Date:2021-11-15

Finish the implementation of the Inner loop Algorithm presented in the reference paper.

Inner loop Algorithm

First of all we need to determine the maximum segments before we get,started with the algorithm, according to the piecewise function of EH model in the picuture below. So to get the maximum number of segments, we need to first figure out which segment of Pth which is the threshold power of PiR belongs to, by using the equation below, we can get the maximum value of the received power PiR then we need to compare it to the piecewise model and figureout the which segment it blongs to then we can get the maximum number of segemnts.

So first use the euqation to obtain the mamimum PiR:

1. Scan the Enabled SWIPT D2D links, set each segemnt's maximum value of received power to the array.

2. Then find the ponint where the received power reches the maximum value among each segemnt's maximum value.

3. Then the corresponding maximum value among all of these "maximum value" will be used to compare to the piecewise model by traversing through the whole threshold power array and the maximum segments will be found.

Then we can start to implement the algorithm with my obtained maximum value of segments:Nmax.

```
function [lambda_ij,P_ij,EE_ij]=Inner(EhaD,S_iD)
syms Initial_QiD Initial_P_iD K_intercept B_intercept optimal_lambda
syms Q
syms EC EH EE
syms TD TC T_min_D T_min_C
syms N0 N1
syms P_th PC PD P_ir P_iR Pmax P_th Pmax
syms hD h_k h_C hB
syms a1 a2 a3 K G H J
syms m1 m2 m3 m4 m5 m6 b1 b2 b3 b4 e_ta  XN YN g phi theta P1
syms s1 s2 s3 s4 s5
syms beta alpha gamma delta in
%initialize step size
s1=1.0000e-05;
s2=1.0000e-05;
s3=1.0000e-05;
s4=1.0000e-05;
s5=1.0000e-05;
%initialize EE
EE_ij=[];
```

```
Nmax=1;
%set the maximum value for each segment of received power P_iR
for i=size(EhaD,2)
    for k=1:size(S_iD,2)
        P_iR_max(i,k)=Pmax*hD(i)+PC(k)*h_k(i)+N0;
    end
end
 [argvalue,argmax]=max(P_iR_max(:));

 %traverse the array Pth to find the maximum number of segemnts Nmax
for p=1:size(P_th,2)
    if argvalue>P_th(p)
        Nmax=Nmax+1;
    elseif argvalue<P_th(p)
        break;
    end
end
%initialize the power splitting ratio and transmission power
optimal_lambda=[];

 %traverse the Enabled SWITPT D2D links array,and for each iteration, loop through the partner selection set
 %we obtained from the prematching algorithm, and loop through each segment
 %until it reaches the maximum value for each iteration.
for i=1:size(EhaD,2)
    for k=1:size(S_iD,2)
        for j=1:Nmax
            %obtain optimal power splitting ratiom and find the maximum
            %value of it
            G=PD(i,1) *hD(i);
            H=PC(k)*h_k(i)+N0;
            K=beta+(Q(i,t)+in)*K_intercept(j)*(G+H);
            J=G*N1*log2(exp(1))*(1+gamma);
            a1=K*(G*H+H*H);
            a2=K*(G+2*H)*N1;
            a3=K*N1*N1-J;
            lam1=(-a2+sqrt(a2*a2-4*a1*a3))/(2*a1);
```

```
            %power. And remember this allowed maximum number of iterations
            %should be established later.
            while t<I
                n=[0 1 2];
                m1=(1+gamma)*hD(i)*log2(exp(1));
                m2=PC(k)*h_k(i)+N0+((N1)/(1-optimal_lambda(i,t)));
                m3=Q(i,t)*(1+(1-optimal_lambda(i,t))*e_ta*hD(i)-(K_intercept(j)*hD(i)))+alpha-(in*optimal_lambda(i,t)
                m4=PC(k)*h_C(k)*hB(i)*log2(exp(1))*delta;
                m5=N0+N1;
                m6=PC(k)*h_C(k);
                b1=m3*hD(i)*hB(i)*hB(i);
                b2=(2*m3*m5+m3*m6)*hD(i)*hB(i)-(m1-m2*m3)*hB(i)^2;
                b3=(m3*m5*m5+m3*m5*m6+m4)*hD(i)-(2*m1*m5+m1*m6-2*m2*m3*m5-m2*m3*m6)*hB(i);
                b4=m2*m4+m2*m3*m5*m5+m2*m3*m5*m6-m1*m5*m5-m1*m5*m6;
                XN=(-m2/(3*m1));
                YN=(2*b2^3)/(27*b1^2)-((b2*b3)/(3*b1))+b4;
                g=2*b1*sqrt((b2*b2-3*b1*b3)/(9*b1*b1))^3;
                phi=(1/3)*acos(-YN/g);
                theta=sqrt((m2*m2-3*m1*m3)/(9*m1*m1));
                P1=piecewise(YN^2>g^2,XN+((-YN+sqrt(YN^2-g^2))/(2*a1))^(1/3),YN^2<g^2,XN+2*theta*cos(phi-n*(2*pi)/3),
                PD(i,t+1)=max(0,P1);

                %update the power splitting ratio for next iteration
                G= PD(i,t+1)*hD(i);
                H=P_kc*h_k(i)+N0;
                K=beta+(QD(i,t)+in)*K_intercept(j)*(G+H);
                J=G*N1*log2(exp(1))*(1+gamma);
                a1=K*(G*H+H*H);
                a2=K*(G+2*H)*N1;
                a3=K*N1*N1-J;
                lam1=(-a2+sqrt(a2*a2-4*a1*a3))/(2*a1);
                lam2=(-a2-sqrt(a2*a2-4*a1*a3))/(2*a1);
                lam3=0;
                optimal_lambda_matrix_i=[lam1 lam2 lam3];
                optimal_lambda(i,t+1)=max(optimal_lambda_matrix_i);

                %Then update the new Throughput with new obtained optimal
                %value of transmission power and and power splitting ratio

```

```

                P_iR(i)=PR(optimal_lambda(i,t+1),PD(i,t+1),h_k(i),hD(i),PC(k));
                EH(i)=Eenergy_harvesting(P_iR(i));
                EC(i)=Energy_Consumption(PD(i,t+1),p_ir,EH(i));
                TD(i)=Throughput_D(optimal_lambda(i,t+1),PD(i,t+1),PC(k),h_k(i),hD(i));
                TC(i)=Throughput_C(PD(i,t+1),hB(i),h_C(k),PC(k))
                if TD(i)-(Q(i,t)*EC(i))>Psi
                    Q(i,t+1)=TD(i)/EC(i);
                    %imagine this as an array which has t elements, and the
                    %point is that we need to figure out which is the
                    %biggest one
                    alpha=max(alpha+s1.*(PD(i)));
                    beta=max(beta+s2.*(optimal_lambda(i)-1));
                    gamma=max(gamma-s3.*(TD(i)-T_min_D));
                    delta=max(delta-s4.*(TC(i)-T_min_C));
                    in=max(in-s5.*(P_iR(i)-P_th(2)));
                    continue
                else
                    PD(i,j)=PD(i,t+1);
                    P_ij=PD(i,j);
                    optimal_lambda(i,j)=optimal_lambda(i,t+1);
                    lambda_ij=optimal_lambda(i,j);
                    EE(i,j)=Q(i,t);
                    EE_ij=EE(i,j);
                end
                t=t+1;
            end

        end
    end
end
%Then find the at which point maximum value of the setted Pir_max array

end
```

Some things need to be finished:

- [ ] Establish the allowed maximum number of iterations later
- [ ] Remember to check some of the multiplication should be dot product instead of element wise multiplication.