NTE NTE Cellular Networking

Michele Luca Puzzo 1783133

April 2022

Exercise 4

Consider a linear array of cells.

- Initially all cells are idle (state $= 0$).
- In each iteration one cell is selected at random among all idle cells (say cell C)
- The selected cell C becomes active, by taking the channel (state $= 1$). Activation is only possible, if neighboring cells of C are NOT active.
- Once C becomes active, its neighboring cells are forced to the blocked state (state $= -1$).
- Repeat iterations until all cells of the array are either active or blocked.

1. Find the average fraction ϕ_{rnd} of cells that become active.
2. Find the maximum values of ϕ that can be achieved with a centralized assignment.

1 Exercise 4

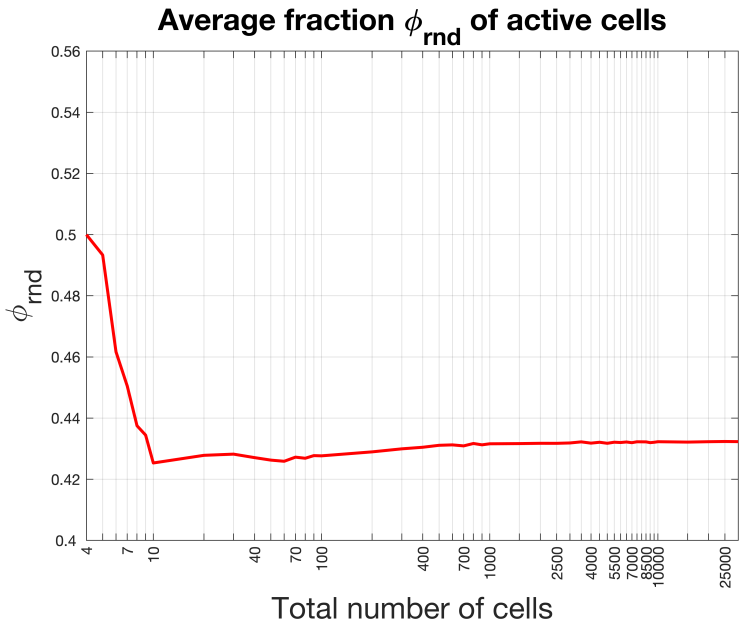
1.1 Question 1

```
%Start timer
tic

%{
    Each element of vector M represents the total number of cell N in the linear array
    for which I have wanted to compute the average number of active cell.
    Since I do not want a vector evenly spaced I have built different vectors
    and then I concatenate them.
%}

M1 = [4:1:10];
M2 = [20:10:100];
M3 = [200:100:1000];
M4 = [1500:500:10000];
M5 = [15000:5000:30000];
M = cat(2, M1, M2, M3, M4, M5);

%number of simulations for each total number of cell N
nsim = 300;

%Risultato will contain the average fraction of active cell for each N
risultato = zeros(length(M), 1);

%for each N
for j = 1:length(M)
    num = M(j);

    %frazione will contain nsim average fractions of active cell for a
    %particular N
    frazione = zeros(1,nsim);

    %for each simulation
    for i = 1:nsim

        %initializing the linear array of cells with all zeros
        lista = zeros(num, 1);

    %'all' function returns 1 when all the elements are nonzero
    while all(lista) ~= 1

        %idle contains the index of idle cell
        idle = find(lista == 0);

        %pick at random an idle cell
        idx = randi([1,length(idle)]);
        elem = idle(idx);

        %the selected cell becomes active
        lista(elem) = 1;

        %{
        The neighbouring are forced to be blocked.
        Cells at the border have just one neighbour
```

```
        for this reason there are these two if (last cell has not a right
        neighbour while left cell has not the left one).
        %}
        if elem ~= M
            lista(elem + 1) = -1;
        end
        if elem ~= 1
            lista(elem - 1) = -1; %#ok<NASGU>
        end
    end

    %to compute the fraction I count the elements equal to one and divide
    %for the length of the array
    frazione(i) = sum(lista(:) == 1)/num;

    end

    %I compute the mean among all the simulations for a given N
    risultato(j) = mean(frazione);
end
risultato;

%stop timer
toc
```

```
Elapsed time is 1956.972576 seconds.
```

```
%I have choosen to use a logarithmic scale on the x-axis and
% a linear scale on the y-axis.
semilogx(M, risultato, "LineWidth",2, "Color", "red")
grid()
xlabel("Total number of cells",'FontSize',20)
ylabel('\phi_{rnd}','FontSize',20)
title('Average fraction \phi_{rnd} of active cells','FontSize',20)
%To have clear and readable xticks I have used this trick (from
%StackOverflow)
xTickLabels = zeros(1,numel(M));% zero cell array the same length as M
xTickLabels(1:3:numel(M)) = M(1:3:numel(M));
xTickLabels = cat(2,string(xTickLabels), ["0" "30000"]);
xTickLabels(xTickLabels=="0")="";
set(gca,'XLim',[4, 30000], 'XTick', M, 'XTickLabel', xTickLabels)
set(gca,'YLim',[0.4, 0.56],'YTickLabel',[0.4:0.02:0.6])
```


Average fraction ϕ_{rnd} of active cells
0.56
0.54
0.52
0.5
0.48
0.46
0.44
0.42
0.4
ϕ_{rnd}
4
7
10
40
70
100
400
700
1000
2500
4000
5500
7000
8500
10000
25000
Total number of cells

So as expected I have a transient when the linear array of cells is short (less than 20 circa) and the fraction of active cells changes quickly. But after that this value stabilizes itself around 0.432 and it does not change anymore. For each length of the linear array I have run 300 simulations and then take the average of the obtained values. I have simulated for length of the linear array up to 30000, but I could stop even to shorter array because there is a steady state.

1.2 Question 2

The maximum value of ϕ_{max} that can be achieved is $\mathbf{\frac{1}{2}}$, in the table below I show why:

0	0	0	0	0	-1	1	-1	0	0	0	0	0
1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1

After choosing the first cell at random, I have the cells that I have highlighted in green that are the optimal choice to have the less number of cells in the blocked state. Indeed in this way the new active cell blocks just one more neighbour and not two. The final result is that active and blocked cells alternate themselves and so I have an active cell every two cells.

Instead the worst possible value of ϕ_{min} is $\mathbf{\frac{1}{3}}$, in the table below I show why:

0	0	0	0	0	-1	1	-1	0	0	0	0	0
1	-1	-1	1	-1	-1	1	-1	-1	1	-1	-1	1

After choosing the first cell at random, I have the cells that I have highlighted in pink that are the worst choice because each cell blocks two other cells and the final result is that I have an active cell every three cells.

Indeed the value ϕ_{rnd} is included between these two values obtained through the centralized assignment.

1.3 Trying to explain simulation results

If I choose alternately active cell in position 'green' and 'pink' I obtained a ϕ_{middle} equals to $\frac{2}{5}$ (two active cells every five cells)

I have found that the ϕ_{rnd} as the weighted average of three situations,
$\phi_{rnd} = 0.5\phi_{max} + 0.25\phi_{min} + 0.25\phi_{middle} = 0.5 \cdot \frac{1}{2} + 0.25 \cdot \frac{1}{3} + 0.25 \cdot \frac{2}{5} = 0.433$
The weight of the ϕ_{max} is the double of the other two weights because it can happen the following situation:

-1	1	-1	0	0	0	0
-1	1	-1	0	-1	1	-1

The fourth cell is forced to be active whenever I choose the green cell as active. So it is like choosing optimal active position helps in keep on choosing optimal active position. On the other hand this does not happen when I choose worst (pink) position for active cells because I could be an active cell in an optimal or not position with the same probability.