MINIMIZING EGG HOLDER FUNCTION WITH GENETIC ALGORITHM

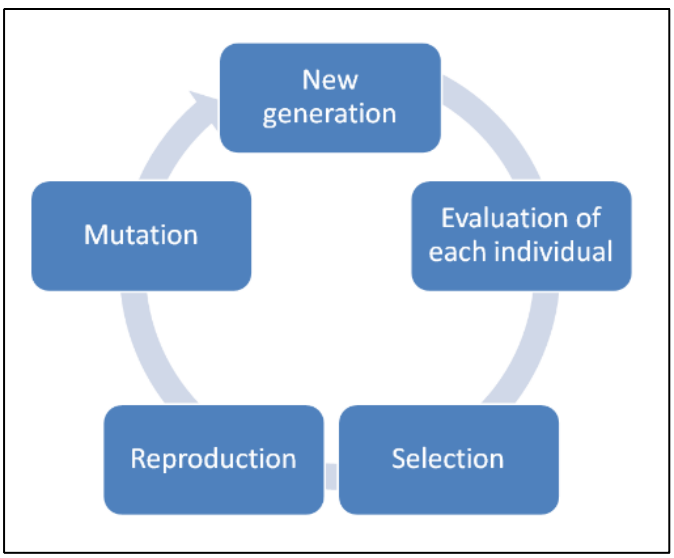
FINAL EXAM
OPERATION RESEARCH II – B
Lecturer: M. Luthfi Shahab, M.Si

Written by:
VENANSIUS RYAN TJAHJONO
06111540000043

A. A BRIEF EXPLANATION ABOUT GENETIC ALGORITHM

Genetic Algorithm (GA) atau algoritma Genetika dikembangkan pertama kali oleh John Holland pada sekitar tahun 1960an. Algoritma ini sebenarnya adalah suatu model abstraksi dari evolusi biologis yang berdasarkan teori Charles Darwin. Holland dan rekannya mengembangkan algoritma ini untuk melakukan *crossover, recombination, mutation,* dan *natural selection* pada sistem buatan (*artificial system*) mereka. Seiring dengan perkembangan waktu, algoritma Genetika menjadi salah satu metode yang sering diterapkan untuk mencari nilai optimal dari suatu permasalahan. Permasalahan yang dimaksud berupa model matematika atau fungsi seperti, program linier, program tak linier, fungsi invarian, pewarnaan *graph, pattern recognition,* fungsi kontinu dan fungsi diskrit. Namun, dibalik kegunaan dari algoritma Genetika, diperlukan keteitian dan kepandaian untuk pendefinisian awal, pemilihan parameter *crossover* dan *mutation,* dan juga banyaknya populasi awal yang harus dibangkitkan.

Setelah itu, algoritma Genetika memiliki empat tahapan, yaitu *generate initial population, selection, crossover,* dan *mutation.*



Berikut adalah penjelasan dari masing-masing tahapan.

1. ***Generate initial population***

   Pada tahap awal, kita harus menentukan banyaknya populasi yang harus dibuat. Hal ini akan mempengaruhi nilai *fitness* atau nilai fungsi objektif tepat sasaran. Selain itu, bentuk solusi yang diinginkan harus terdefinisi dengan baik. Artinya, kita harus menentukan solusi yang diinginkan merupakan bilangan riil, bilangan bulat, ataupun angka biner. Pemilihan teknik representasi ini sangat bergantung pada permasalahan yang dipilih. Misal, untuk optimasi fungsi sebaiknya menggunakan representasi biner, untuk masalah optimasi tata letak barang, sebaiknya menggunakan bilangan bulat, dan untuk masalah graf sebaiknya menggunakan bentuk matrix atau *path*.

2. ***Selection***

   Pada tahap ini, akan diberikan bobot untuk setiap nilai fitness pada setiap individu yang sudah dibangkitkan pada tahap sebelumnya. Ada beberapa cara untuk melakukan *selection*. Cara yang paling terkenal adalah dengan menggunakan sistem *roulette.* Dengan aturan ini, luasan dari bidang *roulette* ditentukan dari hasil pembobotan *fitness.* Lalu, akan dibangkitkan bilangan acak $r \in [0,1]$ untuk menentukan individu mana yang terpilih.

3. ***Crossover/Reproduction***

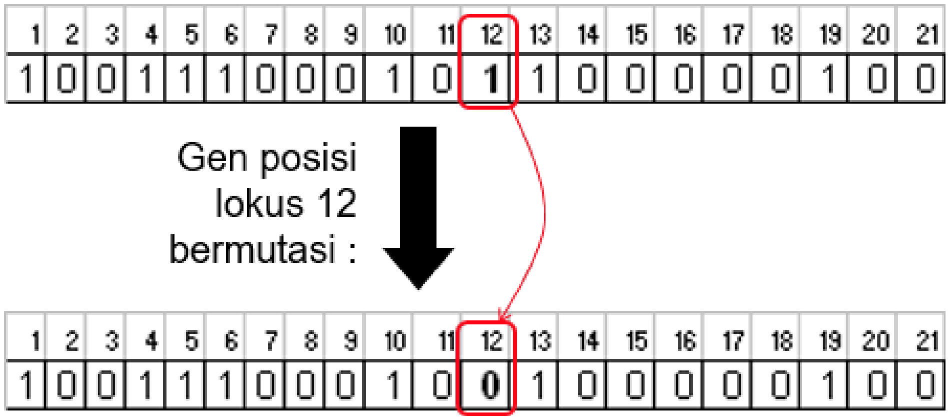
   Setelah menjalani tahap *selection*, dilakukan tahap *crossover* (kawin silang) dengan mengombinasikan dua individu induk untuk menghasilkan dua individu baru. Beberapa teknik dari *crossover* adalah *simple crossover, arithmetical crossover, heuristic crossover,* PMX, OX, dan CX. Namun, pada umumnya digunakan *simple crossover* karena lebih sederhana. *Simple crossover* dilakukan dengan menemukan posisi lokus tertentu tempat terjadi *crossover*. Kromosom terbagi menjadi dua segmen, yaitu dari posisi awal hingga posisi terjadinya *crossover* dan segmen kedua adalah setelah posisi terjadinya *crossover*.

induk	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	1	0	0
4	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1

anak	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
a1	1	0	0	1	1	1	0	0	0	1	0	1	0	0	1	0	1	0	1	0	1
a2	0	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

4. ***Mutation***

   Kemudian, untuk tahapan mutasi, akan diterapkan perubahan nilai gen pada lokus tertentu. Salah satu prosedur mutasi yang sederhana untuk kromosom dengan representasi biner adalah dengan memilih sembarang posisi gen dan mengubah alelnya (1 menjadi 0 dan 0 menjadi 1.



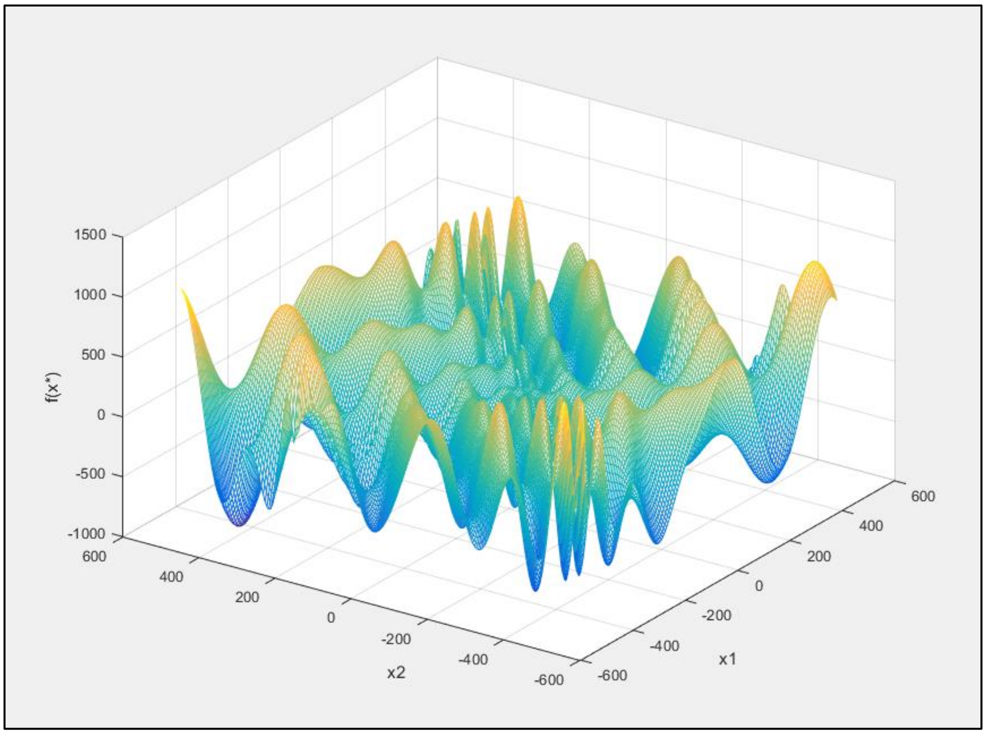
Setelah mengetahui beberapa tahapan algoritma Genetika, dikenalkan pula parameter umum dan *stopping criteria* dari algoritma Genetika. Adapun parameter yang dimaksud adalah jumlah individu pada tiap populasi, peluang *crossover,* dan peluang *mutation.* Kemudian, untuk *stopping criteria*, terdapat tiga cara umum untuk menghentikan proses algoritma Genetika. Pertama dengan menentukan nilai maksimum dari banyaknya generasi yang diinginkan. Pada kasus ini, penentuan banyaknya generasi akan menentukan seberapa baik solusi yang dihasilkan. Kedua, dengan menghentikan proses ketika generasi tidak mengalami perubahan nilai. Untuk cara ini, diperlukan pemahaman yang baik mengenai masalah yang akan diteliti. Ketiga, adalah dengan menetapkan lamanya waktu simulasi. Lalu, bisa juga dengan cara mengombinasikan ketiga cara ini.

B. EGG HOLDER FUNCTION

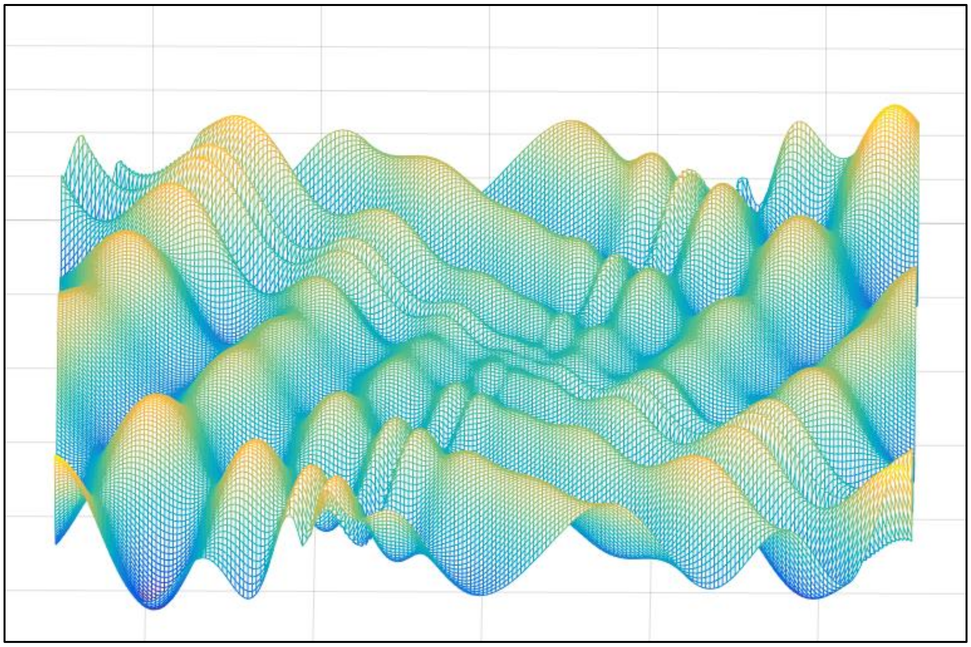
Diberikan suatu ruang vektor V dan f adalah sebuah pemetaan yang didefinsikan oleh $f\colon V \to \mathbb{R}$ dimana

$$f(\boldsymbol{x}^*) = \sum_{i=1}^{m-1} \left[-(x_{i+1} + 47) \sin \sqrt{\left| \frac{x_i}{2} + (x_{i+1} + 47) \right|} - x_i \sin \sqrt{|x_i - (x_{i+1} + 47)|} \right]$$

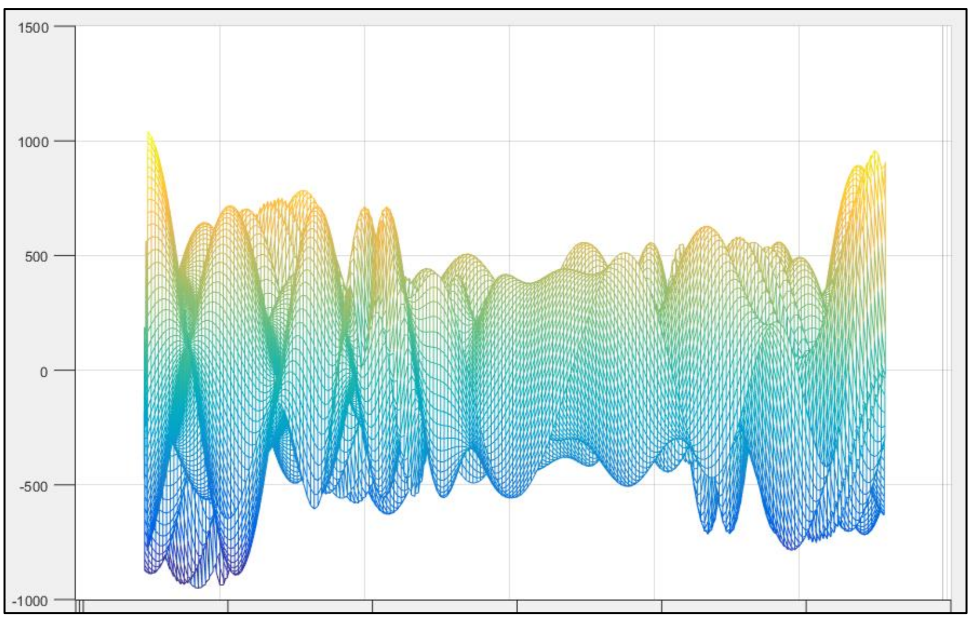
dengan syarat $-512 \leq x_i \leq 512$. Untuk bentuk tiga dimensi, nilai minimum global terletak pada $\boldsymbol{x}^* = (512{,}404{,}2319)$ dan $f(\boldsymbol{x}^*) \approx -959.64$. Interpretasi geometri 3D dari masalah diatas diberikan pada gambar dibawah ini.



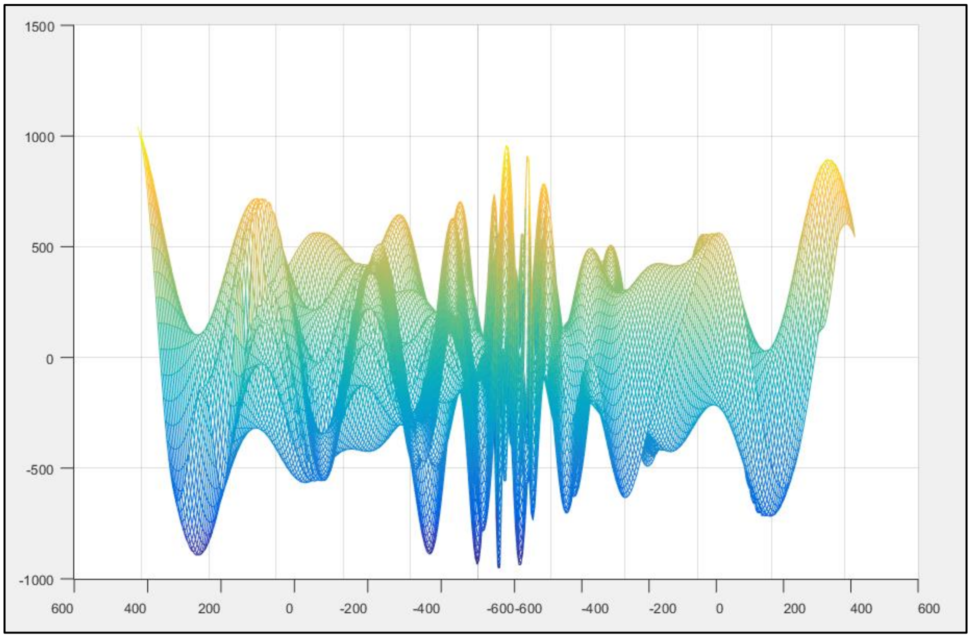
Gambar 1. *Tampak tiga dimensional dari Egg Holder Function*



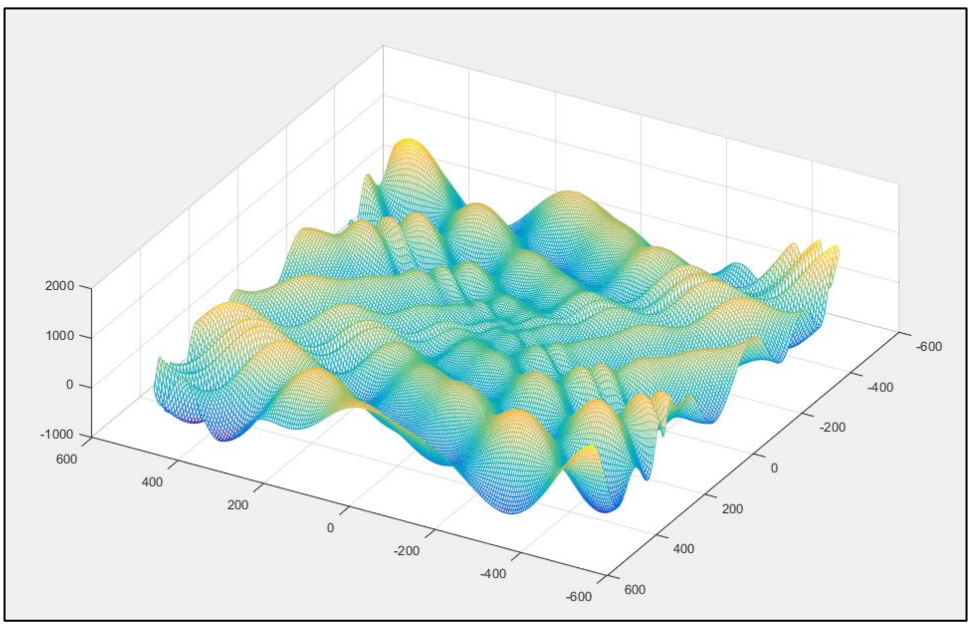
Gambar 2. *Tampak tiga dimensional dari Egg Holder Function*



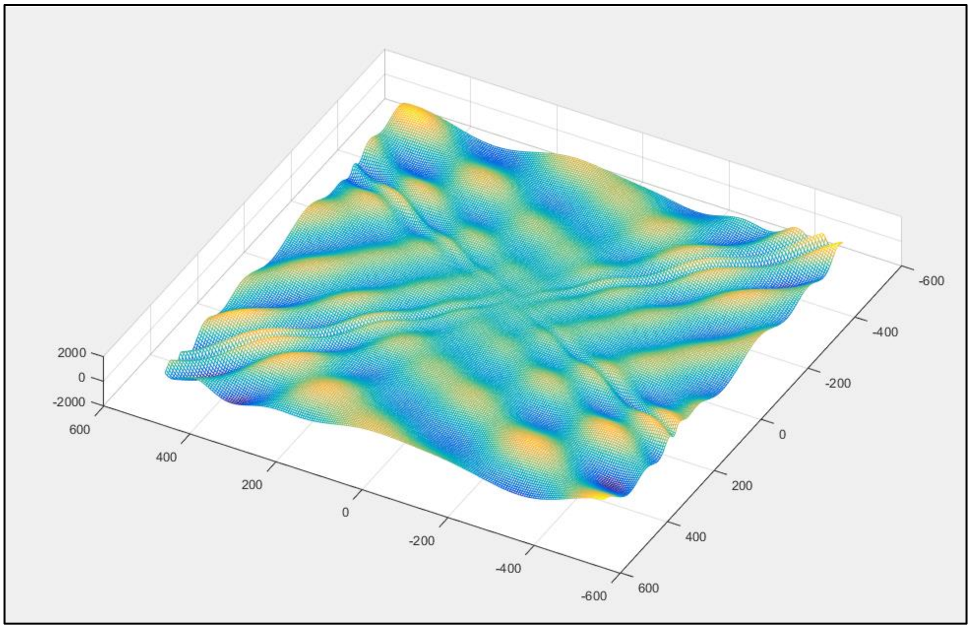
Gambar 3. *Tampak samping tiga dimensional dari Egg Holder Function*



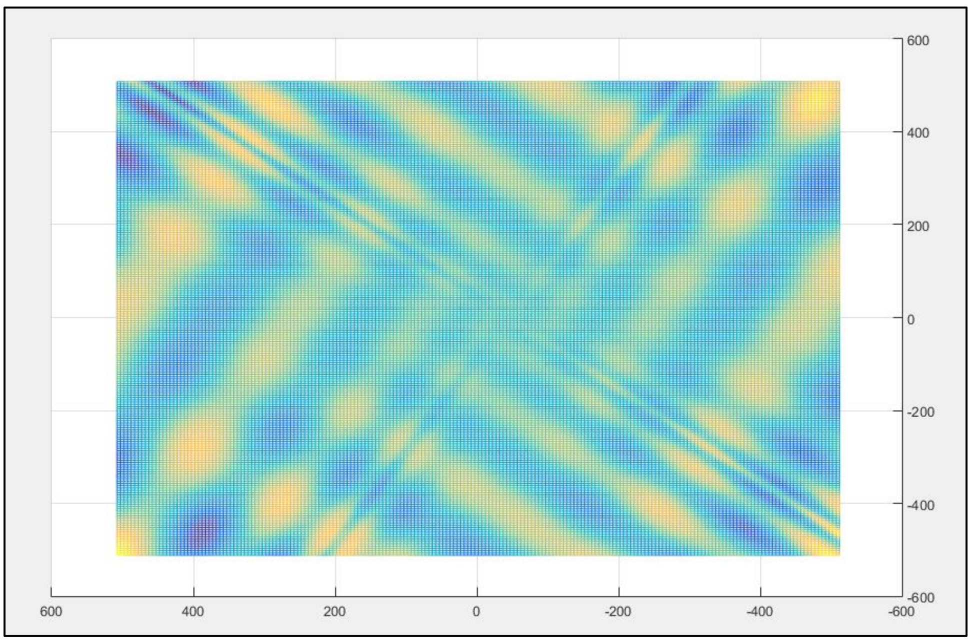
__Gambar 4.__ Tampak samping tiga dimensional dari Egg Holder Function



__Gambar 5.__ Tampak samping tiga dimensional dari Egg Holder Function



Gambar 6. Tampak atas tiga dimensional dari Egg Holder Function



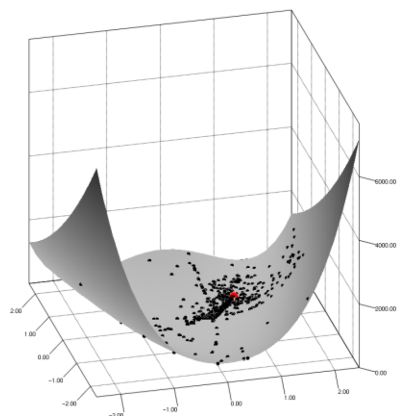
Gambar 7. Tampak atas tiga dimensional dari Egg Holder Function

Berdasarkan hasil plot, tampak banyak titik kritis yang terjadi. Artinya, pemilihan banyaknya populasi akan berpengaruh pada hasil algoritma Genetika. Bila populasi kecil, maka hasil algoritma Genetika yang dihasilkan akan menuju ke titik minimum lokal. Semakin besar, maka hasil algoritma Genetika akan mendekati titik maksimum lokal.

C. CONSTRUCTING INITIAL CONDITION AND DETERMINING PARAMETERS TO APPLY GENETIC ALGORITHM ON EGG HOLDER FUNCTION

1. ***Generate initial population***

   Pada tahap ini, dipilih banyaknya populasi awal adalah **1000**. Lalu, solusi akhir berupa angka biner dengan panjang bit **80**. Alasan mengapa dipilih angka ini adalah banyaknya titik minimum local / stationer pada kurva itu. Artinya, bila dipilih jumlah populasi kecil, tidak menutup kemungkinan bahwa solusi akhir dari algoritma Genetika adalah solusi lokal, bukan solusi global. Dibawah ini adalah ilustrasinya.

   Pada kasus ini, semakin banyak populasi awal, akan semakin bagus hasil yang dihasilkan

2. ***Selection***

   Pada tahap selection, digunakan sistem *roulette* dan pembangkitan bilangan acak $r \in [0,1]$ untuk menentukan individu mana yang akan terseleksi.

3. ***Crossover***

   Untuk tahap *crossover,* digunakan metode *simple crossover* sedemikian hingga akan dipilih individu yang memiliki bobot tertentu. Pemilihan individu akan bergantung pada peluang *crossover* dan suatu bilangan acak yang berdistribusi uniform pada $[0,1]$.

4. ***Mutation***

   Lalu, pada tahapan ini, akan dilakukan suatu mutasi acak dengan cara mengganti nilai alel pada lokus tertentu. Proses ini bergantung pada peluang *mutation* dan suatu bilangan acak yang berdistribusi uniform pada $[0,1]$ juga.

Stopping criteria yang digunakan adalah dengan menetapkan iterasi maksimum, yaitu **2000** iterasi untuk bentuk **dua variabel** dan **1000** iterasi untuk bentuk **lima variabel**.

D. GENETIC ALGORITHM FUNCTION LIST

```
%Mengubah nilai biner menjadi nilai riil ---------------------------
function [num] = bin2num(x)
    ukuran = size(x);
    pop = ukuran(1);
    gen = ukuran(2);
    for i = 1:pop
        temp = 0;
        for j = 1:gen
            temp = temp + x(i,j)*2^(gen-j);
        end
        fitness(i) = temp/(2^gen-1)*1024-512;
    end
    num = fitness';
end


%Pembangkitan populasi awal-----------------------------------------
function [Xbin] = generateEggHolderX(pop,gen)
    x = [];
    for i = 1:pop
        temp = 0;
        for k = 1:gen
            x(i,k) = round(rand(1));
        end
    end
    %Mencari nilai real dari nilai binary
    Xreal = [];
    while true
        for j = 1:pop
            temp = 0;
            for k = 1:gen
                temp = temp + x(j,k)*2^(gen-k);
            end
            Xreal(j) = (temp/(2^gen-1))*1024-512;
        end
        %Mengecek apakah generate population berada
pada -512 s/d 512
        if (sum(-512 <= Xreal(:)) + sum(Xreal(:) <=
512) == 2*pop)
            break;
        end
```

```
    end
    Xbin = x;
end


%Evaluasi Fungsi Fitness Dua Variabel------------------------------
function [fitness] = fitProb29twoVar(x1,x2)
    fitness = (x2+47).*sin(sqrt(abs(x2+x1/2+47))) +
x1.*sin(sqrt(abs(x1-x2-47)));
end

%Evaluasi Fungsi Fitness Lima Variabel-----------------------------
function [fitness] = fitProb29fiveVar(x1,x2,x3,x4,x5)
    ukuran = size(x1);
    x = [x1 x2 x3 x4 x5]; fitness = zeros([ukuran(1)
1]);
    for i = 1:4
        fitness = fitness +
x(:,i+1).*sin(sqrt(abs(x(:,i+1)+x(:,i)./2+47))) ...
                          +
x(:,i).*sin(sqrt(abs(x(:,i)-(x(:,i+1)+47))));
    end
end


%Proses SELECTION--------------------------------------------------
function [X] = selection_EggHolder (x,fitness)
    totFit = sum(abs(fitness));
    fitness = abs(fitness)/totFit;
    ukuran = size(x);
    pop = ukuran(1);

    %Membuat partisi roulette
    q = [];
    q(1) = 0; temp = 0; counterQ = 1;
    for i = 1:pop
        temp = temp + fitness(i);
        q(counterQ+1) = temp;
        counterQ = counterQ + 1;
    end

    %Menentukan individu yang terseleksi
    chosen = []; t = true;
    for counterChosen = 1:pop
```

```
        r = rand();
        counterQ = 1;
        while t || counterQ <= pop
            if ((q(counterQ)<r) && (r<=q(counterQ+1)))
                chosen(counterChosen) = counterQ;
            end
            counterQ = counterQ + 1; t = ~t;
        end
    end
    X = x(chosen,:);
End

%Proses CROSSOVER---------------------------------------------------
function [X] = crossover_EggHolder(x,crossRate)
    ukuran = size(x);
    pop = ukuran(1);
    gen = ukuran(2);
    for i = 1:pop
        r = rand(1);
        if r<crossRate
            while true
                parent1 = round((pop-1)*rand(1)+1);
                parent2 = round((pop-1)*rand(1)+1);
                if (parent1 ~= parent2)
                    tempparent = [x(parent1,:);
x(parent2,:)];
                    break;
                end
            end

            %mendapatkan indeks yang akan disilangkan
            posCross = round((gen-1)*rand(1)+1);

            %CrossOver
            for k = posCross:gen
                tempparent([2*k-1 2*k]) =
tempparent([2*k 2*k-1]);
            end
            x(parent1,:) = tempparent(1,:);
            x(parent2,:) = tempparent(2,:);
        end
    end
    X = x;
end
```

```
%Proses MUTATION----------------------------------------------------
function [X] = mutation_EggHolder(x,mutationRate)
    ukuran = size(x);
    pop = ukuran(1);
    gen = ukuran(2);
    for i = 1:pop
        %Mendapatkan individu yang akan mutasi
        posMutPop = round((pop-1)*rand(1)+1);

        %Mendapatkan posisi gen yang akan termutasi
        posMutGen = round((gen-1)*rand(1)+1);
        pMutRand = rand(1);

        %Mutasi
        if pMutRand < mutationRate
            x(posMutPop,posMutGen) = 1-
x(posMutPop,posMutGen);
        end
    end
    X = x;
end
```

E. SOURCE CODE GENETIC ALGORITHM TO FIND MINIMUM GLOBAL OF EGG HOLDER FUNCTION WITH TWO VARIABLES

```
clc; clear; tic;

% Alert Sound
WarnWave = [sec(1:.15:500)];
Audio = audioplayer(WarnWave, 22050);
play(Audio);

%% INITIALIZATION
gen = 80; pop = 1000;

%Pembangkitan populasi awal berupa angka biner
x1 = generateEggHolderX(pop,gen);
x2 = generateEggHolderX(pop,gen);

%Menterjemahkan nilai biner menjadi nilai riil
x1val = bin2num(x1);
x2val = bin2num(x2);

%Mengevaluasi nilai fitness
fitness = fitProb29twoVar(bin2num(x1),bin2num(x2));

pcross = 0.6; %Peluang crossover
pmut = 0.4; %Peluang mutation
max_Iteration = 2000; %maksimum iterasi/generasi

%% GENETIC ALGORITHM
for iterasi = 1:max_Iteration
    iterasi
    %% SELECTION %%
    x1new = selection_EggHolder(x1,fitness);
    x2new = selection_EggHolder(x2,fitness);

    %% CROSS OVER %%
    x1new = crossover_EggHolder(x1new, pcross);
    x2new = crossover_EggHolder(x2new, pcross);

    %% MUTATION %%
    x1new = mutation_EggHolder(x1new, pmut);
    x2new = mutation_EggHolder(x2new, pmut);
```

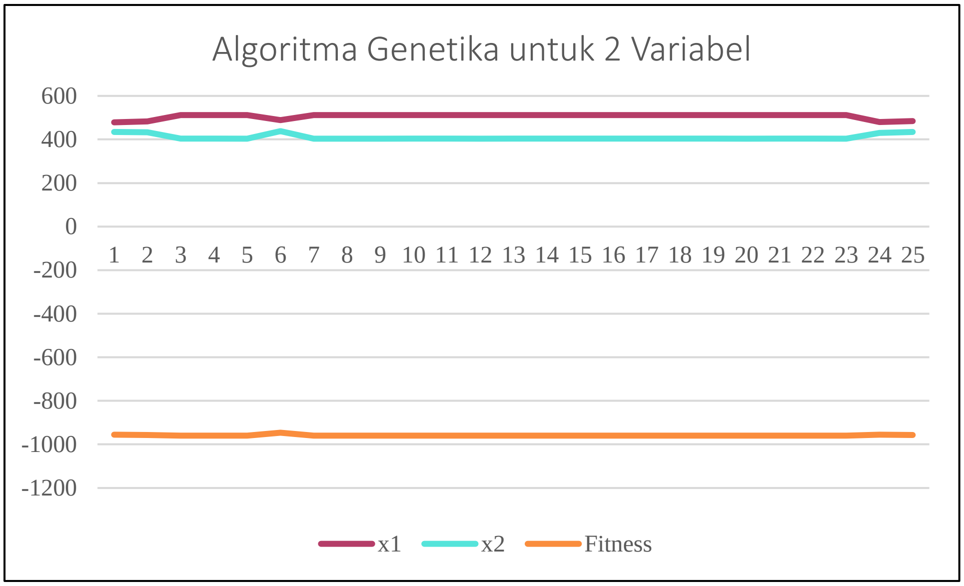
```
    %% NEW GENERATION %%
    fitnessbaru =
fitProb29twoVar(bin2num(x1new),bin2num(x2new));
    for i = 1:pop
        if fitnessbaru(i)>fitness(i)
            x1(i,:) = x1new(i,:);
            x2(i,:) = x2new(i,:);
        end
    end
    fitness = fitProb29twoVar(bin2num(x1),bin2num(x2));
    temp(iterasi) = max(fitness);
    posisi = find(fitness==max(fitness));
    tempx1(iterasi) = max(bin2num(x1(posisi(1),:)));
    tempx2(iterasi) = max(bin2num(x2(posisi(1),:)));
end

%% EVALUASI POPULASI
plot(1:max_Iteration,temp); hold on;
plot(1:max_Iteration,tempx1); hold on;
plot(1:max_Iteration,tempx2); hold on;
legend('fitness','x1','x2');

result = find(fitness == max(fitness));
index = result(1);
fitness = -
fitProb29twoVar(bin2num(x1(index,:)),bin2num(x2(index,:
)))
[bin2num(x1(index,:)) bin2num(x2(index,:))]
toc;

%Alert sound
WarnWave = [sec(1:.15:750),tan(1:.45:750)];
Audio = audioplayer(WarnWave, 22050);
play(Audio);
```

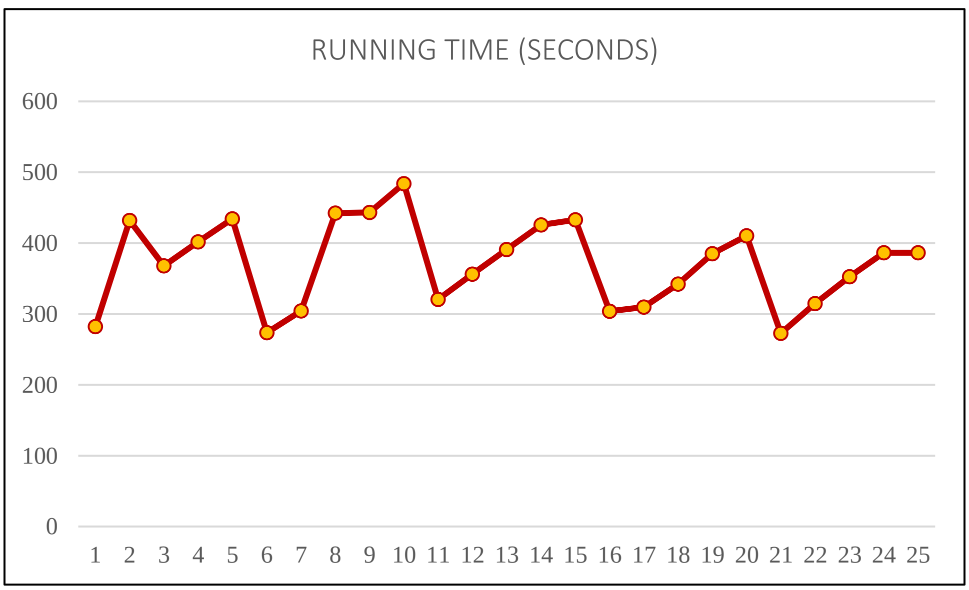
F. RESULT OF GENETIC ALGORITHM TO FIND MINIMUM GLOBAL OF EGG HOLDER FUNCTION WITH TWO VARIABLES

Tabel 1. *Hasil Algoritma Genetika Egg Holder Function Dua Variabel*

ITERATION	CROSSOVER RATE	MUTATION RATE	x_1	x_2	FITNESS	RUNNING TIME (SECONDS)
2000	0.1	0.1	478.7278	434.1875	-955.2360	282.278518
	0.2	0.1	482.3123	432.8379	-956.9179	432.030481
	0.3	0.1	512	404	-959.5797	368.066764
	0.4	0.1	512	404	-959.5797	401.589102
	0.5	0.1	512	404.2318	-959.6407	434.140071
	0.1	0.2	488.1959	438.2517	-946.2238	273.510772
	0.2	0.2	512	404.25	-959.6403	304.098096
	0.3	0.2	512	404.2318	-959.6407	442.330724
	0.4	0.2	512	404.2318	-959.6407	443.126006
	0.5	0.2	512	404	-959.5797	484.131475
	0.1	0.3	512	404	-959.5797	320.589552
	0.2	0.3	512	404.25	-959.6403	356.142505
	0.3	0.3	512	404	-959.5797	391.021893
	0.4	0.3	512	404	-959.5797	425.923911
	0.5	0.3	512	404	-959.5797	432.755949
	0.1	0.4	512	404	-959.5797	303.756481
	0.2	0.4	512	404	-959.5797	309.700877
	0.3	0.4	512	404	-959.5797	342.375381
	0.4	0.4	512	404	-959.5797	385.122469
	0.5	0.4	512	404.2318	-959.6407	410.181720
	0.1	0.5	512	404	-959.5797	272.709739
	0.2	0.5	512	404	-959.5797	314.476942
	0.3	0.5	512	404.25	-959.6403	352.397931
	0.4	0.5	479.9805	430.5	-955.8534	386.634612
	0.5	0.5	484	434.5172	-956.4100	386.634612

Setelah melakukan *running* pada algoritma Genetika yang sudah dibuat, diperoleh tabel 1. Pada tabel tersebut dapat ditinjau perilaku dari masing-masing variabel fungsi *fitness,* dan *running time* yang disajikan pada grafik berikut secara bersesuaian dengan hasil iterasi.





Apabila kita amati hasil diatas, nilai dari x_1, x_2, dan fitness konvergen ke suatu titik. Artinya, algoritma Genetika yang dirancang sudah mendekati nilai sebenarnya. Berjalan dari hal ini, diperoleh

$$x^* = (512, 404.2318) \text{ dan } f(x^*) = -959.6407$$

Jawaban ini sudah tervalidasi sesuai dengan jawaban pada textbook.

G. SOURCE CODE GENETIC ALGORITHM TO FIND MINIMUM GLOBAL OF EGG HOLDER FUNCTION WITH FIVE VARIABLES – WITH GUI

```
function varargout = EggHolderSimulation(varargin)
% EGGHOLDERSIMULATION MATLAB code for EggHolderSimulation.fig
%      EGGHOLDERSIMULATION, by itself, creates a new
EGGHOLDERSIMULATION or raises the existing
%      singleton*.
%
%      H = EGGHOLDERSIMULATION returns the handle to a new
EGGHOLDERSIMULATION or the handle to
%      the existing singleton*.
%
%      EGGHOLDERSIMULATION('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in EGGHOLDERSIMULATION.M with the given
input arguments.
%
%      EGGHOLDERSIMULATION('Property','Value',...) creates a new
EGGHOLDERSIMULATION or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before EggHolderSimulation_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to EggHolderSimulation_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
EggHolderSimulation

% Last Modified by GUIDE v2.5 09-Dec-2018 21:39:59

% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@EggHolderSimulation_OpeningFcn, ...
                   'gui_OutputFcn',
@EggHolderSimulation_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before EggHolderSimulation is made visible.
function EggHolderSimulation_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to EggHolderSimulation (see
VARARGIN)

% Choose default command line output for EggHolderSimulation
handles.output = hObject;
axes(handles.axes1); imshow('mtk.png');

axes(handles.axes2);
xlabel ('ITERATION(s)','FontSize',12,'FontWeight','bold','color','k');
ylabel ('VALUE','FontSize',12,'FontWeight','bold','color','k');
set(handles.axes2,'XColor','k');
set(handles.axes2,'YColor','k');
grid on;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes EggHolderSimulation wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = EggHolderSimulation_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tic; cla; grid on;
WarnWave = [sec(1:.15:500)];
Audio = audioplayer(WarnWave, 22050);
play(Audio);

set(handles.edit6,'String','');
set(handles.edit7,'String','');
set(handles.edit8,'String','');
set(handles.edit9,'String','');
set(handles.edit10,'String','');
set(handles.edit11,'String','');
set(handles.edit12,'String','');

pop = str2num(get(handles.edit1,'String'));
gen = str2num(get(handles.edit2,'String'));
pcross = str2num(get(handles.edit3,'String'));
pmut = str2num(get(handles.edit4,'String'));
max_Iteration = str2num(get(handles.edit5,'String'));

%% INITIALIZATION

x1 = generateEggHolderX(pop,gen);
x2 = generateEggHolderX(pop,gen);
x3 = generateEggHolderX(pop,gen);
```

```
x4 = generateEggHolderX(pop,gen);
x5 = generateEggHolderX(pop,gen);

x1val = bin2num(x1);
x2val = bin2num(x2);
x3val = bin2num(x3);
x4val = bin2num(x4);
x5val = bin2num(x5);

fitness =
fitProb29fiveVar(bin2num(x1),bin2num(x2),bin2num(x3),bin2num(x4),bin2n
um(x5));

%% GENETIC ALGORITHM

for iterasi = 1:max_Iteration
    pause(0.01);
    set(handles.edit13,'String',num2str(iterasi));
    %% SELECTION %%
    x1new = selection_EggHolder(x1,fitness);
    x2new = selection_EggHolder(x2,fitness);
    x3new = selection_EggHolder(x3,fitness);
    x4new = selection_EggHolder(x4,fitness);
    x5new = selection_EggHolder(x5,fitness);

    %% CROSS OVER %%
    x1new = crossover_EggHolder(x1new,pcross);
    x2new = crossover_EggHolder(x2new,pcross);
    x3new = crossover_EggHolder(x3new,pcross);
    x4new = crossover_EggHolder(x4new,pcross);
    x5new = crossover_EggHolder(x5new,pcross);

    %% MUTATION %%
    x1new = mutation_EggHolder(x1new,pmut);
    x2new = mutation_EggHolder(x2new,pmut);
    x3new = mutation_EggHolder(x3new,pmut);
    x4new = mutation_EggHolder(x4new,pmut);
    x5new = mutation_EggHolder(x5new,pmut);

    %% NEW GENERATION %%
    fitnessbaru =
fitProb29fiveVar(bin2num(x1new),bin2num(x2new),bin2num(x3new),bin2num(
x4new),bin2num(x5new));
    for i = 1:pop
```

```
        if fitnessbaru(i)>fitness(i)
            x1(i,:) = x1new(i,:);
            x2(i,:) = x2new(i,:);
            x3(i,:) = x3new(i,:);
            x4(i,:) = x4new(i,:);
            x5(i,:) = x5new(i,:);
        end
        if fitnessbaru == fitness
            break;
        end
    end
    fitness =
fitProb29fiveVar(bin2num(x1),bin2num(x2),bin2num(x3),bin2num(x4),bin2n
um(x5));
    temp(iterasi) = max(fitness);
    posisi = find(fitness==max(fitness));
    tempx1(iterasi) = max(bin2num(x1(posisi(1),:)));
    tempx2(iterasi) = max(bin2num(x2(posisi(1),:)));
    tempx3(iterasi) = max(bin2num(x3(posisi(1),:)));
    tempx4(iterasi) = max(bin2num(x4(posisi(1),:)));
    tempx5(iterasi) = max(bin2num(x5(posisi(1),:)));
    plot(iterasi,temp(iterasi),'.','linewidth',1.5,'color','k'); hold
on;
    plot(iterasi,tempx1(iterasi),'.','linewidth',1.5,'color','r');
hold on;
    plot(iterasi,tempx2(iterasi),'.','linewidth',1.5,'color','b');
hold on;
    plot(iterasi,tempx3(iterasi),'.','linewidth',1.5,'color','g');
hold on;
    plot(iterasi,tempx4(iterasi),'.','linewidth',1.5,'color','c');
hold on;
    plot(iterasi,tempx5(iterasi),'.','linewidth',1.5,'color','y');
hold on;
end

%% EVALUASI POPULASI
cla;
plot(1:max_Iteration,temp  ,'linewidth',1.5,'color','k'); hold on;
plot(1:max_Iteration,tempx1,'linewidth',1.5,'color','r'); hold on;
plot(1:max_Iteration,tempx2,'linewidth',1.5,'color','g'); hold on;
plot(1:max_Iteration,tempx3,'linewidth',1.5,'color','b'); hold on;
plot(1:max_Iteration,tempx4,'linewidth',1.5,'color','c'); hold on;
plot(1:max_Iteration,tempx5,'linewidth',1.5,'color','y'); hold on;
legend('fitness','x1','x2','x3','x4','x5');
```

```
result = find(fitness == max(fitness));
index = result(1);
fitness = -
fitProb29fiveVar(bin2num(x1(index,:)),bin2num(x2(index,:)),bin2num(x3(
index,:)),bin2num(x4(index,:)),bin2num(x5(index,:)));

set(handles.edit6,'String',num2str(bin2num(x1(index,:))));
set(handles.edit7,'String',num2str(bin2num(x2(index,:))));
set(handles.edit8,'String',num2str(bin2num(x3(index,:))));
set(handles.edit9,'String',num2str(bin2num(x4(index,:))));
set(handles.edit10,'String',num2str(bin2num(x5(index,:))));
set(handles.edit11,'String',num2str(fitness));
set(handles.edit12,'String',round(toc*1000)/1000);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit1,'String','');
set(handles.edit2,'String','');
set(handles.edit3,'String','');
set(handles.edit4,'String','');
set(handles.edit5,'String','');
set(handles.edit6,'String','');
set(handles.edit7,'String','');
set(handles.edit8,'String','');
set(handles.edit9,'String','');
set(handles.edit10,'String','');
set(handles.edit11,'String','');
set(handles.edit12,'String','');
set(handles.edit13,'String','');
cla;
legend('hide');

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
```

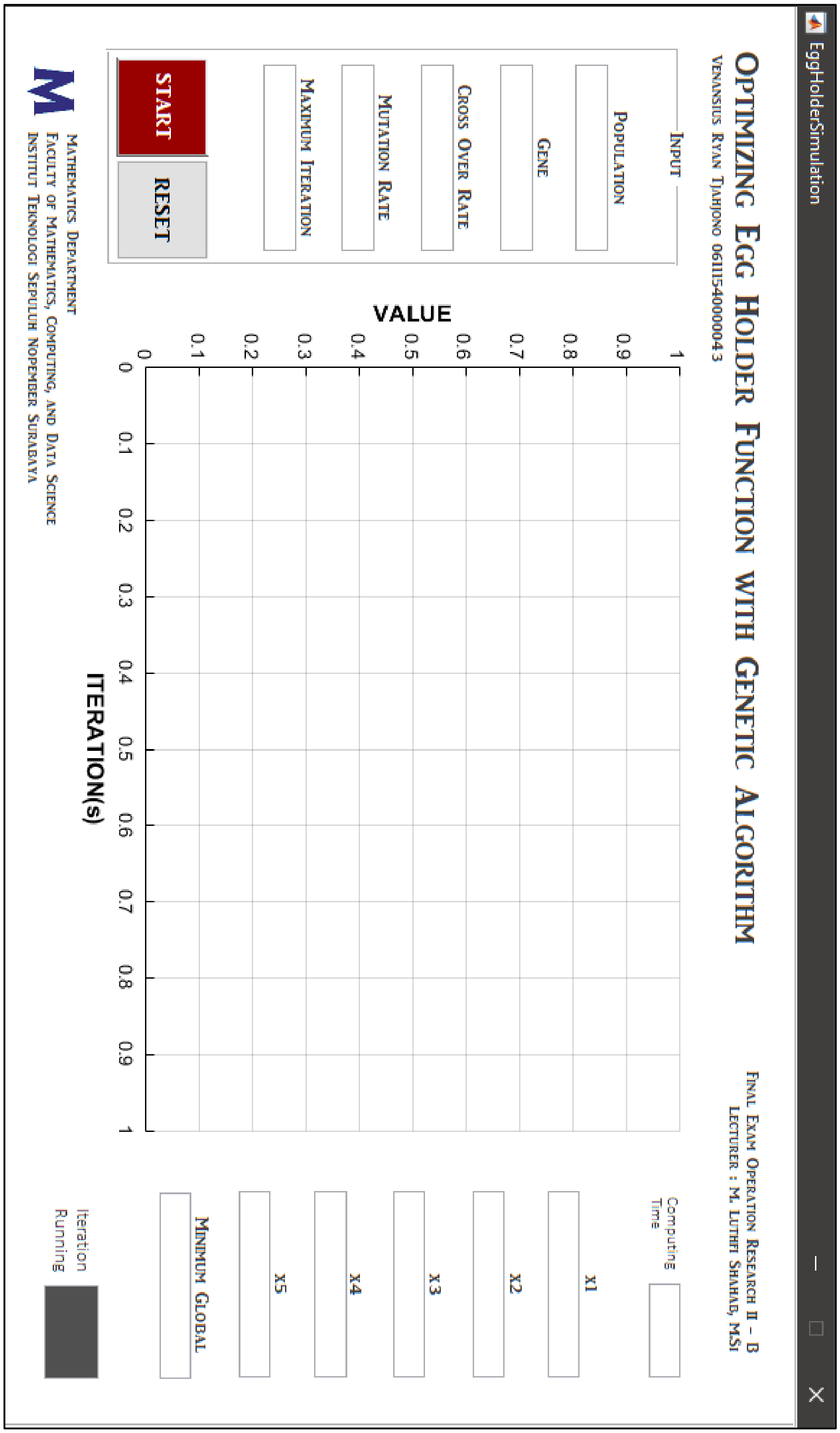
```
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

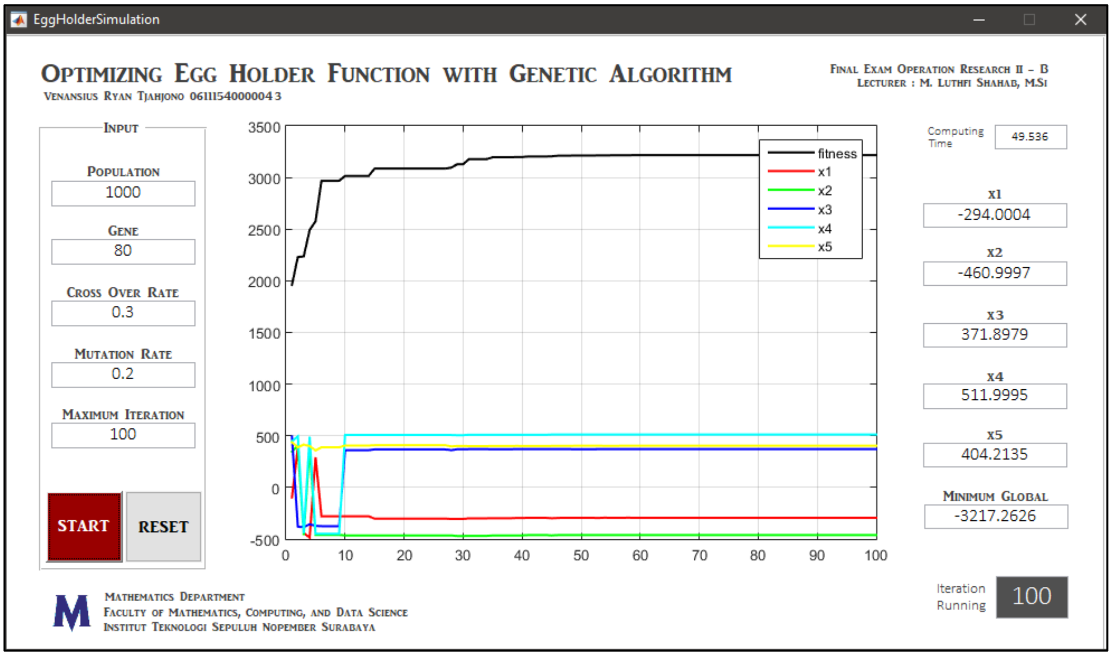
```
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

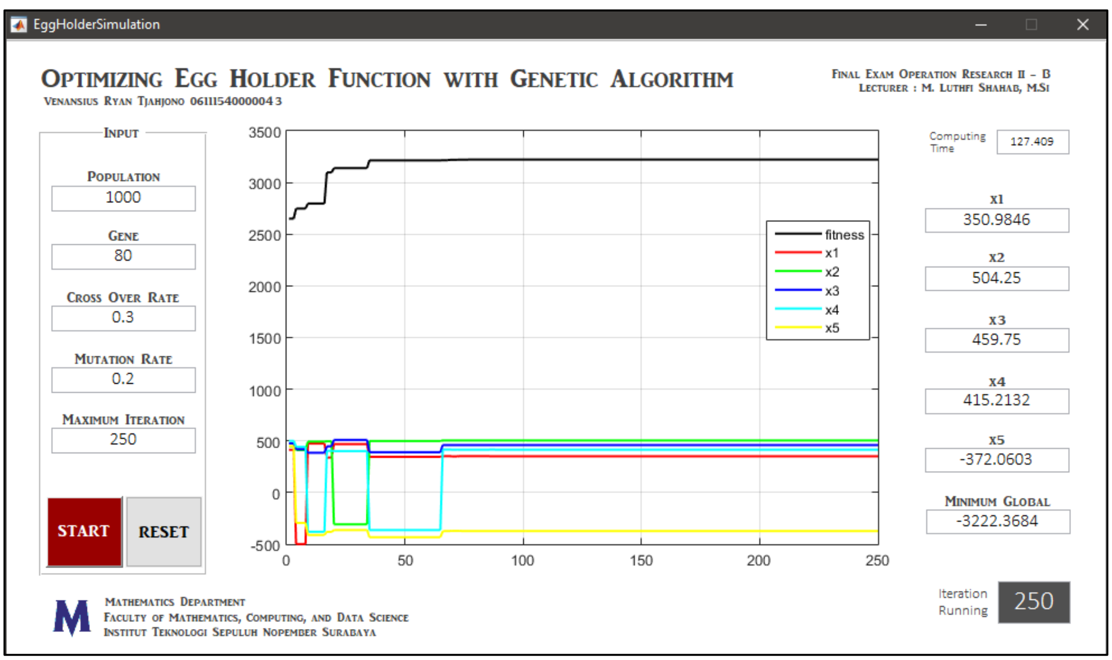
```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13
as a double

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

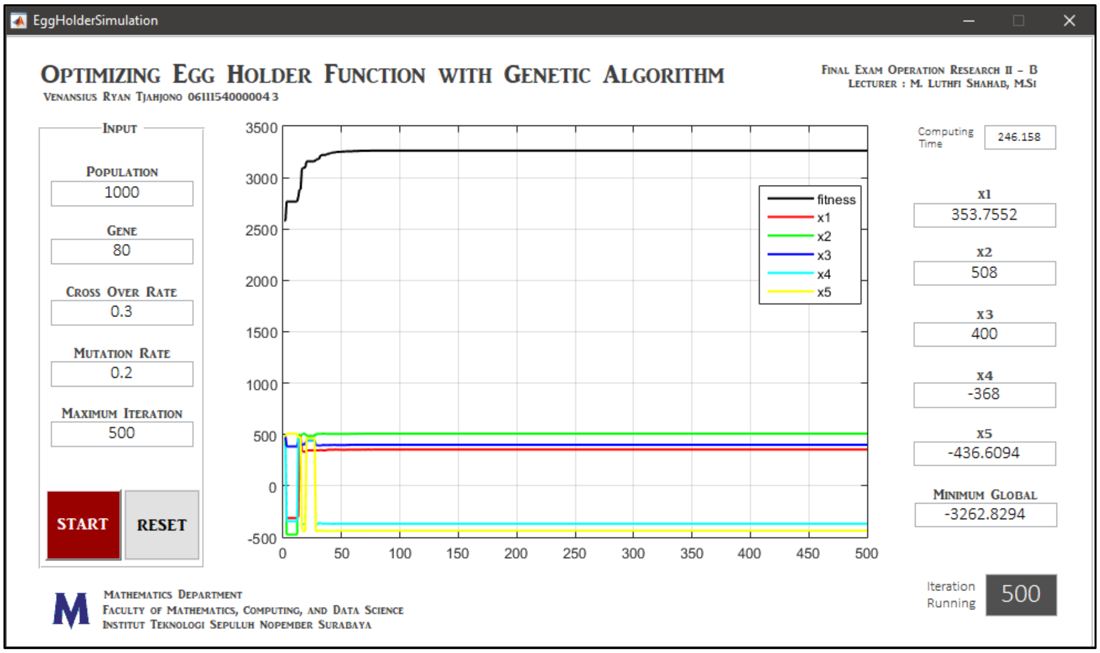
TAMPILAN GUI

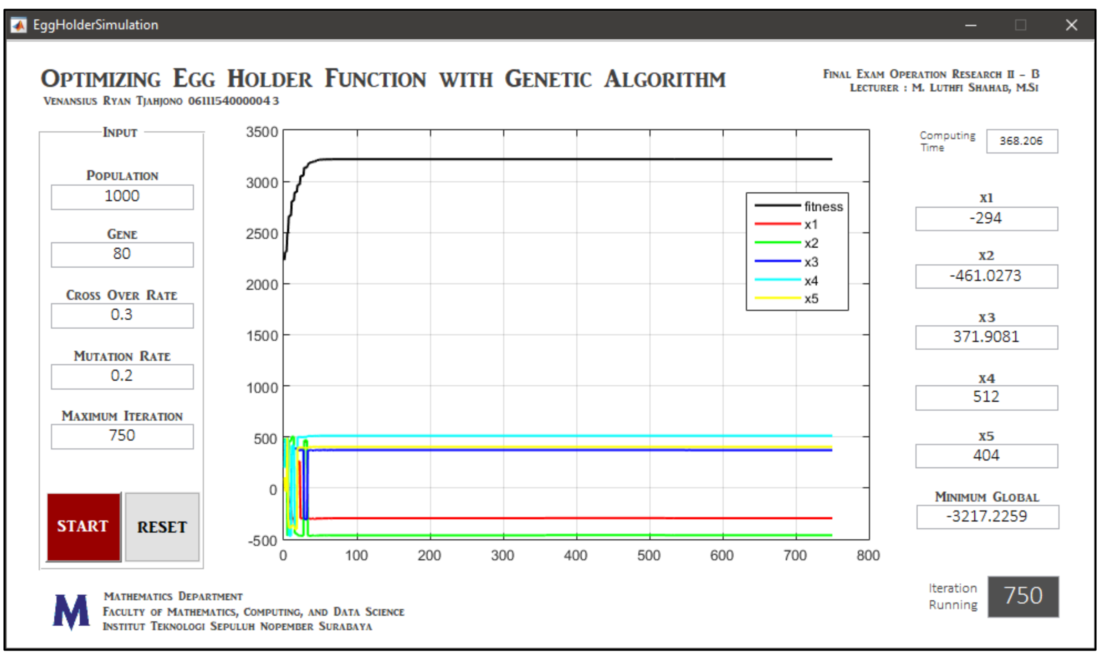


Berikut akan disajikan bentuk hasil *running* dari GUI yang sudah dibuat. Grafik yang ditampilkan adalah perubahan nilai x_1, x_2, x_3, x_4, x_5, dan fitness setiap iterasi hingga maksimum iterasi tercapai.






EggHolderSimulation
OPTIMIZING EGG HOLDER FUNCTION WITH GENETIC ALGORITHM
VENANSIUS RYAN TJAHJONO 06111540000043
FINAL EXAM OPERATION RESEARCH II - B
LECTURER : M. LUTHFI SHAHAB, M.SI
INPUT
POPULATION
1000
GENE
80
CROSS OVER RATE
0.3
MUTATION RATE
0.2
MAXIMUM ITERATION
500
START
RESET
fitness
x1
x2
x3
x4
x5
Computing Time
246.158
x1
353.7552
x2
508
x3
400
x4
-368
x5
-436.6094
MINIMUM GLOBAL
-3262.8294
Iteration Running
500
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS, COMPUTING, AND DATA SCIENCE
INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA

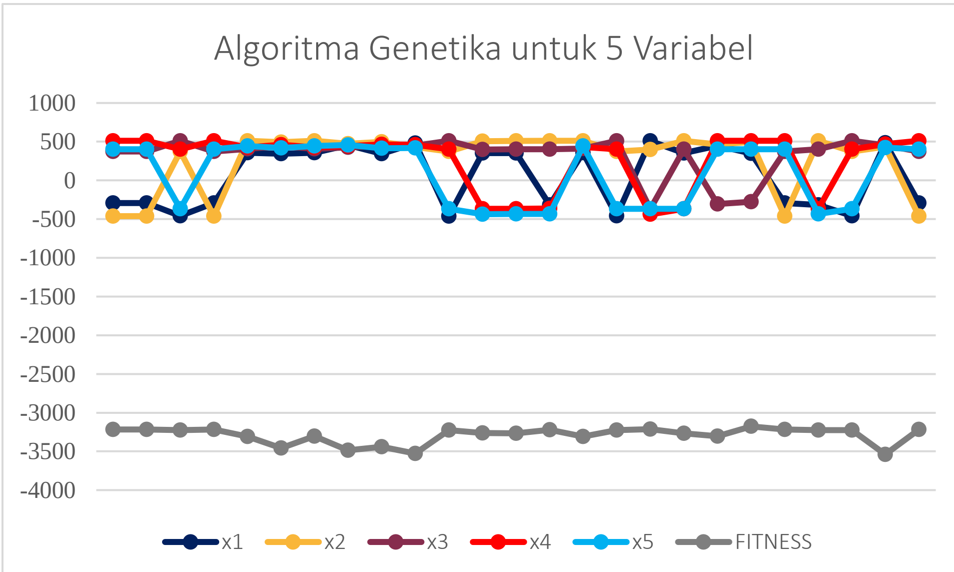

EggHolderSimulation
OPTIMIZING EGG HOLDER FUNCTION WITH GENETIC ALGORITHM
VENANSIUS RYAN TJAHJONO 06111540000043
FINAL EXAM OPERATION RESEARCH II - B
LECTURER : M. LUTHFI SHAHAB, M.SI
INPUT
POPULATION
1000
GENE
80
CROSS OVER RATE
0.3
MUTATION RATE
0.2
MAXIMUM ITERATION
750
START
RESET
fitness
x1
x2
x3
x4
x5
Computing Time
368.206
x1
-294
x2
-461.0273
x3
371.9081
x4
512
x5
404
MINIMUM GLOBAL
-3217.2259
Iteration Running
750
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS, COMPUTING, AND DATA SCIENCE
INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA

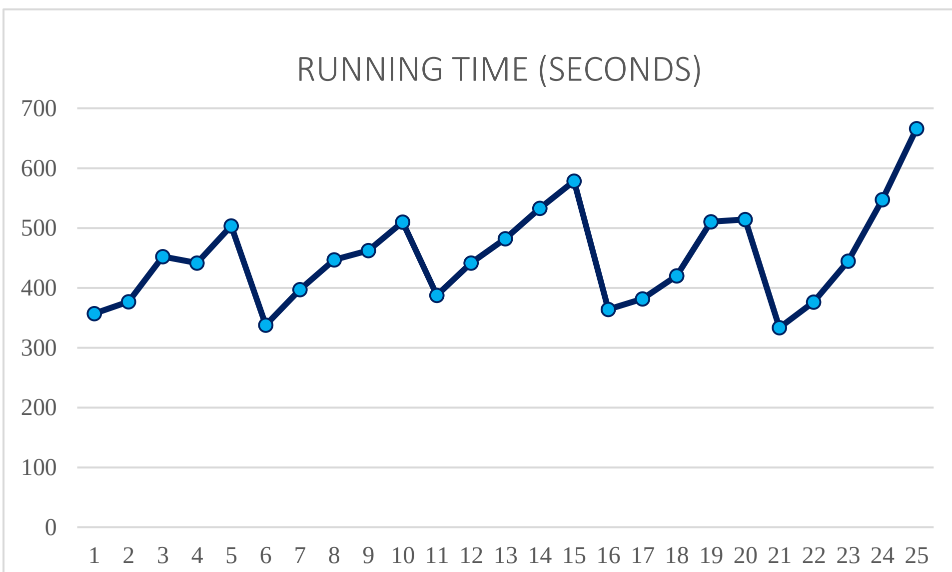
H. RESULT OF GENETIC ALGORITHM TO FIND MINIMUM GLOBAL OF EGG HOLDER FUNCTION WITH FIVE VARIABLES – IN TABLE REPRESENTATION

Tabel 2. *Hasil Algoritma Genetika Egg Holder Function Lima Variabel*

ITERATION	CROSSOVER RATE	MUTATION RATE	x_1	x_2	x_3	x_4	x_5	FITNESS	RUNNING TIME (SECONDS)
1000	0.1	0.1	-293.5	-460.7832	371.8867	512	404.1889	-3.2173e+03	357.028194
	0.2	0.1	-294	-461.0352	372	512	404.1875	-3.2173e+03	376.759630
	0.3	0.1	-459.6228	371.7793	512	403.6875	-368.8074	-3.2274e+03	452.437051
	0.4	0.1	-293.6824	-460.8828	372	512	404.1890	-3.2173e+03	441.534225
	0.5	0.1	356.75	512	408.7068	428	444.0052	-3.3064e+03	503.574079
	0.1	0.2	344.2206	494.9829	445.5	462	417.5442	-3.4581e+03	337.715035
	0.2	0.2	356.7446	512	408	428	444	-3.3055e+03	397.036524
	0.3	0.2	457.4159	473	428.5625	443.8208	457.8575	-3.4855e+03	446.978714
	0.4	0.2	346.25	497.5	448.0283	464.875	415.5	-3.4401e+03	462.312191
	0.5	0.2	481.75	432.2344	447.4999	462.5	418	-3.5266e+03	509.789642
	0.1	0.3	-460	371.8142	512	403.6951	-368.8096	-3.2273e+03	387.231739
	0.2	0.3	353.75	508	400	-368	-436.6092	-3.2628e+03	441.520948
	0.3	0.3	354.4524	508.9375	400.9641	-365.5	-434.25	-3.2678e+03	482.376234
	0.4	0.3	-315.3563	512	403.2126	-363.8036	-432	-3.2210e+03	532.794230
	0.5	0.3	356.7446	512	409.3154	429.75	445.5	-3.3086e+03	578.146469
	0.1	0.4	-459.6228	371.7793	512	403.6951	-368.8095	-3.2274e+03	364.141560
	0.2	0.4	509.5635	400	-369	-438.75	-368.4628	-3.2116e+03	381.352953
	0.3	0.4	354.4794	508.9738	401	-365.75	-365.75	-3.2678e+03	420.352654
	0.4	0.4	450	464.9342	-304	512	404	-3.3046e+03	510.418221
	0.5	0.4	347.7807	499.8750	-275.7995	512	404.1875	-3.1776e+03	514.039639
	0.1	0.5	-293.6764	-460.8750	372	512	404.1875	-3.2173e+03	333.316463
	0.2	0.5	-315.3563	512	403.3678	-365.4149	-434	-3.2249e+03	376.511915
	0.3	0.5	-459.6228	371.7793	512	403.6951	-368.8095	-3.2274e+03	444.769559
	0.4	0.5	486.2500	436.7979	452.1741	468	423.5156	-3.5388e+03	547.221686
	0.5	0.5	-293.6692	-460.8656	371.8934	512	404.1889	-3.2173e+03	666.037126

Setelah melakukan *running* pada algoritma Genetika yang sudah dibuat, diperoleh tabel 1. Pada tabel tersebut dapat ditinjau perilaku dari masing-masing variabel fungsi *fitness,* dan *running time* yang disajikan pada grafik berikut secara bersesuaian dengan hasil iterasi.





Apabila kita amati hasil diatas, nilai dari x_1, x_2, x_3, x_4, x_5, dan fitness konvergen ke suatu titik. Namun, masih bentuk yang dihasilkan masih belum stabil. Hal ini dikarenakan banyaknya titik ekstrim pada egg holder function. Artinya, diperlukan populasi yang lebih besar lagi agar hasilnya lebih akurat. Hasil yang diperoleh dari algoritma Genetika ini adalah

$$x^* = (486.25, 436.7979, 452.1741, 468, 423.5156) \text{ dan } f(x^*) \approx -3538.8$$