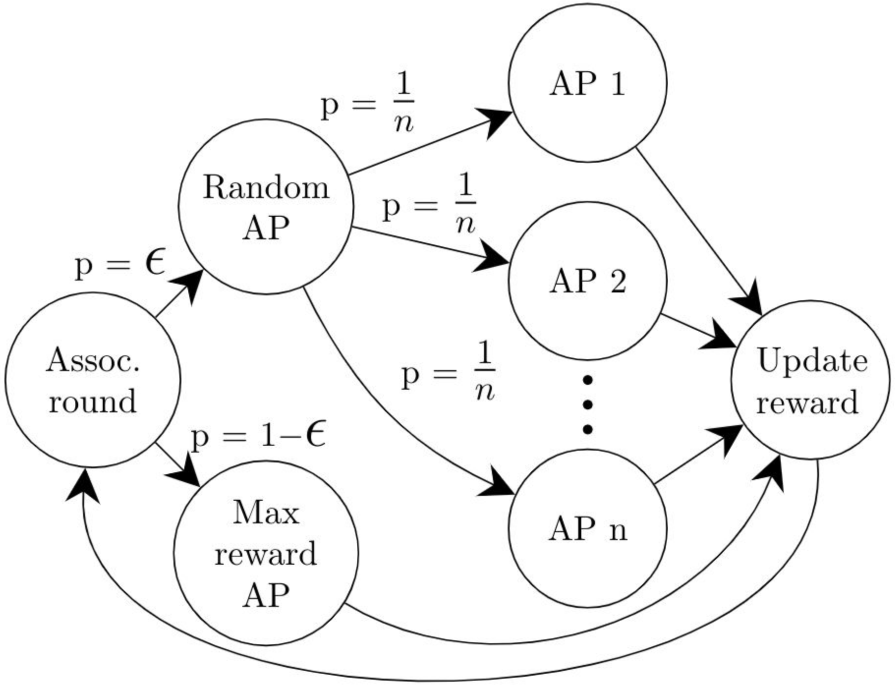
Machine Learning for Networking

Lab 2

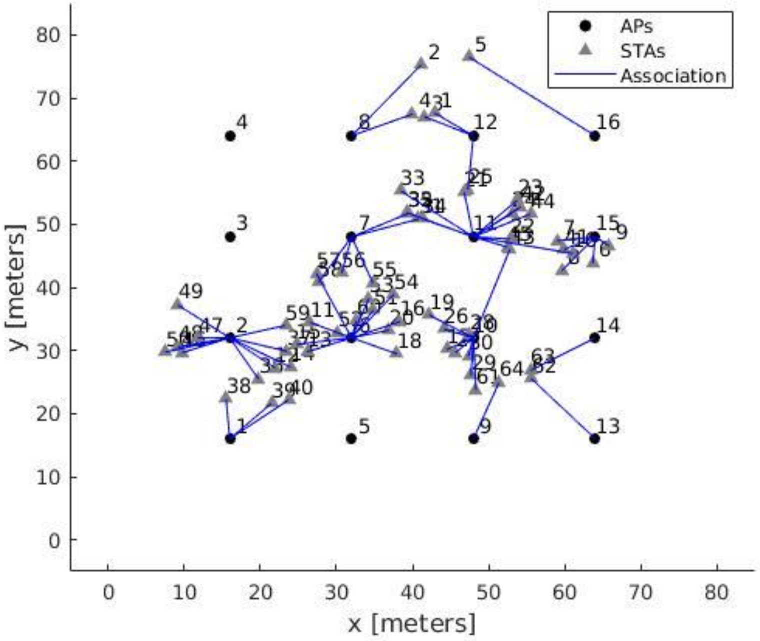
Marc Carrascosa Zamacois: marc.carrascosa@upf.edu

ε-greedy



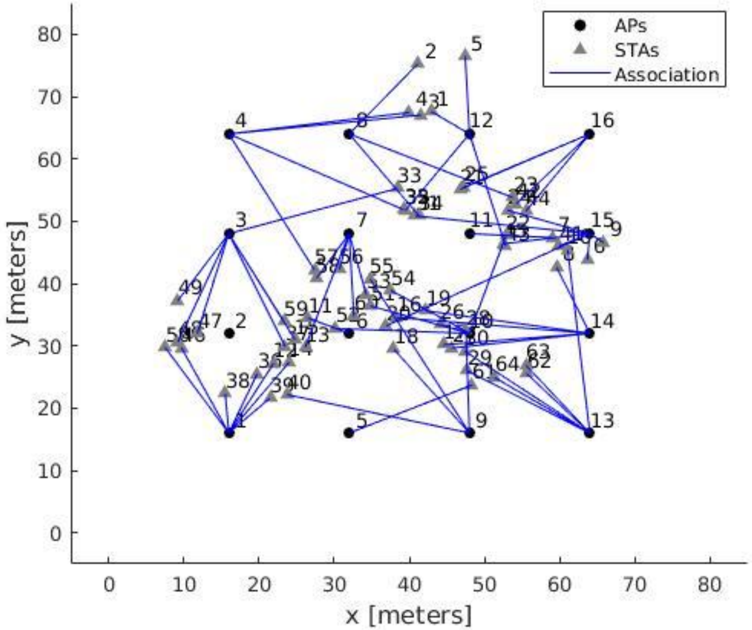
- We adapt ε-greedy to our AP selection problem
- Each STA has an ε-greedy agent
- Each AP in range is an arm for the agent
- We use the throughput achieved as a reward

Using ε-greedy



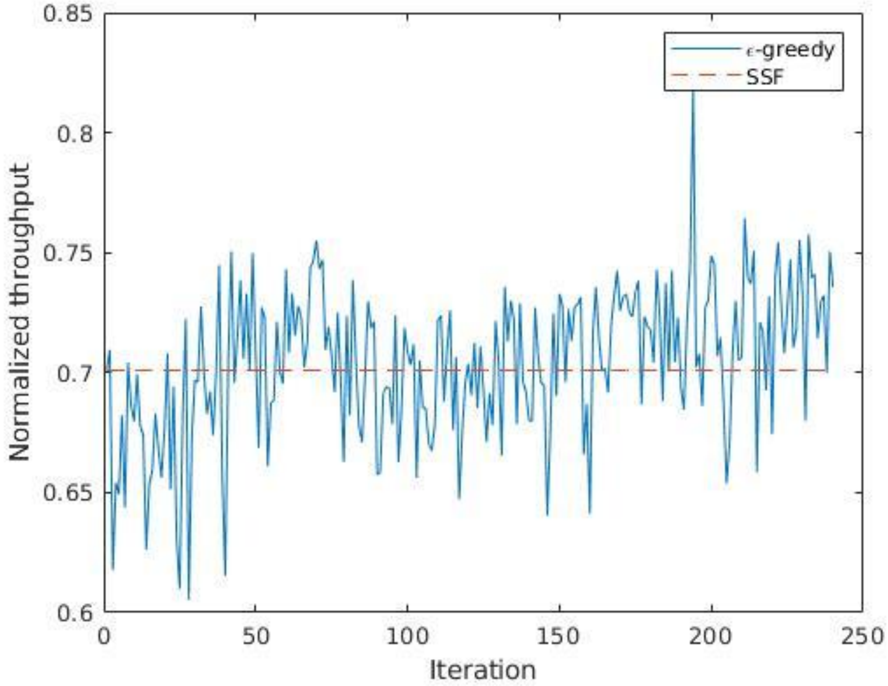
- We create a topology and use the default association
  - STAs select AP based on higher RSSI
- Note APs 3, 4 and 5

Using ε-greedy



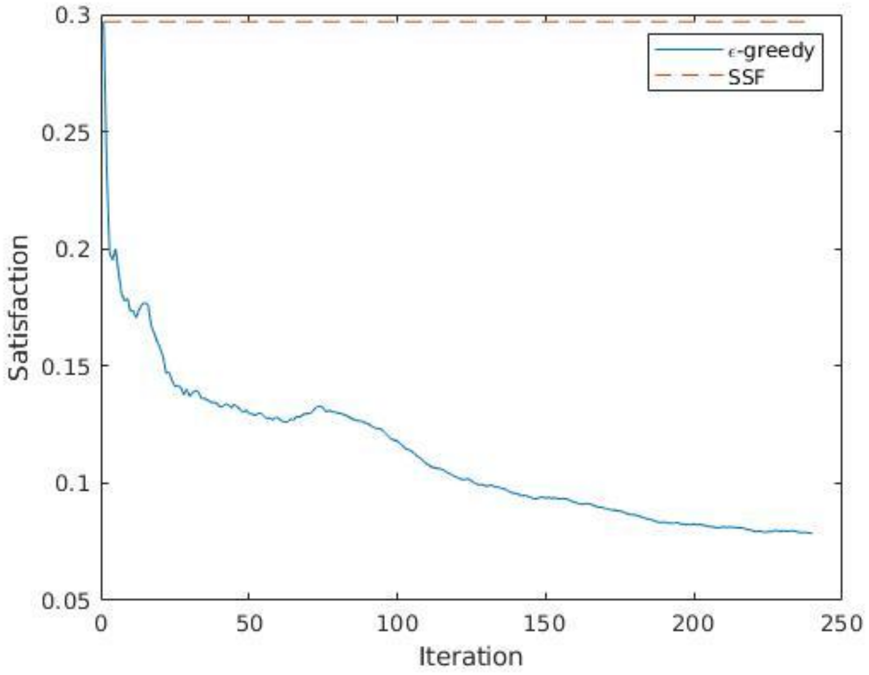
- We then use ε-greedy to select a new AP every time interval *t*
- For every time interval, the STA selects a new AP based on its own metrics
- We then recalculate the network performance with the new configuration
- Then we select a new AP once again
- After 240 intervals we achieve this result
- Now APs 3, 4 and 5 have STAs associated, and other APs relieve heavy areas like that of AP 6

Using ε-greedy



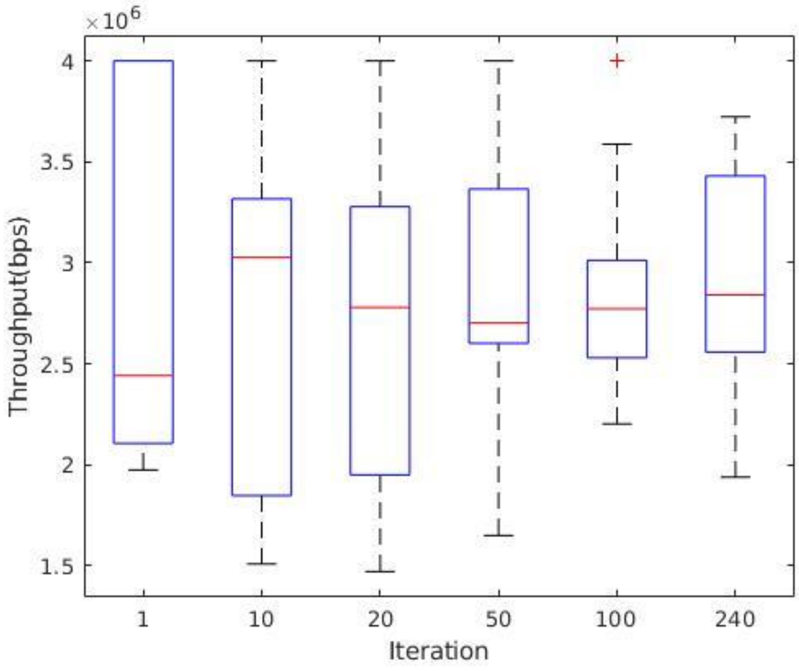
- SSF = Strongest Signal First
- ε-greedy keeps learning and obtains better throughput over time
- Since the first decisions are made without data, initially we can expect randomness (in this case a worse performance)

Using ε-greedy



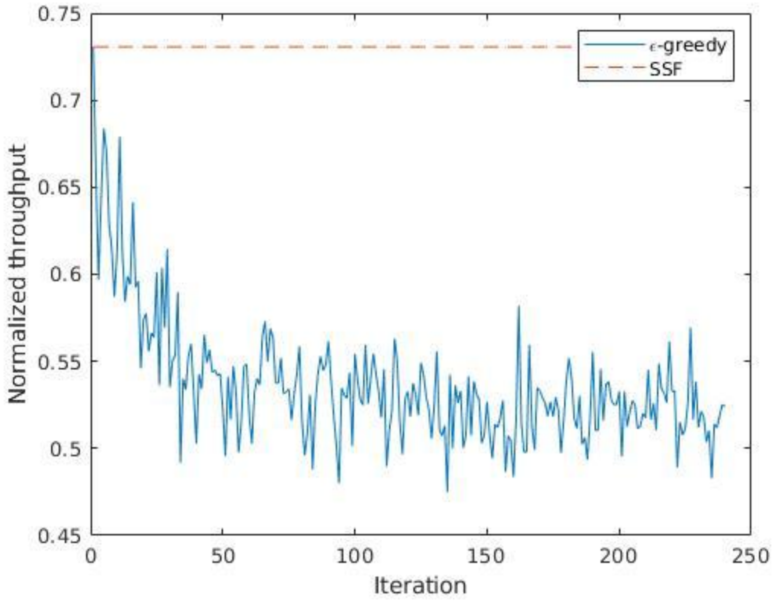
- Satisfaction is binary
  - 1 if STA gets all its load
  - 0 otherwise
- A higher throughput does not necessarily mean higher satisfaction
- We use throughput as a reward, so it takes precedence

Using ε-greedy

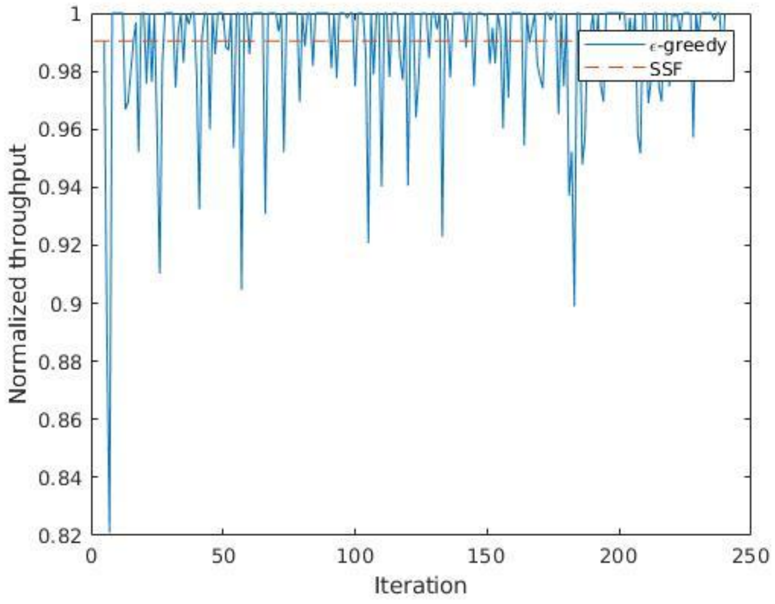


- Iteration 1 is using strongest signal association, others use ε-greedy
- The range of the throughput decreases over time
- But median and minimum increase

Scenarios matter

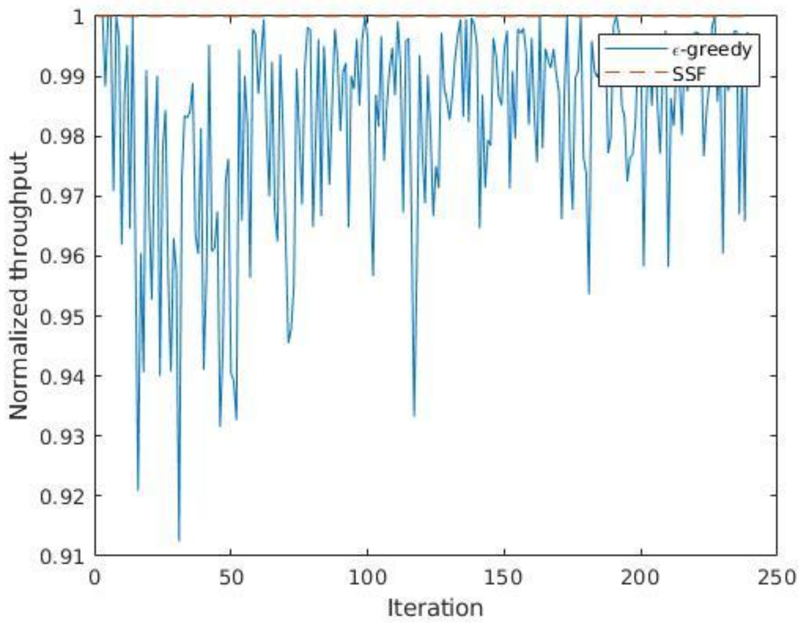


16 APs, 100 STAs, 4 Mbps

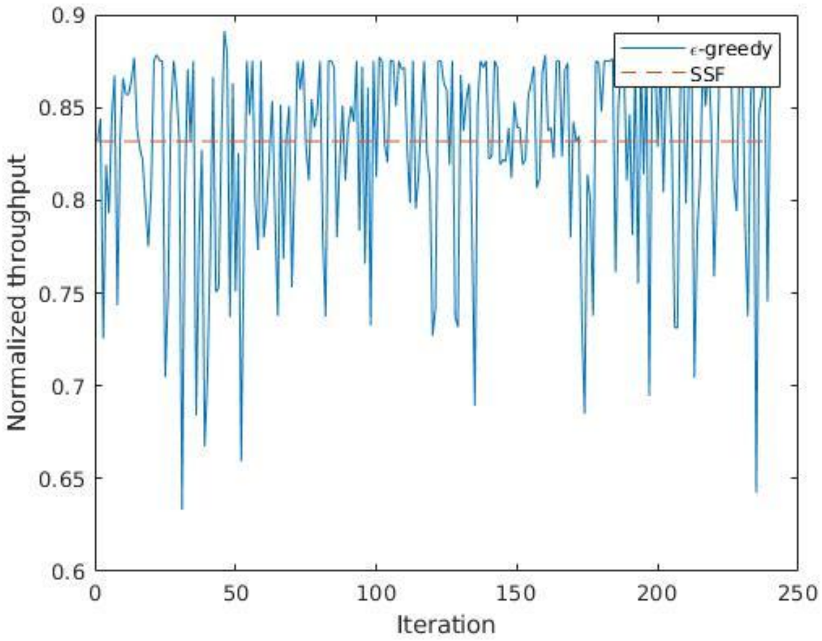


16 APs, 24 STAs, 4 Mbps

Scenarios matter

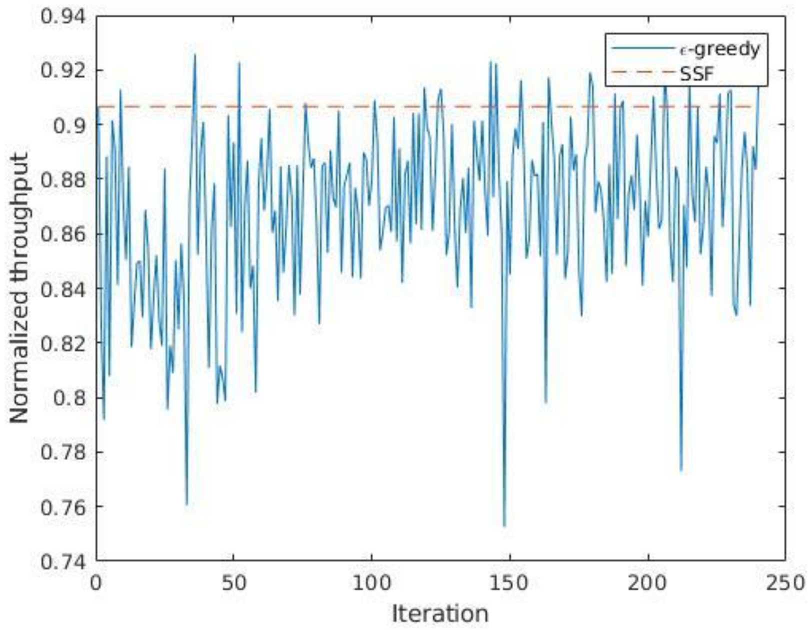


16 APs, 100 STAs, 1.5 Mbps

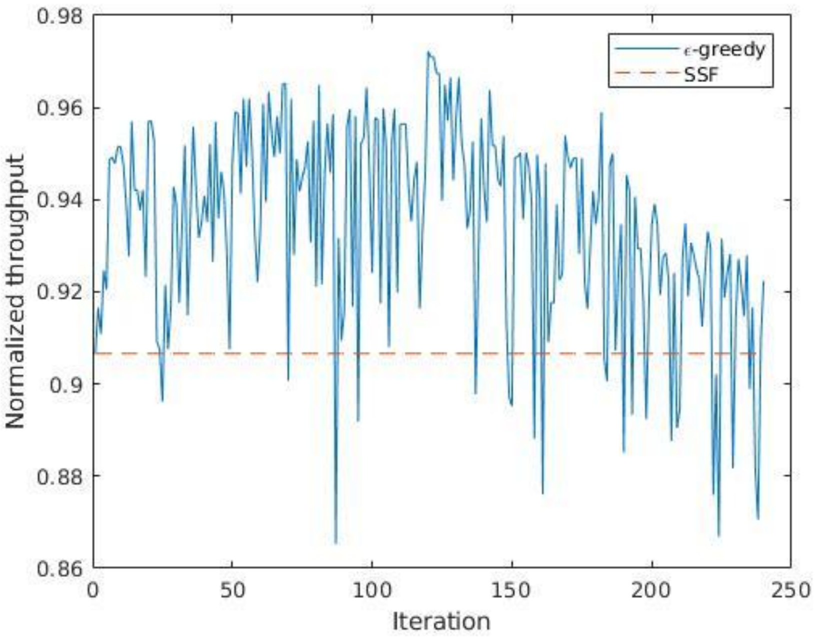


4 APs, 30 STAs, 4 Mbps

Scenarios matter



32 APs, 100 STAs, 4 Mbps, e = 0.1



32 APs, 100 STAs, 4 Mbps, e = 0.02

Tuning the algorithm

- Some cases can be improved by changing the epsilon parameter
- Some cases can be so unbalanced that exploring only makes it worse (Strongest Signal can be the best solution)
- Sometimes there is nothing to improve, so exploring just wastes time
- Sometimes we find a solution in iteration 100 that is better than the one on iteration 240

Lab 2

- Execute file Main.m. You should get the following outputs:
  - A scatter plot showing the position of the AP and STAs.
  - A plot of the average network throughput over time.
  - A plot of the STA satisfaction over time.
  - A boxplot of the network throughput over multiple iterations.
- For this initial simulation, consider the scatter plots of the default association (Figure 1) and the association after using ε-greedy (Figure 101), how has the association changed? What is the main reason for the improved network performance?
- Why is the satisfaction of the network lower despite the throughput increasing?
- Identify where ε-greedy is implemented in the code.
- Try different values for ε. What impact does it have on the results? Is a higher ε better or worse? Which value do you think is optimal?
- Modify the value of ε so that it decreases over time, you can use $\frac{1}{\sqrt{iteration}}$ or $\frac{1}{iteration}$ for instance. What impact does it have on ε-greedy ?
- Identify where the reward of ε-greedy is computed in *nodeload.m* and change it from an average reward to a cumulative one. Run the simulation again, are the results now better or worse? Why?

Lab 3

- Lab 1 and Lab 2 give you (up to) 6 points
- Lab 3 is worth 4 points
- For lab 3 you will have to implement your own solution to the AP selection problem
- You can use a different MAB (Thompson sampling, UCB…)
- You can modify ε-greedy to improve its performance
- You can create a heuristic outside of ML
- You will be graded mainly on your idea, performance is not as important

The code

- You only need to look at 3 files:
  - Main.m
  - nodeLoad.m
  - epsilon_greedy.m
- New things in main:

```
%%%%%%%%%%%%%%%%%%%%%%%%%% SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rng('default');
seed=1;

MaxSim=1; %Number of seeds to use
CCA=-82;    %Clear Channel Assessment
N_APs=16;  %Number of APs
N_STAs=64;    %Number of STAs
L=12000;    %Packet size
CWmin=16;   %Minimum contention window, for every node
ThrReq= 4E06; % Throughput required by STAs (bps)
SLOT=9E-6;  %OFDM time slot
MaxIter=239; %Number of e-greedy iterations
greedy=1;    % 1 for e-greedy, 0 for standard association
epsilon=0.1; % Epsilon value
clustered=1; % 1 for clustered STAs, 0 for random uniform placement

%%%%%%%%%%%%%%%%%%%%%%%%%% SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The code

```
%%%%%%%%%%%%%%%%%%%%%%%%%% SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rng('default');
seed=1;

MaxSim=1; %Number of seeds to use
CCA=-82;     %Clear Channel Assessment
N_APs=16;  %Number of APs
N_STAs=64;   %Number of STAs
L=12000;     %Packet size
CWmin=16;    %Minimum contention window, for every node
ThrReq= 4E06; % Throughput required by STAs (bps)
SLOT=9E-6;   %OFDM time slot
MaxIter=239; %Number of e-greedy iterations
greedy=1;    % 1 for e-greedy, 0 for standard association
epsilon=0.1; % Epsilon value
clustered=1; % 1 for clustered STAs, 0 for random uniform placement

%%%%%%%%%%%%%%%%%%%%%%%%%% SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

You can get smoother curves on your results if you use multiple scenarios

There is a relationship between APs, STAs and ThrReq

239 iterations of greedy decisions + first association based on RSSI = 240

This changes the behaviour of ε-greedy

The code (main)

```
for i=1:MaxIter

    [AP,STA]=nodeLoad(AP,STA,NodeMatrix,i,CWmin);
    if(greedy == 1)
        [STA]=epsilon_greedy(STA,i,epsilon);
    end
end
%%%% Checks satisfaction after last decision %%%%

[AP,STA]=nodeLoad(AP,STA,NodeMatrix,MaxIter+1,CWmin);
```

Main loop in which we calculate the network performance (nodeload) and then apply ε-greedy

We check one last time after the last decision!

The code (ε-greedy)

```
function [STA] = epsilon_greedy(STA,iter,Epsilon)
N_STAs=length(STA);


for i=1:N_STAs
    t = rand();
    if(t < Epsilon)
        %        disp('Explore---------------');
        STA(i).associated_AP = STA(i).APs_range(ceil(length(STA(i).APs_range)*rand));

    else
        %        disp('Exploit---------------');
        [x,index]=max(STA(i).APs_reward);

        if(x~=0)
            if(sum(STA(i).APs_reward==x)>1)
                indexes = find(STA(i).APs_reward==x);
                STA(i).associated_AP = randsample(indexes, 1);


            else

                STA(i).associated_AP = index;
            end

        else
            STA(i).associated_AP = STA(i).APs_range(ceil(length(STA(i).APs_range)*rand));

        end
        STA(i).expl(iter)=STA(i).expl(iter)+1;
    end

    STA(i).APSel(iter+1)=STA(i).associated_AP;

end
```

- STA(i).associated_AP stores new AP
- If two APs have max reward, we select one at random
- The STA structure is the only thing that is needed

The code (nodeLoad)

```
% ------------ Received Bandwidth ---------------

for i = 1:N_STAs
    if(STA(i).associated_AP~=0)
        airtime = RequiredAirtimeUser(STA(i).B,STA(i).L,NodeMatrix(i+N_APs,STA(i).associated_AP),CWmin);
        if(STA(i).nAPs > 0)
            if(AP(STA(i).associated_AP).airtime <= 1)

                STA(i).Be = STA(i).B;
                STA(i).satisfaction = STA(i).satisfaction + 1;

                STA(i).APs_rew(STA(i).associated_AP,STA(i).APs_rewIt(STA(i).associated_AP)+1) = 1; % Update reward history vector
                STA(i).APs_reward(STA(i).associated_AP) = mean(nonzeros(STA(i).APs_rew(STA(i).associated_AP,:))); % Update reward used for selection
                STA(i).APs_rewIt(STA(i).associated_AP)= STA(i).APs_rewIt(STA(i).associated_AP)+1; % Number of times this choice has been made

            else
                STA(i).Be = STA(i).B*(airtime / AP(STA(i).associated_AP).airtime)/airtime;
                STA(i).satisfaction = STA(i).satisfaction + 0;

                STA(i).APs_rew(STA(i).associated_AP,STA(i).APs_rewIt(STA(i).associated_AP)+1) = STA(i).Be/STA(i).B;
                STA(i).APs_reward(STA(i).associated_AP) = mean(nonzeros(STA(i).APs_rew(STA(i).associated_AP,:)));
                STA(i).APs_rewIt(STA(i).associated_AP)= STA(i).APs_rewIt(STA(i).associated_AP)+1;

            end
            STA(i).satisf(it)=STA(i).satisfaction;
            STA(i).accB(it)=STA(i).Be;
        else
            STA(i).Be = 0;
        end
    else
        % Nothing
    end
end
```

The code (nodeLoad)

```
% ------------ Received Bandwidth ---------------

for i = 1:N_STAs
    if(STA(i).associated_AP~=0)
        airtime = RequiredAirtimeUser(STA(i).B,STA(i).L,NodeMatrix(i+N_APs,STA(i).associated_AP),CWmin);
        if(STA(i).nAPs > 0)
            if(AP(STA(i).associated_AP).airtime <= 1)

                STA(i).Be = STA(i).B;
                STA(i).satisfaction = STA(i).satisfaction + 1;

                STA(i).APs_rew(STA(i).associated_AP,STA(i).APs_rewIt(STA(i).associated_AP)+1) = 1; % Update reward history vector
                STA(i).APs_reward(STA(i).associated_AP) = mean(nonzeros(STA(i).APs_rew(STA(i).associated_AP,:))); % Update reward used for selection
                STA(i).APs_rewIt(STA(i).associated_AP)= STA(i).APs_rewIt(STA(i).associated_AP)+1; % Number of times this choice has been made

            else
                STA(i).Be = STA(i).B*(airtime / AP(STA(i).associated_AP).airtime)/airtime;
                STA(i).satisfaction = STA(i).satisfaction + 0;

                STA(i).APs_rew(STA(i).associated_AP,STA(i).APs_rewIt(STA(i).associated_AP)+1) = STA(i).Be/STA(i).B;
                STA(i).APs_reward(STA(i).associated_AP) = mean(nonzeros(STA(i).APs_rew(STA(i).associated_AP,:)));
                STA(i).APs_rewIt(STA(i).associated_AP)= STA(i).APs_rewIt(STA(i).associated_AP)+1;

            end
            STA(i).satisf(it)=STA(i).satisfaction;
            STA(i).accB(it)=STA(i).Be;
        else
            STA(i).Be = 0;
        end
    else
        % Nothing
    end
end
```

If the STA gets all its load, our reward for this slot is 1

If the STA does not get all its load, our reward is the proportion of throughput/load that arrives

The code (nodeLoad)

The reward used by the algorithm is the average of all the slots

```
% ------------ Received Bandwidth ---------------

for i = 1:N_STAs
    if(STA(i).associated_AP~=0)
        airtime = RequiredAirtimeUser(STA(i).B,STA(i).L,NodeMatrix(i+N_APs,STA(i).associated_AP),CWmin);
        if(STA(i).nAPs > 0)
            if(AP(STA(i).associated_AP).airtime <= 1)

                STA(i).Be = STA(i).B;
                STA(i).satisfaction = STA(i).satisfaction + 1;

                STA(i).APs_rew(STA(i).associated_AP,STA(i).APs_rewIt(STA(i).associated_AP)+1) = 1; % Update reward history vector
                STA(i).APs_reward(STA(i).associated_AP) = mean(nonzeros(STA(i).APs_rew(STA(i).associated_AP,:))); % Update reward used for selection
                STA(i).APs_rewIt(STA(i).associated_AP)= STA(i).APs_rewIt(STA(i).associated_AP)+1; % Number of times this choice has been made

            else
                STA(i).Be = STA(i).B*(airtime / AP(STA(i).associated_AP).airtime)/airtime;
                STA(i).satisfaction = STA(i).satisfaction + 0;

                STA(i).APs_rew(STA(i).associated_AP,STA(i).APs_rewIt(STA(i).associated_AP)+1) = STA(i).Be/STA(i).B;
                STA(i).APs_reward(STA(i).associated_AP) = mean(nonzeros(STA(i).APs_rew(STA(i).associated_AP,:)));
                STA(i).APs_rewIt(STA(i).associated_AP)= STA(i).APs_rewIt(STA(i).associated_AP)+1;

            end
            STA(i).satisf(it)=STA(i).satisfaction;
            STA(i).accB(it)=STA(i).Be;
        else
            STA(i).Be = 0;
        end
    else
        % Nothing
    end
end
```

Vectors

STA(1).APs_rew

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0.6143	0.6143	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0.4659	0.5995	0.4125	0.7161	0.5317	0.3866	0.4630	0.4661	0.4767	0.5820	0.4753	0.3446	0.6351	0.4312	0.5492	0
12	0.6267	0.6267	0.4776	0.6756	0.5112	0.6756	0.6756	0.7000	0.7000	0.7615	0.5412	0.4742	0.5773	0.3667	0.6185	0.6185
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0.4483	0.6222	0.7885	0.4494	0.6066	0.4037	0.6066	0	0	0	0	0	0	0	0	0

This vector stores the instantaneous reward, meaning the reward obtained for every particular case in which an AP is selected, so the eighth time we took AP 12, we obtained 0.7

Vectors

STA(1).APs_reward

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0.6143	0	0	0.5024	0.7091	0	0	0	0.5608

The average reward for AP 12 is 0.7091

STA(1).APs_rewIt

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	2	0	0	15	216	0	0	0	7

We selected AP 12 216 times