Optimum Design of Power System Stabilizer based on Improved Ant Colony Optimization Algorithm

Ruba AL-MulaHumadi
Assistant Professor
Mechanical Engineering Dept.
College of Engineering,
University of Baghdad, Iraq.
rubamkh@yahoo.com

Dr. Nizar Hadi Abbas
Assistant Professor
Electrical Engineering Dept.
College of Engineering,
University of Baghdad, Iraq.
drnizaralmsaodi@gmail.com

Mohanad Azeez Joodi
Assistant Lecturer
Electrical Engineering Dept.
College of Engineering,
University of Baghdad, Iraq.
eng_muhanad73@yahoo.com

ABSTRACT

This paper presents an improved technique on Ant Colony Optimization (ACO) algorithm. The procedure is applied on Single Machine with Infinite Bus (SMIB) system with power system stabilizer (PSS) at three different loading regimes. The simulations are made by using MATLAB software. The results show that by using Improved Ant Colony Optimization (IACO) the system will give better performance with less number of iterations as it compared with a previous modification on ACO. In addition, the probability of selecting the arc depends on the best ant performance and the evaporation rate.

Keywords: SMIB, PSS, ACO.

التصميم الامثل لـــ مثبت نظام القدرة المبني على خوارزمية مستعمرة النمل المطورة

ربى الملا حمادي
استاذ مساعد
قسم هندسة الميكانيك
كلية الهندسة
جامعة بغداد

د. نزار هادي عباس
استاذ مساعد
قسم هندسة الكهرباء
كلية الهندسة
جامعة بغداد

مهند عزيز جودي
مدرس مساعد
قسم هندسة الكهرباء
كلية الهندسة
جامعة بغداد

الخلاصة

يقدم البحث تقنية محسنة لخوارزمية مستعمرة النمل (ACO).الفكرة مطبقة على نظام SMIB مع مثبت نظام القدرة (PSS) في ثلاثة احمال مختلفة حيث تمت المحاكاة باستخدام برنامج MATLAB. بينت النتائج ان النظام باستخدام تقنية مستعمرة النمل المتطورة يعطي اداء افضل مع اقل عدد من التكرارات بالمقارنة مع تحويرات سابقة اجريت على مستعمرة النمل. كذلك ان احتمالية اختيار او تحديد مسار النمله يعتمد ايضا على افضل اداء للنمل ومعدل التبخر لمادة الفيرومون التي تفرزها النمله.

1. INTRODUCTION

In the last years, much effort has been invested in improving the damping performance of power systems using Power system stabilizers (PSS). PSS provides a supplementary excitations control signal that enhances the damping capabilities of synchronous machines. The choice of parameters for the PSS is important as it affects the overall dynamic performance of the power system, there are various forms of PSS controllers, and the famous types are lead-lag compensator (i.e., classical PSS) and PID.

Kundur, 1993, illustrated the construction, function, and operation of a PSS that uses auxiliary stabilizing signals to control the excitation system so as to improve power system dynamic performance. Commonly used input signals to the power system stabilizer are shaft speed, terminal frequency, and power. PSS will add a component of electrical torque in phase with the rotor speed.

A robust PID stabilizer was proposed by **Otaru, et al., 2004**. The authors used a genetic algorithm to enhance the performance of the considered system which is a synchronous generator connected to an infinite bus, this model is sufficient for low-frequency oscillations studies, the PID stabilizer gains are designed optimally using Genetic Algorithm (GA) to arrive at the optimal setting of the controller. Another PID-PSS was proposed by **Hosseini, et al., 2007,** where the gain setting of PID-PSS is optimized by minimizing an objective function using GA. The dynamic response was compared with other two stabilizers named FUZZY-PSS and LQR-PSS. The authors revealed that their stabilizer gave better performance. They applied the proposed stabilizer on single machine infinite bus (SMIB) system and the simulations were made in MATLAB.

Abdul-Ghaffar, et al., 2013, used hybrid particle swarm bacteria foraging optimization (PSO-BFA) in tuning the PID parameters. This research considers the stabilization of a synchronous machine connected to an infinite bus via a PID. Simulation results were presented with and without the proposed controller then compared with the classical PID. They applied the controller to SMIB system. The results showed that using of hybrid control gave better performance.

Duman and Ozturk, 2010, presented a Real Coded Genetic Algorithm (RCGA) based PID controller to improve power system dynamic stability applied on (SMIB). Different controllers' structures are presented; conventional power system stabilizer (CPSS), optimized PSS and RCGAPID are used to improve the stability. Two performance indexes; integral absolute error (IAE) and integral squared error (ISE) were used as objective functions. Different loading conditions were studied. The results show that the ISE is better than IAE for the optimization problem.

A Harmony Search Algorithm (HSA) approach is presented by **Abdul Hameed, et al., 2014**, for the robust and optimal design of PID controller to PSS for damping low-frequency power oscillation. Also, they applied their technique to infinite bus single machine system, Eigenvalue analysis by using genetic algorithm based PSS (GAPSS) under different operating conditions reveals that under-damped and lightly damped oscillation modes are shifted to a specific stable zone in the S- Plane.

Boroujeni, et al., 2011, demonstrated a different type of stabilizers to generate supplementary damping control signals for the excitation system to damp the low frequency oscillation of the electric power system, they applied a new optimal technique PID type PSS based on (PSO-PSS) to a typical single machine infinite bus power system. The simulation results demonstrated that these design is showing the guarantee of the robust stability and robust performance of the power system to some conditions.

A new technique in designing a power system stabilizer PSS was presented by **Mahmoud and Soliman, 2012**, based on a combination of Particle Swarm Optimization (PSO) and Linear Matrix Inequality (LMI) in order to eliminate the number of variables. They applied their idea on (SMIB) using MATLAB environment. Finally, they concluded that their method was effective and convergence as the system confirms better performance under different loading conditions.
Soliman, et al., 2008, demonstrated another type of PSS that minimizes the maximum overshoot in order to alleviate the generator shaft fatigue. They used PSO algorithm in order to fix the gain of the PSS and lead time compensator. They applied their stabilizer to SMIB system at different loading conditions, the results showed the effectiveness and robustness of the proposed technique.
The rest of this paper is organized as follows: the mathematical model is illustrated in the next section. Ant colony optimization and its' modifications are demonstrated in the third section while the fourth section contains the results and discussion. Finally, a conclusion is demonstrated in the last section.

2. MATHEMATICAL MODEL

In this section, the mathematical model of the system with PSS and finding the transfer function are described.

2.1 SMIB

A synchronous machine with infinite bus system was taken in this research as a test system, as shown in Fig.1. The state space model "A, B, C, and D" are shown in appendix **Abdul-Ghaffar, et al., 2013,** and **Mahmoud** and **Soliman, 2012,** with the machine data in pu are X_d=1.6,$\dot{X}_d = 0.32$ X_q=1.55, ω_d=2*π*50 rad/sec, $T_{do}^{'} = 6\ sec,$ and M=10, transmission line reactance X_e=0.4, r_e=0. The machine constants k_1 to k_6 are as ref. **Mahmoud and Soliman, 2012,** and their values are depending on the loading conditions.
The transfer functions of the system for three loading conditions are tabulated in Table 1.

2.2 Power System Stabilizer (PSS)

The transfer function of power system stabilizer is

$$pss\,TF = K_i\,\frac{sT_w}{1+sT_w}\left[\frac{1+sT_1}{1+sT_2}\,\frac{1+sT_3}{1+sT_4}\right] \tag{1}$$

Where K_i represents the gain of the PSS

$\frac{sT_w}{1+sT_w}$ washout

$\left[\frac{1+sT_1}{1+sT_2}\,\frac{1+sT_3}{1+sT_4}\right]$ lead-lag compensator.

The limits for T_1 to T_4, T_W and K_i as ref. **Mahdiyeh, et al., 2010.**

$0.01 < T_1, T_2, T_3, \& T_4 < 2$ sec.

$1 < K_i < 50$

2.3 System with PSS

After connecting the PSS with the system as shown in **Fig.1**, the signal flow graph of the overall system will be as shown in **Fig.2**.

3. CLASSICAL ANT COLONY OPTIMIZATION (CACO) ALGORITHM

CACO is kind of optimization that is based on the behavior of ants in searching of the food process. The flow chart of CACO is illustrated in **Fig.3**.

3.1 Modified Ant Colony Optimization (MACO) Algorithm

The flowchart of this method is shown in **Fig.4**. A modification was made by **Mathiyalagan, et al., 2010**, that deal with the process of updating the pheromone. This equation is illustrated in the flowchart of **Fig.4**, as one can see there the new pheromone depends on the pheromone evaporation rate (ρ). In addition to this, the initial pheromone is entered in a random way with a dependency on the pheromone evaporation rate.

3.2 Improved Ant Colony Optimization (IACO) Algorithm

The proposed ACO algorithm in this research work is represented by a new improvement through modifying the updating pheromone equation; by adding the pheromone deposited by the best ant ($\zeta * f_{best}$ / f_{worst}) multiplied by (k) in case more than one ant take the best path. The flowchart of this process is illustrated in **Fig. 5**.

4. RESULTS AND DISCUSSION

The results of this research are simulated by MATLAB R2013 environment executed on the core (TM) i5, 2.5GHz and 4 RAM system. 8 overall proper transfer functions (SMIB with PSS) were taken for 3 loading regimes (heavy load, nominal load, and light load). 4 design variables were taken; T_1, T_2, T_W, and gain of PSS. Two different methods of optimization were taken; modified ant colony optimization (MACO) introduced by **Mathiyalagan, et al., 2010,** and proposed improved ant colony optimization (IACO), 4 ants were taken for each variable and two evaporation rate (ρ= 0.5 and 0.2). Table 1, shows the results; the system alone, the system with PSS applying MACO and the system with PSS applying IACO at ρ=0.5.

Table 2, shows the results of the system with PSS applying MACO and the system with PSS based on IACO at ρ=0.2.

In spite of getting the same results sometimes, IACO algorithm is better than the MACO algorithm that because of two reasons; the number of iterations in the first method (MACO) is between 10-500 iterations while it is between 1-15 iterations in the second method (IACO). This means the second method is faster.

The second reason deals with the process of updating the pheromone that affects the probability of selecting the arc.

MACO $\tau_{1j}^{(2)} = [\{\rho + \left(\frac{1-\rho}{1+\rho}\right)\}\tau_{1j}^{old}] + [\{\rho - \left(\frac{\rho}{1+\rho}\rho\right\} * \Delta\tau_{1j}]$ (2)

$$\text{IACO}\ \tau_{1j}^{(2)} = \left[\left\{\rho + \left(\frac{1-\rho}{1+\rho}\right)\right\} * \tau_{1j}^{old}\right] + \left[\left\{\rho - \left(\frac{\rho}{1+\rho}\right)\right\} * k * \frac{\xi f_{best}}{f_{worst}}\right] \qquad (3)$$

It is clear from the 1st equation that the last term ($\Delta\tau_{1j}$) is constant while the last term in the 2nd equation takes the effect of the best ant that makes the probability of selecting the arc closer to the optimal solution then the number of iteration will be less

5. CONCLUSIONS

Many researchers studied the stability problem of power system with the existence of PSS, and many methods were taken in order to analyze the system performance. One of these methods is ACO, an additional improvement to a previous modification of ACO algorithm is proposed in this paper. It is clear from the results that the system gives better performance when using IACO than when using MACO. That because the number of iterations is less and the simulation process will be faster. The process of updating the pheromone depends on the effect of the best ant.

Detailed illustrations about the used programs and the followed execution procedure:

The pare presents 3 flowcharts; Fig.3 shows Conventional ACO, Fig.4 shows Modified ACO, **Fig.5** Improved ACO. Illustrations of MATLAB codes for Modified ACO and Improved ACO are presented. While Conventional ACO was presented in a previous paper by another researcher.

Program 1/ no flowchart presented in this paper.

```
clear
clc
% from paper Soliman, et al, 2008.
% system alone (machine data)
xq=1.55; xd=1.6; xd_dash=0.32;M=10; wo=100*pi;Tdo_dash=6;D=0;
xe=0.4;re=0; Te=0.05; Ke=25; V=1;
%  three load conditions are considered
% ======================================
% heavy load    nominal load    light load
% -----------[]--------------[]-----------
%   P = 1    []   P=0.7      []  P=0.4
%   Q=0.5    []   Q=0.3      []  Q=0.1
% ----------------------------------------
P=input('input active power')
Q=input('input reactive power')
% calculations of c1-c7 and k1-k6 constants
c1=V^2/(xe+xq);
c2=(xd_dash+xe)/(xd+xe);
c3=c1*((xq-xd_dash)/(xe+xd_dash));
c4=V/(xe+xd_dash);
c5=(xd-xd_dash)/(xe+xd_dash);
c6=c1*xq*((xq-xd_dash)/(xe+xq));
c7=xe/(xe+xd_dash);

K1=c3*(P^2/(P^2+(Q+c1)^2))+Q+c1;
K2=c4*(P/(sqrt(P^2+(Q+c1)^2)));
K3=c2;
K4=c5*(P/(sqrt(P^2+(Q+c1)^2)));
K5=c4*xe*(P/(V^2+Q*xe))*(c6*((c1+Q)/(P^2+(c1+Q)^2))-xd_dash);
```

```
K6=c7*((sqrt(P^2+(Q+c1)^2))/(V^2+Q*xe))*(xe+(c1*xq*(c1+Q))/(P^2+(c1+Q)^2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%___________________________%%%%%%%%%%%%%

% x=[delta(delta)  delta(w)  delta(Eq')  delta(Efd)]
A=[  0             wo     0                     0
   -K1/M        -D/M    -K2/M                   0
   -K4/Tdo_dash    0    -1/(K3*Tdo_dash)       1/Tdo_dash
   -K5*Ke/Te       0    -K6*Ke/Te              -1/Te];


B=[0  0  0   Ke/Te]';
C=[0   1  0  0];
D=0   ;
[num,Den]=ss2tf(A,B,C,D);
disp('sys alone')
G=tf(num,Den)% tf of sys alone
eig(G);
s=stepinfo(G)
     tr=s.RiseTime;
     ts=s.SettlingTime;
     Mp=s.Overshoot;
     Ess=abs(1-dcgain(G))
figure(1);
step(G);
title('sys alone');

%  pss
syms s
 % T1 T2 T3 T4 compensators time
 T1= [0.02 0.05 0.11   0.2   0.3  0.4   0.55   0.15 ];
 T3=T1;
 T2 =[0.01 0.03  0.07  0.09  0.12  0.22  0.45  0.065 ];
 T4=T2;
 ks=[1   7   11   17   26   35   40   45];    % pss gain
 Tw=[1   2   6    8    10   12   14   16] ;   % washout time
for i=1:8   % 8 is the number of states of T1 T2 T3 T4

G1=ks(i)*(s*Tw(i)/(1+s*Tw(i)))*((1+s*T1(i))/(1+s*T2(i)))*((1+s*T3(i))/(1+s*T4
(i)));  %pss TF
    [N1,D1]=numden(G1);  % of pss
    n1=sym2poly(N1);
    d1=sym2poly(D1);
    disp('TF of PSS');
    GG1=tf(n1,d1); % TF of PSS
%figure(2)
%step(GG1)
%title('pss alone')
              %%% pss with sys
     [n_sp,d_sp]=feedback(num,Den,n1,d1,+1);
     disp('sys with pss');
     i
     G_sp=tf(n_sp,d_sp);
     figure;
     step(G_sp);
     title('sys with pss');
     S=stepinfo(G_sp)
     tr=S.RiseTime;
     ts=S.SettlingTime;
     Mp=S.Overshoot;
     Ess=abs(1-dcgain(G_sp))
     % the objective function
     fitt(i)= (0.25*tr) + (0.5*ts) + (0.25*Ess);
     TF(i)=G_sp;
end
fitt;TF;
     %%%%%% this is the end %%%%%%
```

How to execute the programs:
First of all, execute the program 1 by entering active and reactive powers according to the load regimes (heavy, nominal and light) loads. Taking, for example, the first case (heavy load condition; active power is 1 and reactive power is 0.5. The program will continue its execution until getting the transfer function of the system alone as:

```
                       -8.132 s
    G = ---------------------------------------------------------
          s^4 + 20.46 s^3 + 98.45 s^2 + 922.7 s + 2363
```

ts, tr and ESS can be obtained from "`stepinfo`" MATLAB command as:

```
s =
        RiseTime: 0
    SettlingTime: 156.2869
     SettlingMin: -0.0114
     SettlingMax: 0.0084
       Overshoot: Inf
      Undershoot: Inf
            Peak: 0.0114
        PeakTime: 0.4433
Ess =
     1
```

We have 4 variables (T_1, T_2, Ks, and Tw) each one of 8 values as stated in the program. At the end of the program, we will get 8 transfer functions (`TF`) and 8 values of the objective function (`fitt`).

Now go to the second program

Program 2: flowchart of Fig.4 in this paper depending on equation of ref. **Mathiyalagan, et al., 2010.**

```
%% Modified Ant colony optimization Mathiyalagan, et al., 2010.
% applied on SMIB WITH PSS
clear
clc
N= input('the no. of ants') % write N=4; no_of_ants for each variable
p=input('no. of states')% write 8; no_of_states
% for min < T1, T2, ks, Tw < max, from program1
n=input('no. of design') % write 1 for each time u execute the program
% because we have4 design variables are PSS gain, T1 T2 Tw
% x is a matrix of 8 overall TF obtained from program1
syms TF1,syms TF2, syms TF3, syms TF4, syms TF5, syms TF6, syms TF7, syms TF8
```

```
x=[TF1  TF2  TF3  TF4  TF5  TF6  TF7  TF8]

%fx=fitt results from program1
disp('1:heavy load,2:nominal load, 3:light load')
load=input('load')
if load==1
   fx= [55.4714  18.2018  9.3514  2.3952  1.1983  3.3588  52.3429  1.2541] %
heavy load
elseif load==2
   fx=[18.5437   10.7712    6.8578    2.3386    1.2461    2.6462   11.7511
1.3481]% nominal load
elseif load==3
   fx =[11.0834    7.5520    5.4010    2.2261    1.0898    1.9588    5.4122
1.5132] %light load
end

jj=1:8  % the order at the 8 states

roo=input('roo');  % roo=0.2   0.5   0.9
iteration=1

tao=1-roo*rand;    % initial pheromone
for i=1:p
    tao1(i)=tao;
end

p1j=tao/sum(tao1);

x1=[0    p1j   2*p1j  3*p1j  4*p1j  5*p1j   6*p1j   1]

syms ant1, syms ant2, syms ant3, syms ant4
ant=[ ant1    ant2     ant3      ant4]

r=rand(1,N)  %N is the number of ants
for i=1:N
    for j=1:p
        if r(i)> x1(j) & r(i)<x1(j+1)
           xx(i)=x(j);    % to present the TF
           fx1(i)=fx(j);  % to present the fitness
           order(i)=jj(j); % to present the index
        end
    end
end
xx;
fx1, order
fbest=min(fx1)
fworst=max(fx1)
% to print ant number and x
k=0;
for i=1:N
    if fbest==fx1(i)
        best_ant =[xx(i),   ant(i),   order(i)]
        k=k+1;
    elseif fworst==fx1(i)
        worst_ant=[xx(i),   ant(i),   order(i)]
    else
    end
```

```
end
k

t1_j=tao

while k<N
   % step 4
   % ants return home and start again in search of food
   iteration=iteration+1

   % modification
   %t_new=t1_j=(roo+((1-roo)/(1+roo)))*t1_j+(roo-(roo/(1+roo)))*0.2; from
ref. Mathiyalagan, et al., 2010.

   t1_j=(roo+((1-roo)/(1+roo)))*t1_j+(roo-(roo/(1+roo)))*0.2 % new value
   % go to step 2
   for j=1:p
       t(j)=t1_j;
   end
   sum(t);
   p1_j=t/sum(t);
   p1_j


    % to prepare x11:x18 in range 0-1
   x11(1)=0;
   x1=0;
   for i=2:7
       x1=x1+p1_j(i-1);
       x11(i)=x1;
   end
   x11(8)=1;
   x11

   r=rand(1,N)
   for i=1:N
       for j=1:p
           if r(i)> x11(j) & r(i)<x11(j+1)
              xx(i)=x(j);
              fx1(i)=fx(j);
              order(i)=jj(j);
           end
       end
   end
   xx
   fx1, order
   fbest=min(fx1)
   fworst=max(fx1)
   % to print ant number and x
   k=0; % k is the no. of best ants
   for i=1:N
       if fbest==fx1(i)
          best_ant= [xx(i), ant(i), order(i)]
          k=k+1;
       elseif fworst==fx1(i)
          worst_ant=[ xx(i), ant(i), order(i)]
       else
```

```
        end
    end
    k

end
```

To execute program 2 input

```
N=4;
P=8;
n=1;
```

Remove the lines:

```
syms TF1,syms TF2, syms TF3, syms TF4, syms TF5, syms TF6, syms TF7, syms TF8
x=[TF1  TF2  TF3  TF4  TF5  TF6  TF7  TF8]
```

Write `x=TF,` to let x=8 transfer function that described in program 1.
Then input load 1, to execute for heavy load case.
Input `roo=0.2;`
After that, the Ant Colony will be operated with the fitness function according to that presented in **Mathiyalagan, et al., 2010.**
Wait for the final results. All the ants will follow the same direction. And the program will give you the best transfer function also the program will give you the number of iterations.
The results are shown in Table2.
Now repeat for `roo=0.5,` the results are shown in Table 3.

Finally, follow the same procedure as described above but this time on program 3. The difference is the analysis is done according to our fitness equation.

Program 3/ flowchart of Fig.5 in this paper depending on our equation

```
%% Modified Ant colony optimization   "our equations"
% applied on SMIB WITH PSS
clear
clc
N= input('the no. of ants') % write N=4; no_of_ants for each variable
p=input('no. of states')% write 8; no_of_states
% for  min < T1, T2, ks,Tw < max, from program1
n=input('no. of design') % write 1 for each time u execute the program
% because we have4 design variables are PSS gain, T1 T2 Tw
% x is a matrix of 8 overall TF obtained from program1
syms TF1,syms TF2, syms TF3, syms TF4, syms TF5, syms TF6, syms TF7, syms TF8
x=[TF1  TF2  TF3  TF4  TF5  TF6  TF7  TF8]

%fx=fitt results from program1
disp('1:heavy load,2:nominal load, 3:light load')
load=input('load')
if load==1
   fx= [55.4714  18.2018  9.3514  2.3952  1.1983  3.3588  52.3429  1.2541] %
heavy load
elseif load==2
   fx=[18.5437    10.7712    6.8578    2.3386    1.2461    2.6462    11.7511
1.3481]% nominal load
elseif load==3
   fx =[11.0834    7.5520    5.4010    2.2261    1.0898    1.9588    5.4122
1.5132] %light load
end

jj=1:8  % the order at the 8 states
```

```
roo=input('roo');  % roo=0.2   0.5   0.9

iteration=1

tao=1-roo*rand;     % initial pheromone
for i=1:p
    tao1(i)=tao;
end

p1j=tao/sum(tao1);

x1=[0    p1j   2*p1j  3*p1j  4*p1j  5*p1j   6*p1j   1]

syms ant1, syms ant2, syms ant3, syms ant4
ant=[ ant1    ant2     ant3      ant4]

r=rand(1,N)  %N is the number of ants
for i=1:N
    for j=1:p
        if r(i)> x1(j) & r(i)<x1(j+1)
            xx(i)=x(j);     % to present the TF
            fx1(i)=fx(j);  % to present the fitness
            order(i)=jj(j); % to present the index
        end
    end
end
xx;
fx1, order
fbest=min(fx1)
fworst=max(fx1)
% to print ant number and x
k=0;
for i=1:N
    if fbest==fx1(i)
        best_ant =[xx(i),   ant(i),   order(i)]
        k=k+1;
    elseif fworst==fx1(i)
        worst_ant=[xx(i),   ant(i),   order(i)]
    else
    end
end
k

t1_j=tao;
t_2=tao;

while k<N
   % step 4
   % ants return home and start again in search of food
   iteration=iteration+1

   % modification
   %t_new=(p+(1-p)/(1+p))*t_old+(p-p/(1+p))*sum(delta(t)) our equation
   t1_j=(roo+((1-roo)/(1+roo)))*t1_j % old value
   zeta=2; % scaling parameter
```

```
    sum_delta_t=k*zeta*fbest/fworst
    t_2=(roo+((1-roo)/(1+roo)))*t_2+sum_delta_t

    % go to step 2
    for j=1:p
        if j==best_ant(3)  % at j=3 best ant for the first case
           t(j)=t_2;
        else
           t(j)=t1_j;
        end
    end
    sum(t);
    p1_j=t/sum(t);
    p1_j

     % to prepare x11:x18 in range 0-1
    x11(1)=0;
    x1=0;
    for i=2:7
        x1=x1+p1_j(i-1);
        x11(i)=x1;
    end
    x11(8)=1;
    x11

    r=rand(1,N)
    for i=1:N
        for j=1:p
            if r(i)> x11(j) & r(i)<x11(j+1)
               xx(i)=x(j);
               fx1(i)=fx(j);
               order(i)=jj(j);
            end
        end
    end
    xx
    fx1, order
    fbest=min(fx1)
    fworst=max(fx1)
    % to print ant number and x
    k=0; % k is the no. of best ants
    for i=1:N
        if fbest==fx1(i)
           best_ant= [xx(i), ant(i), order(i)]
           k=k+1;
        elseif fworst==fx1(i)
           worst_ant=[ xx(i), ant(i), order(i)]
        else
        end
    end
    k

end
```

The same is done for nominal and light loads. The results are shown in Tables 2 and 3.

REFERENCES:

- Abdul Hameed, K., and Palani, S., 2014, *Robust Design of Power System Stabilizer using Harmony Search Algorithm*, Automatika, Vol. 55, No. 2, PP. 162–169.
- Abdul-Ghaffar, H. I., Ebrahim, E. A., and Azzam, M., 2013, *Design of PID Controller for Power System Stabilization Using Hybrid Particle Swarm-Bacteria Foraging Optimization*, WSEAS Transactions on Power Systems, Issue 1, Vol. 8, PP. 12-23.
- Boroujeni, S. M. S., Hemmati, R., Delafkar, H., and Boroujeni, A. S., 2011, *Optimal PID Power System Stabilizer Tuning based on Particle Swarm Optimization*, Indian Journal of Science and Technology, Vol. 4, No. 4, PP. 379-383.
- Duman, S., and Öztürk, A., 2010, *Robust Design of PID Controller for Power System Stabilization by Using Real Coded Genetic Algorithm*, International Review of Electrical Engineering (I.R.E.E.), Vol.5, No. 5, PP. 2159-2170.
- Hosseini, S. H., Reza, R., and Hamed, K., 2007, *Application of Genetic Algorithm to Design PID Controller for Power System Stabilization*, 5th International Conference on Electrical and Electronics Engineering (ELECO 2007), Bursa, Turkey.
- Kundur., P., 1993, *power system stability and control*, McGraw-Hill, New York.
- Mahdiyeh Eslami, Hussain Shareef, Azah Mohamed and S. P. Ghoshal,2010, *Tuning of power system stabilizers using particle swarm optimization with passive congregation*, International Journal of the Physical Sciences Vol. 5(17), PP. 2574-2589, Available online at http://www.academicjournals.org/IJPS
  ISSN 1992 - 1950 ©2010 Academic Journals.
- Mahmoud, M. S., and Soliman, H. M., 2012, *Design of Robust Power System Stabilizer Based on Particle Swarm Optimization*, Circuits and Systems, Vol.3, PP. 82-89.
- Mathiyalagan, P., Dhepthie, U. R., and Sivanandam, S. N., 2010, *Enhanced Hybrid PSO-ACO Algorithm for Grid Scheduling*, ICTACT journal on soft computing, Issue:01, PP. 54-59.
- Otaru, M. U., Al-Musabi, N. A., and Al-Baiyat, S. A., 2004, *Robust PID Stabilizer Design using Genetic Algorithms*, Proceedings of the 2nd IEEE GCC Conference, Bahrain, PP. 33-36 ,
- Rao, S. S., 2009, *Engineering Optimization Theory and Practice*, 4th edition, John Wiley & Sons. INC.
- Soliman, H. M., Bayoumi, E. H. E., and Hassan, M. F., 2008, *PSO–Based Power System Stabilizer for Minimal Overshoot and Control Constraints*, Journal of Electrical Engineering, Vol. 59, No. 3, PP. 153–159.

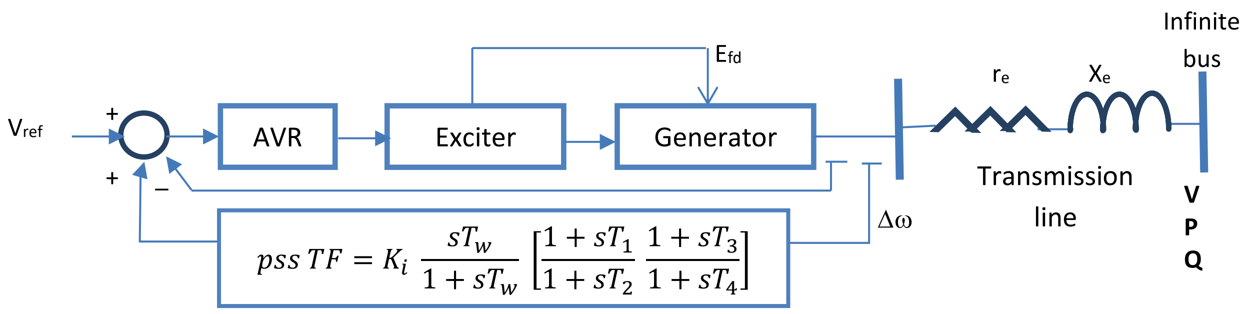
Infinite bus
Efd
re
Xe
Vref
+
+
−
AVR
Exciter
Generator
Transmission line
Δω
V
P
Q
$pss\,TF = K_i \frac{sT_w}{1+sT_w}\left[\frac{1+sT_1}{1+sT_2}\,\frac{1+sT_3}{1+sT_4}\right]$

Figure 1. A synchronous machine with the infinite bus.

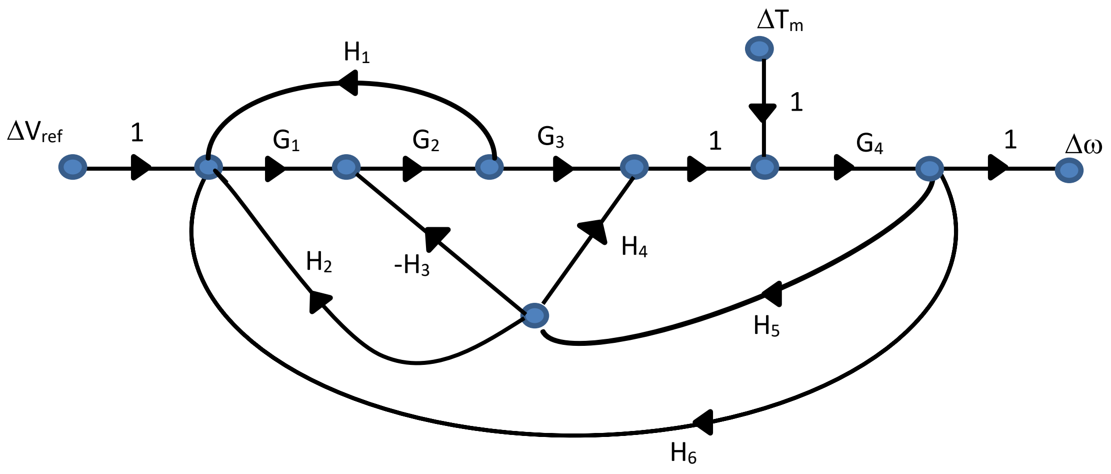

Figure 2. Signal flow graph of the system with PSS.

Where:

$$G_1 = \frac{K_E}{1+sT_E}, \quad G_2 = \frac{K_3}{1+sT_3}, \quad G_3 = K_2, \quad G_4 = \frac{1}{M}$$

$$H_1 = K_6, \quad H_2 = K_5, \quad H_3 = K_4, \quad H_4 = K_1, \quad H_5 = \frac{314}{s}, \quad H_6 = P_{ss}$$

♦♦1 the no. of paths or arcs is the permissible discrete values (X_{11}, X_{12}, X_{13}, …, X_{1P}) within a selected range.

♦♦2 the design variables are T_1, T_2, T_W, and G_{PSS}.

♦♦3 the fitness function =0.25*tr+0.5*ts+0.25*Ess

♦♦4 generate 4 random numbers, one for each ant.

♦♦5 K is the number of best ant ζ is the scaling factor assumed to be 2. **Rao, 2009.** $\tau_{1j}^{old}=(1-\rho)*\tau_{1j}^{(1)}$, ρ=0.5 pheromone decay factor **Rao, 2009.**

♦♦6 if the no. of best ants (k)=the no. of ants (N), that means all the ants follow the same path

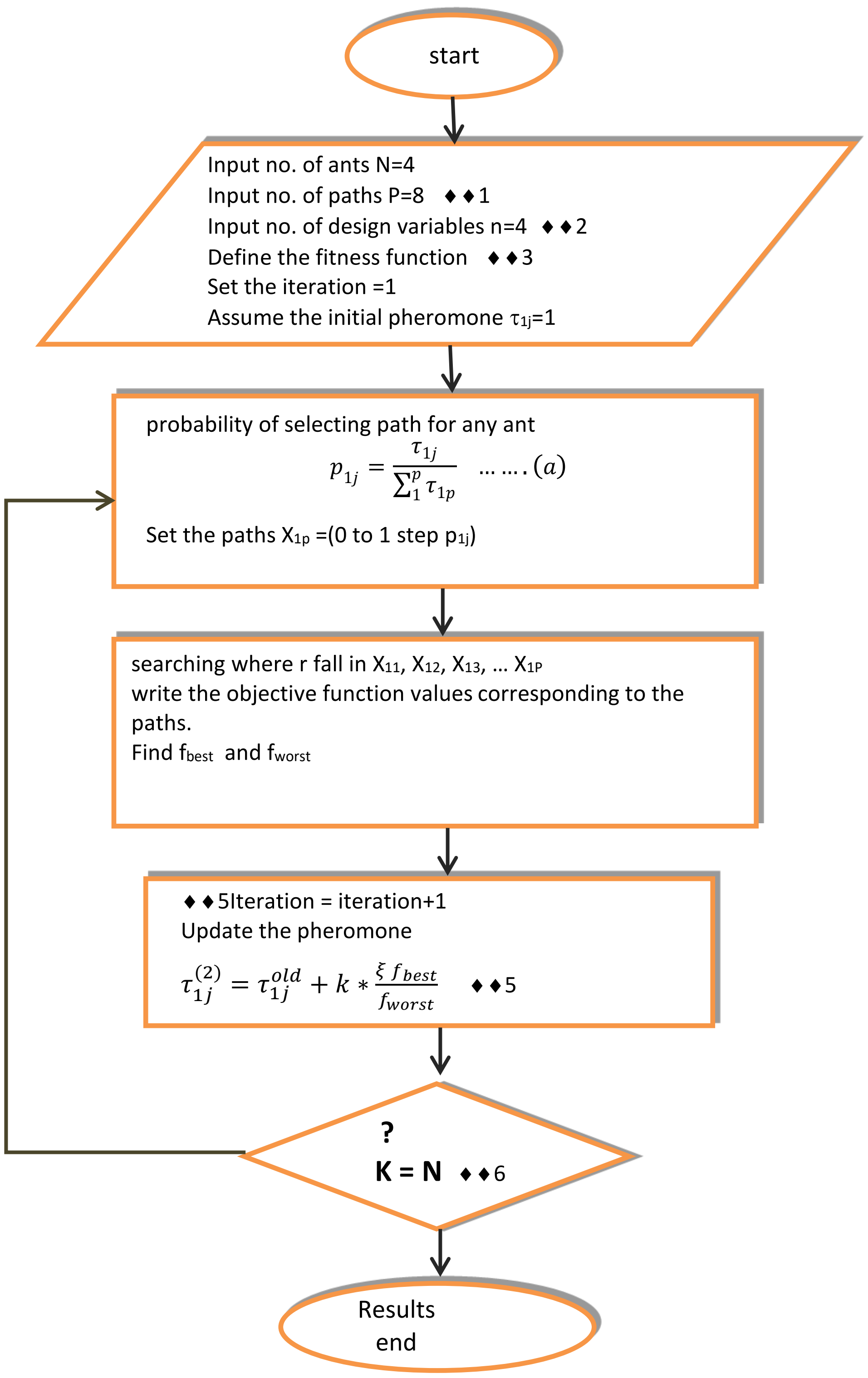
start
Input no. of ants N=4
Input no. of paths P=8 ♦♦1
Input no. of design variables n=4 ♦♦2
Define the fitness function ♦♦3
Set the iteration =1
Assume the initial pheromone τ_{1j}=1
probability of selecting path for any ant
$p_{1j} = \frac{\tau_{1j}}{\sum_1^p \tau_{1p}}$ … ….(a)
Set the paths X_{1p} =(0 to 1 step p_{1j})
searching where r fall in X_{11}, X_{12}, X_{13}, … X_{1P}
write the objective function values corresponding to the paths.
Find f_{best} and f_{worst}
♦♦5Iteration = iteration+1
Update the pheromone
$\tau_{1j}^{(2)} = \tau_{1j}^{old} + k * \frac{\xi f_{best}}{f_{worst}}$ ♦♦5
?
K = N ♦♦6
Results
end

Figure 3. The flowchart of CACO algorithm.

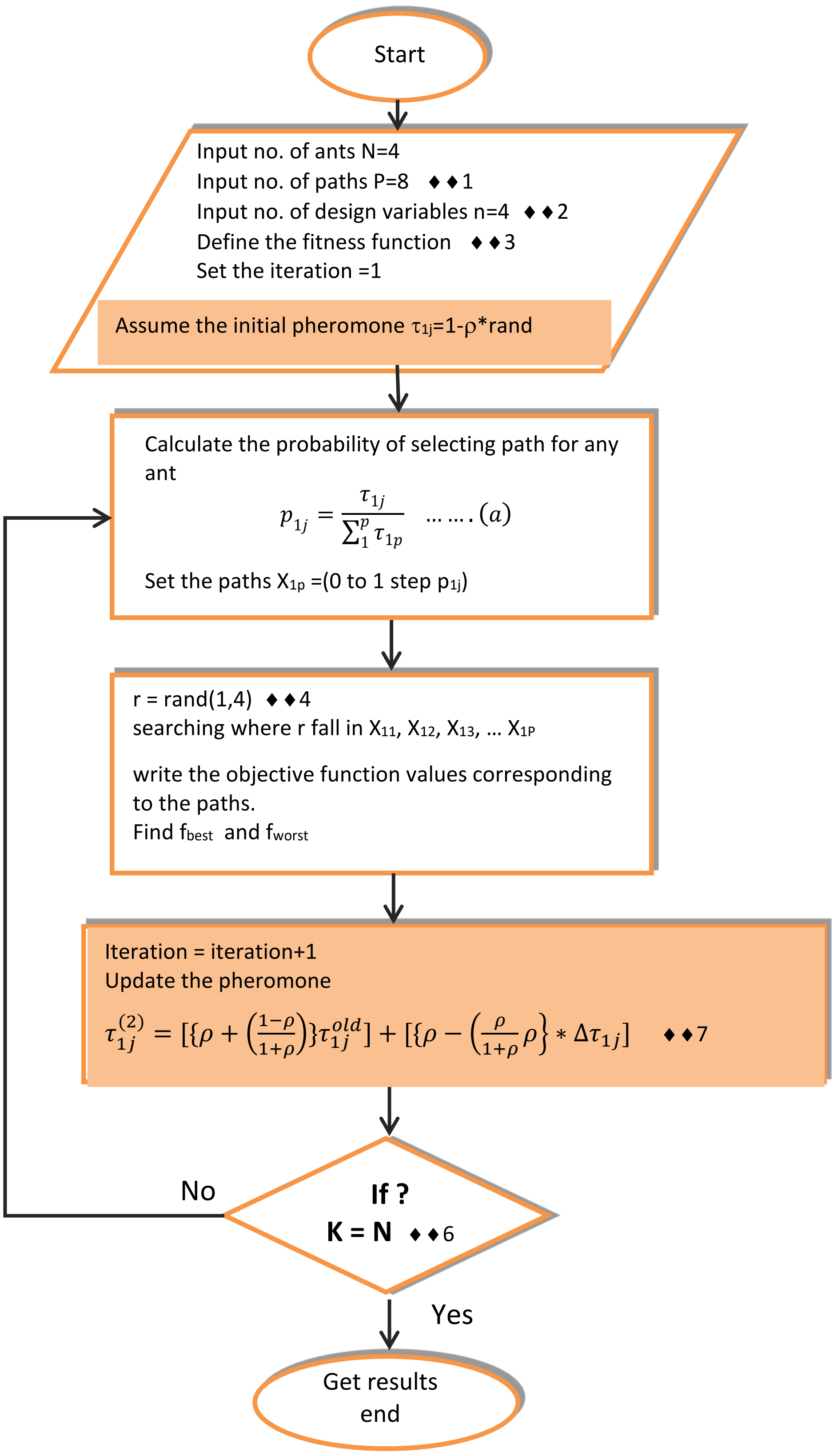
Start
Input no. of ants N=4
Input no. of paths P=8 ♦♦1
Input no. of design variables n=4 ♦♦2
Define the fitness function ♦♦3
Set the iteration =1
Assume the initial pheromone τ_{1j}=1-ρ*rand
Calculate the probability of selecting path for any ant
$p_{1j} = \frac{\tau_{1j}}{\sum_1^p \tau_{1p}} \quad \dots\dots (a)$
Set the paths X_{1p} =(0 to 1 step p_{1j})
r = rand(1,4) ♦♦4
searching where r fall in X_{11}, X_{12}, X_{13}, … X_{1P}
write the objective function values corresponding to the paths.
Find f_{best} and f_{worst}
Iteration = iteration+1
Update the pheromone
$\tau_{1j}^{(2)} = [\{\rho + \left(\frac{1-\rho}{1+\rho}\right)\}\tau_{1j}^{old}] + [\{\rho - \left(\frac{\rho}{1+\rho}\rho\right\} * \Delta\tau_{1j}]$ ♦♦7
No
If ?
K = N ♦♦6
Yes
Get results
end

Figure 4. The flowchart of MACO algorithm.

♦♦7 $\Delta\tau_{1j}$=0.2 constant value according to ref. **Mathiyalagan, et al., 2010**.

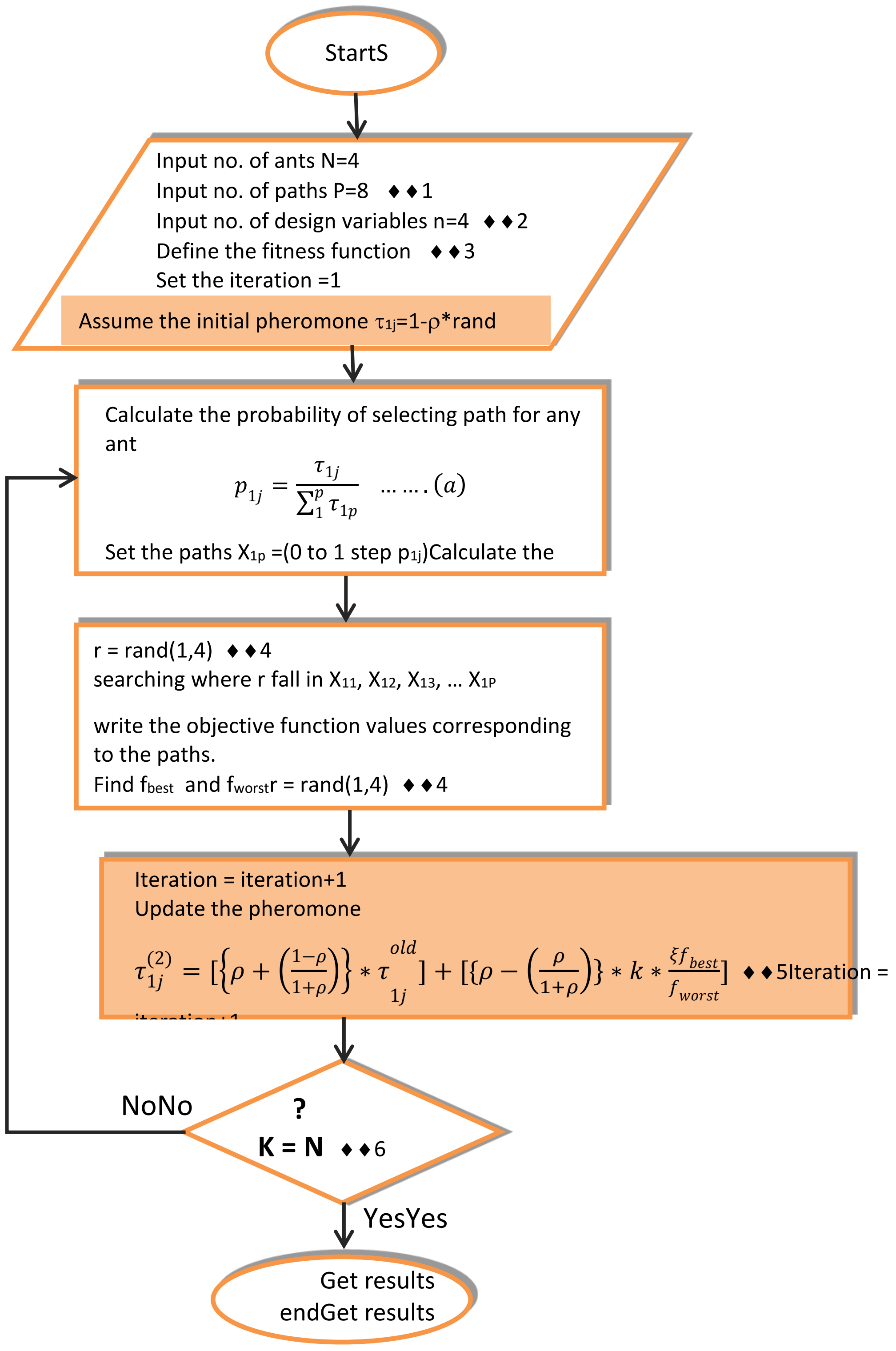
StartS
Input no. of ants N=4
Input no. of paths P=8 ♦♦1
Input no. of design variables n=4 ♦♦2
Define the fitness function ♦♦3
Set the iteration =1
Assume the initial pheromone τ_{1j}=1-ρ*rand
Calculate the probability of selecting path for any ant
$p_{1j} = \frac{\tau_{1j}}{\sum_1^p \tau_{1p}}$ …….(a)
Set the paths X_{1p} =(0 to 1 step p_{1j})Calculate the
r = rand(1,4) ♦♦4
searching where r fall in X_{11}, X_{12}, X_{13}, … X_{1P}
write the objective function values corresponding to the paths.
Find f_{best} and f_{worst}r = rand(1,4) ♦♦4
Iteration = iteration+1
Update the pheromone
$\tau_{1j}^{(2)} = [\{\rho + (\frac{1-\rho}{1+\rho})\} * \tau_{1j}^{old}] + [\{\rho - (\frac{\rho}{1+\rho})\} * k * \frac{\xi f_{best}}{f_{worst}}]$ ♦♦5Iteration = iteration+1
NoNo
?
K = N ♦♦6
YesYes
Get results endGet results

Figure 5. The flowchart of IACO algorithm.

Table 1. Loading conditions and transfer functions

Loading condition	Transfer function
Light load	$\dfrac{-6.326\,s}{s^4 + 20.46\,s^3 + 79.4\,s^2 + 558.4\,s + 1242}$
Nominal load	$\dfrac{-7.553\,s}{s^4 + 20.46\,s^3 + 89.15\,s^2 + 756.9\,s + 1795}$
Heavy load	$\dfrac{-8.132\,s}{s^4 + 20.46\,s^3 + 98.45\,s^2 + 922.7\,s + 2363}$

Table 2. The system time response performance at ρ=0.5

	System alone	MACO ρ=0.5	IACO ρ=0.5
Heavy load	ts =156.28 sec	ts=1.89	ts=1.89
P=1	tr=0 sec	tr=0	tr=0
Q=0.5	Ess=1	Ess=1	Ess=1
	Fig. 6	**Fig. 7,** TF5	**Fig. 7,** TF5
Nominal load	ts=41.40	ts=1.99	ts=1.99
P=0.7	tr=0	tr=0	tr=0
Q=0.3	Ess=1	Ess=1	Ess=1
	Fig. 8	**Fig. 9,** TF5	**Fig. 9,** TF5
Light load	ts=23.58	ts=3.41	ts=1.67
P=0.4	tr=0	tr=0	tr=0
Q=0.1	Ess=1	Ess=1	Ess=1
	Fig. 10	**Fig. 11,** TF6	**Fig. 12,** TF5

Table 3. The system time response performance at ρ=0.2

	MACO ρ=0.2	IACO ρ=0.2
Heavy load	ts=18.20	ts=1.89
P=1	tr=0	tr=0
Q=0.5	Ess=1	Ess=1
	Fig. 13, TF3	**Fig. 7,** TF5
Nominal load	ts=1.99	ts=1.99
P=0.7	tr=0	tr=0
Q=0.3	Ess=1	Ess=1
	Fig. 9, TF5	**Fig. 9,** TF5
Light load	ts=1.67	ts=1.67
P=0.4	tr=0	tr=0
Q=0.1	Ess=1	Ess=1
	Fig. 12, TF5	**Fig. 12,** TF5

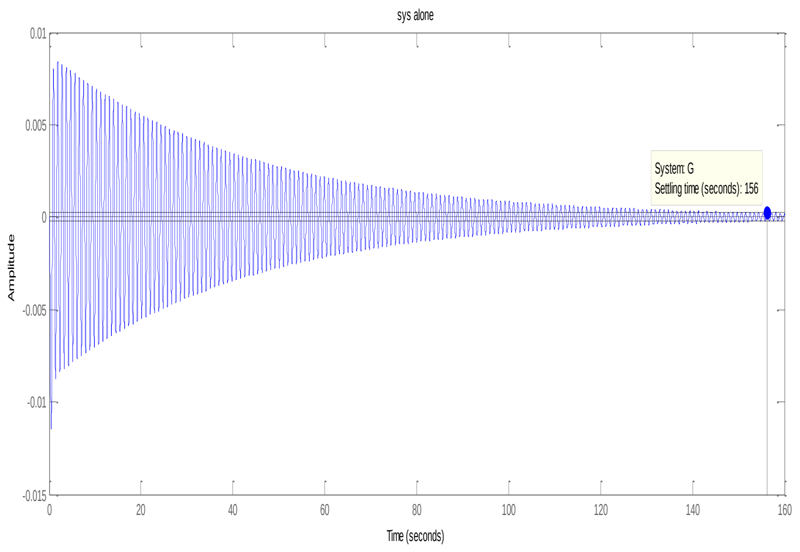

Figure 6. System alone (without PSS) for heavy load regime.

$$TF = \frac{-8.132\ s}{s^4 + 20.46\ s^5 + 98.45\ s^2 + 922.7\ s + 2363}$$

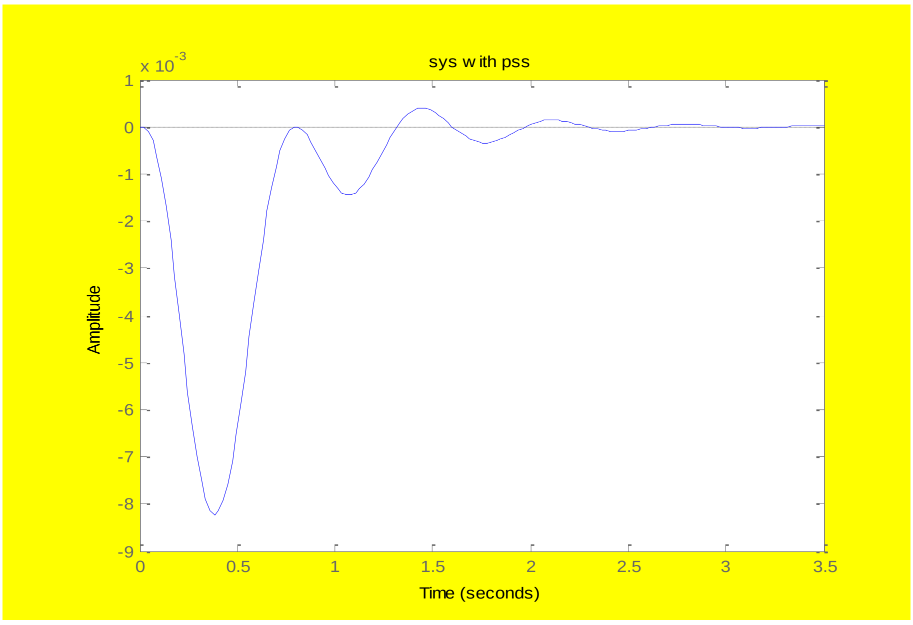

Figure 7. The system with PSS for heavy load regime based on MACO and IACO at ρ=0.5 & 0.2.

$$TF= \frac{-731.9\ s^4 - 1.227*10^4\ s^3 - 5.204*10^4\ s^2 - 5082\ s}{90\ s^7 + 3351\ s^6 + 4.614e04\ s^5 + 4.821*10^5\ s^4 + 3.041*10^6\ s^3 + 1.085*10^7\ s^2 + 1.57*10^7\ s + 1.477*10^6}$$

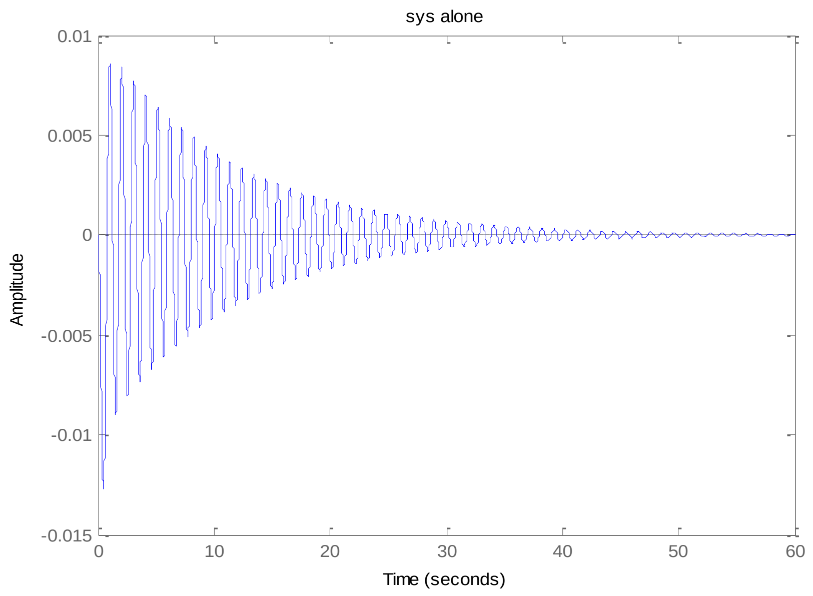

Figure 8. System alone (without PSS) for nominal load regime.

$$TF = \frac{-7.553\, s}{s^4 + 20.46\, s^3 + 89.15\, s^2 + 756.9\, s + 1795}$$

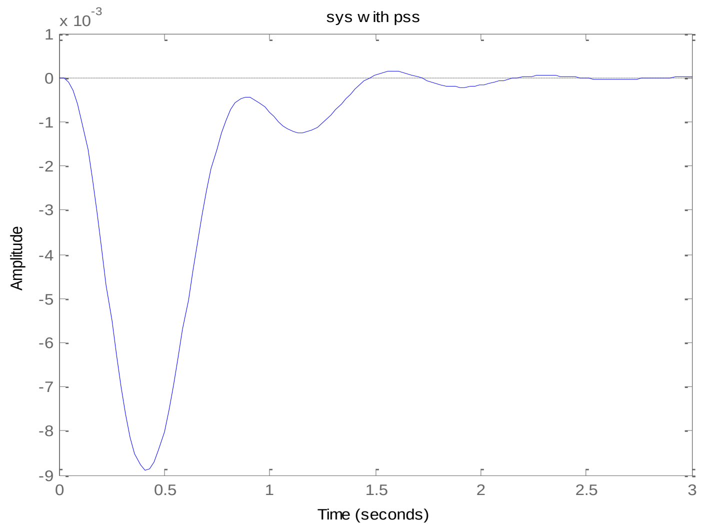

Figure 9. The system with PSS for nominal load regime based on M1CO and IACO at ρ=0.5 & 0.2.

$$TF = \frac{-679.8\, s^4 - 1.14*10^4\, s^3 - 4.834*10^4\, s^2 - 4720\, s}{90\, s^7 + 3351\, s^6 + 4.53*10^4\, s^5 + 4.447*10^5\, s^4 + 2.623*10^6\, s^3 + 8.836*10^6\, s^2 + 1.196*10^7\, s + 1.122*10^6}$$

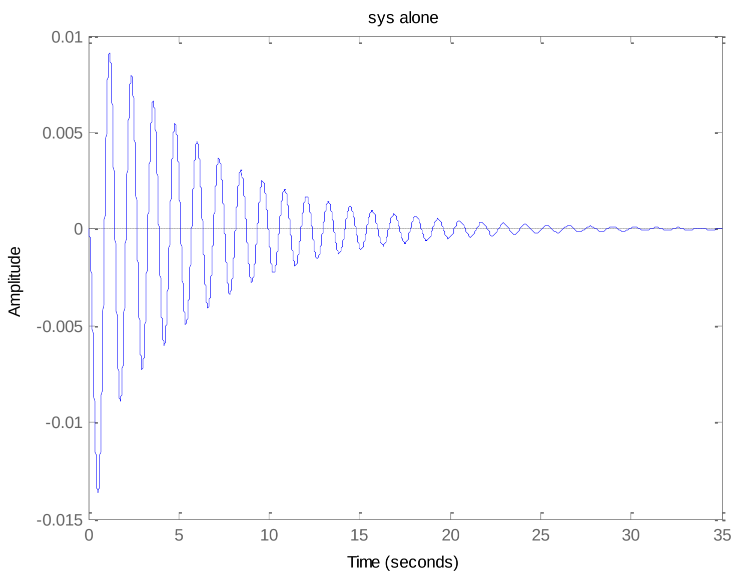

Figure 10. The system alone (without PSS) for light load regime.

$$TF= \frac{-6.326\, s}{s^4 + 20.46\, s^3 + 79.4\, s^2 + 558.4\, s + 1242}$$

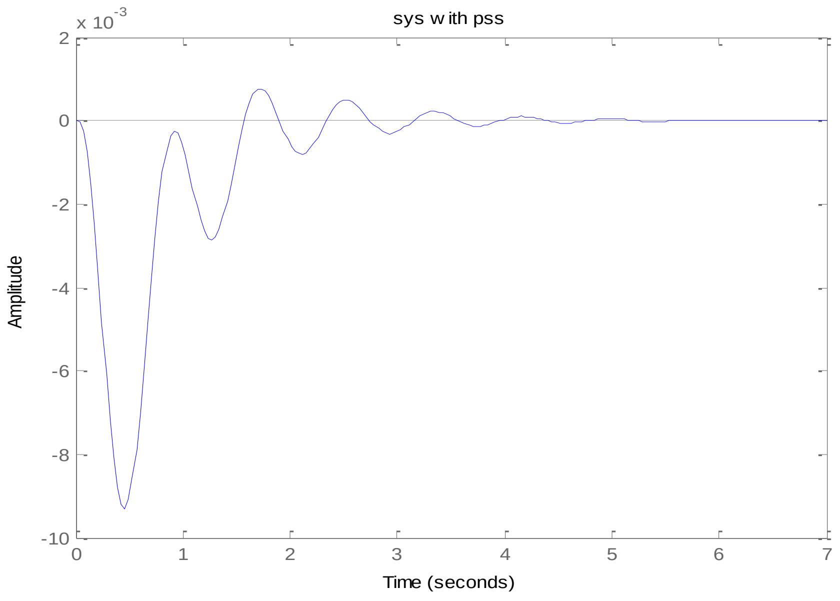

Figure 11. The system with PSS for light load regime based on MACO at ρ=0.5.

$$TF= \frac{-9186\, s^4 - 8.427*10^4\, s^3 - 1.967*10^5\, s^2 - 1.582*10^4\, s}{1452\, s^7 + 4.303*10^4\, s^6 + 4.19*10^5\, s^5 + 3.57*10^6\, s^4 + 1.707*10^7\, s^3 + 4.074*10^7\, s^2 +}$$

$4.001*10^7 s + 3.104*10^6$

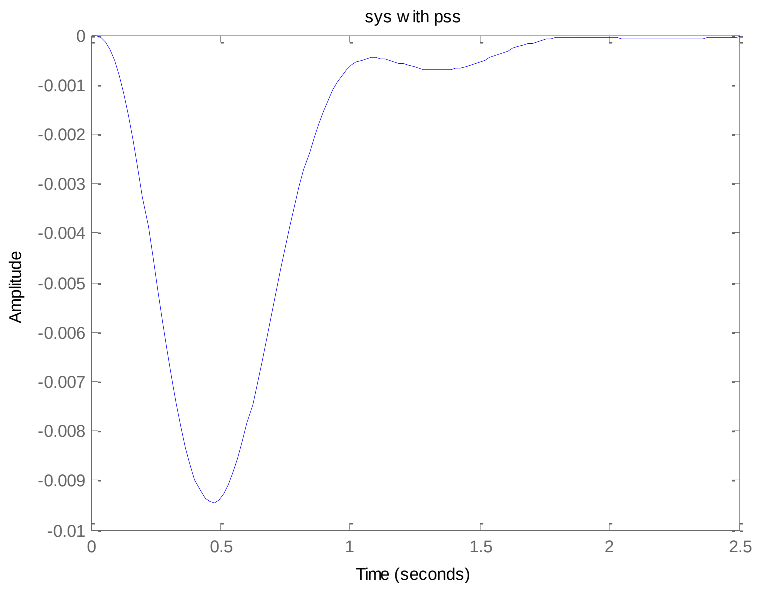

Figure 12. The system with PSS for light load regime based on MACO & IACO at ρ=0.5 & 0.2.

$$TF=\frac{-569.4 s^4 - 9546 s^3 - 4.049*10^4 s^2 - 3954 s}{90 s^7 + 3351 s^6 + 4.442*10^4 s^5 + 3.942*10^5 s^4 + 2.092*10^6 s^3 + 6.525*10^6 s^2 + 8.295*10^6 s + 7.76*10^5}$$

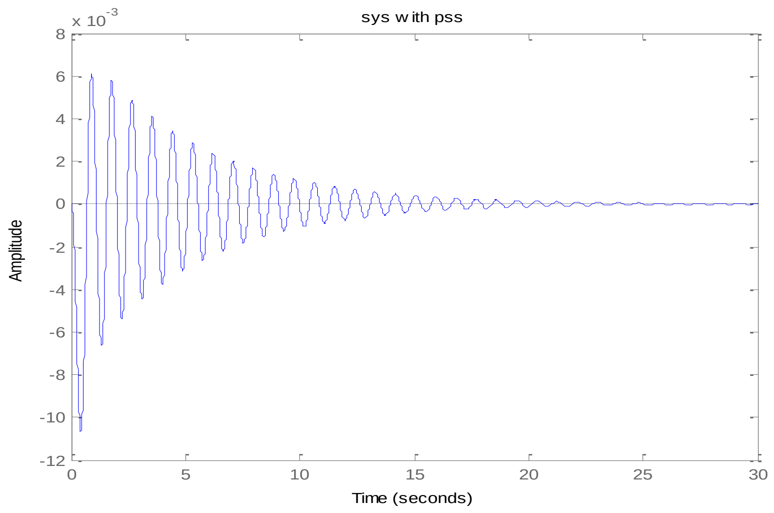

Figure 13. The system with PSS for heavy load regime based on MACO at ρ=0.2.

$$TF=\frac{-2391 s^\wedge 4 - 6.871e04 s^\wedge 3 - 4.993e05 s^2 - 8.132e04 s}{294 s^7 + 1.447*10^4 s^6 + 2.632*10^5 s^5 + 2.434*10^6 s^4 + 1.592*10^7 s^3 + 8.297*10^7 s^2 + 1.543*10^8 s}$$

$+2.363*10^7$

Appendix:

The system A, B, C, and D are **Mahmoud, and Soliman, 2012**.

$$A=\begin{bmatrix} 0 & w_o & 0 & 0 \\ -\frac{k_1}{M} & 0 & -\frac{k_2}{M} & 0 \\ -\frac{k_4}{T'_{do}} & 0 & -\frac{1}{T} & -\frac{1}{T'_{do}} \\ -\frac{k_E k_5}{T_E} & 0 & -\frac{k_E k_6}{T_E} & -\frac{1}{T_E} \end{bmatrix}$$

$$x=\begin{bmatrix}\Delta\delta & \Delta w & \Delta E'_q & \Delta E_{fd}\end{bmatrix}$$

$B=\begin{bmatrix}0\\0\\0\\\frac{k_E}{T_E}\end{bmatrix}$ C=[0 1 0 0], $T=k_3\,T'_{do}$