LAMPIRAN

Lampiran 1 Listing Program Aliran Daya Pada Sistem IEEE 30 Bus Setelah Penempatan STATCOM Dengan Optimasi BFO

```
clc;
clear;
basemva = 100;  accuracy = 10^-8; accel = 1.8; maxiter = 1000;
% Bus Data IEEE30
% Line Data IEEE30
%% Bacterial Foraging Optimization
Nvar =3;       % Jumlah STATCOM
NBac = 20;     % Number of bacteria in the colony
NChem = 10;    % Number of chemotactic steps
NSwim = 4;     % Number of swim steps
NRep = 10;      % Number of reproductive steps
NEDi = 10;      % Number of elimination and dispersal steps
Sr = NBac/2;  % The number of bacteria reproductions (splits) per
generation
PEDi = 0.5;    % The probability that each bacteria will be
eliminated/dispersed
RLeng(:,1)=25*ones(NBac,1); % the run length unit (the size of the step
taken in each run or tumble)
Rb1 = 1;       % Batas bawah interval Lokasi
Ra1 = max(busdata(:,1));      % Batas akhir interval Lokasi
Rb2 =-5;       % Batas bawah interval MVAR
Ra2 =100;      % Batas atas interval MVAR

% the initital posistions
for nn = 1 : NBac
    X1(:,nn,1,1,1) = randperm(Ra1,Nvar);
    X2(:,:,1,1,1) = ((rand(NBac,Nvar)*(Ra2-Rb2))+Rb2)';
end

for lll = 1:NEDi
    % (3) Reproduction loop
    for kkk = 1:NRep
        % (4) Chemotaxis (swim/tumble) loop
        for jjj = 1:NChem
            % (4.1) Chemotatic step
            for iii = 1:NBac
            % (4.2) Fitness function
            Fit(iii,jjj,kkk,lll) =
EvaluasiIndividu(busdata,linedata,X1(:,iii,jjj,kkk,lll),X2(:,iii,jjj,kkk,
lll),Nvar);
            % (4.3) Jlast
            Fitlast=Fit(iii,jjj,kkk,lll);
            % (4.4) Tumble
            Delta(:,iii) = unifrnd(-1,1,Nvar,1);
            % (4.5) Move
            X1(:,iii,jjj+1,kkk,lll)= randperm(Ra1,Nvar);
            X2(:,iii,jjj+1,kkk,lll)=
X2(:,iii,jjj,kkk,lll)+RLeng(iii,kkk)*Delta(:,iii)/sqrt(Delta(:,iii)'*Delt
a(:,iii));
            for chck=1:Nvar
                if X2(chck,iii,jjj+1,kkk,lll) > Ra2
                   X2(chck,iii,jjj+1,kkk,lll) = Ra2;
                elseif X2(chck,iii,jjj+1,kkk,lll) < Rb2
```

```
                    X2(chck,iii,jjj+1,kkk,lll) = Rb2;
                end
            end
            % (4.6) New fitness function
            Fit(iii,jjj+1,kkk,lll) =
EvaluasiIndividu(busdata,linedata,X1(:,iii,jjj+1,kkk,lll),X2(:,iii,jjj+1,
kkk,lll),Nvar);
            % (4.7) Swimming
            m=0; % counter for swim length
            while m < NSwim
                m = m + 1;
                 if Fit(iii,jjj+1,kkk,lll)> Fitlast
                    Fitlast = Fit(iii,jjj+1,kkk,lll);
                    X1(:,iii,jjj+1,kkk,lll) = randperm(Ra1,Nvar);
                    X2(:,iii,jjj+1,kkk,lll) =
X2(:,iii,jjj+1,kkk,lll)+RLeng(iii,kkk)*Delta(:,iii)/sqrt(Delta(:,iii)'*De
lta(:,iii));
                    for chck=1:Nvar
                        if  X2(chck,iii,jjj+1,kkk,lll) > Ra2
                            X2(chck,iii,jjj+1,kkk,lll) = Ra2;
                        elseif X2(chck,iii,jjj+1,kkk,lll) < Rb2
                            X2(chck,iii,jjj+1,kkk,lll) = Rb2;
                        end
                    end
   Fit(iii,jjj+1,kkk,lll) =
EvaluasiIndividu(busdata,linedata,X1(:,iii,jjj+1,kkk,lll),X2(:,iii,jjj+1,
kkk,lll),Nvar);
                 else
                    m = NSwim;
                 end
            end
            Fit(iii,jjj,kkk,lll) = Fitlast;
            end % (4.8) Next bacterium
        end     % (5) if j < Nc, chemotaxis
        % (6) Reproduction
        % (6.1) Health
        Fithealth = sum(Fit(:,:,kkk,lll),2);      % Set the health of
each of the S bacteria
        [Fithealth,sortind] = sort(Fithealth,'descend');% Sorts bacteria
in order of descending values
        X1(:,:,1,kkk+1,lll) = X1(:,sortind,NChem+1,kkk,lll);
        X2(:,:,1,kkk+1,lll) = X2(:,sortind,NChem+1,kkk,lll);
        RLeng(:,kkk+1) = RLeng(sortind,kkk);          % Keeps the
chemotaxis parameters with each bacterium at the next generation
        % (6.2) Split the bacteria
        for spl=1:Sr % Sr??
                X1(:,spl+Sr,1,kkk+1,lll) = X1(:,spl,1,kkk+1,lll); % The
least fit do not reproduce, the most fit ones split into two identical
copies
                X2(:,spl+Sr,1,kkk+1,lll) = X2(:,spl,1,kkk+1,lll);
                RLeng(spl+Sr,kkk+1) = RLeng(spl,kkk+1);
        end
    end % (7) Loop to go to the next reproductive step
% (8) Elimination-dispersal
    for mmm = 1:NBac
        if  PEDi>rand % % Generate random number
            X1(:,mmm,1,1,lll+1) = randperm(Ra1,Nvar);
            X2(:,mmm,1,1,lll+1) = ((rand(1,Nvar)*(Ra2-Rb2))+Rb2)';
        else
            X1(:,mmm,1,1,lll+1)=X1(:,mmm,1,NRep+1,lll); % Bacteria that
are not dispersed
```

```
            X2(:,mmm,1,1,lll+1)=X2(:,mmm,1,NRep+1,lll);
        end
    end
end
reproduction = Fit(:,1:NChem,NRep,NEDi);
[bc,abc]=max(reproduction,[],2);
[~,man]=max(bc);
BestX2=X2(:,man,abc(man,:),kkk,lll);
BestX1=X1(:,man,abc(man,:),kkk,lll);

for i=1:Nvar
    if busdata(BestX1(i),2)==0
        busdata(BestX1(i),11) = BestX2(i);
    end
end
```

Lampiran 2 Listing Program Aliran Daya Metode *Newton Raphson* Pada Sistem JAMALI 500kV Tanpa STATCOM

```
clear
clc
format long
tic
basemva = 1000;  accuracy = 10^-4; accel = 1.8; maxiter = 100;
% Bus Data jamali 500kV
Bus  Bus     Voltage   Angle    Load          Generator
No  Code    Magnitude  Degree   MW   MVAR     MW    MVAR
1     1     1.020       0     147.86  77.44   2766  1144%suralaya
2     0     1.000       0     100.23 193.24    0      0%cilegon
3     0     1.000       0     122    144       0      0%kembangan
4     0     1.000       0     571.28 -31.2     0      0%Gandul
5     0     1.000       0     367.32 231.14    0      0%cibinong
6     0     1.000       0     523    108       0      0%cawang
7     0     1.000       0     740    -7        0      0%Bekasi
8     2     1.000       0      0     0        767 244.4%Muaratawar
9     0     1.000       0     947    410       0      0%cibatu
10    2     1.000       0     667.07 261.07   422   141%cirata
11    2     1.025       0     0      0      566.93 111.71%saguling
12    0     1.000       0     557    261       0     0%bandungsltn
13    0     1.000       0     67.96  66.79     0     0 %mandiracan
14    0     1.000       0     654    410       0     0 %ungaran
15    2     1.000       0     292    55      2230  201%tanjungjati
16    0     1.000       0     932.85 339.71    0     0%surabayabrt
17    2     1.000       0     102.3  118.77 515.16 157.34%gresik
18    0     1.000       0     503.99 108.38    0     0 %depok
19    0     1.000       0     285     92       0     0%tasikmalaya
20    0     1.000       0     674    231       0     0%pedan
21    0     1.000       0     615.4345 178.8   0     0%kediri
22    2     1.000       0     869      210  4038.1 715.060%Paiton
23    2     1.025       0     447.4    280.3 764   100.05%grati
24    0     1.000       0     574      164     0     0%ngimbang
25    0     1.000       0     577      184     0     0%balaraja
26    0     1.000       0     25.73    4.18    0   0];%ujungberung

% Line Data JAMALI 500 kV
Bus   Bus      R              X          1/2 B
nl    nr       pu             pu         pu          Tap Setting
1      2       0.000626496   0.007008768  0.0011414290     1
1      25      0.003677677   0.035333317  0.0002264160     1
2      5       0.013133324   0.146925792  0.0000544495     1
```

```
3      4       0.001513179   0.016928308  0.0004726210     1
4      18      0.000694176   0.006669298  0.0023990310     1
5      7       0.00444188    0.0426754    0.0001874620     1
5      8       0.0062116     0.059678     0.0001340530     1
5      11      0.00411138    0.04599504   0.0001739310     1
6      7       0.001973648   0.01896184   0.0004219000     1
6      8       0.0056256     0.054048     0.0001480170     1
8      9       0.002822059   0.027112954  0.0002950630     1
9      10      0.00273996    0.026324191  0.0003055450     1
10     11      0.001474728   0.014168458  0.0005646350     1
11     12      0.0019578     0.0219024    0.0003652570     1
12     13      0.00699098    0.0671659    0.0001191080     1
12     26      0.000385      0.003703                0     1
13     14      0.013478      0.12949      0.0000617810     1
13     26      0.000868      0.0097008               0     1
14     15      0.01353392    0.15140736   0.0001056750     1
14     16      0.01579856    0.1517848    0.0000279497     1
14     20      0.00903612    0.0868146    0.0000921505     1
16     17      0.00139468    0.0133994    0.0005945430     1
16     23      0.003986382   0.044596656  0.0001793860     1
18     5       0.000818994   0.00786848   0.0009126260     1
18     19      0.014056      0.15724802   0.0000508750     1
19     20      0.015311      0.0171288    0.0000467055     1
20     21      0.010291      0.0115128    0.0000694880     1
21     22      0.010291      0.0115128    0.0000694880     1
22     23      0.004435823   0.049624661  0.0001695470     1
24     14      0.023479613   0.225580588  0.0000354641     1     24
16      0.005966652   0.057324466  0.0001393620     1
25     4       0.002979224   0.02862292   0.0002794970     1];
%  This program obtains th Bus Admittance Matrix for power flow solution
%  Copyright (c) 1998 by H. Saadat
j=sqrt(-1); i = sqrt(-1);
nl = linedata(:,1); nr = linedata(:,2); R = linedata(:,3);
X = linedata(:,4); Bc = j*linedata(:,5); a = linedata(:, 6);
nbr=length(linedata(:,1)); nbus = max(max(nl), max(nr));
Z = R + j*X; y= ones(nbr,1)./Z;         %branch admittance
for n = 1:nbr
if a(n) <= 0  a(n) = 1; else end
Ybus=zeros(nbus,nbus);     % initialize Ybus to zero
               % formation of the off diagonal elements
for k=1:nbr;
       Ybus(nl(k),nr(k))=Ybus(nl(k),nr(k))-y(k)/a(k);
       Ybus(nr(k),nl(k))=Ybus(nl(k),nr(k));
    end
end
              % formation of the diagonal elements
for  n=1:nbus
     for k=1:nbr
         if nl(k)==n
         Ybus(n,n) = Ybus(n,n)+y(k)/(a(k)^2) + Bc(k);
         elseif nr(k)==n
         Ybus(n,n) = Ybus(n,n)+y(k) +Bc(k);
         else, end
     end
end
clear Pgg
%   Power flow solution by Newton-Raphson method
%   Copyright (c) 1998 by  H. Saadat
ns=0; ng=0; Vm=0; delta=0; yload=0; deltad=0;
nbus = length(busdata(:,1));
for k=1:nbus
```

```
n=busdata(k,1);
kb(n)=busdata(k,2); Vm(n)=busdata(k,3); delta(n)=busdata(k, 4);
Pd(n)=busdata(k,5); Qd(n)=busdata(k,6); Pg(n)=busdata(k,7); Qg(n) =
busdata(k,8);
Qmin(n)=busdata(k, 9); Qmax(n)=busdata(k, 10);
Qsh(n)=busdata(k, 11);
    if Vm(n) <= 0  Vm(n) = 1.0; V(n) = 1 + j*0;
    else delta(n) = pi/180*delta(n);
         V(n) = Vm(n)*(cos(delta(n)) + j*sin(delta(n)));
         P(n)=(Pg(n)-Pd(n))/basemva;
         Q(n)=(Qg(n)-Qd(n)+ Qsh(n))/basemva;
         S(n) = P(n) + j*Q(n);
    end
end
for k=1:nbus
if kb(k) == 1, ns = ns+1; else, end
if kb(k) == 2 ng = ng+1; else, end
ngs(k) = ng;
nss(k) = ns;
end
Ym=abs(Ybus); t = angle(Ybus);
m=2*nbus-ng-2*ns;
maxerror = 1; converge=1;
iter = 0;
% Start of iterations
clear A  DC   J  DX
while maxerror >= accuracy & iter <= maxiter % Test for max. power
mismatch
for i=1:m
for k=1:m
   A(i,k)=0;      %Initializing Jacobian matrix
end, end
iter = iter+1;
for n=1:nbus
nn=n-nss(n);
lm=nbus+n-ngs(n)-nss(n)-ns;
J11=0; J22=0; J33=0; J44=0;
   for i=1:nbr
     if nl(i) == n | nr(i) == n
        if nl(i) == n,  l = nr(i); end
        if nr(i) == n,  l = nl(i); end
        J11=J11+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
        J33=J33+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
        if kb(n)~=1
        J22=J22+ Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
        J44=J44+ Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
        else, end
        if kb(n) ~= 1  & kb(l) ~=1
        lk = nbus+l-ngs(l)-nss(l)-ns;
        ll = l -nss(l);
      % off diagonalelements of J1
        A(nn, ll) =-Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
              if kb(l) == 0  % off diagonal elements of J2
              A(nn, lk) =Vm(n)*Ym(n,l)*cos(t(n,l)- delta(n) +
delta(l));end
              if kb(n) == 0  % off diagonal elements of J3
              A(lm, ll) =-Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)-
delta(n)+delta(l)); end
              if kb(n) == 0 & kb(l) == 0  % off diagonal elements of  J4
              A(lm, lk) =-Vm(n)*Ym(n,l)*sin(t(n,l)- delta(n) +
delta(l));end
```

```
        else end
      else , end
    end
    Pk = Vm(n)^2*Ym(n,n)*cos(t(n,n))+J33;
    Qk = -Vm(n)^2*Ym(n,n)*sin(t(n,n))-J11;
    if kb(n) == 1 P(n)=Pk; Q(n) = Qk; end    % Swing bus P
      if kb(n) == 2  Q(n)=Qk;
          if Qmax(n) ~= 0
            Qgc = Q(n)*basemva + Qd(n) - Qsh(n);
            if iter <= 7                    % Between the 2th & 6th
iterations
               if iter > 2                  % the Mvar of generator buses
are
                 if Qgc  < Qmin(n),         % tested. If not within limits
Vm(n)
                 Vm(n) = Vm(n) + 0.01;      % is changed in steps of 0.01 pu
to
                 elseif Qgc  > Qmax(n),     % bring the generator Mvar
within
                 Vm(n) = Vm(n) - 0.01;end % the specified limits.
               else, end
            else,end
          else,end
      end
    if kb(n) ~= 1
      A(nn,nn) = J11;  %diagonal elements of J1
      DC(nn) = P(n)-Pk;
    end
    if kb(n) == 0
      A(nn,lm) = 2*Vm(n)*Ym(n,n)*cos(t(n,n))+J22;  %diagonal elements of
J2
      A(lm,nn)= J33;         %diagonal elements of J3
      A(lm,lm) =-2*Vm(n)*Ym(n,n)*sin(t(n,n))-J44;  %diagonal of elements
of J4
      DC(lm) = Q(n)-Qk;
    end
end
DX=A\DC';
for n=1:nbus
  nn=n-nss(n);
  lm=nbus+n-ngs(n)-nss(n)-ns;
    if kb(n) ~= 1
    delta(n) = delta(n)+DX(nn); end
    if kb(n) == 0
    Vm(n)=Vm(n)+DX(lm); end
 end
  maxerror=max(abs(DC));
     if iter == maxiter & maxerror > accuracy
   fprintf('\nWARNING: Iterative solution did not converged after ')
   fprintf('%g', iter), fprintf(' iterations.\n\n')
   fprintf('Press Enter to terminate the iterations and print the results
\n')
   converge = 0; pause, else, end

end

if converge ~= 1
   tech= ('                      ITERATIVE SOLUTION DID NOT CONVERGE');
else,
   tech=('                   Power Flow Solution by Newton-Raphson
Method');
```

```
end
V = Vm.*cos(delta)+j*Vm.*sin(delta);
deltad=180/pi*delta;
i=sqrt(-1);
k=0;
for n = 1:nbus
     if kb(n) == 1
     k=k+1;
     S(n)= P(n)+j*Q(n);
     Pg(n) = P(n)*basemva + Pd(n);
     Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
     Pgg(k)=Pg(n);
     Qgg(k)=Qg(n);      %june 97
     elseif  kb(n) ==2
     k=k+1;
     S(n)=P(n)+j*Q(n);
     Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
     Pgg(k)=Pg(n);
     Qgg(k)=Qg(n);  % June 1997
  end
yload(n) = (Pd(n)- j*Qd(n)+j*Qsh(n))/(basemva*Vm(n)^2);
end
busdata(:,3)=Vm'; busdata(:,4)=deltad';
Pgt = sum(Pg);  Qgt = sum(Qg); Pdt = sum(Pd); Qdt = sum(Qd); Qsht =
sum(Qsh);

%clear A DC DX  J11 J22 J33 J44 Qk delta lk ll lm
%clear A DC DX  J11 J22 J33  Qk delta lk ll lm
%  This program prints the power flow solution in a tabulated form
%  on the screen.
%
%  Copyright (C) 1998 by H. Saadat.

%clc
disp(tech)
fprintf('                       Maximum Power Mismatch = %g \n', maxerror)
fprintf('                             No. of Iterations = %g \n\n', iter)
head =['    Bus  Voltage  Angle    ------Load------    ---Generation---
Injected'
       '    No.  Mag.     Degree     MW       Mvar       MW       Mvar
Mvar '
       '
'];
disp(head)
for n=1:nbus
     fprintf(' %5g', n), fprintf(' %7.3f', Vm(n)),
     fprintf(' %8.3f', deltad(n)), fprintf(' %9.3f', Pd(n)),
     fprintf(' %9.3f', Qd(n)),  fprintf(' %9.3f', Pg(n)),
     fprintf(' %9.3f ', Qg(n)), fprintf(' %8.3f\n', Qsh(n))
end
    fprintf('      \n'), fprintf('    Total              ')
    fprintf(' %9.3f', Pdt), fprintf(' %9.3f', Qdt),
    fprintf(' %9.3f', Pgt), fprintf(' %9.3f', Qgt), fprintf(' %9.3f\n\n',
Qsht)
%  This program is used in conjunction with lfgauss or lf Newton
%  for the computation of line flow and line losses.
%
%  Copyright (c) 1998 H. Saadat
SLT = 0;
fprintf('\n')
fprintf('                            Line Flow and Losses \n\n')
```

```
fprintf('     --Line--  Power at bus & line flow    --Line loss--
Transformer\n')
fprintf('     from  to    MW      Mvar     MVA       MW      Mvar
tap\n')

for n = 1:nbus
busprt = 0;
   for L = 1:nbr;
       if busprt == 0
       fprintf('   \n'), fprintf('%6g', n), fprintf('      %9.3f',
P(n)*basemva)
       fprintf('%9.3f', Q(n)*basemva), fprintf('%9.3f\n',
abs(S(n)*basemva))

       busprt = 1;
       else, end
       if nl(L)==n      k = nr(L);
       In = (V(n) - a(L)*V(k))*y(L)/a(L)^2 + Bc(L)/a(L)^2*V(n);
       Ik = (V(k) - V(n)/a(L))*y(L) + Bc(L)*V(k);
       Snk = V(n)*conj(In)*basemva;
       Skn = V(k)*conj(Ik)*basemva;
       SL  = Snk + Skn;
       SLT = SLT + SL;
       elseif nr(L)==n  k = nl(L);
       In = (V(n) - V(k)/a(L))*y(L) + Bc(L)*V(n);
       Ik = (V(k) - a(L)*V(n))*y(L)/a(L)^2 + Bc(L)/a(L)^2*V(k);
       Snk = V(n)*conj(In)*basemva;
       Skn = V(k)*conj(Ik)*basemva;
       SL  = Snk + Skn;
       SLT = SLT + SL;
       else, end
         if nl(L)==n | nr(L)==n
         fprintf('%12g', k),
         fprintf('%9.3f', real(Snk)), fprintf('%9.3f', imag(Snk))
         fprintf('%9.3f', abs(Snk)),
         fprintf('%9.3f', real(SL)),
             if nl(L) ==n & a(L) ~= 1
             fprintf('%9.3f', imag(SL)), fprintf('%9.3f\n', a(L))
             else, fprintf('%9.3f\n', imag(SL))
             end
         else, end
  end
end
SLT = SLT/2;
fprintf('   \n'), fprintf('    Total loss                          ')
fprintf('%9.3f', real(SLT)), fprintf('%9.3f\n', imag(SLT))
clear Ik In SL SLT Skn Snk
```

Lampiran 3 Listing Program Aliran Daya Pada Sistem JAMALI 500kV Setelah Penempatan STATCOM Dengan Optimasi BFO

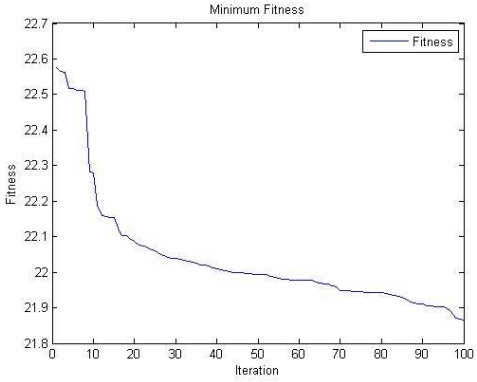
```
clear;
clc;
format long
tic
basemva = 1000;  accuracy = 10^-4; accel = 1.8; maxiter = 1000;
% Bus Data jamali 500kV
% Line Data JAMALI 500 kV
```

```
%% Bacterial Foraging Optimization
Nvar =3;       % Jumlah STATCOM
NBac = 20;     % Number of bacteria in the colony
NChem = 10;    % Number of chemotactic steps
NSwim = 4;     % Number of swim steps
NRep = 10;      % Number of reproductive steps
NEDi = 10;      % Number of elimination and dispersal steps
Sr = NBac/2;  % The number of bacteria reproductions (splits) per
generation
PEDi = 0.5;      % The probability that each bacteria will be
eliminated/dispersed
RLeng(:,1)=25*ones(NBac,1);   % the run length unit (the size of the step
taken in each run or tumble)
Rb1 = 1;        % Batas bawah interval Lokasi
Ra1 = max(busdata(:,1));       % Batas akhir interval Lokasi
Rb2 =-450;     % Batas bawah interval MVAR
Ra2 =600;      % Batas atas interval MVAR

% the inititial posistions
for nn = 1 : NBac
    X1(:,nn,1,1,1) = randperm(Ra1,Nvar);
    X2(:,:,1,1,1) = ((rand(NBac,Nvar)*(Ra2-Rb2))+Rb2)';
end

for lll = 1:NEDi
    % (3) Reproduction loop
    for kkk = 1:NRep
        % (4) Chemotaxis (swim/tumble) loop
        for jjj = 1:NChem
            % (4.1) Chemotatic step
            for iii = 1:NBac
            % (4.2) Fitness function
            Fit(iii,jjj,kkk,lll) =
EvaluasiIndividu(busdata,linedata,X1(:,iii,jjj,kkk,lll),X2(:,iii,jjj,kkk,
lll),Nvar);
            % (4.3) Jlast
            Fitlast=Fit(iii,jjj,kkk,lll);
            % (4.4) Tumble
            Delta(:,iii) = unifrnd(-1,1,Nvar,1);
            % (4.5) Move
            X1(:,iii,jjj+1,kkk,lll)= randperm(Ra1,Nvar);
            X2(:,iii,jjj+1,kkk,lll)=
X2(:,iii,jjj,kkk,lll)+RLeng(iii,kkk)*Delta(:,iii)/sqrt(Delta(:,iii)'*Delt
a(:,iii));
            for chck=1:Nvar
                if X2(chck,iii,jjj+1,kkk,lll) > Ra2
                   X2(chck,iii,jjj+1,kkk,lll) = Ra2;
                elseif X2(chck,iii,jjj+1,kkk,lll) < Rb2
                   X2(chck,iii,jjj+1,kkk,lll) = Rb2;
                end
            end
            % (4.6) New fitness function
            Fit(iii,jjj+1,kkk,lll) =
EvaluasiIndividu(busdata,linedata,X1(:,iii,jjj+1,kkk,lll),X2(:,iii,jjj+1,
kkk,lll),Nvar);
            % (4.7) Swimming
            m=0; % counter for swim length
            while m < NSwim
                m = m + 1;
                 if Fit(iii,jjj+1,kkk,lll)> Fitlast
                    Fitlast = Fit(iii,jjj+1,kkk,lll);
```

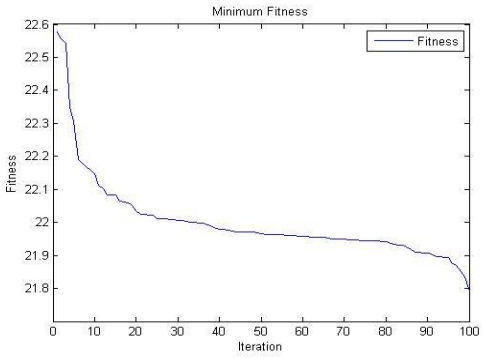
```
                    X1(:,iii,jjj+1,kkk,lll) = randperm(Ra1,Nvar);
                    X2(:,iii,jjj+1,kkk,lll) =
X2(:,iii,jjj+1,kkk,lll)+RLeng(iii,kkk)*Delta(:,iii)/sqrt(Delta(:,iii)'*De
lta(:,iii));
                    for chck=1:Nvar
                        if  X2(chck,iii,jjj+1,kkk,lll) > Ra2
                            X2(chck,iii,jjj+1,kkk,lll) = Ra2;
                        elseif X2(chck,iii,jjj+1,kkk,lll) < Rb2
                            X2(chck,iii,jjj+1,kkk,lll) = Rb2;
                        end
                    end
                    Fit(iii,jjj+1,kkk,lll) =
EvaluasiIndividu(busdata,linedata,X1(:,iii,jjj+1,kkk,lll),X2(:,iii,jjj+1,
kkk,lll),Nvar);
                 else
                    m = NSwim;
                 end
            end
            Fit(iii,jjj,kkk,lll) = Fitlast;
            end % (4.8) Next bacterium
        end     % (5) if j < Nc, chemotaxis
        % (6) Reproduction
        % (6.1) Health
        Fithealth = sum(Fit(:,:,kkk,lll),2);      % Set the health of
each of the S bacteria
        [Fithealth,sortind] = sort(Fithealth,'descend');% Sorts bacteria
in order of descending values
        X1(:,:,1,kkk+1,lll) = X1(:,sortind,NChem+1,kkk,lll);
        X2(:,:,1,kkk+1,lll) = X2(:,sortind,NChem+1,kkk,lll);
        RLeng(:,kkk+1) = RLeng(sortind,kkk);          % Keeps the
chemotaxis parameters with each bacterium at the next generation
        % (6.2) Split the bacteria
        for spl=1:Sr % Sr??
                X1(:,spl+Sr,1,kkk+1,lll) = X1(:,spl,1,kkk+1,lll); % The
least fit do not reproduce, the most fit ones split into two identical
copies
                X2(:,spl+Sr,1,kkk+1,lll) = X2(:,spl,1,kkk+1,lll);
                RLeng(spl+Sr,kkk+1) = RLeng(spl,kkk+1);
        end
    end % (7) Loop to go to the next reproductive step
% (8) Elimination-dispersal
    for mmm = 1:NBac
        if  PEDi>rand % % Generate random number
            X1(:,mmm,1,1,lll+1) = randperm(Ra1,Nvar);
            X2(:,mmm,1,1,lll+1) = ((rand(1,Nvar)*(Ra2-Rb2))+Rb2)';
        else
            X1(:,mmm,1,1,lll+1)=X1(:,mmm,1,NRep+1,lll); % Bacteria that
are not dispersed
            X2(:,mmm,1,1,lll+1)=X2(:,mmm,1,NRep+1,lll);
        end
    end
en
```

Lampiran 4 Hasil Pengujian

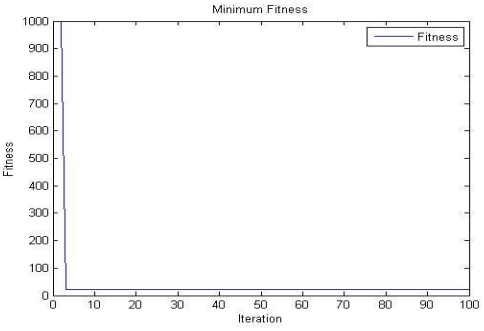
Data IEEE 30 Bus dengan Kapasitor Bank



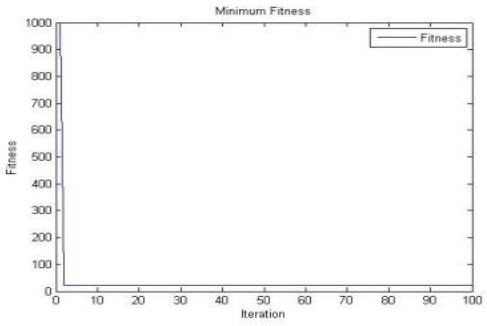
(1)



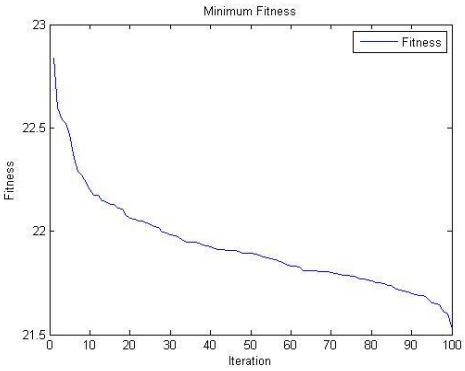
(2)



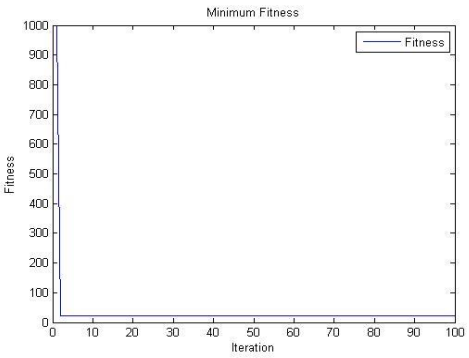
(3)



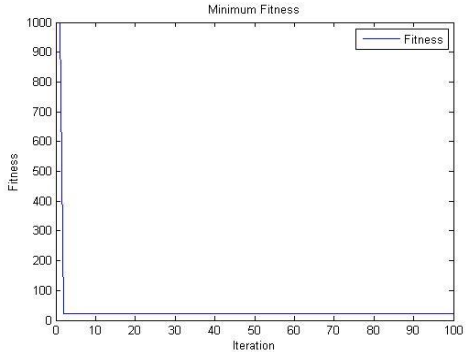
(4)



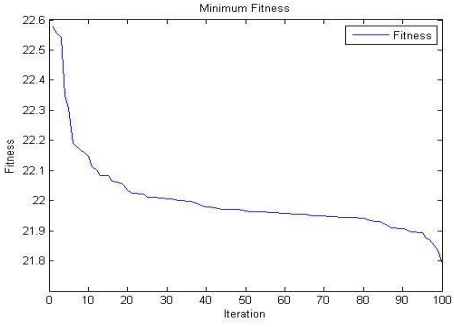
(5)



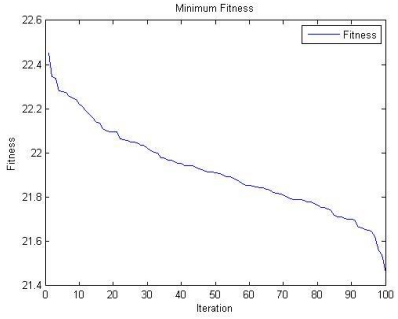
(6)



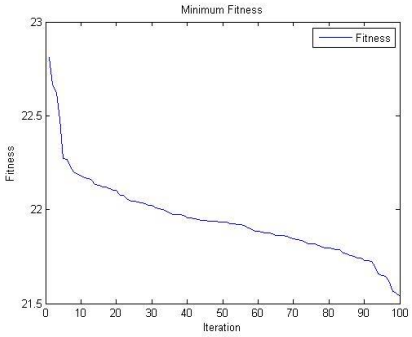
(7)



(8)

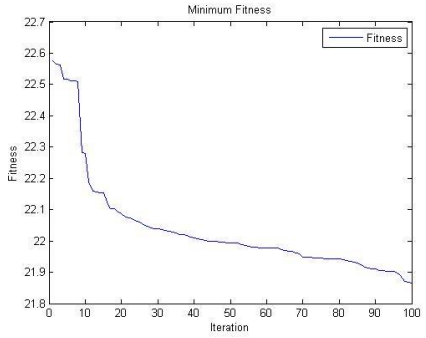


(9)

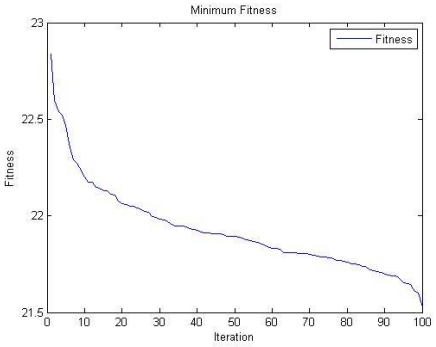


(10)

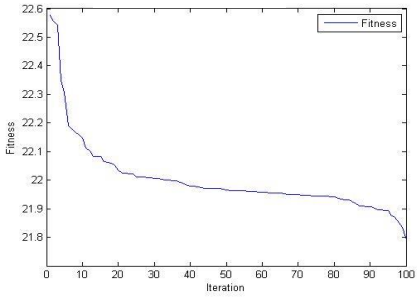
Data IEEE 30 Bus dengan STATCOM



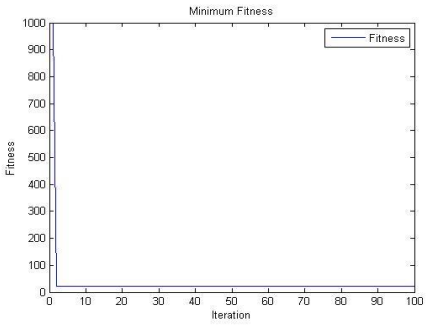
(1)



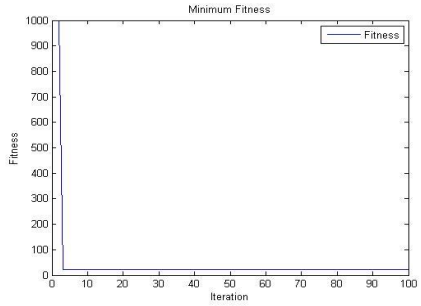
(4)



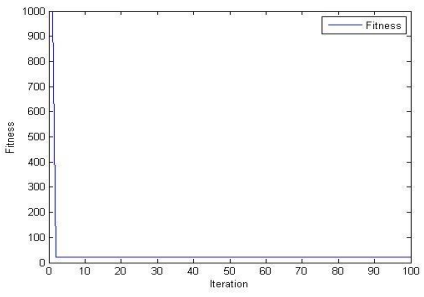
(2)



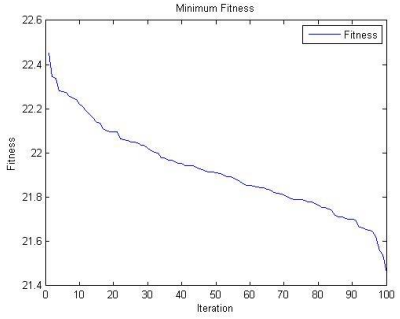
(5)



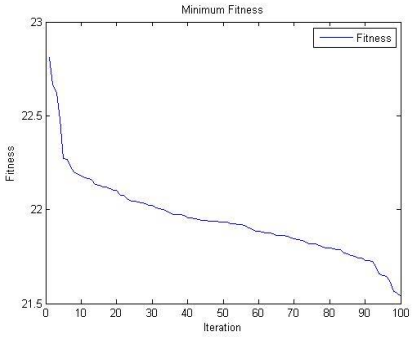
(3)



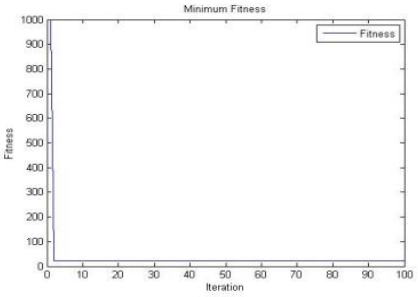
(6)



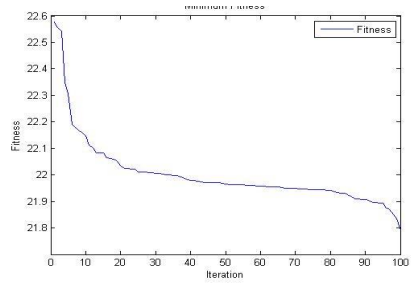
(7)



(9)

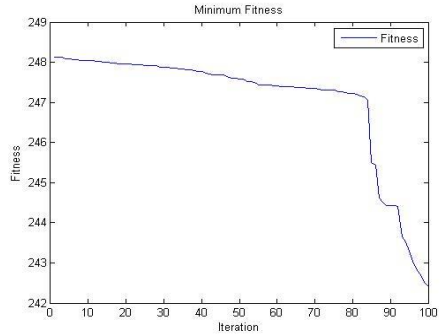


(8)

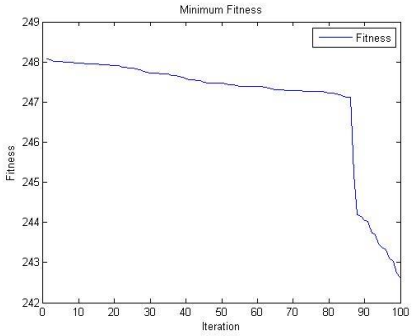


(10)

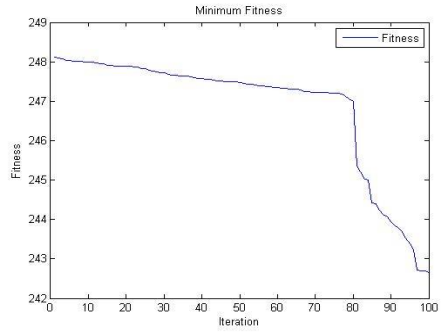
Data JAMALI 500 kV Bus dengan kapasitor bank



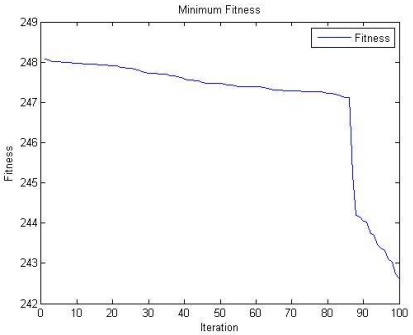
(1)



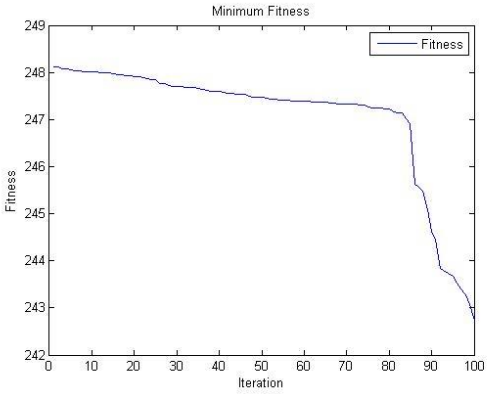
(3)



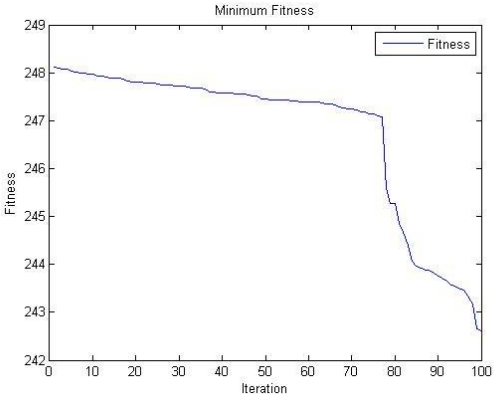
(2)



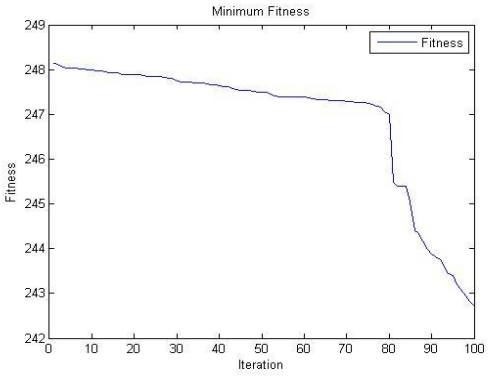
(4)



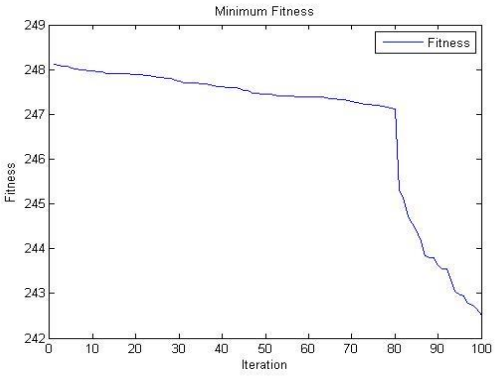
(5)



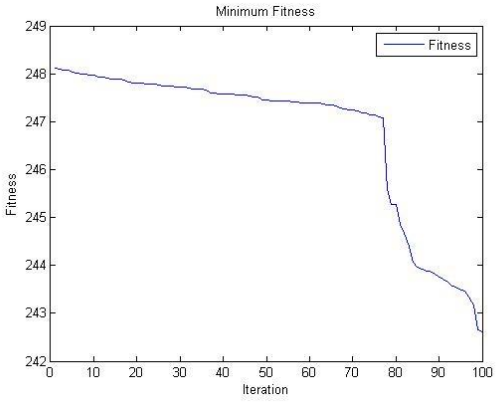
(6)



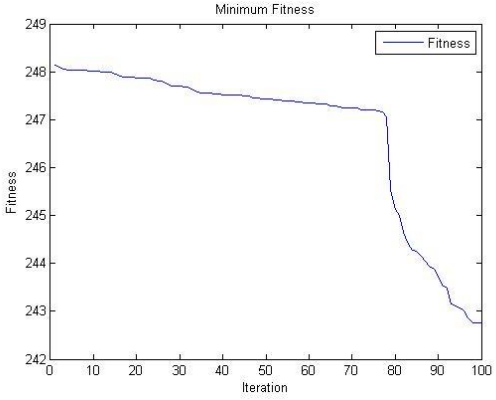
(7)



(8)

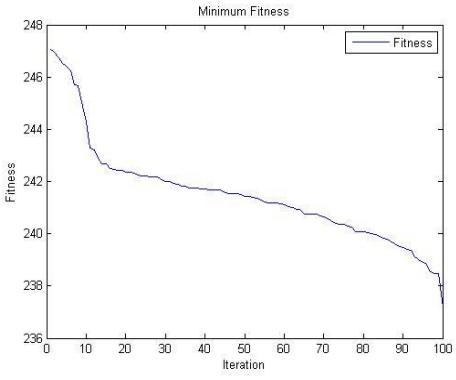


(9)

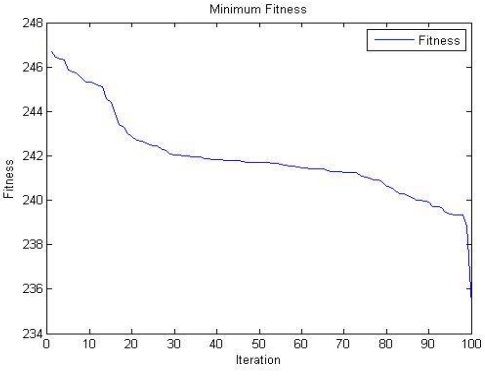


(10)

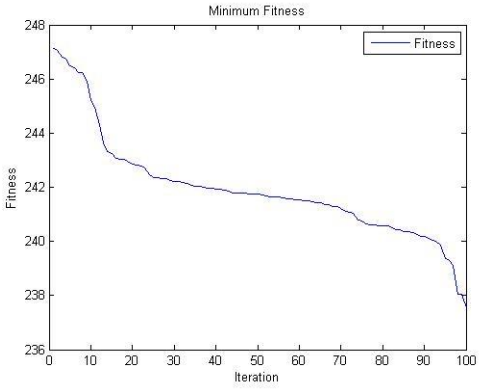
Data JAMALI 500 kV Bus dengan STATCOM



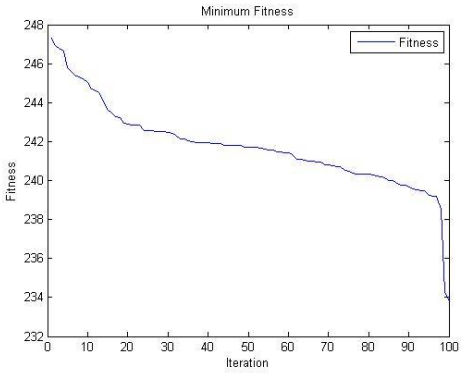
(1)



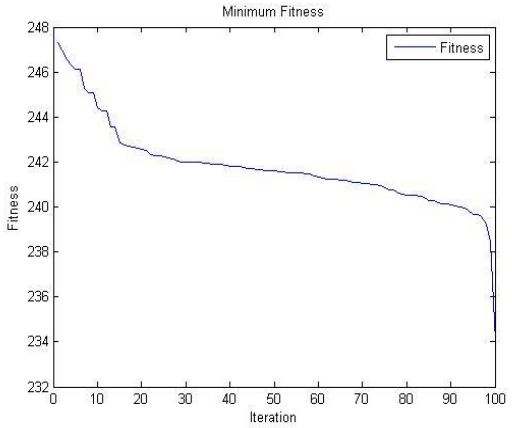
(2)


Minimum Fitness
Fitness
Fitness
Iteration

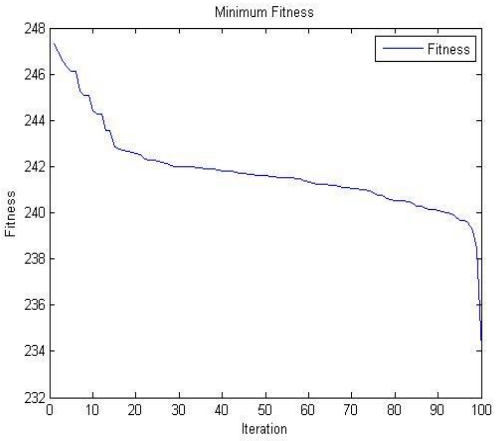
(3)


Minimum Fitness
Fitness
Fitness
Iteration

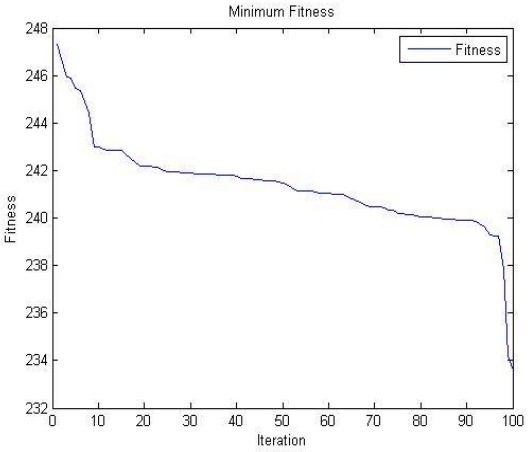
(4)


Minimum Fitness
Fitness
Fitness
Iteration

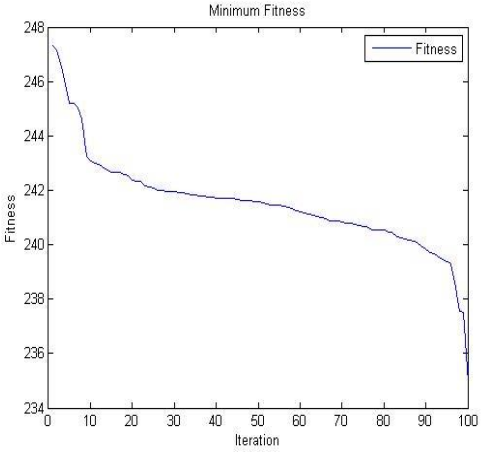
(5)


Minimum Fitness
Fitness
Fitness
Iteration

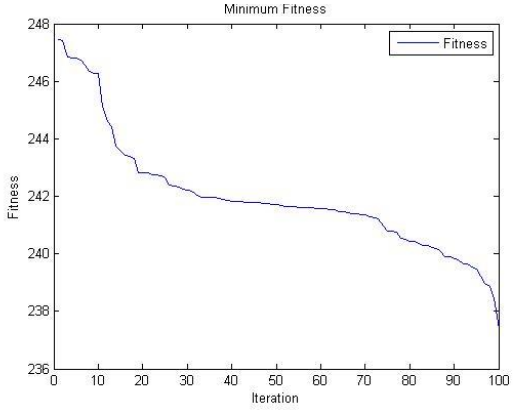
(6)


Minimum Fitness
Fitness
Fitness
Iteration

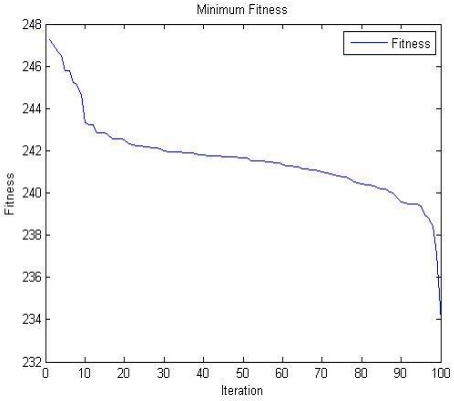
(7)


Minimum Fitness
Fitness
Fitness
Iteration

(8)


Minimum Fitness
Fitness
Fitness
Iteration

(9)


Minimum Fitness
Fitness
Fitness
Iteration

(10)