «**fminunc**» a amélioré efficacement la précision de la solution.

Additional information

Optimization procedure by Matlab Genetic Algorithm

The following example shows how to write an M-file for the function you want to optimize. Suppose that you want to minimize the function The M-file that computes this function must accept a vector x of length 2, corresponding to the variables x1 and x2, and return a scalar equal to the value of the function at x.

To write the M-file, do the following steps:Select New from the MATLAB File menu.Select M-File. This opens a new M-file in the editor. In the M-file ‘‘my_fun’’, enter the following two lines of code:

```
function z = my_fun(x)
z = x(1)^2 - 2*x(1)*x(2) + 6*x(1) + x(2)^2 - 6*x(2);
```

Save the M-file in a directory on the MATLAB path. To check that the M-file returns the correct value, enter on the Matlab workspace:

```
>> my_fun([2 3])
ans =     -5
```

Create and save M-file ‘‘ProgramGA’’ in a directory on the MATLAB path.

```
[x fval] = ga(@my_fun, 2)
```

Run this file to get the following result:

```
x =     0     3
fval =    -9
```

Help of Soubroutine Genetic Algorithm

Type on the Matlab workspace:

```
>>help ga
```

```
GA    Constrained optimization using genetic algorithm.
   GA attempts to solve problems of the form:
       min F(X)  subject to:  A*X  <= B, Aeq*X  = Beq (linear constraints)
        X                     C(X) <= 0, Ceq(X) = 0 (nonlinear constraints)
                            LB <= X <= UB

   X = GA(FITNESSFCN,NVARS) finds a local unconstrained minimum X to the
   FITNESSFCN using GA. NVARS is the dimension (number of design
   variables) of the FITNESSFCN. FITNESSFCN accepts a vector X of size
   1-by-NVARS, and returns a scalar evaluated at X.

   X = GA(FITNESSFCN,NVARS,A,b) finds a local minimum X to the function
   FITNESSFCN, subject to the linear inequalities A*X <= B. Linear
   constraints are not satisfied when the PopulationType option is set to
   'bitString' or 'custom'. See the documentation for details.

   X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq) finds a local minimum X to the
   function FITNESSFCN, subject to the linear equalities Aeq*X = Beq as
   well as A*X <= B. (Set A=[] and B=[] if no inequalities exist.) Linear
   constraints are not satisfied when the PopulationType option is set to
   'bitString' or 'custom'. See the documentation for details.

   X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq,LB,UB) defines a set of lower and
   upper bounds on the design variables, X, so that a solution is found in
   the range LB <= X <= UB. Use empty matrices for LB and UB if no bounds
```

exist. Set LB(i) = -Inf if X(i) is unbounded below; set UB(i) = Inf if
X(i) is unbounded above. Linear constraints are not satisfied when the
PopulationType option is set to 'bitString' or 'custom'. See the
documentation for details.

X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq,LB,UB,NONLCON) subjects the
minimization to the constraints defined in NONLCON. The function
NONLCON accepts X and returns the vectors C and Ceq, representing the
nonlinear inequalities and equalities respectively. GA minimizes
FITNESSFCN such that C(X)<=0 and Ceq(X)=0. (Set LB=[] and/or UB=[] if
no bounds exist.) Nonlinear constraints are not satisfied when the
PopulationType option is set to 'bitString' or 'custom'. See the
documentation for details.

X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq,LB,UB,NONLCON,options) minimizes
with the default optimization parameters replaced by values in the
structure OPTIONS. OPTIONS can be created with the GAOPTIMSET function.
See GAOPTIMSET for details.

X = GA(PROBLEM) finds the minimum for PROBLEM. PROBLEM is a structure
that has the following fields:
```
    fitnessfcn: <Fitness Function>
        nvars: <Number of design variables>
      options: <Options structure created with GAOPTIMSET>
        Aineq: <A matrix for inequality constraints>
        Bineq: <B vector for inequality constraints>
          Aeq: <A matrix for equality constraints>
          Beq: <B vector for equality constraints>
           LB: <Lower bound on X>
           UB: <Upper bound on X>
     nonlcon: <nonlinear constraint function>
   randstate: <Optional field to reset rand state>
  randnstate: <Optional field to reset randn state>
```

[X,FVAL] = GA(FITNESSFCN, ...) returns FVAL, the value of the fitness
function FITNESSFCN at the solution X.

[X,FVAL,EXITFLAG] = GA(FITNESSFCN, ...) returns EXITFLAG which
describes the exit condition of GA. Possible values of EXITFLAG and the
corresponding exit conditions are

```
  1 Average change in value of the fitness function over
    options.StallGenLimit generations less than options.TolFun and
    constraint violation less than options.TolCon.
  3 The value of the fitness function did not change in
    options.StallGenLimit generations and constraint violation less
    than options.TolCon.
  4 Magnitude of step smaller than machine precision and constraint
    violation less than options.TolCon. This exit condition applies
    only to nonlinear constraints.
  5 Fitness limit reached and constraint violation less than
    options.TolCon.
  0 Maximum number of generations exceeded.
```

-1 Optimization terminated by the output or plot function.
-2 No feasible point found.
-4 Stall time limit exceeded.
-5 Time limit exceeded.

[X,FVAL,EXITFLAG,OUTPUT] = GA(FITNESSFCN, ...) returns a
structure OUTPUT with the following information:
randstate: <State of the function RAND used before GA started>
randnstate: <State of the function RANDN used before GA started>
generations: <Total generations, excluding HybridFcn iterations>
funccount: <Total function evaluations>
maxconstraint: <Maximum constraint violation>, if any
message: <GA termination message>

[X,FVAL,EXITFLAG,OUTPUT,POPULATION] = GA(FITNESSFCN, ...) returns the
final POPULATION at termination.

[X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES] = GA(FITNESSFCN, ...) returns
the SCORES of the final POPULATION.

Example:
Unconstrained minimization of 'rastriginsfcn' fitness function of
numberOfVariables = 2
x = ga(@rastriginsfcn,2)

Display plotting functions while GA minimizes
options = gaoptimset('PlotFcns',...
{@gaplotbestf,@gaplotbestindiv,@gaplotexpectation,@gaplotstopping});
[x,fval,exitflag,output] = ga(@rastriginsfcn,2,[],[],[],[],[],[],[],options)

An example with inequality constraints and lower bounds
A = [1 1; -1 2; 2 1]; b = [2; 2; 3]; lb = zeros(2,1);
% Use mutation function which can handle constraints
options = gaoptimset('MutationFcn',@mutationadaptfeasible);
[x,fval,exitflag] = ga(@lincontest6,2,A,b,[],[],lb,[],[],options);

FITNESSFCN can also be an anonymous function:
x = ga(@(x) 3*sin(x(1))+exp(x(2)),2)

If FITNESSFCN or NONLCON are parameterized, you can use anonymous
functions to capture the problem-dependent parameters. Suppose you want
to minimize the fitness given in the function myfit, subject to the
nonlinear constraint myconstr, where these two functions are
parameterized by their second argument a1 and a2, respectively. Here
myfit and myconstr are M-file functions such as

```
function f = myfit(x,a1)
f = exp(x(1))*(4*x(1)^2 + 2*x(2)^2 + 4*x(1)*x(2) + 2*x(2) + a1);
```

and

```
function [c,ceq] = myconstr(x,a2)
c = [1.5 + x(1)*x(2) - x(1) - x(2);
```

```
        -x(1)*x(2) - a2];
    % No nonlinear equality constraints:
     ceq = [];
```

To optimize for specific values of a1 and a2, first assign the values to these two parameters. Then create two one-argument anonymous functions that capture the values of a1 and a2, and call myfit and myconstr with two arguments. Finally, pass these anonymous functions to GA:

```
a1 = 1; a2 = 10; % define parameters first
% Mutation function for constrained minimization
options = gaoptimset('MutationFcn',@mutationadaptfeasible);
x = ga(@(x)myfit(x,a1),2,[],[],[],[],[],[],@(x)myconstr(x,a2),options)
```

See also gaoptimset, fitnessfunction, gaoutputfcntemplate, patternsearch