Machine Learning for Networking

Reinforcement Learning

Session 8 – Multi-armed Bandits III
(Exp3, Contextual-MABs, States)

Boris Bellalta: boris.bellalta@upf.edu

Contents

- The adaptive routing problem in multi-path transport protocols
- Non-stationary environments
- Multi-armed Bandits
  - Exp3 → MAB for non-stationary cases / adversarial
- Contextual Multi-armed Bandits
- Definition of an state
  - State = set of characteristics / parameters that define a situation
  - The same environment can be represented in different ways using states

Exp3

- Gamma → trade-off between exploring and exploiting
- Exponential weights → try to promote good arms, specially if they were unlikely to be selected
- Note that $x_j(t)/p_j(t)$ is higher if it comes from an arm with previously low probability!
- It is a conservative algorithm, that instead of finding the best arm, it tries to avoid the worse ones.

Parameters: Real $\gamma \in (0,1]$

Initialisation: $\omega_i(1) = 1$ for $i = 1, \ldots, K$

For each t = 1, 2, ..., T

1. Set $p_i(t) = (1-\gamma)\frac{\omega_i(t)}{\sum_{j=1}^{K} \omega_j(t)} + \frac{\gamma}{K} \qquad i = 1, \ldots, K$
2. Draw i_t randomly according to the probabilities $p_1(t), \ldots, p_K(t)$
3. Receive reward $x_{i_t}(t) \in [0,1]$
4. For $j = 1, \ldots, K$ set:

$$\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t \\ 0, & \text{otherwise} \end{cases}$$

$$\omega_j(t+1) = \omega_j(t) exp(\gamma \hat{x}_j(t)/K)$$

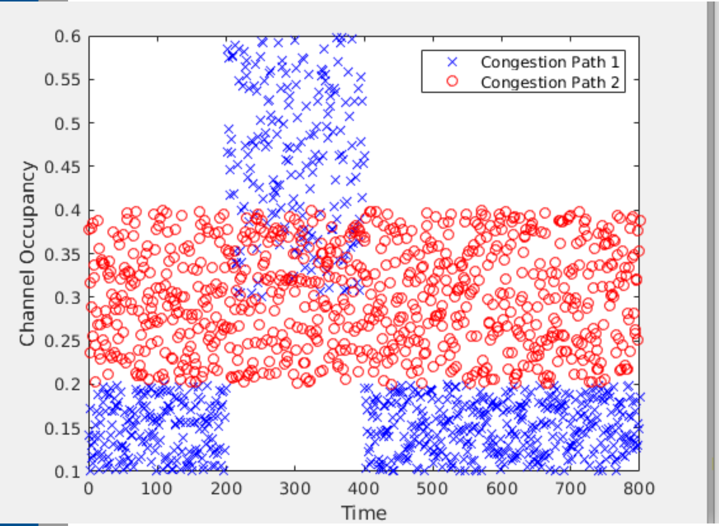
(image from Wikipedia)

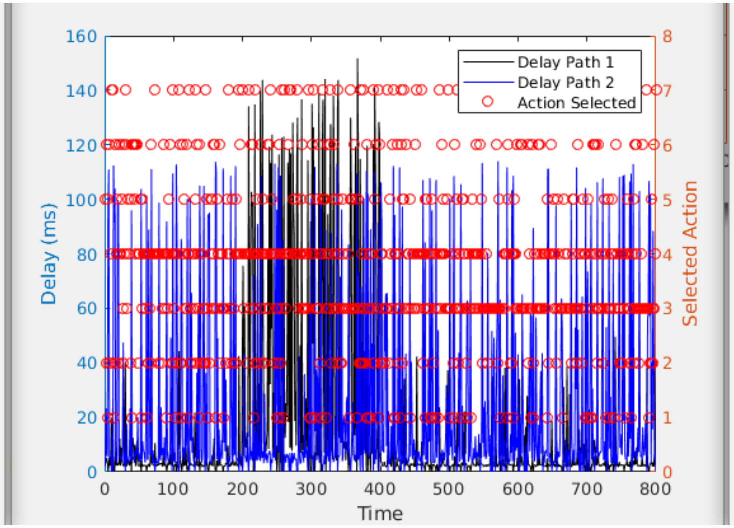
Code Exp3

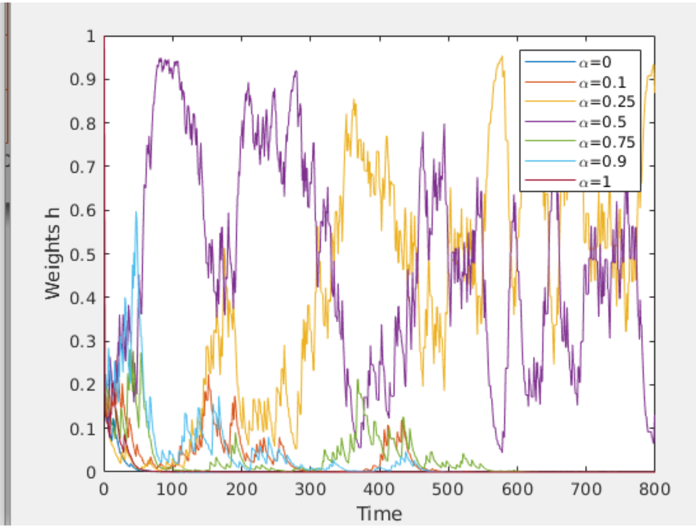
```
%---------------- EXP3 -----------------


Q_E3_disp = zeros(K,TimeHorizon);
Q_E3 = zeros(K,1);

R_E3 = zeros(1,TimeHorizon);
alfa_E3 = zeros(1,TimeHorizon);
alfa_selected_E3 = zeros(1,K);

% First Iteration

n_sel = randi(K,1); % Action Selected, random
alfa_E3(1) = n_sel;
alfa_selected_E3(n_sel) = 1;

% First Path
[ED1(1),ELosses1(1)] = PathDelay(KS,alfa(n_sel)*B,L,C1,w(1,1));
% Second Path
[ED2(1),ELosses2(1)] = PathDelay(KS,(1-alfa(n_sel))*B,L,C2,w(1,1));

% Reward (taking only the delay into account)
R_E3(1) = alfa(n_sel)*ED1(1)+(1-alfa(n_sel))*ED2(1); % Mean
%R_EG(1) = max([ED1(1) ED2(1)]); % Max

% Update Q-table
Q_E3(n_sel)=R_E3(1);
Q_E3_Disp(:,1) = Q_E3;

alfa_selected_E3(n_sel) = 1;

Sigma = 0.6; % Controls de Exploration

h=ones(K,TimeHorizon); % weights

p = zeros(K,TimeHorizon);
```

```
p = zeros(K,TimeHorizon);

for t=2:TimeHorizon

    % Calculate distribution

    for a=1:K
        p(a,t)=(1-Sigma)*h(a,t-1)/(sum(h(:,t-1))) + Sigma/K;
    end

    % Selects an action

    P=cumsum(p(:,t));
    aux=rand;
    n_sel = find(aux<P,1,'first'); % find the 1st index s.t. r<D(i);

    alfa_E3(t) = n_sel;
    alfa_selected_E3(n_sel)=alfa_selected_E3(n_sel)+1;

    % First Path
    [ED1(t),ELosses1(t)] = PathDelay(KS,alfa(n_sel)*B,L,C1,w(1,t));
    % Second Path
    [ED2(t),ELosses2(t)] = PathDelay(KS,(1-alfa(n_sel))*B,L,C2,w(2,t));

    % Reward (taking only the delay into account)
    R_E3(t) = abs(ED1(t)-ED2(t));

    Q_E3(n_sel)=(Q_E3(n_sel)*(alfa_selected_E3(n_sel)-1) + R_E3(t))/alfa_selected_E3(n_sel); % Av

    Q_E3_disp(:,t)=Q_E3(:); % For display purposes

    normQ = (1-(Q_E3(n_sel)/sum(Q_E3)));
    normQ_est = normQ/p(n_sel,t);

    % Update weights

    for a=1:K
        if(a==n_sel)
            h(a,t)=h(a,t-1)*exp(Sigma*normQ_est/K);
        else
            h(a,t)=h(a,t-1);
        end
    end

    h(:,t)=h(:,t)/sum(h(:,t));

end
```

Example Exp3







This is certainly not a good case for Exp3!
It explores too much as there are good (not the best) actions that are promoted just in case

Adding Contexts

Do you take the same actions (route) when moving in BCN when it is sunny or raining?

- In our adaptive routing problem, the 'network' can be in two different states:
  - State 1 (normal): a1=0.1; b1=0.2; a2=0.2, b2=0.4;
  - State 2 (busy-hour): a1=0.5; b1=0.7; a2=0.2, b2=0.4;
- If we know in which state we are (normal or busy-hour), can we use that information to improve the network performance?
  - Yes, just use a different MAB, or the same but take different actions in each state, switching between them when the environment changes from one state to another.
- Challenge: how to know in which state the environment is?
  - What about an external source of data? Why do you check the weather before planning a trip? To know the state of the environment, and so take the best actions.
  - ML + Big Data is about that, combining multiple sources of information.

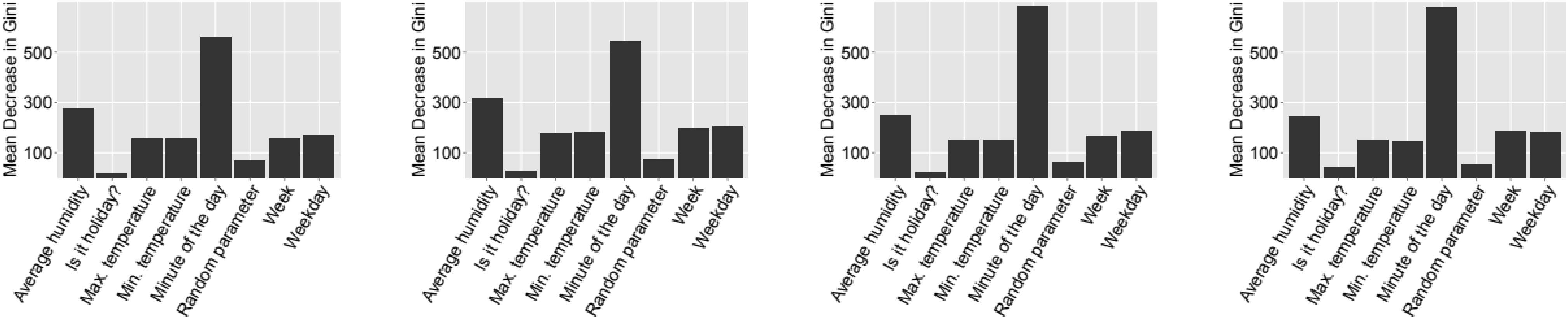
SAI Intelligent Systems Conference 2015
November 10-11, 2015 | London, UK

Predicting Occupancy Trends in Barcelona's Bicycle Service Stations Using Open Data

https://arxiv.org/pdf/1505.03662.pdf

Gabriel Martins Dias, Boris Bellalta and Simon Oechsner

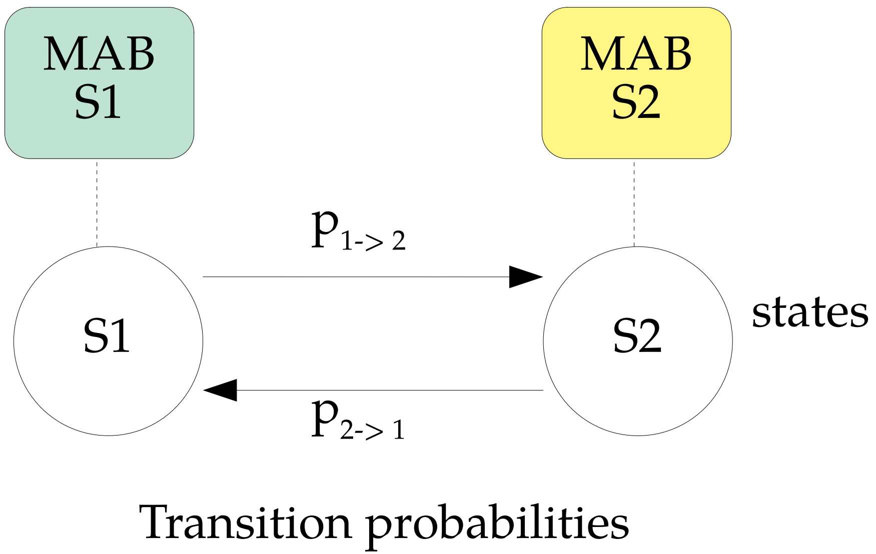
Department of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain
Email: {gabriel.martins, boris.bellalta, simon.oechsner}@upf.edu



(a) Station 50 (b) Station 124 (c) Station 92 (d) Station 305

Fig. 2: Importance of the external factors in the number of bikes at the observed stations.

Adding contexts



- We can have different algorithms (TS, UCB) in each ‘state’.
- We can have different parameters (i.e., exploration rate), etc.

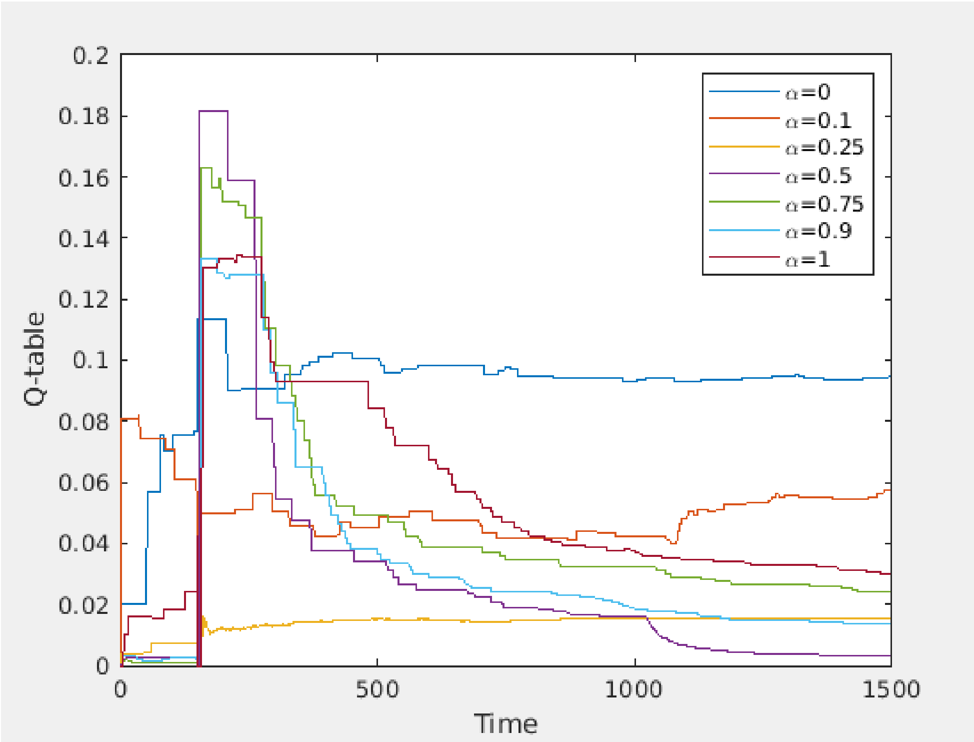
Epsilon-greedy with contexts

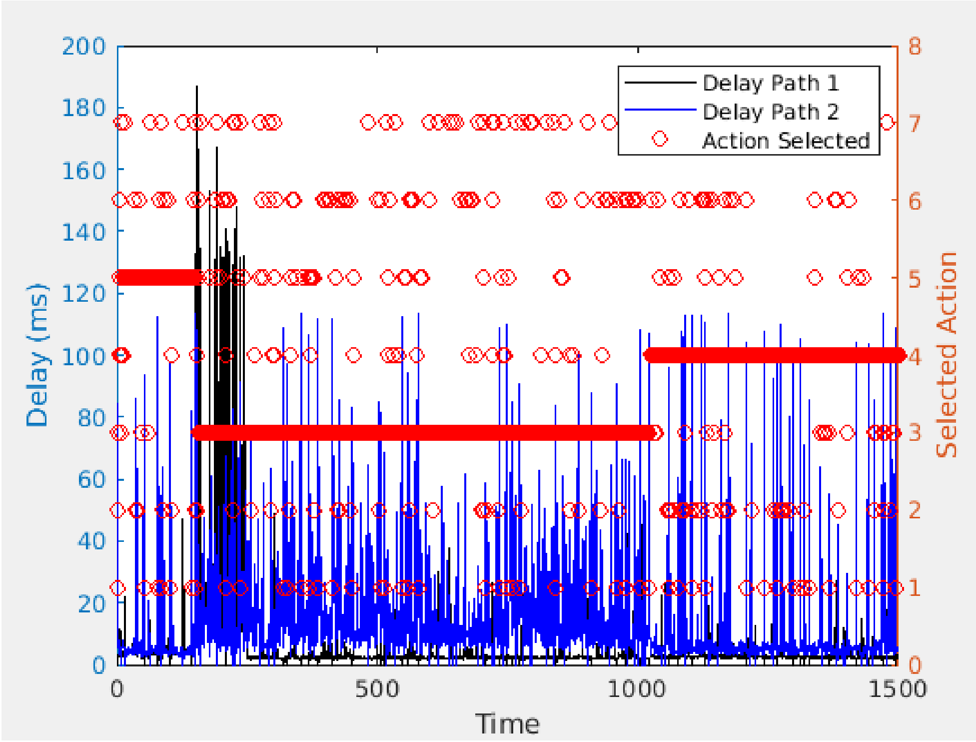
```
C1 = 10E6;
C2 = 10E6;
L=8000;
KS=100;
B=8E6;

% Sim parameters
TimeHorizon=400;
TimeChange=150;
TimeChange2=250;

N=2; % number of paths

a1=zeros(1,TimeHorizon);
a1(1:TimeChange)=0.1.*ones(1,TimeChange);
a1(TimeChange+1:TimeChange2)=0.5.*ones(1,TimeChange2-TimeChange);
a1(TimeChange2+1:TimeHorizon)=0.1.*ones(1,TimeHorizon-TimeChange2);

b1=zeros(1,TimeHorizon);
b1(1:TimeChange)=0.2.*ones(1,TimeChange);
b1(TimeChange+1:TimeChange2)=0.7.*ones(1,TimeChange2-TimeChange);
b1(TimeChange2+1:TimeHorizon)=0.2.*ones(1,TimeHorizon-TimeChange2);


a2=0.2;
b2=0.4;

w(1,:)=a1 + (b1-a1).*rand(1,TimeHorizon);
w(2,:)=a2 + (b2-a2).*rand(1,TimeHorizon);

% Weights - Actions
%alfa = [0:0.05:1];
%alfa = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
alfa = [0 0.1 0.25 0.5 0.75 0.9 1];
%alfa = [0.5];

K = length(alfa);
```

```
%----------------- Epsilon greedy ------------------

epsilon = 0.1;
Delta = 0.0;

%epsilon = 1;
%Delta = 0.02;

Q_EG_disp = zeros(K,TimeHorizon);
Q_EG = zeros(K,1);
Q_EG_context1 = zeros(K,TimeHorizon);

R_EG = zeros(1,TimeHorizon);
alfa_EG = zeros(1,TimeHorizon);
alfa_selected_EG = zeros(1,K);

alfa_selected_EG_context1 = zeros(1,K);


% First Iteration

n_sel = randi(K,1); % Action Selected, random
alfa_EG(1) = n_sel;
alfa_selected_EG(n_sel) = 1;

% First Path
[ED1(1),ELosses1(1)] = PathDelay(KS,alfa(n_sel)*B,L,C1,w(1,1));
% Second Path
[ED2(1),ELosses2(1)] = PathDelay(KS,(1-alfa(n_sel))*B,L,C2,w(1,1));

% Reward (taking only the delay into account)
R_EG(1) = alfa(n_sel)*ED1(1)+(1-alfa(n_sel))*ED2(1); % Mean
%R_EG(1) = max([ED1(1) ED2(1)]); % Max

% Update Q-table
Q_EG(n_sel)=R_EG(1);
Q_EG_Disp(:,1) = Q_EG;

alfa_selected_EG(n_sel) = 1;
```

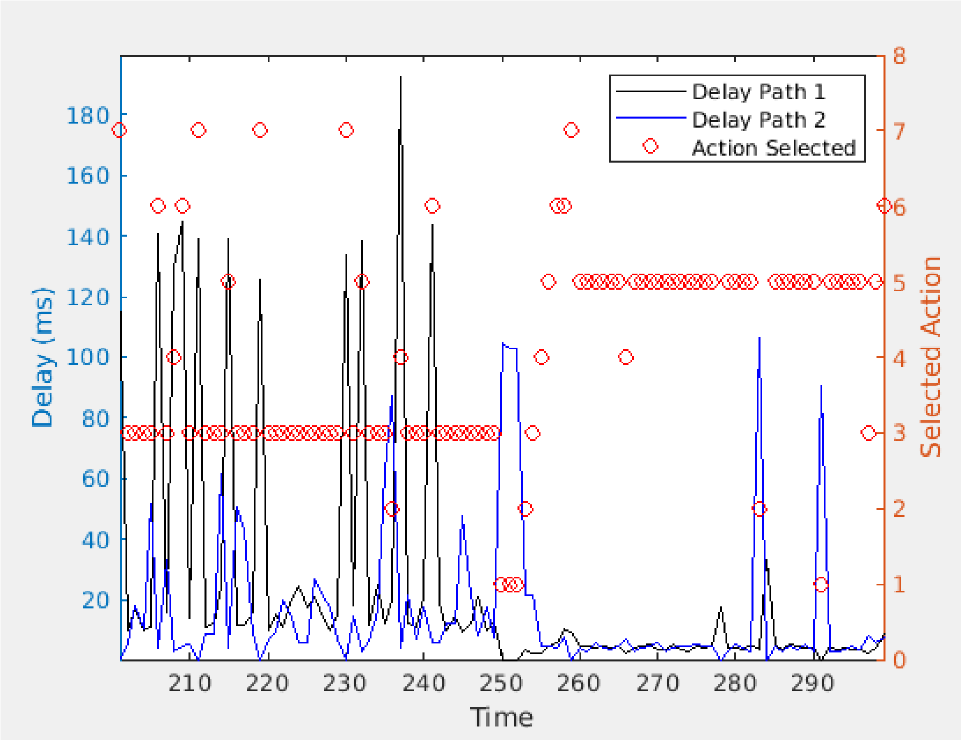
```
for t=2:TimeHorizon

    if(rand < epsilon) % explore
        n_sel=randi(K,1);
    else % exploit
        [~,n_sel]=min(Q_EG(:)); % Maximize the reward --> minimize the delay!!!
    end

    alfa_EG(t) = n_sel;
    alfa_selected_EG(n_sel)=alfa_selected_EG(n_sel)+1;

    % First Path
    [ED1(t),ELosses1(t)] = PathDelay(KS,alfa(n_sel)*B,L,C1,w(1,t));
    % Second Path
    [ED2(t),ELosses2(t)] = PathDelay(KS,(1-alfa(n_sel))*B,L,C2,w(2,t));

    % Reward (taking only the delay into account)
    R_EG(t) = abs(ED1(t)-ED2(t));

    Q_EG(n_sel)=(Q_EG(n_sel)*(alfa_selected_EG(n_sel)-1) + R_EG(t))/alfa_selected_EG(n_sel); % Av

    Q_EG_disp(:,t)=Q_EG(:); % For display purposes

    % Update Learning
    epsilon = epsilon - Delta;

    % Reset ----------------------------

    if(t==TimeChange+1)

       Q_EG_context1 = Q_EG;
       alfa_selected_EG_context1 = alfa_selected_EG;

       % Reset (since I don't have previous info from current context)
       alfa_selected_EG = zeros(1,K);
       Q_EG = zeros(K,1);

    end

    if(t == TimeChange2+1)
       % Update Context: I detect the system moves to context 1, so

       alfa_selected_EG = alfa_selected_EG_context1;
       Q_EG = Q_EG_context1;

       % The Reset case
       %alfa_selected_EG = zeros(1,K);
       %Q_EG = zeros(K,1);

    end

end
```

We do nothing...

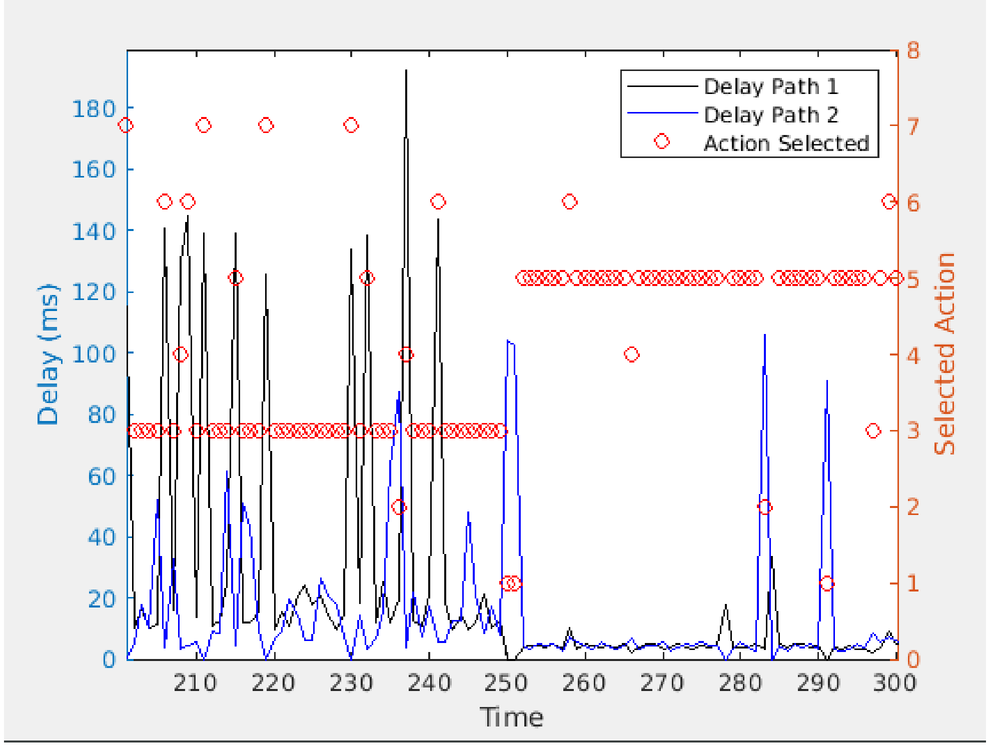




We detect a change of the context



RESET



Q-table REUSE

Activity

- Download Example8.zip
- Update how the exploration is done: set Epsilon to 1, and try different values of Delta when there is a context change. Is it a good strategy?