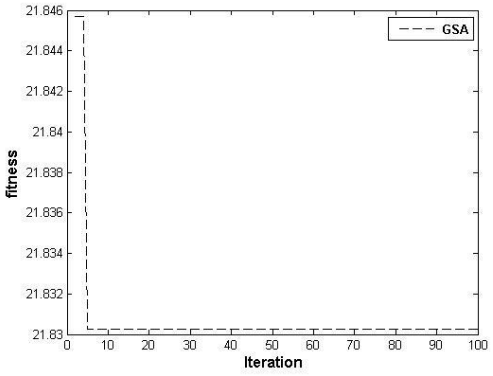
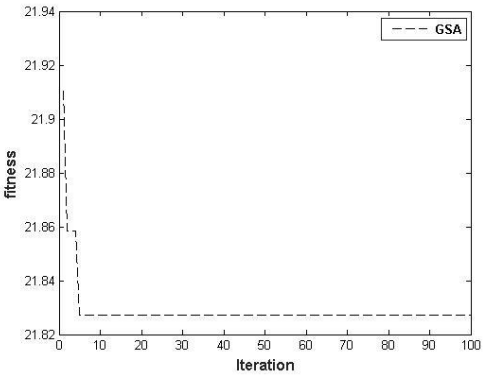
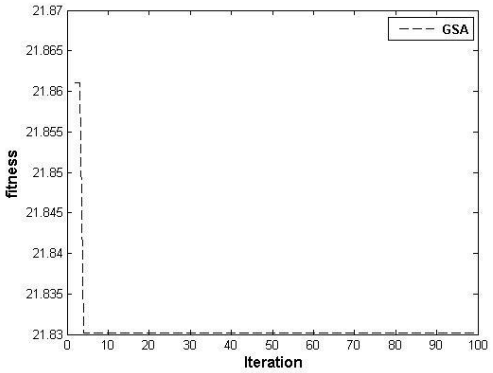
Lampiran 1 Grafik Konvergensi Simulasi Optimasi Lokasi dan *Rating* Kapasitor Bank pada Sistem Standar IEEE 30 *Bus* Menggunakan GSA



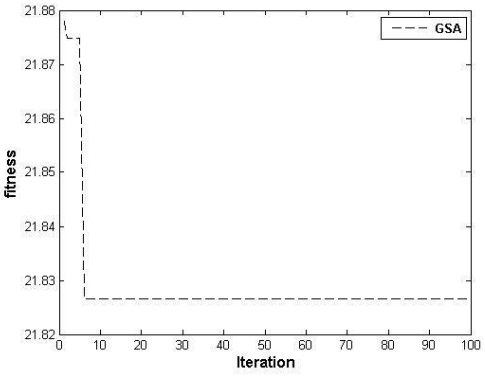
1



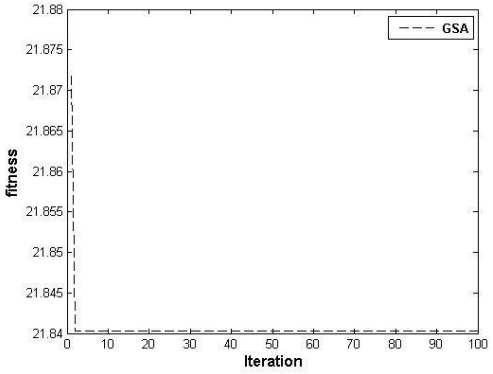
2



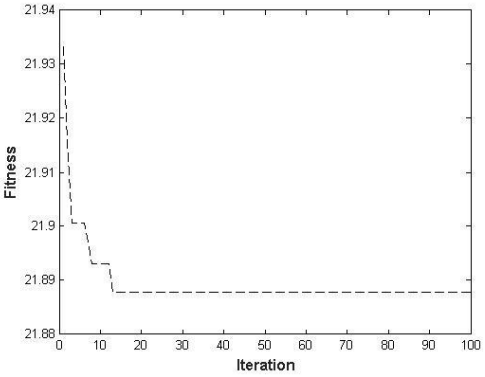
3



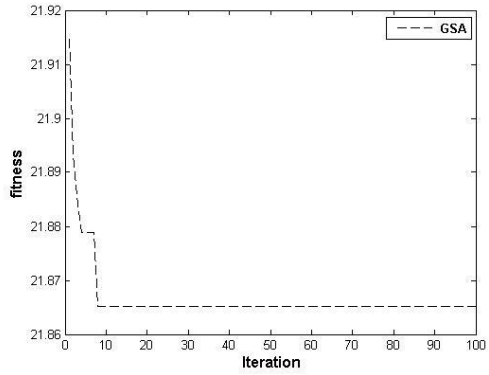
4



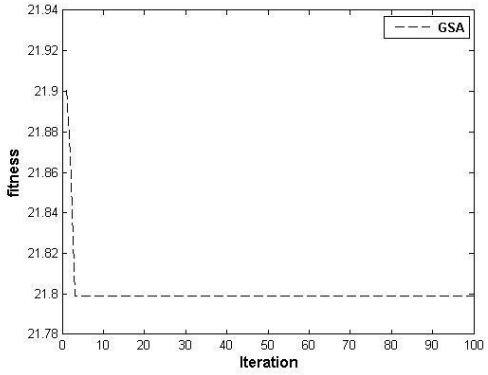
5



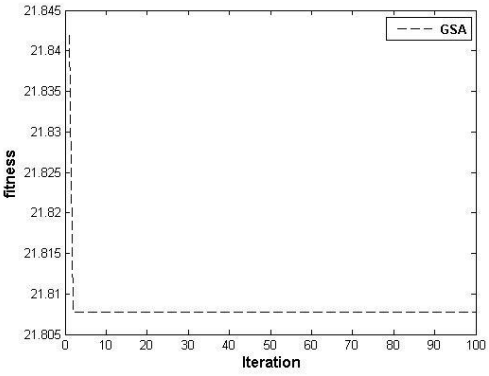
6



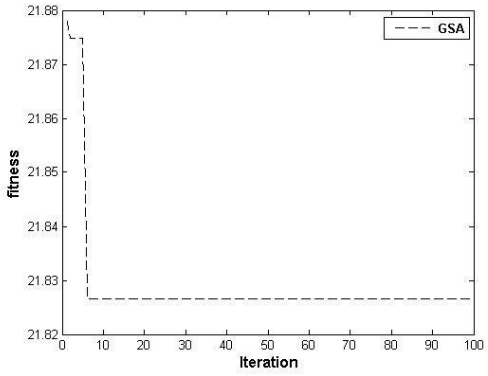
7



8

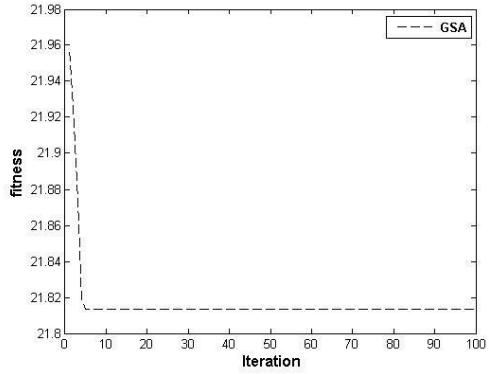


9

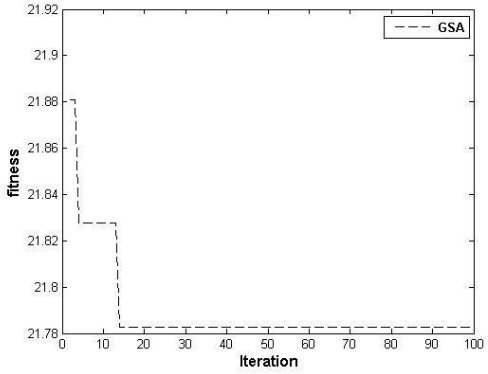


10

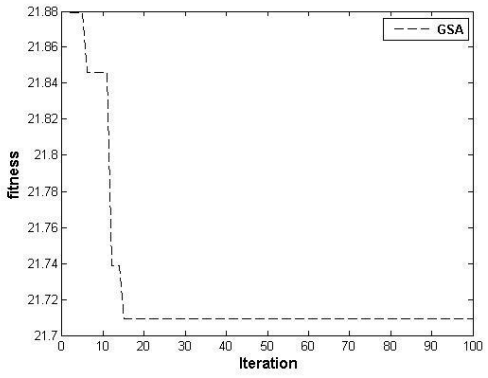
Lampiran 2 Grafik Konvergensi Simulasi Optimasi Lokasi dan *Rating* SSSC pada Sistem Standar IEEE 30 *Bus* Menggunakan GSA



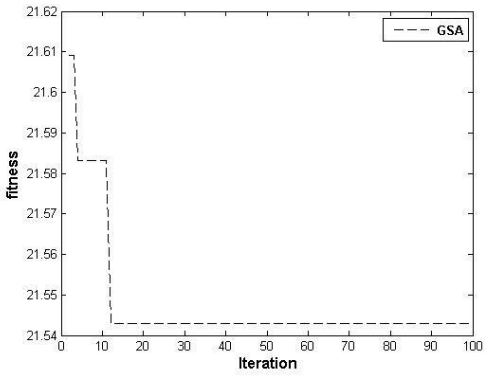
1



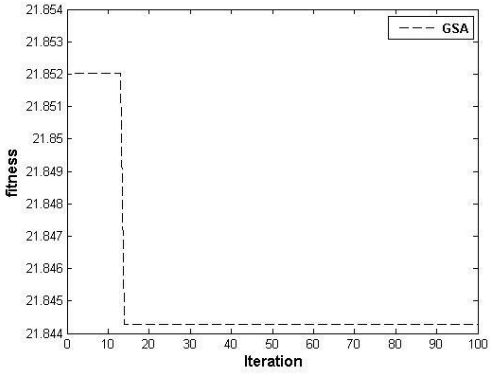
2


GSA
fitness
Iteration
21.88
21.86
21.84
21.82
21.8
21.78
21.76
21.74
21.72
21.7
0 10 20 30 40 50 60 70 80 90 100

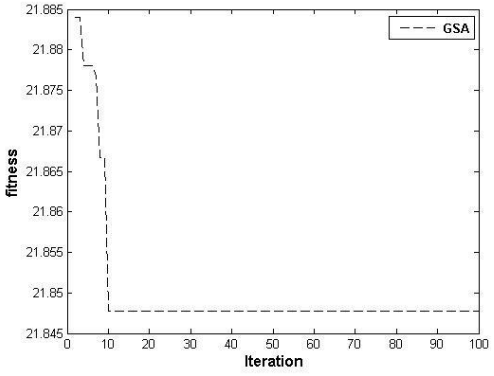
3


GSA
fitness
Iteration
21.62
21.61
21.6
21.59
21.58
21.57
21.56
21.55
21.54
0 10 20 30 40 50 60 70 80 90 100

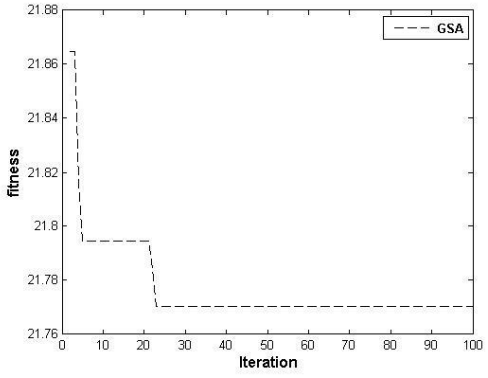
4


GSA
fitness
Iteration
21.854
21.853
21.852
21.851
21.85
21.849
21.848
21.847
21.846
21.845
21.844
0 10 20 30 40 50 60 70 80 90 100

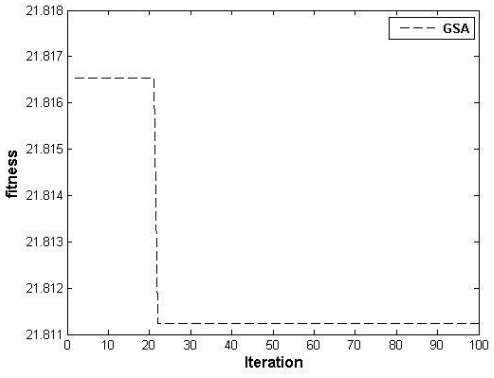
5


GSA
fitness
Iteration
21.885
21.88
21.875
21.87
21.865
21.86
21.855
21.85
21.845
0 10 20 30 40 50 60 70 80 90 100

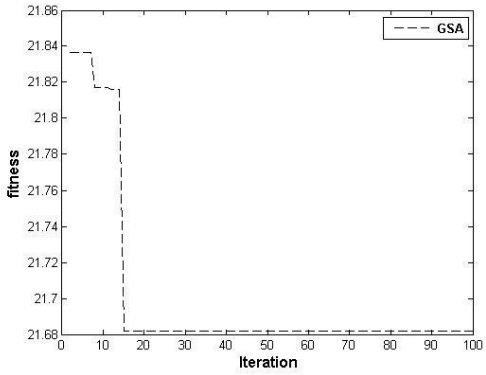
6


GSA
fitness
Iteration
21.88
21.86
21.84
21.82
21.8
21.78
21.76
0 10 20 30 40 50 60 70 80 90 100

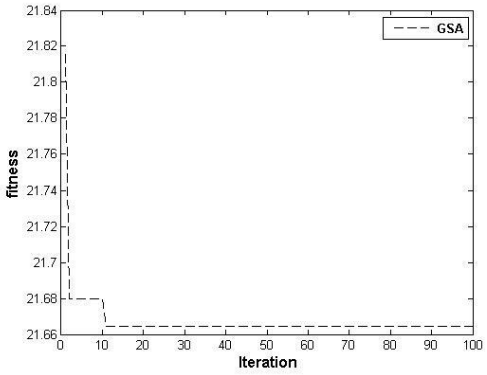
7


GSA
fitness
Iteration
21.818
21.817
21.816
21.815
21.814
21.813
21.812
21.811
0 10 20 30 40 50 60 70 80 90 100

8

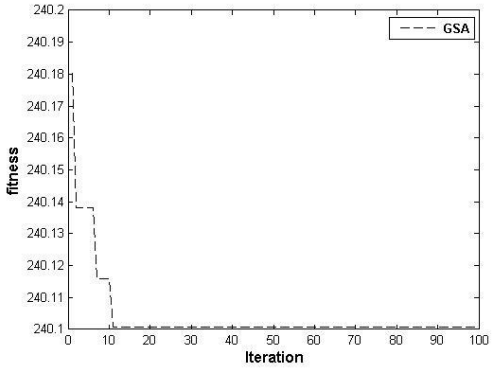


9

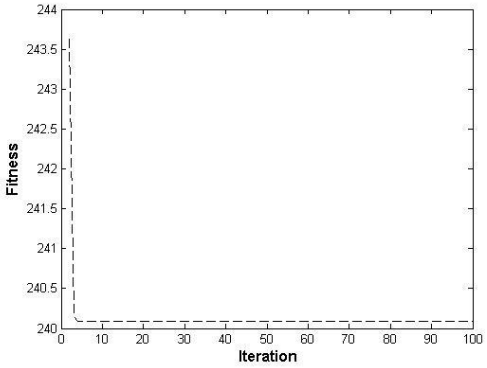


10

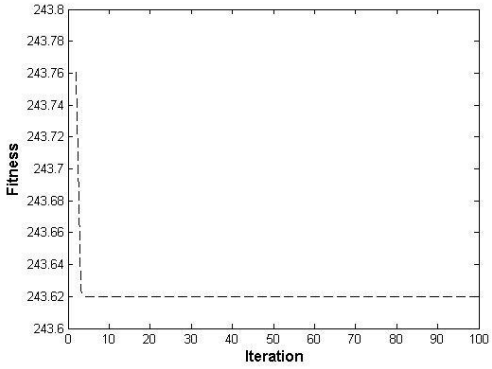
Lampiran 3 Grafik Konvergensi Simulasi Optimasi Lokasi dan *Rating* Kapasitor Bank pada Sistem Transmisi Jawa Bali 500 kV Menggunakan GSA



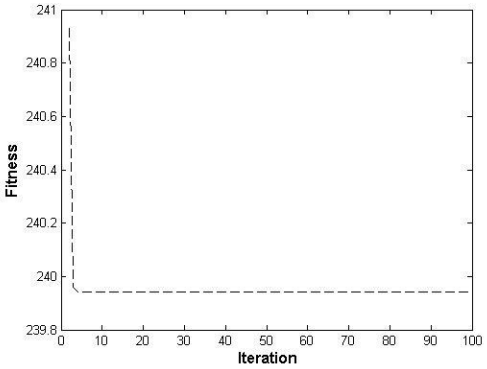
1



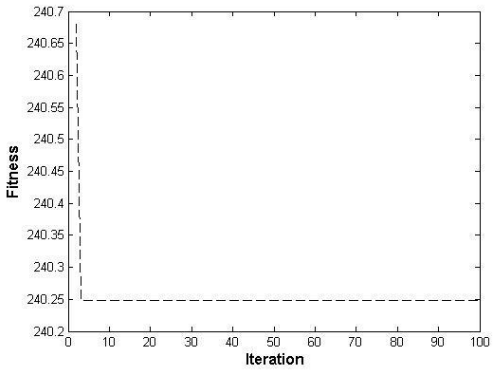
2



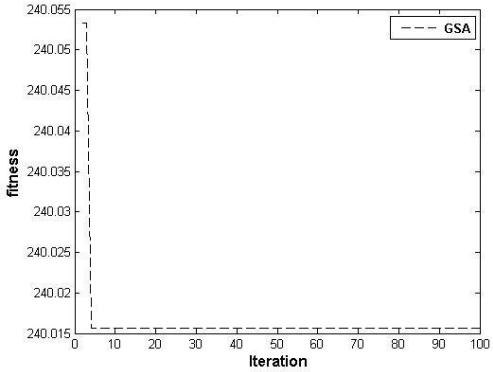
3



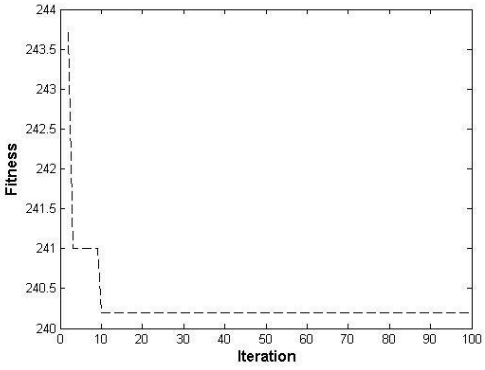
4


Fitness
Iteration

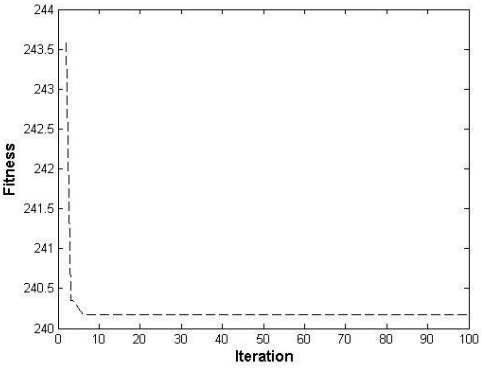
5


GSA
fitness
Iteration

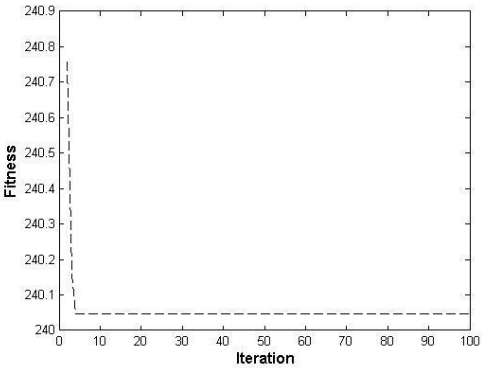
6


Fitness
Iteration

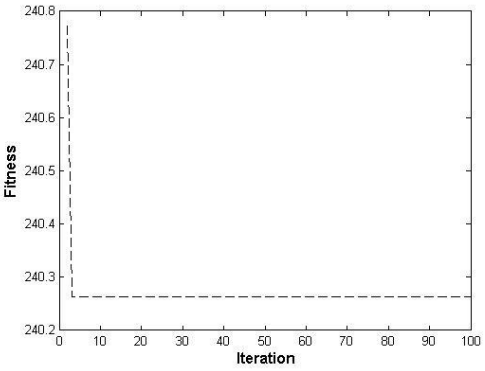
7


Fitness
Iteration

8

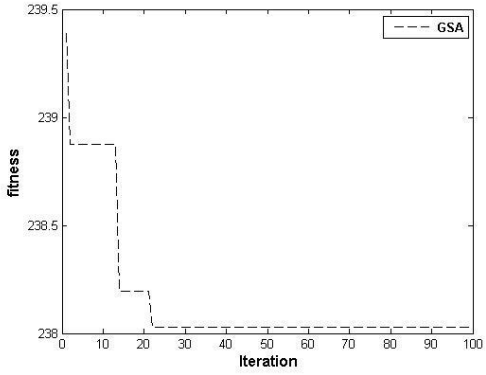

Fitness
Iteration

9

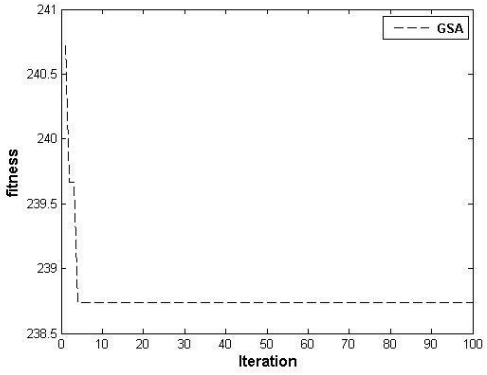

Fitness
Iteration

10

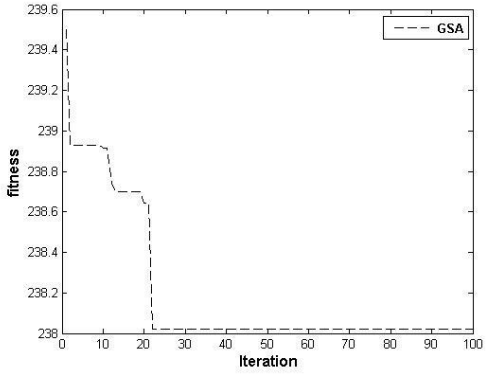
Lampiran 4 Grafik Konvergensi Simulasi Optimasi Lokasi dan *Rating* SSSC pada Sistem Transmisi Jawa Bali 500 kV Menggunakan GSA



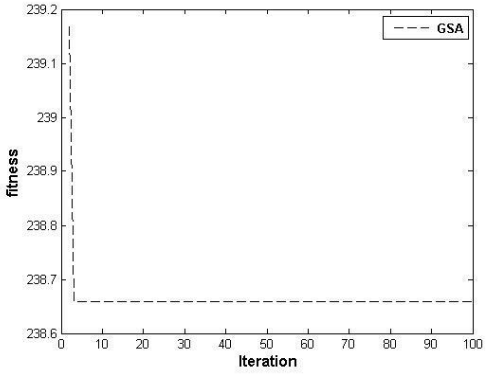
1



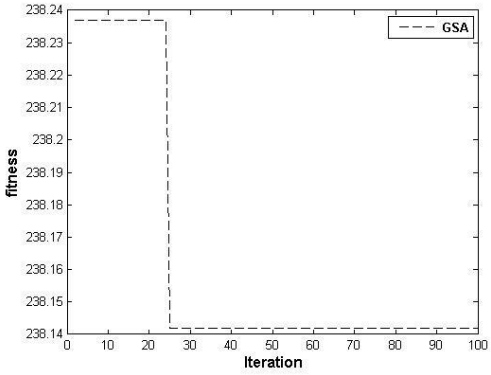
2



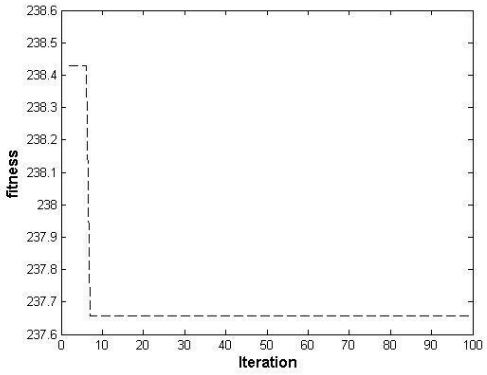
3



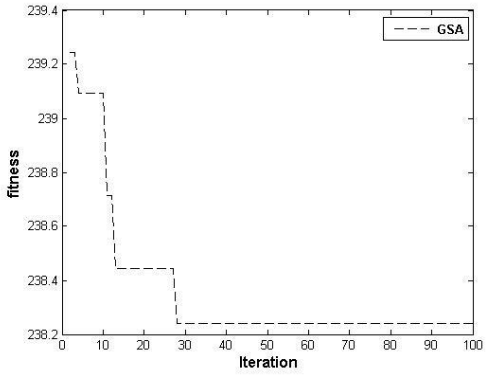
4



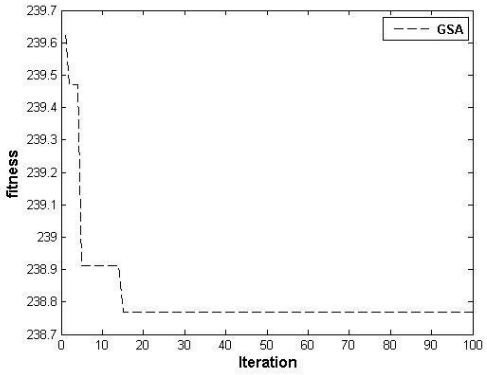
5



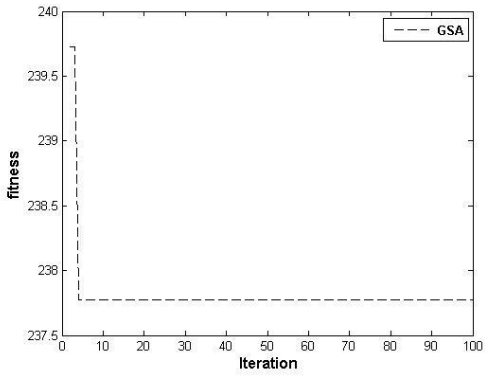
6



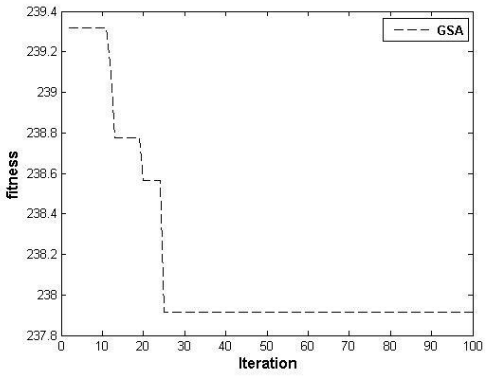
7



8



9



10

Lampiran 5 *Listing* Program Optimasi SSSC Dengan Menggunakan GSA

```
%% Parameter loadflow
current;
%currentieee
%% Parameter SSSC
Rb1 = -0.15;                          %Batas bawah V SSSC
Ra1 = 0.15;                           %Batas atas V SSSC
Rb2 = 1;
Ra2 = max(size(linedata(:,1)));        %Kandidat lokasi
Rb3 = 90*pi/180;                       %Sudut


%% Parameter GSA
N = 100;
dim = 3;
Rpower = 1;
Rnorm = 2;
alfa = 20;
G0 = 100;
ElitistCheck = 1;
final_per = 2;%In the last iteration, only 2 percent of agents apply
force to the others.
max_it= 100;
%% Inisiasi
    x1 = rand(N,dim).*(Ra1-Rb1)+Rb1;    %Vq
    x2 = round(rand(N,dim).*(Ra2-Rb2)+Rb2); %Lokasi
```

```
%%
BestChart=[];
Velocity1=zeros(N,dim);
Velocity2=zeros(N,dim);
%%
for iteration=1:max_it
    %%
    iterasi=iteration
    [N,dim]=size(x1);
    for i=1:N

Tp=x1(i,:)>Ra1;Tm=x1(i,:)<Rb1;x1(i,:)=(x1(i,:).*(~(Tp+Tm)))+((rand(1,dim)
.*(Ra1-Rb1)+Rb1).*(Tp+Tm));
    end
    [N,dim]=size(x2);
    for i=1:N

Tp2=x2(i,:)>Ra2;Tm2=x2(i,:)<Rb2;x2(i,:)=round((x2(i,:).*(~(Tp2+Tm2)))+((r
and(1,dim).*(Ra2-Rb2)+Rb2).*(Tp2+Tm2)));
    end

    %%
    for i=1:N
        ii=1:dim;
        Qinj(i,ii) = x1(i,ii).*(Iline(x2(i,ii),1)')*sin(Rb3)*basemva;
        Lokasi(i,ii) = linedata(x2(i,ii),1)
    end
%%
    for nk=1:N
        nkk=1:dim;
        fit(nk)=
EvaluasiIndividu(busdata,linedata,Lokasi(nk,nkk),Qinj(nk,nkk),dim);
    end
function Fitness = EvaluasiIndividu(busdata,linedata,Lokasi,Qinj,dim)
basemva = 1000;  accuracy = 10^-4; accel = 1.6; maxiter = 1000;
%Input Lokasi dan MVAR SSSC
for i=1:dim
     if busdata(Lokasi(i),2)==0;
        busdata(Lokasi,6) = Qinj;
    end
 end
Lfybus
%  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

Lfnewton
%   Power flow solution by Newton-Raphson method
%   Copyright (c) 1998 by  H. Saadat
%   Revision 1 (Aug. 99) To include two or more parallel lines
ns=0; ng=0; Vm=0; delta=0; yload=0; deltad=0;
nbus = length(busdata(:,1));
kb=[];Vm=[]; delta=[]; Pd=[]; Qd=[]; Pg=[]; Qg=[]; Qmin=[]; Qmax=[];  %
Added (6-8-00)
Pk=[]; P=[]; Qk=[]; Q=[]; S=[]; V=[]; % Added (6-8-00)
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;
%%%% added for parallel lines (Aug. 99)
mline=ones(nbr,1);
for k=1:nbr
      for m=k+1:nbr
         if((nl(k)==nl(m)) & (nr(k)==nr(m)));
            mline(m)=2;
         elseif ((nl(k)==nr(m)) & (nr(k)==nl(m)));
         mline(m)=2;
         else, end
      end
   end
%%%   end of statements for parallel lines (Aug. 99)

% Start of iterations
clear A  DC   J  DX
```

```
while maxerror >= accuracy & iter <= maxiter % Test for max. power
mismatch
for ii=1:m
for k=1:m
   A(ii,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 ii=1:nbr
    if mline(ii)==1   % Added to include parallel lines (Aug. 99)
        if nl(ii) == n | nr(ii) == n
            if nl(ii) == n ,  l = nr(ii); end
            if nr(ii) == n , l = nl(ii); 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
      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
for i=1:nbus
    if Vm(i)<=0.95
        Vm(i)=Vm(i)+0.03;
    end
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

Lineflow
%  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')
indrecord = 1;
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;
       if busprt == 1
           CurrentRecord(indrecord,:) = [n k In];
           indrecord = indrecord+1;                          %modified
        end
       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
%clear Ik In SL Skn Snk

TVD = 0;
for i=1:nbus
    VD(i) = abs(Vm(i)-1.00);
    TVD = TVD + VD(i);
end
Fitness = real(SLT)+TVD;
for i=1:nbus
    if Vm(i)<=0.95;
        Vm(i)=0.95;
    else
        Vm(i)>=1.05;
        Vm(i)=1.05;
    end
end

 %%
[best, best_X]=min(fit); %minimization.
    if iteration==1
       Fbest=best;
       x1best=x1(best_X,:);
       x2best=x2(best_X,:);
    end

    if best<Fbest  %minimization.
       Fbest=best;
       x1best=x1(best_X,:);
       x2best=x2(best_X,:);
    end

    BestChart =[BestChart Fbest];
    %%
 Fmax=max(fit);
 Fmin=min(fit);
 Fmean=mean(fit);

   if Fmax==Fmin
      M=ones(N,1);
   else
      best=Fmin;
      worst=Fmax; %eq.17-18.
      M=(fit-worst)./(best-worst); %eq.15,
   end
```

```
M=M./sum(M);
%%
G=G0*exp(-alfa*iteration/max_it);

%%
%%%%total force calculation
 if ElitistCheck==1
     kbest=final_per+(1-iteration/max_it)*(100-final_per); %kbest in eq.
21.
     kbest=round(N*kbest/100);
 else
     kbest=N; %eq.9.
 end
    [Ms ds]=sort(M,'descend');

 for i=1:N
     E1(i,:)=zeros(1,dim);
     E2(i,:)=zeros(1,dim);
     for ii=1:kbest
         j=ds(ii);
         if j~=i
            R1=norm(x1(i,:)-x1(j,:),Rnorm); %Euclidian distanse.
            R2=norm(x2(i,:)-x2(j,:),Rnorm);
            for k=1:dim
             E1(i,k)=E1(i,k)+rand*(M(j))*((x1(j,k)-
x1(i,k))/(R1^Rpower+eps));
             E2(i,k)=E2(i,k)+round(rand*(M(j))*((x2(j,k)-
x2(i,k))/(R2^Rpower+eps)));%note that Mp(i)/Mi(i)=1
         end
         end
     end
 end

%%acceleration
a1=E1.*G; %note that Mp(i)/Mi(i)=1
a2=round(E2.*G);
%%
Velocity1=rand(N,dim).*Velocity1+a1; %eq. 11.
Velocity2=round(rand(N,dim).*Velocity2+a2);
x1=x1+Velocity1;
x2=x2+Velocity2;
end
%%
Fitnees_Optimal = Fbest
Ukuran_SSSC_Optimal = x1best
Lokasi_SSSC_saluran = x2best
for i=1:dim
    Qinjeksibest(i) =
Ukuran_SSSC_Optimal(i).*(Iline(Lokasi_SSSC_saluran(i),1)')*basemva;
    Placebest(i) = linedata(Lokasi_SSSC_saluran(i),1);
end
Qinjeksibest
Placebest
%%
plot(BestChart,'--k');
 xlabel('\fontsize{12}\bf Iteration');ylabel('\fontsize{12}\bf fitness
');
 legend('\fontsize{10}\bf GSA',1);
 %%
```