دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش تمرین دوم (HW2_OPC)

نگارش

جابر بابکی

استاد

دکتر مهدی راستی

فروردین ۱۳۹۸

In this homework the OPC algorithm is applied to a simple single-cell and a two-cell wireless network. The following parameters are fxed for both cases:

- Background noise power 10^-10
- OPC constant 0.05
- Path gain 0.1d^-3

1 Single-Cell Networks

Simulate the system under the above conditions, for 5 number of users and cell radius of 250m. The users should be uniformly distributed in the cell.

```
clc;
close all;
clear all;

NOISE=1e-10;
OPCConstant=0.05;
powerInitiate=rand(1,5)*1e-4;

distnceVector=FuncPosition();%create system model and calculate
distance;
pathGainVector=FuncPathGain(distnceVector);%calculate pathgaine
disp(pathGainVector);

OPC=FuncOPC(OPCConstant,pathGainVector,powerInitiate,NOISE);
FuncFigure(OPC);
```

سیستم را با شرایط خواسته شده ایجاد کردیم در مورد توابع استفاده شده در زیر توضیح می دهم:

تابع تعیین موقعیت FuncPosition:

```
function [ distnceVector ] = FuncPosition()
    clear all;
    BsCOVRAGE=500;%base station coverage area
    Bs1POSITION=[250,250];
    userJoinBs1=ones(5,1);
    for n=1:5
        x=rand(1)*BsCOVRAGE;
        y=rand(1)*BsCOVRAGE;
        if(x==Bs1POSITION(1)&&y==Bs1POSITION(2))
            x=x+1;
            y=y+1;
        end
        userJoinBs1(n,1)=complex(y,x);
    end
    distnceVector=FuncDistance(userJoinBs1,Bs1POSITION);
end
```

در تابع تعیین موقعیت، موقعیت هر کاربر به صورت یکنواخت در محدوده پوشش سلول که شعاع ۲۵۰ هست ایجاد شده است.

تابع محاسبه گین FuncPathGain:

```
function [ pGain ] = FuncPathGain( dis )
    pGain=0.1*(dis.^-3);
end
```

در این تابع براساس رابطه داده شده گین هر کاربر محاسبه می شود.

تابع FuncOPC:

```
function [ OPC ] = FuncOPC(eta,pathGainVector,powerInitiate,NOISE )
    iteration=1;
    finished=0;
    while (iteration <=200 && finished==0)
        for i=1:5
            interference(i)=sum(powerInitiate.*pathGainVector)-
(powerInitiate(i)*pathGainVector(i));

sinrVector(i)=(powerInitiate(i)*pathGainVector(i))/(interference(i)+NOIS
E);
            powerUpdateVector(i)=(eta*sinrVector(i))/powerInitiate(i);
            errorOPC(i)=abs(powerUpdateVector(i)-powerInitiate(i));
            sinrVectorOPC(iteration,i)= sinrVector(i);
            powerVectorOPC(iteration,i)=powerInitiate(i);
            powerInitiate(i)=powerUpdateVector(i);
        end
        temp=errorOPC<(10^-6);
        if sum(temp)==5
            finished=1;
        else
            iteration=iteration+1;
        end
    end
    OPC{1}=powerVectorOPC;
    OPC{2}=sinrVectorOPC;
    OPC{3}=iteration;
end
```

در این تابع در واقع الگوریتم OPC اجرا می شود، برای اجرای این الگوریتم نیاز به این داریم که در بازه تکراری الگوریتم اجرا و توان اپدیت شود به همین جهت در حلقه while با تکرار ۲۰۰ قرار داده شده البته، البته مقدار

خطایی هم جهت پایان برنامه در نظر گرفته شده است. در هر با اجرا ما باید توان کاربران را تعیین کنیم به همین جه نیاز به محاسبه تداخل و SNRکاربران داریم که در داخل حلقه For محاسبه می شود و در نهایت power وSNR را به عنوان خروجی برمی گرداند.

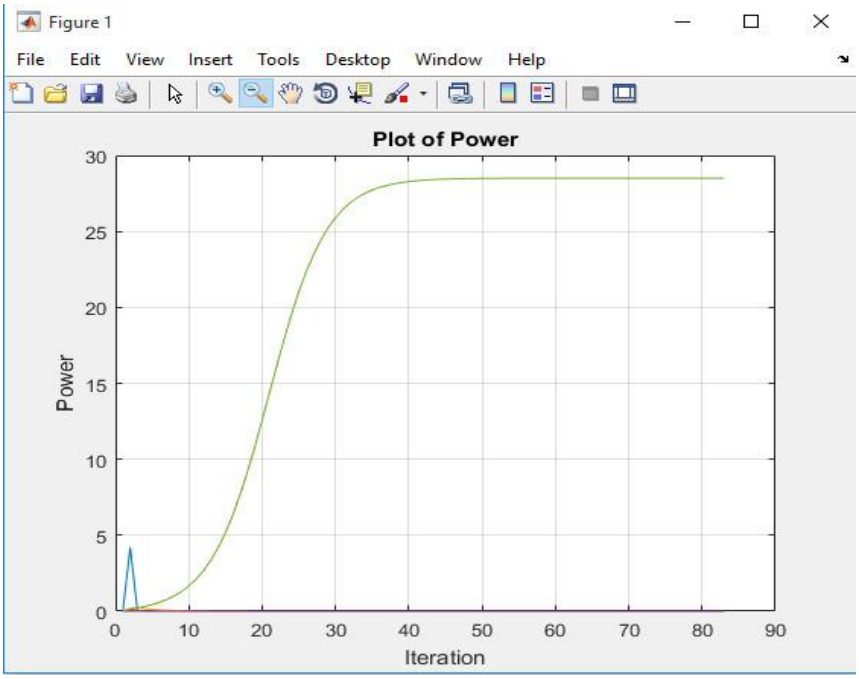
تابع رسم نمودارد FuncFigure:

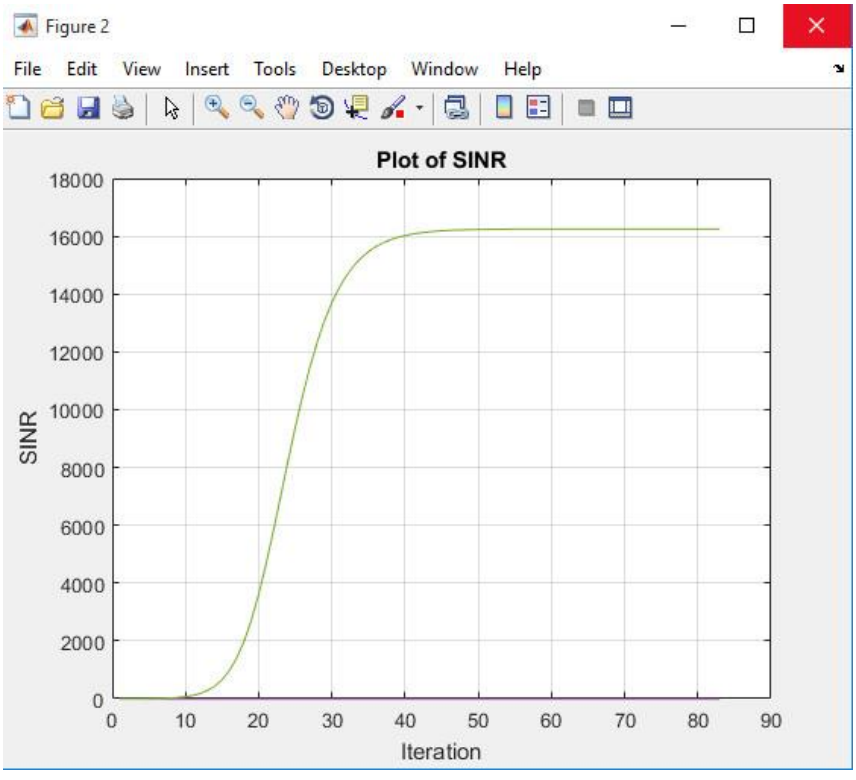
```
function FuncFigure( OPC )
    Power=OPC{1};
    gamma=OPC{2};
    j=OPC{3};
    x=1:(j);
    figure(1)
    plot(x,Power),grid on;

    xlabel('Iteration')
    ylabel('Power')
    title('Plot of Power')
    figure(2)
    plot(x,gamma),grid on;
    xlabel('Iteration')
    ylabel('SINR')
    title('Plot of SINR')
end
```

در نهایت powerو SNR را در نمودار نمایش می دهد.

1-1 Plot SINR and power of each user versus the number of iterations (a measure of time).





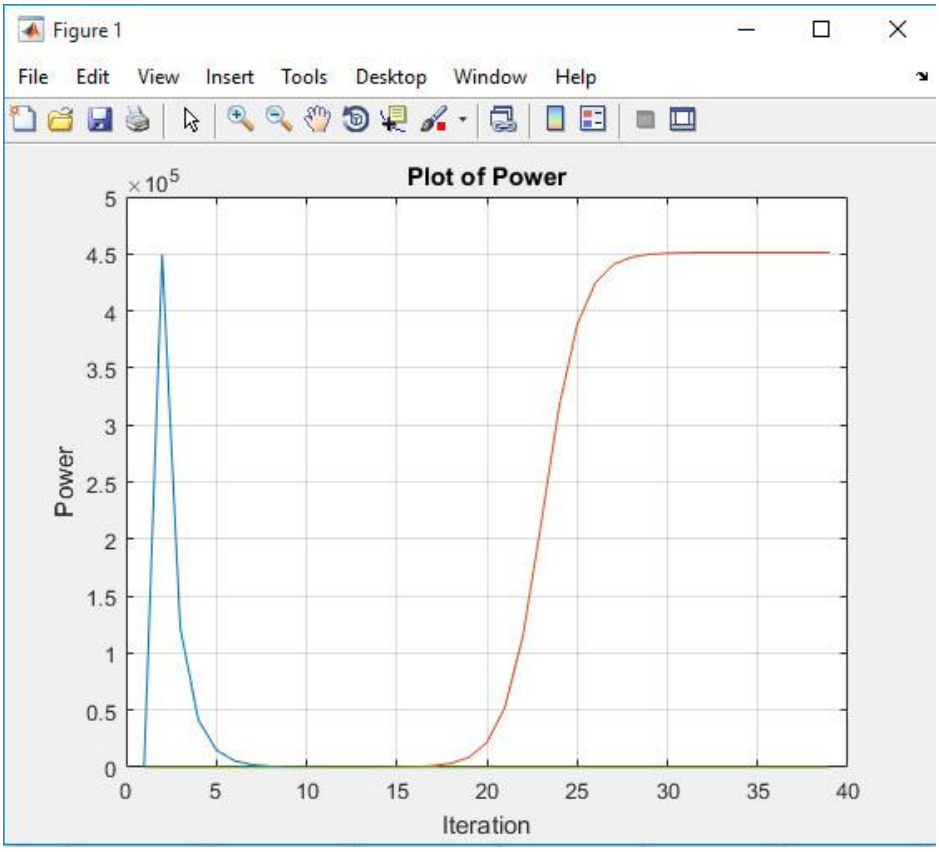
همان طور که در شکل نیز مشخص است، کاربری که نمودار آن به رنگ سبز می باشد، حداکثر توان ارسالی را دارد. در OPC جهت حداکثر کردن مجموع گذردهی فقط یک کاربر در توان max به ارسال

داده می پردازد و همین کاربر به رنگ سبز است که میتواند به SINR قابل خود هم برسد. اما مقدار این توان و SINR به مقدار توان اولیه وابسته نمی باشد. چون در این شبیه سازی توان اولیه به صورت رندوم ایجاد شده است اما باز هم کاربر سبز در توان max بوده است. در OPC مقدار توان ارسالی هر کاربر به وضعیت کانال و تداخل ها وابسته می باشد. هر چه وضعیت کانال کاربر بهتر باشد، سریع تر می تواند به حداکثر توان ارسالی مربوط به خود برسد. توان سایر کاربران نیز به مرور زمان به min مقدار خود میل می کند.

1-2 Which users transmit at high power levels? Does it depend on initial transmit power vector?

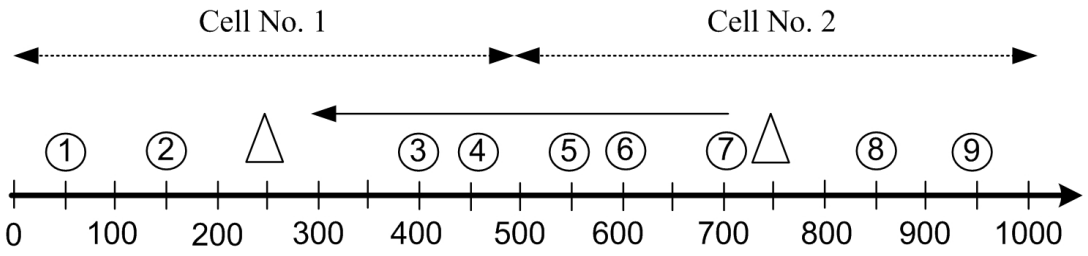
بستگی به توان اولیه ندارد، چون الگوریتم توزیعشده است و توان را بر اساس توان لحظه قبل که تابعی از تداخل بر روی گین است تنظیم می کند. کاربری که وضعیت کانال کاربر بهتری داشته باشد، سریع تر می تواند به حداکثر توان ارسالی مربوط به خود برسد. توان سایر کاربران نیز به مرور زمان به min مقدار خود میل می کند. در شکل مشاهده می شود که کاربر ابی رنگ ابتدا توان خوبی داشته اما توان اولیه نمی تواند تضمین باشد چون در اپدیت های بعدی توان واقعی هر کاربر مشخص می شود.

1-3 Increase or decrease the OPC constant and see its impact on performance of the OPC algorithm.



بر اساس مشاهدات انجام شده پارامتر eta به دلیل وجود در صورت کسر باعث می شود در ابتدا توان کاربران زیاد شود ولی به مرور زمان به دلیل بد شدن وضعیت کانالی توان ارسالی کاربران کاهش می یابد. پس اگر مقدار eta بالا باشد کاربران در توان بالا ابتدا ارسال دارند و اگر مقدار eta کم باشد در توان پایین ارسال دارند در واقع eta میزان فرورفتگی را نشان می دهد.

2-



Distribution of users and base stations in a two-cell wireless network. Users are marked by “O” , and base stations are marked by “^”. Users 1 to

6, 8 and 9 are _xed, and user 7 at t = 0 starts moving from the starting-point x = 700 m in cell No. 2 towards the end-point x = 300 in cell No. 1 along the illustrated line at a uniform speed of 20 m/s (72 km/h).

برای حل این مسئله از کد های زیر استفاده کردم:

تابع اصلی main:

```
clc;
close all;
clear all;

NOISE=10e-10;
OPCConstant=0.05;
IterNumber=1000;
powerInitiate=zeros(IterNumber,9);
powerInitiate(1,:)=rand(1,9)*1e-4;
Bs1POSITION=[250,250];
Bs2POSITION=[750,250];

positionVector=FuncPosition();%create system model and calculate
distance;

for time = 1:20000
    positionVector{2}(3,1) = complex(700 - (20*time/1000),250);

distnceVector=FuncDistance(positionVector{1},positionVector{2},Bs1POSITI
ON,Bs2POSITION);
    pathGainVector=FuncPathGain(distnceVector);
    if(time >= 10000)
        g5=pathGainVector(7,1);
        g6=pathGainVector(7,2);
        g7=pathGainVector(7,3);
        g8=pathGainVector(7,4);
        pathGainVector(7,1)=pathGainVector(7,5);
        pathGainVector(7,2)=pathGainVector(7,6);
        pathGainVector(7,3)=pathGainVector(7,7);
        pathGainVector(7,4)=pathGainVector(7,8);
        pathGainVector(7,5)=g5;
        pathGainVector(7,6)=g6;
        pathGainVector(7,7)=g7;
        pathGainVector(7,8)=g7;
    end
    OPC=FuncOPC(pathGainVector,powerInitiate(1,:),NOISE,OPCConstant);
    powerInitiate(time,:) = OPC{1};
    SIR(time,:) = OPC{2};
end
FuncFigure(powerInitiate,SIR);
```

کارکرد کلی کد به این صورت هست که بر اساس پارامتر های و موقعیت های کاربران که به صورت ثابت تعریف شده است سیستم را اجاد کردیم، سپس در یک حلقه به تکرار ۲۰۰۰۰ که این عدد را به صورت دستی جوری قرار دادم تا طبق گفته مسئله بعد از تکرار ها کاربر ۷ به نقطه ۳۰۰ برسد. در هر حلقه هر بار مسافتی با مقداری برابر با ۲۰ متر بر ثانیه را از کاربر ۷ کم میکنیم و سپس الگوریتم OPC را با تکرار ۱۰۰۰ اجرا میکنیم و خواهیم دید که الگوریتم به سرعت اپدیت می شود.

تابع تعیین موقعیت FuncPosition:

```
function [ distnceVector ] = FuncPosition()
    clear all;
    BsCOVRAGE=500;%base station coverage area
    Bs1POSITION=[250,250];
    Bs2POSITION=[750,250];

    userJoinBs1=ones(4,1);
    userJoinBs2=ones(5,1);

    userJoinBs1(1,1)=complex(50,250);
    userJoinBs1(2,1)=complex(150,250);
    userJoinBs1(3,1)=complex(400,250);
    userJoinBs1(4,1)=complex(450,250);

    userJoinBs2(1,1)=complex(550,250);
    userJoinBs2(2,1)=complex(600,250);
    userJoinBs2(3,1)=complex(700,250);
    userJoinBs2(4,1)=complex(850,250);
    userJoinBs2(5,1)=complex(950,250);

    distnceVector{1}=userJoinBs1;
    distnceVector{2}=userJoinBs2;

%distnceVector=FuncDistance(userJoinBs1,userJoinBs2,Bs1POSITION,Bs2POSIT
ION);
end
```

بر اساس تمرین موقعیت ها ثابت داده شده است .

تابع تعیین فاصله FuncDistance:

```
function [ dis ] = FuncDistance(
userJoinBs1,userJoinBs2,Bs1POSITION,Bs2POSITION )

    A=complex(Bs1POSITION(1),Bs1POSITION(2));
    B=complex(Bs2POSITION(1),Bs2POSITION(2));
    for s=1:4
        for m=1:4
            temp=userJoinBs1(s,1)- A;
            dis(s,m)=abs(temp);
        end
```

```
        for n=5:9
            temp=userJoinBs1(s,1)- B;
            dis(s,n)=abs(temp);
        end
    end

    for s=1:5
        for m=1:4
            temp=userJoinBs2(s,1)- A;
            dis((s+4),m)=abs(temp);
        end
        for n=5:9
            temp=userJoinBs2(s,1)- B;
            dis((s+4),n)=abs(temp);
        end
    end

end
```

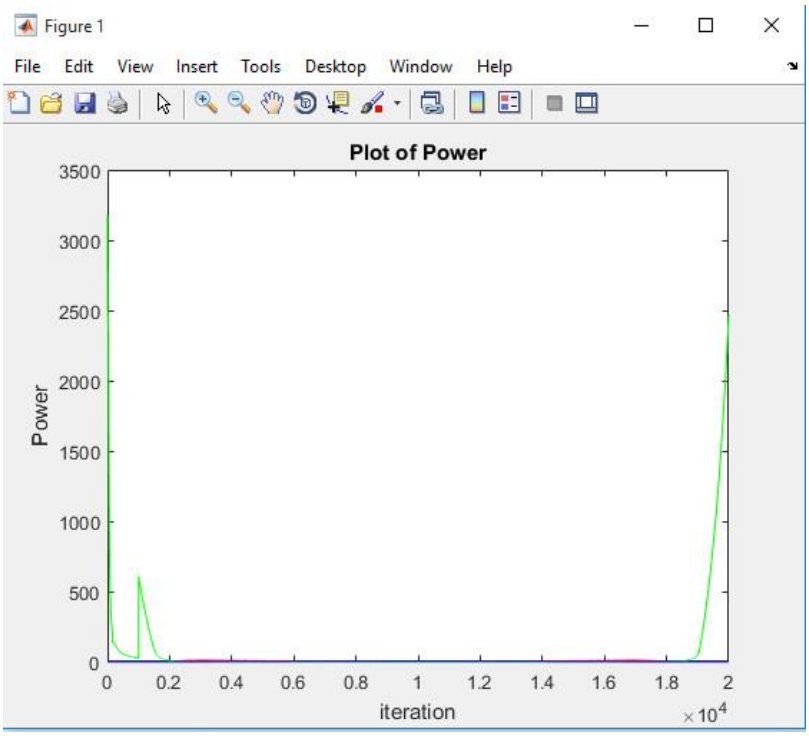
تابع محاسبه PathGain:

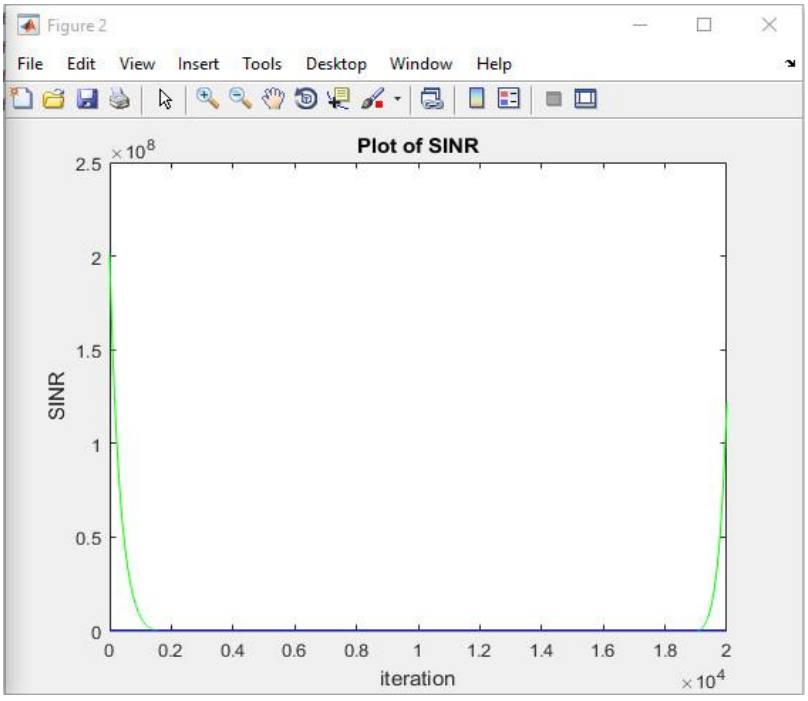
```
function [ pGain ] = FuncPathGain( dis )
    pGain=0.1*(dis.^-3);
end
```

تابع محاسبه SNR:

```
function [ gamma ] = FuncSINR( pathGainVector,randomPower,NOISE )
for i=1:9
    t=0;
    for j=1:9
        if (i~=j)
             t=t+pathGainVector(j,i)*randomPower(j);
        end
    end
    t=t+NOISE;
    gamma(i)=(randomPower(i)*pathGainVector(i,i))/(t);
end
end
```

1-1Plot the transmit-power levels and the received SIRs versus time for users 2, 7, and 8.





Command Window

```
    0.0108    0.0391    0.0176    0.0109    0.0031    0.0103    0.1264    0.0583    0.0090

    0.0108    0.0391    0.0176    0.0109    0.0031    0.0103    0.0761    0.0583    0.0090
```

2-1 Discuss and interpret the results.

برای توضیح نمودارهای مربوط به این قسمت تمرین، می توان این طور تفسیر کرد که کاربر ۷ که در ابتدا در سلول دوم قرار دارد به دلیل نزدیکی به BS متناظر خود که در موقعیت ۷۵۰ قرار دارد، ارسال خود را با حداکثر توان خود انجام می دهد. اما با شروع به حرکت هرچقدر از BS خود دورتر می شود، میزان توان ارسالی آن کاهش می یابد. که این مسئله در شکل نمودار توان نیز قابل مشاهده هست. درحقیقت با دور شدن کاربر ۷ ازBS متناظر خود در سلول دوم، PathGain این کاربر بدتر شده و کاربر ۷ جز کاربران ضعیف به حساب می آید و توان ارسالی کمی نیز پیدا می کند. درواقع موقعه ای که کاربر ۷ در موقعیت ۵۰۰ قرار میگیرد بعدترین شرایط حاکم را دارد چون نسبت به BS ها دور هست واین مسئله در نموردا توان وSINR کاملا مشاهده می شود. با ورود این کاربر به سلول ۱ و در ابتدای ورود هنوز به دلیل فاصله با BSمتناظر در سلول اول، جز کاربران ضعیف سلول اول هست. اما هر چقدر به حرکت خود ادامه می دهد و به BSسلول اول که در موقعیت ۲۵۰ قرار دارد نزدیکتر می شود، PathGain آن بهتر خواهد شد و در نتیجه می تواند با توان بیشتری ارسال خود را انجام دهد و به نوعی جز کاربران خوب سلول اول به حساب بیاید.

شکل اول مربوط به power نیز همین موضوع را اثبات می کند زیرا در ابتدا توان یوزر ۷ توان مناسبی است و بعد به مرور زمان توان کاهش می یابد و می بینیم که شکل نمودار نزولی می شود و سپس بعد از ورود به سلول اول و ادامه مسیر خود و نزدیکتر شدن به BS موجود در سلول اول این توان افزایش می یابد. در نمودار نیز پس از نزول، نمودار دوباره حالت صعودی پیدا کرده است که نشان دهنده همین مطلب است. نمودار مربوط بهSINR نیز این موضوع را اثبات می کند.

شکل سوم مقدار SNR را نشان می دهد که در گام های کاربر ۷ مقدار SINR اش کم شده است.