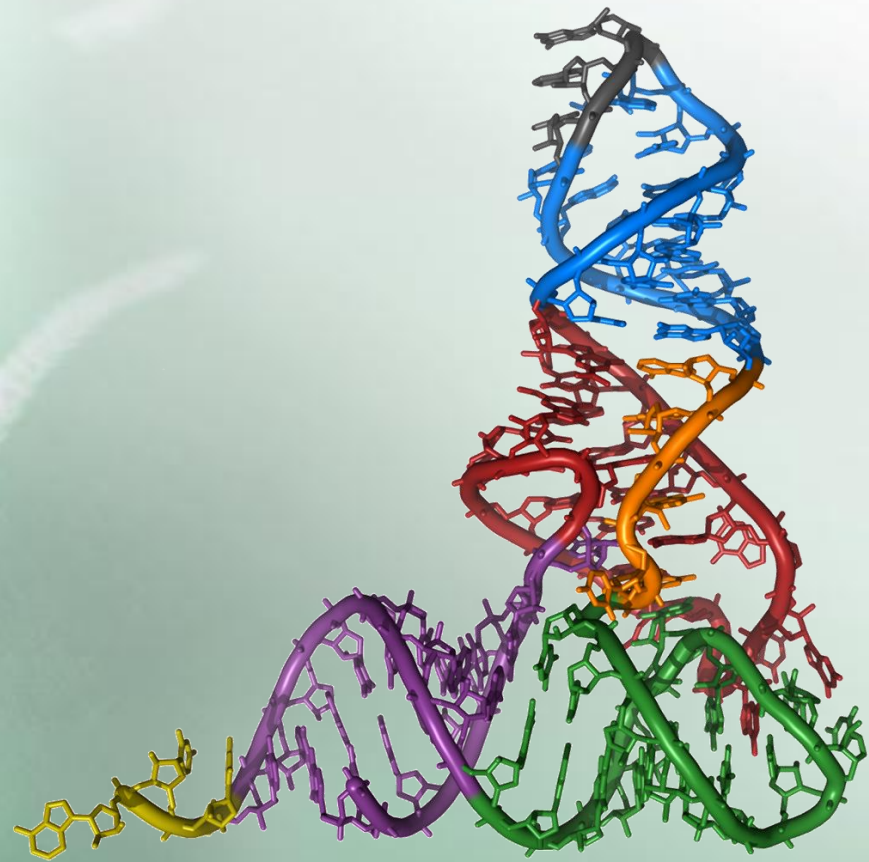


ПЛ-5: Оптимизација флексибилних технолошких процеса применом генетичких алгоритама



- **Пример #1:**

- ✓ Оптимизација функције $f(x)=x^2$ – “ручна” симулација;
-

- **Пример #2:**

- ✓ Оптимизација “*drop wave*” функције применом *Genetic Algorithm Toolbox*-а;
-

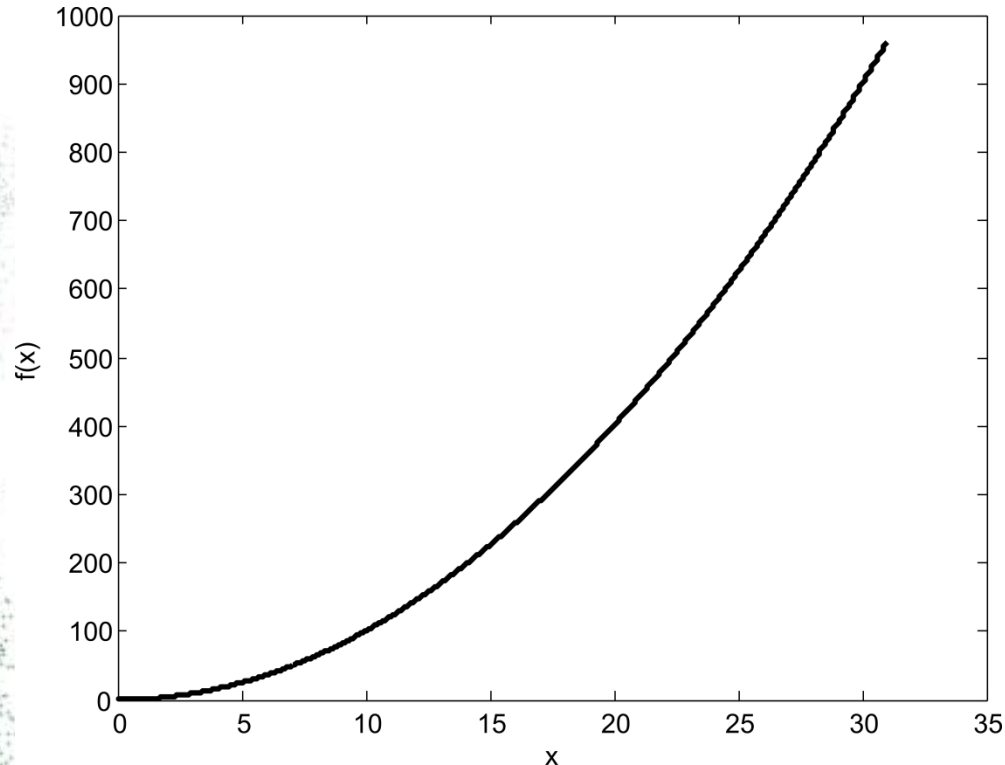
- **Пример #3:**

- ✓ Оптимизација технолошких процеса – програмирање у *Matlab* софтверском пакету;
-

Пример#1

поставка проблема

- потребно је одредити максимум функције $f(x)=x^2$ на интервалу $[0,31]$;



Код у Matlab-y:

```
clc; clear all; close all;  
  
% x - argument  
x = 0:31;  
  
% y – vrednost funkcije  
y = x.^2;  
  
%crtanje grafika  
plot(x,y,'-k','LineWidth',2)  
%oznacavanje osa  
xlabel('x')  
ylabel('f(x)')
```

Пример#1

- представљање бројева у бинарном и децималном нумеричком систему;

$$X = \sum_{i=-m}^{n-1} k_i N^i = k_{n-1} N^{n-1} + k_{n-2} N^{n-2} + \dots + k_0 N^0 + k_{-1} N^{-1} + \dots + k_{-m} N^{-m}$$

- ✓ m – број цифара у разломљеном делу;
- ✓ n – број цифара у целобројном делу;
- ✓ k – цифре бројног система
- ✓ на пример: $k=(0,1,2,3,4,5,6,7,8,9)$ за децимални бројни систем, $k=(0,1)$ за бинарни бројни систем, $k=(1,0,-1)$ за тернарни бројни систем, итд.

- број **32,45** представљен у децималном бројном систему:

$$3 \cdot 10^1 + 2 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2} = 32.45$$

- број **25** представљен у бинарном бројном систему (11001):

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 25$$

- користећи петобитне цифре, могуће је добити бројеве између 0 (00000) и 31 (11111);

Пример#1

- кодирање хромозома: четири случајно генерисана стринга са бинарним вредностима

- декодираније хромозома: за стринг под редним бројем 3 (01000)
 $0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 8$

- функција циља (*fitness*)
 $f(x) = 8^2 = 64$

| бр. стринга | иницијална популација | x у дец. систему | $f(x) = x^2$ | $\frac{f_i}{\sum f}$ | $\frac{f_i}{\bar{f}}$ | рулет селекција |
|---------------------------|-----------------------|------------------|--------------|----------------------|-----------------------|-----------------|
| 1 | 01101 | 13 | 169 | 0,14 | 0,58 | 1 |
| 2 | 11000 | 24 | 576 | 0,49 | 1,97 | 2 |
| 3 | 01000 | 8 | 64 | 0,06 | 0,22 | 0 |
| 4 | 10011 | 19 | 361 | 0,31 | 1,23 | 1 |
| сума $\sum f$ | | | 1170 | 1,00 | 4,00 | 4,0 |
| средња вредност \bar{f} | | | 293 | 0,25 | 1,00 | 1,0 |
| максимална вредност | | | 576 | 0,49 | 1,97 | 2,0 |

Пример#1

- рулет селекција (*roulette wheel selection*) је рангирајућа селекција код које је вероватноћа одабира јединке пропорционална функцији циља (*fitness*);
- за пример из табеле, највећи ранг и вероватноћу да “преживи” има јединка под редним бројем 2, а затим јединке под редним бројем 3 и 4;

- вероватноћа селекције сваке од четири јединке;

| бр. стринга | иницијална популација | x у дец. систему | $f(x) = x^2$ | $\frac{f_i}{\sum f}$ | $\frac{f_i}{\bar{f}}$ | рулет селекција |
|---------------------------|-----------------------|------------------|--------------|----------------------|-----------------------|-----------------|
| 1 | 01101 | 13 | 169 | 0,14 | 0,58 | 1 |
| 2 | 11000 | 24 | 576 | 0,49 | 1,97 | 2 |
| 3 | 01000 | 8 | 64 | 0,06 | 0,22 | 0 |
| 4 | 10011 | 19 | 361 | 0,31 | 1,23 | 1 |
| сума $\sum f$ | | | 1170 | 1,00 | 4,00 | 4,0 |
| средња вредност \bar{f} | | | 293 | 0,25 | 1,00 | 1,0 |
| максимална вредност | | | 576 | 0,49 | 1,97 | 2,0 |

Пример#1

- резултати примене оператора укрштања након само једне генерације;
- и средњи и максимални резултат су побољшани у новој популацији;
 - ✓ средња вредност функције циља се са 293 повећала на 439;
 - ✓ максимална вредност се повећала са 576 на 729;

| хромозоми након укрштања | парови | позиција укрштања | нова популација | x | $f(x) = x^2$ |
|-----------------------------|--------|----------------------|--------------------|-----|--------------|
| 0 1 1 0 1 1 | 2 | 4 | 0 1 1 0 0 | 12 | 144 |
| 1 1 0 0 1 0 | 1 | 4 | 1 1 0 0 1 | 25 | 625 |
| 1 1 1 0 0 0 | 4 | 2 | 1 1 0 1 1 | 27 | 729 |
| 1 0 1 0 1 1 | 3 | 2 | 1 0 0 0 0 | 16 | 256 |
| сума $\sum f$ | | | | | 1754 |
| средња вредност \bar{f} | | | | | 439 |
| максимална вредност | | | | | 729 |

Пример#1

- резултати након примене оператора мутације;
- средњи резултат је побољшани у новој популацији;
 - ✓ средња вредност функције циља се са 439 повећала на 588,5;
 - ✓ најбољи стринг у 1. генерацији (11000) добија две копије у 2. генерацији; укрштањем са стрингом (10011) у случајно изабраној тачки укрштања 2, један од насталих потомака (11011) показује већу вредност функције циља $f(x)$ у односу на оба родитеља;

| бр. стринга | потомци након укрштања | потомци након мутације | x | $f(x) = x^2$ |
|---------------------------|------------------------|------------------------|----|--------------|
| 1 | 0 1 1 0 0 | 1 1 1 0 0 | 26 | 676 |
| 2 | 1 1 0 0 1 | 1 1 0 0 1 | 25 | 625 |
| 3 | 1 1 0 1 1 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 0 0 0 | 1 0 1 0 0 | 18 | 324 |
| сума $\sum f$ | | | | 2354 |
| средња вредност \bar{f} | | | | 588,5 |
| максимална вредност | | | | 729 |

Пример#2

поставка проблема

- потребно је одредити максимум “*drop wave*” функције

$$f(x) = -\frac{1 + \cos(12 - \sqrt{(x_1^2 + x_2^2)})}{\frac{1}{2}(x_1^2 + x_2^2)}$$

- изразито *мултимодална* тест функција;
- границе решења су у домену:

$$-5.12 \leq x_1 \leq 5.12, \quad -5.12 \leq x_2 \leq 5.12$$

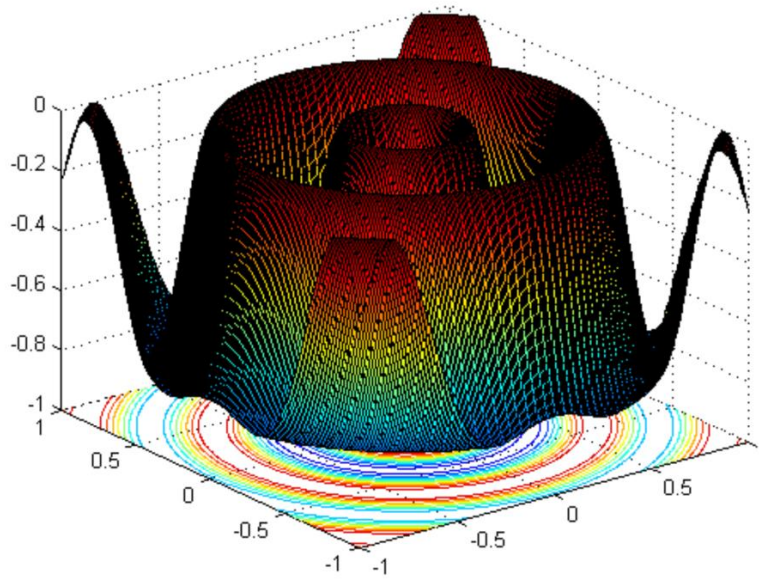
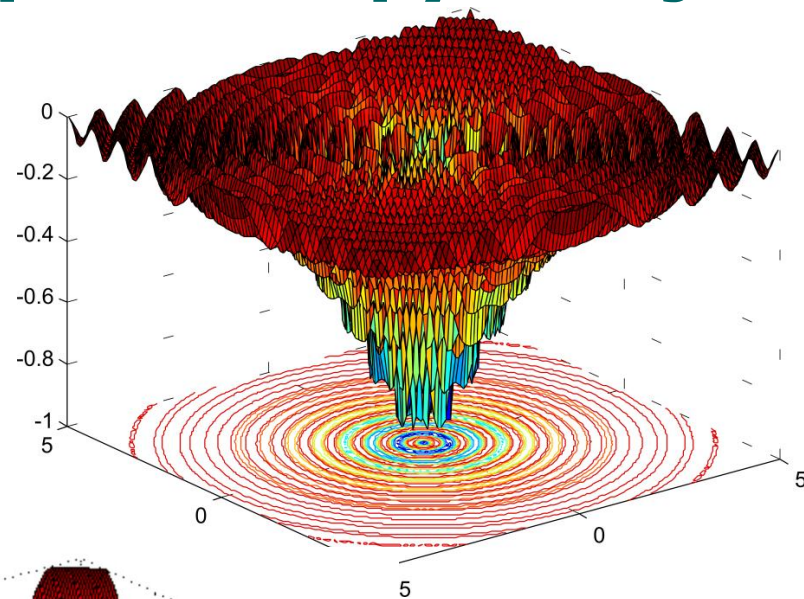
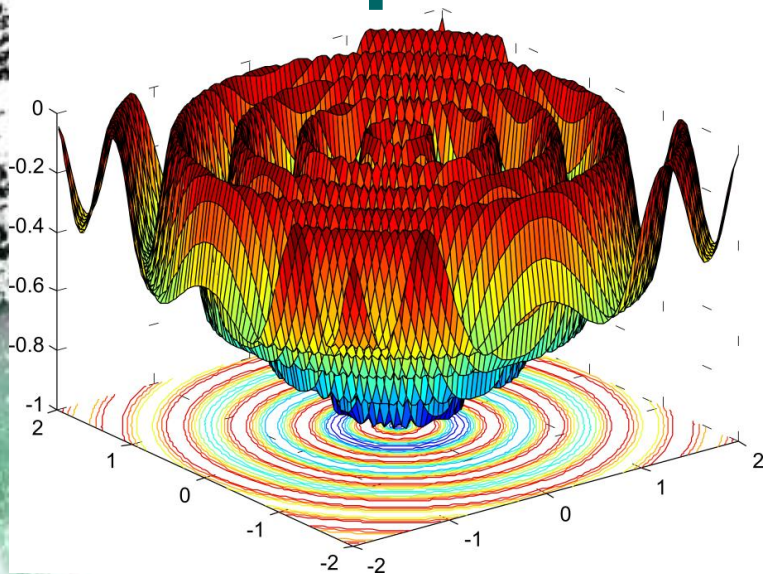
- глобални минимум функције је:

$$f(x^*) = -1$$

$$x^* = (0,0)$$

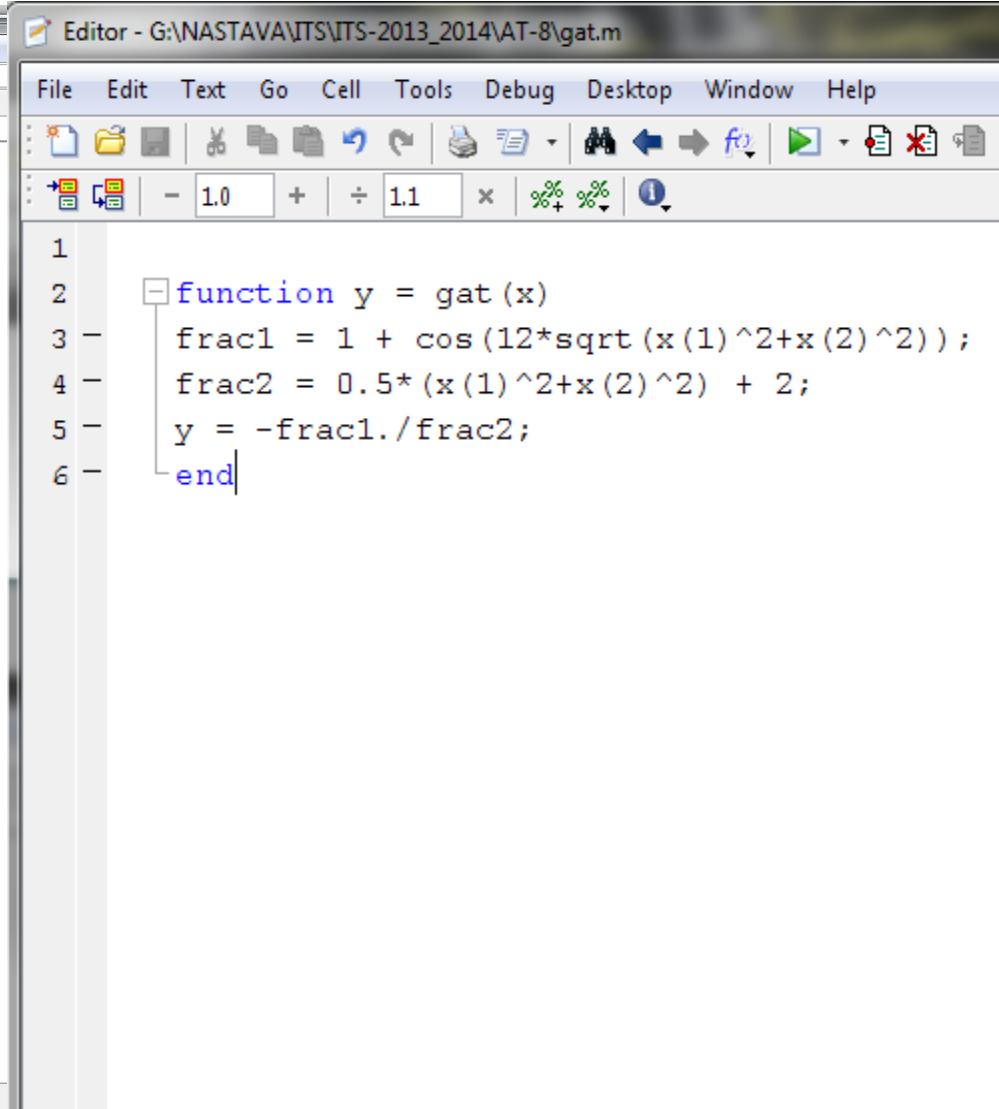
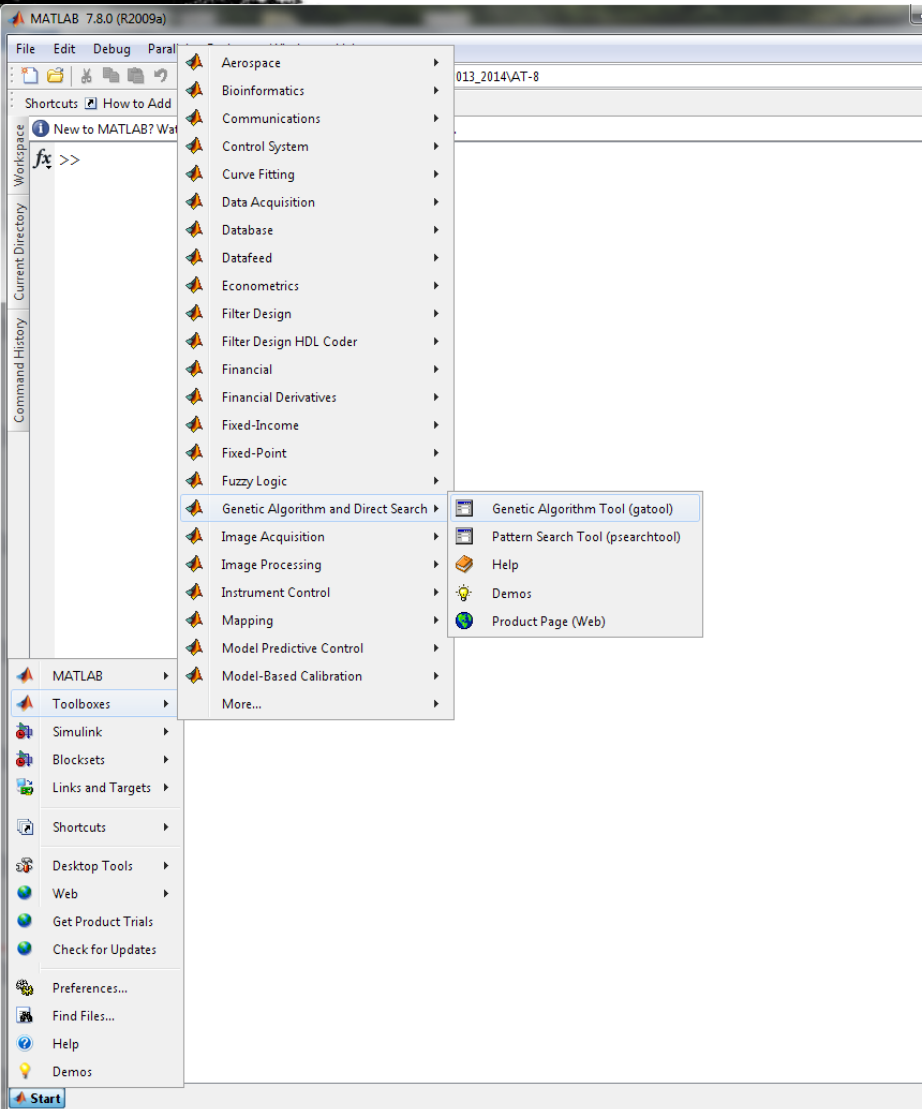
Пример#2

3D приказ "Drop wave" функције



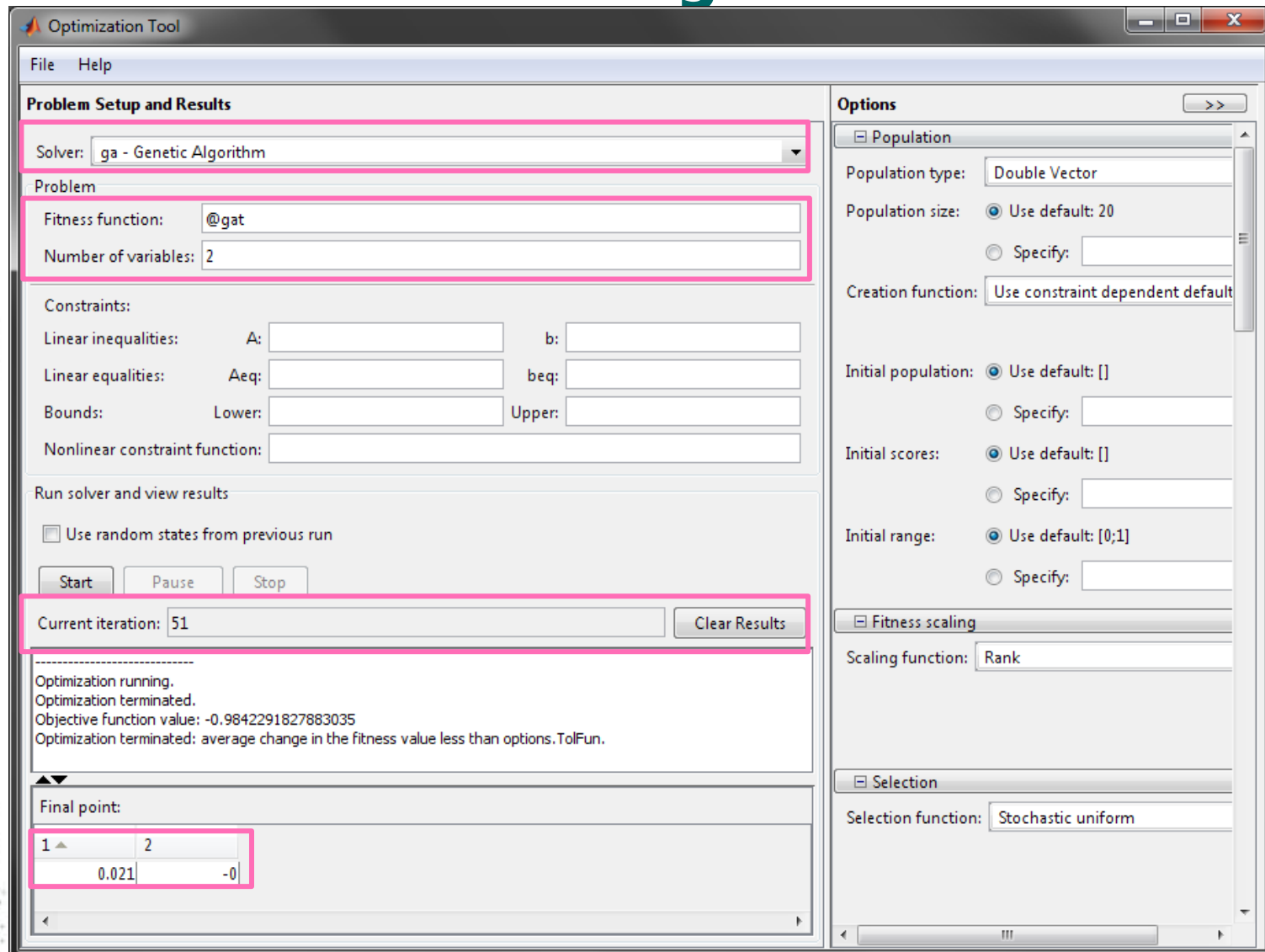
Пример#2

Genetic Algorithm toolbox



Пример#2

Genetic Algorithm toolbox



The screenshot displays the 'Optimization Tool' window, which is divided into several sections. The 'Problem Setup and Results' section on the left contains the following fields:

- Solver: ga - Genetic Algorithm
- Problem
- Fitness function: @gat
- Number of variables: 2
- Constraints:
 - Linear inequalities: A: [] b: []
 - Linear equalities: Aeq: [] beq: []
 - Bounds: Lower: [] Upper: []
 - Nonlinear constraint function: []
- Run solver and view results
 - Use random states from previous run
 - Start [] Pause [] Stop []
 - Current iteration: 51 [] Clear Results []
- Optimization running.
Optimization terminated.
Objective function value: -0.9842291827883035
Optimization terminated: average change in the fitness value less than options.TolFun.
- Final point:

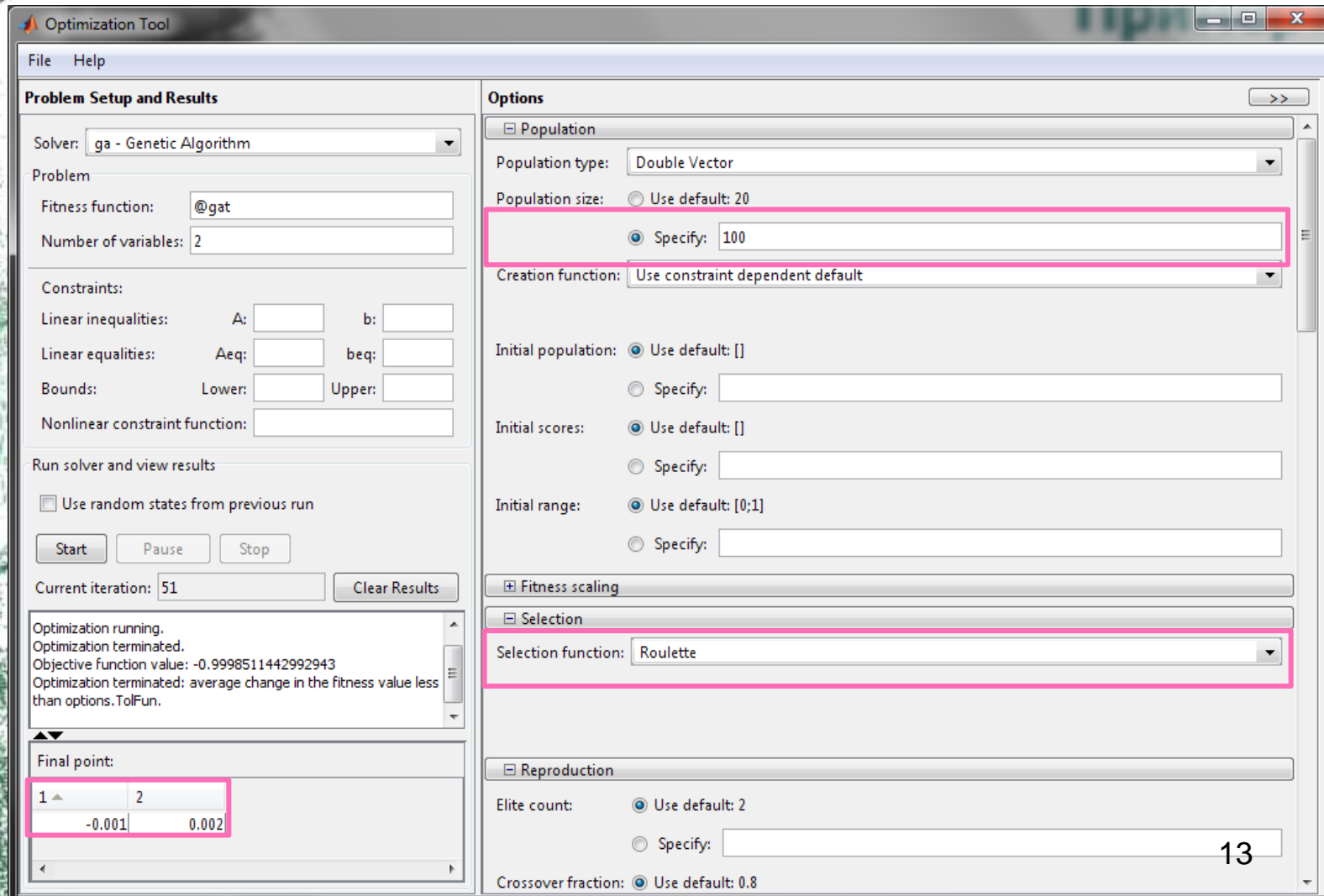
| 1 | 2 |
|-------|----|
| 0.021 | -0 |

The 'Options' section on the right is expanded to show the 'Population' settings:

- Population type: Double Vector
- Population size: Use default: 20 Specify: []
- Creation function: Use constraint dependent default
- Initial population: Use default: [] Specify: []
- Initial scores: Use default: [] Specify: []
- Initial range: Use default: [0;1] Specify: []
- Fitness scaling: Scaling function: Rank
- Selection: Selection function: Stochastic uniform

Пример#2

Genetic Algorithm toolbox



The screenshot displays the 'Optimization Tool' window, which is divided into two main sections: 'Problem Setup and Results' and 'Options'.

Problem Setup and Results:

- Solver:** ga - Genetic Algorithm
- Problem:**
 - Fitness function: @gat
 - Number of variables: 2
- Constraints:**
 - Linear inequalities: A: [] b: []
 - Linear equalities: Aeq: [] beq: []
 - Bounds: Lower: [] Upper: []
 - Nonlinear constraint function: []
- Run solver and view results:**
 - Use random states from previous run
 - Buttons: Start, Pause, Stop
 - Current iteration: 51
 - Clear Results button
- Output Log:**
 - Optimization running.
 - Optimization terminated.
 - Objective function value: -0.9998511442992943
 - Optimization terminated: average change in the fitness value less than options.TolFun.
- Final point:**

| | |
|--------|-------|
| 1 | 2 |
| -0.001 | 0.002 |

Options:

- Population:**
 - Population type: Double Vector
 - Population size: Specify: 100
 - Creation function: Use constraint dependent default
- Initial population:** Use default: []
- Initial scores:** Use default: []
- Initial range:** Use default: [0;1]
- Fitness scaling:** (Expanded)
- Selection:** Selection function: Roulette
- Reproduction:**
 - Elite count: Use default: 2
 - Crossover fraction: Use default: 0.8

Пример#2

Genetic Algorithm toolbox

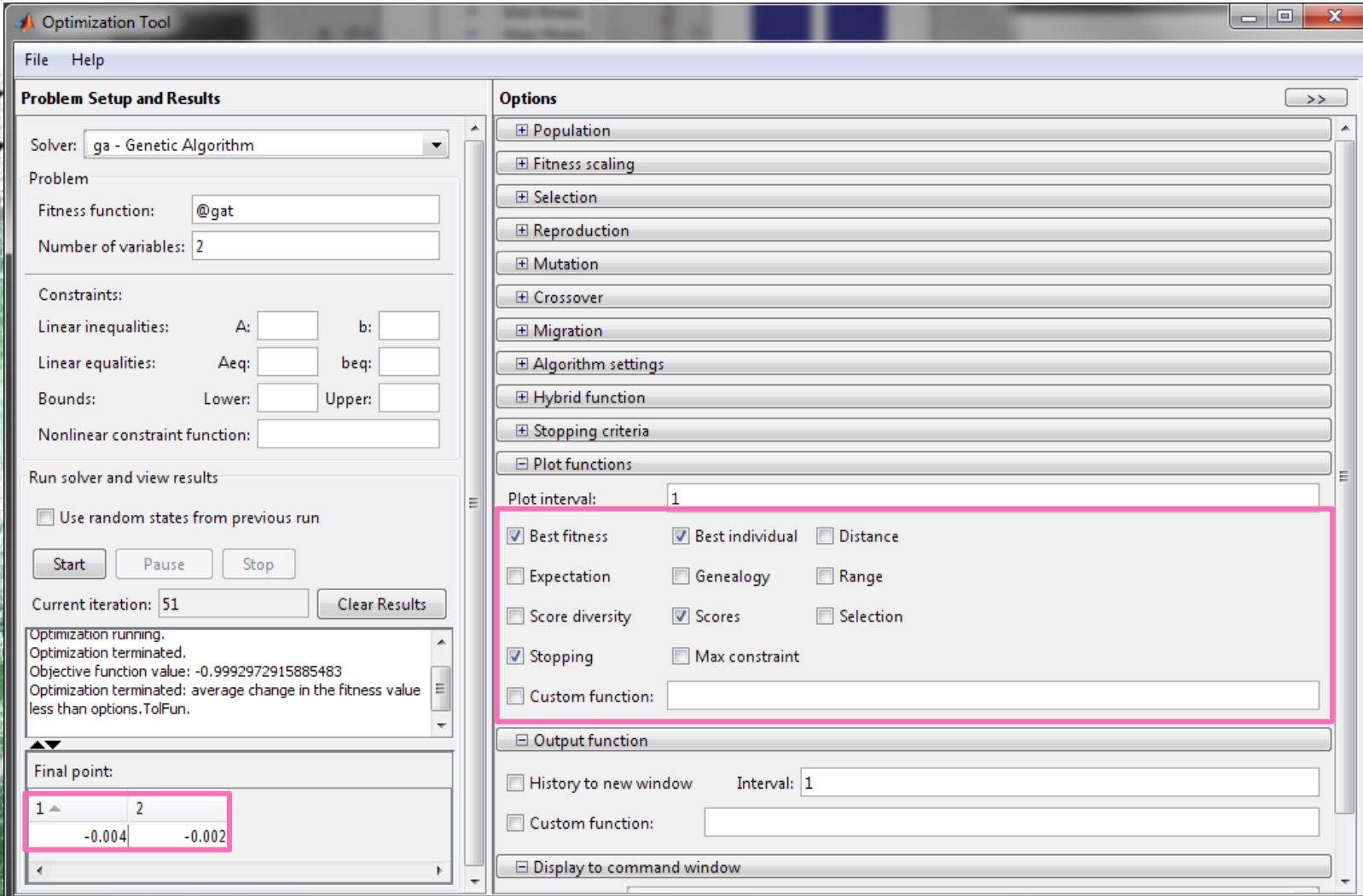
The screenshot displays the 'Optimization Tool' window with the following configuration:

- Problem Setup and Results:**
 - Solver: ga - Genetic Algorithm
 - Problem: @gat
 - Number of variables: 2
 - Linear inequalities: A: [], b: []
 - Linear equalities: Aeq: [], beq: []
 - Bounds: Lower: [], Upper: []
 - Nonlinear constraint function: []
 - Current iteration: 51
 - Optimization terminated. Objective function value: -0.9997026758974772
 - Optimization terminated: average change in the fitness value less than options.TolFun.
 - Final point table:
- Options:**
 - Elite count: Specify: 6
 - Crossover fraction: Specify: 0.9
 - Mutation function: Gaussian
 - Scale: Use default: 1.0
 - Shrink: Use default: 1.0
 - Crossover function: Single point

| 1 | 2 |
|-------|-------|
| 0.003 | 0.001 |

Пример#2

Genetic Algorithm toolbox



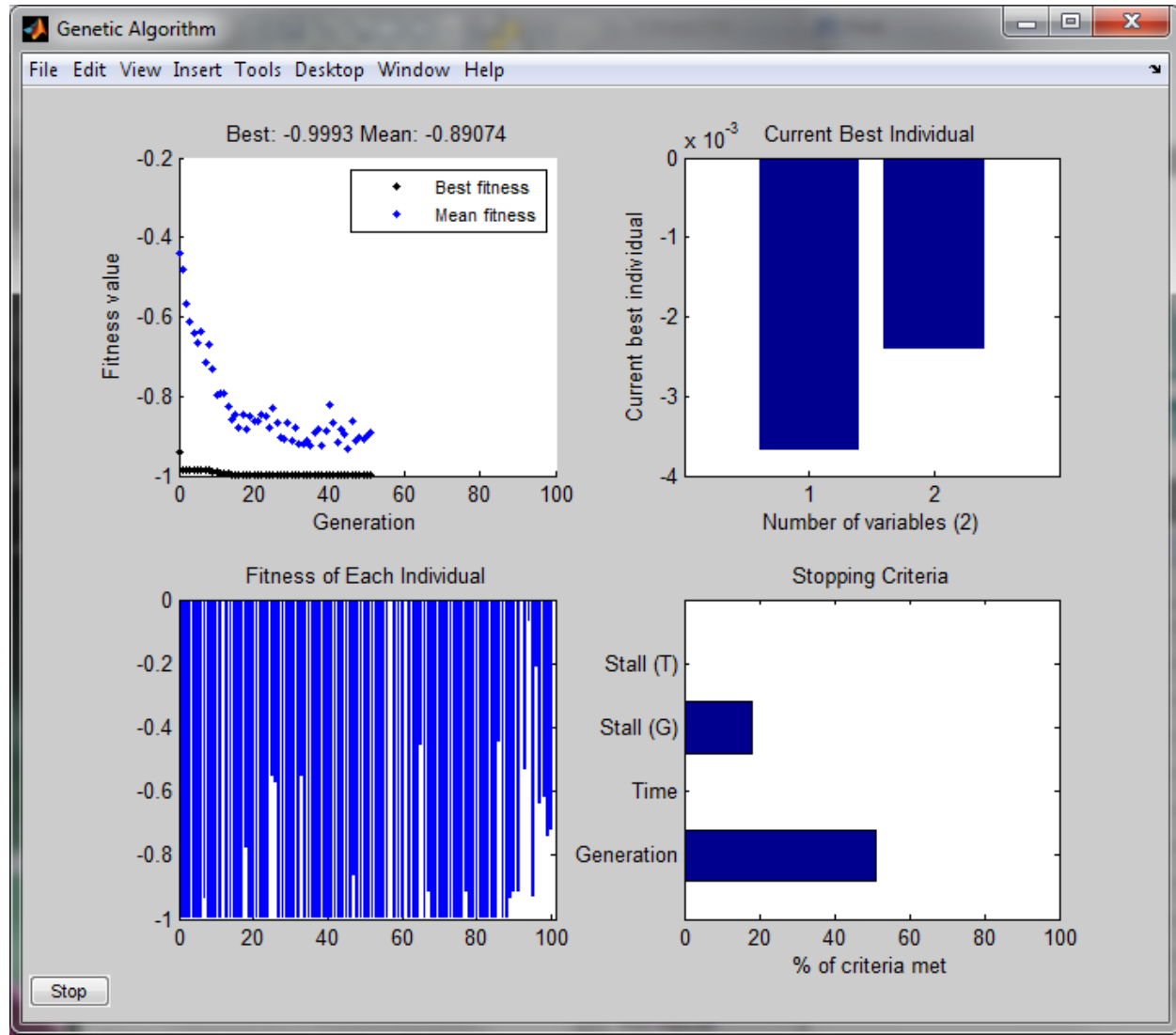
The screenshot displays the Optimization Tool interface, which is divided into several sections:

- Problem Setup and Results:**
 - Solver: ga - Genetic Algorithm
 - Problem: Fitness function: @gat, Number of variables: 2
 - Constraints: Linear inequalities, Linear equalities, Bounds, and Nonlinear constraint function fields.
 - Run solver and view results: Includes a checkbox for "Use random states from previous run", Start/Pause/Stop buttons, and a "Clear Results" button.
 - Current iteration: 51
 - Optimization running status: Optimization terminated. Objective function value: -0.9992972915885483. Optimization terminated: average change in the fitness value less than options.TolFun.
 - Final point: A table showing the final values for the two variables.
- Options:**
 - Expandable sections for Population, Fitness scaling, Selection, Reproduction, Mutation, Crossover, Migration, Algorithm settings, Hybrid function, Stopping criteria, and Plot functions.
 - Plot interval: 1
 - Checked options: Best fitness, Best individual, Stopping.
 - Other options: Distance, Expectation, Genealogy, Range, Score diversity, Scores, Selection, Max constraint.
 - Output function: History to new window (Interval: 1), Custom function.
 - Display to command window.

| 1 | 2 |
|--------|--------|
| -0.004 | -0.002 |

Пример#2

Genetic Algorithm toolbox



Пример#3

иницијална популација

- генерисање иницијалне популације

```
clc; clear all; close all;
n = 4; %job number
p1 = [1 3 2 3 3 3 8 8; 1 2 2 3 3 3 8 8; 1 5 2 3 3 3 8 8];%deo1
p2 = [1 5 5 3 6 3 8 9; 1 6 5 3 6 3 8 9; 1 5 5 3 6 3 8 8];%deo2
p3 = [1 3 2 6 3 5 8 5; 1 3 2 6 3 6 8 5; 1 4 2 4 3 6 8 5];%deo3
p4 = [1 9 2 5 8 6 0 0; 1 9 2 6 8 6 0 0; 5 4 6 2 7 3 8 2];%deo4
p = [p1; p2; p3; p4];
q = 4;% q maksimalan broj operacija za sve tehnoloske procese
sp = [4 4 4; 4 4 4; 4 4 4; 3 3 4];%broj operacija za sve tehn. procese
job = []; ini = [];
N = 10; %N - broj hromozoma tj. velicin populacije;
for h = 1 : N
    r = [randnumber(1,3) randnumber(1,3) randnumber(1,3) randnumber(1,3)];
    initalPPstring(h,:) = r;
    for i = 1 : n
        j = sp(i,(r(1,i)));
        ini = [ini ones(1,j)*i];
        if i == 4
            diff = q*n-size(ini,2);
            ini = [ini zeros(1,diff)];
        end
    end
    INI = ini(:,randperm(size(ini,2)));
    initalPopulation(h,:) = INI;
    ini = [];
end
```

end

Пример#3

мутација

- први оператор мутације - двопозициона „swapping“ мутација

```
function [population] = mutation1Scheduling(population)

population = ...
    [4 1 3 2 6 0 5 1 2 6 0 4 5 3 2 5 0 4 6 0 1 5 2 3 5 1 3 0 6 0];
%     1 0 4 3 5 6 0 4 3 5 2 1 4 0 2 1 6 5 3 0 1 0 2 6 0 3 6 5 0 0]

r1 = randnumber(1,size(population,2));
r2 = randnumber(1,size(population,2));

while population(1,r1) == population(1,r2);
    r1 = randnumber(1,size(population,2));
    r2 = randnumber(1,size(population,2));
end

t1 = population(1,r1);
t2 = population(1,r2);
population(1,r2) = t1;
population(1,r1) = t2;
```

Пример#3

мутација

- други оператор мутације – промена алт. техн. процеса

```
function [population,ppstring_mutation2] = ...
mutation2Scheduling(population,ppstring,sp)
population = ...
    [4 1 3 2 6 0 5 1 2 6 0 4 5 3 2 5 0 4 6 0 1 5 2 3 5 1 3 0 6 0;
    1 0 4 3 5 6 0 4 3 5 2 1 4 0 2 1 6 5 3 0 1 0 2 6 0 3 6 5 0 0]
ppstring = [1 2 2 3]; sp = [4 4 4; 4 4 4; 4 4 4; 3 3 4];
r = randnumber(1,size(ppstring,2));
if ppstring(1,r) == 1;
    index = 1; g = [2 3]; rr = randnumber(1,2); gg = g(1,rr);
elseif ppstring(1,r) == 2;
    index = 2; g = [1 3]; rr = randnumber(1,2); gg = g(1,rr);
else
    index = 3; g = [1 2]; rr = randnumber(1,2); gg = g(1,rr);
end
ppstring(1,r) = gg; ppstring_mutation2 = ppstring;
if sp(r,gg) > sp(r,index);
    f = find(population(1,:) == 0);
    rrr = randnumber(1,size(f,2));
    population(1,f(1,rrr)) = r;
else sp(r,gg) < sp(r,index);
    f = find(population(1,:) == r);
    rrr = randnumber(1,size(f,2));
    population(1,f(1,rrr)) = 0;
end
```

Питања?

Хвала на пажњи!

