Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks

Prepared By: Hamidreza Aliakbary

What is Multichannel access problem?

The Problem is to assign proper channels for each user. Challenge is to cope with correlation of channels to solve it.

Simplified solutions

01

Myopic Policy

Ignores dependency and premises channels are i.i.d

02

Whittle index

Assumes channels are independent But folowwing different Markovian chains

What is Bad Then?

Ignoring Correlation will result in Divergence.

Solution?

Deep Reinforcement Learning

Why Deep reinforcement learning?

Formulation

Problem Choosing Best channel out of N-channels (action)

States 2^N-state and each state is a length-N vector showing goodness of each channel

Reward +1 if chose correctly and -1 if choice is wrong

The problem Falls into framework of POMDP.

- We have no sense about transitions or combinations.
- Our belief should update in each observation.

SO,

What is Belief?

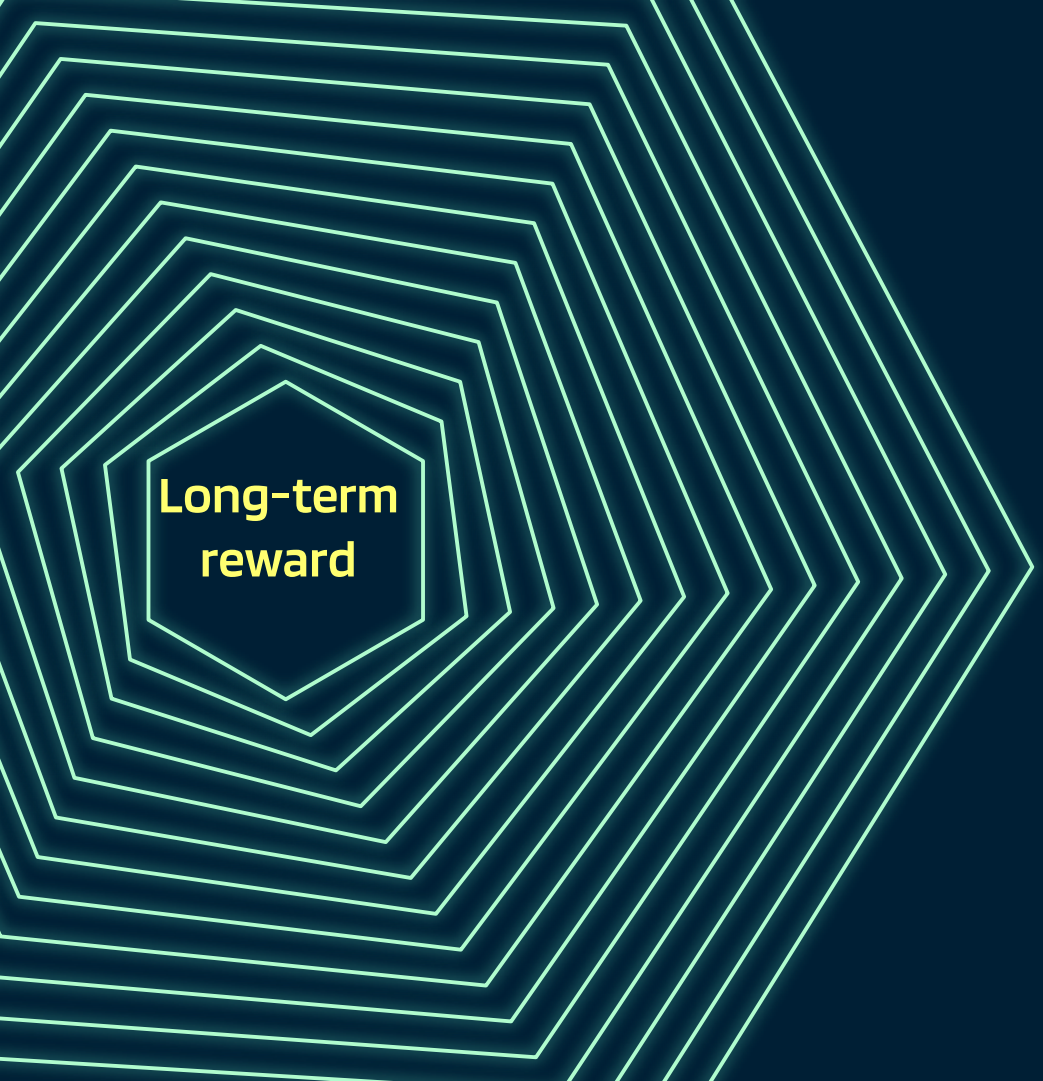
$$\Omega(t) = [\omega_{s_1}(t), \dots, \omega_{s_2^N}(t)]$$

$$\omega_{s_i}(t) = P(S = s_i | All\ prev.decisions\ and\ observations)$$

How to update Belief?

$$\widehat{\omega}_{s_i}(t) = \begin{cases} \dfrac{(\omega_{s_i}(t)\mathbb{I}(s_{ik}(t)=1))}{\sum_{i=1}^{2^N}\omega_{s_i}(t)\mathbb{I}(s_{ik}(t)=1))} & a(t)=k, o(t)=1 \\ \dfrac{(\omega_{s_i}(t)\mathbb{I}(s_{ik}(t)=0))}{\sum_{i=1}^{2^N}\omega_{s_i}(t)\mathbb{I}(s_{ik}(t)=0))} & a(t)=k, o(t)=0 \end{cases}$$

a(t): denotes which channel to sense in timeslot t, $\in [1, N]$
o(t): denotes state of channel in timeslot t , $\in \{0,1\}$



$$\mathbb{E}_{\pi}[\sum_{t=1}^{\infty} \gamma^{t-1} R_{\pi(\Omega(t))}\ (t)|\Omega(1)]$$

$0 \leq \gamma < 1$

$\pi(\Omega(t))$: is the channel sensing action at time slot t given belief $\Omega(t)$.

What is our Objective?

$finding\ \pi^*$

$$\pi^* = arg\max_{\pi} \mathbb{E}_{\pi}[\sum_{t=1}^{\infty} \gamma^{t-1} R_{\pi(\Omega(t))}\ (t)|\Omega(1)]$$

there are many existing works on finding the optimal/near-optimal policy

Myopic Policy

A Myopic policy only focuses on the immediate reward obtained from an action and ignores its effects in the future. Thus the user always tries to select a channel which gives the maximized expected immediate reward.

2

Whittle index

When channels are independent, the dynamic multichannel access problem can also be considered as a restless multi-armed bandit problem (RMAB)if each channel is treated as an arm. An index policy assigns a value to each arm based on its current state and chooses the arm with the highest index at each time slot. The index policy is not guaranteed to be optimal in general.

3

Deep Q-Learning

It is good enough to get answer!
The goal of Q-learning is to find
an optimal policy, i.e., a sequence of actions that maximizes the long-term expected accumulated discounted reward.

Classic Q-table update in Q-learning

$$Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) - Q(x_t, a_t)]$$

Why should we use DQN?

1- The state space size in
this work grows exponentially

2- Q-learning
can capture enough information for learning.

Inputs and target of DQN

Input

Pairs of state-action

Target

Output is corresponding Q-value and target is to minimize function below:

$$L_i(\theta_i) = \mathbb{E}\left[\left(y_i - Q(x, a; \theta_i)\right)^2\right] \rightarrow \theta_i : weights\ of\ network\ in\ i-th\ itteration$$

OPTIMAL POLICY FOR KNOWN FIXED-PATTERN CHANNEL SWITCHING

Algorithm 1 Optimal policy

At the beginning of time slot 0, choose a channel in the initial activated subset C_1
for $n = 1, 2, \ldots$ **do**
    At the beginning of time slot n,
    if $0.5 \leq p \leq 1$ **then**
        if The previous chosen channel is good **then**
            Choose a channel in the next activated subset according to the subset activation order
        else
            Stay in the same channel
    else
        if The previous chosen channel is good **then**
            Stay in the same channel
        else
            Choose a channel in the next activated subset according to the subset activation order

DQN Hyper Parameter

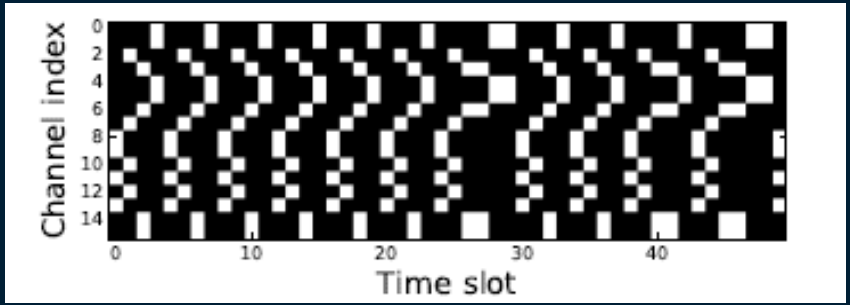
Hyperparameters	Values
ϵ	0.1
Minibatch size	32
Optimizer	Adam
Activation Function	ReLU
Learning rate	10^{-4}
Experience replay size	1,000,000
γ	0.9

DQN Architecture

- Fully connected neural network
- 2 hidden layers
- Each layer has 200 neurons
- Exploration-exploitation balance is with ϵ-greedy policy with fixed amount of 0.1

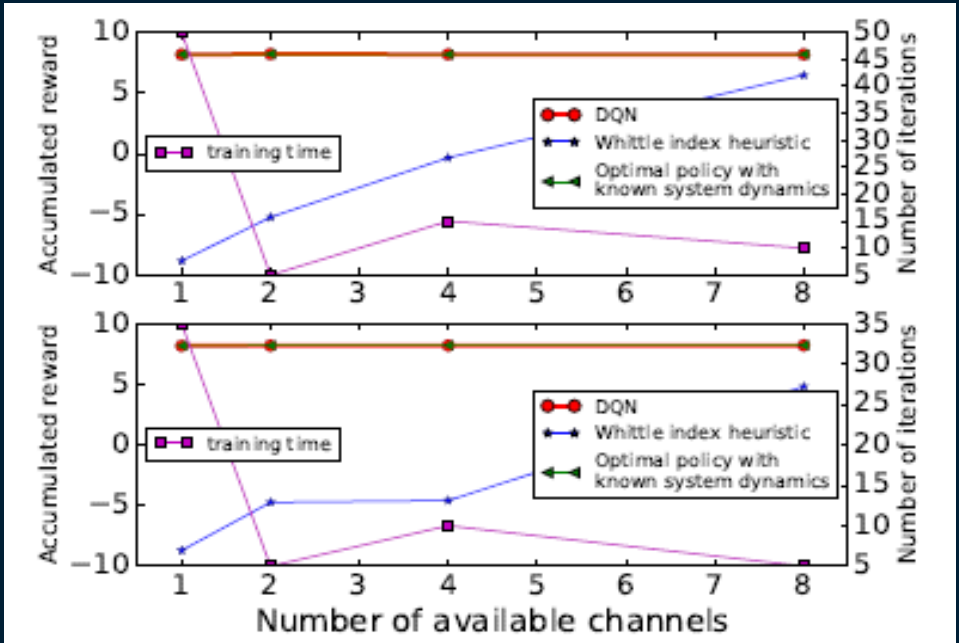
Experiment

In this experiment author consider a system with 16 channels are evenly divided into several subsets that take turns to become available with a switching probability fixed at p = 0:9. A capture of good channel visualized with pixel method:

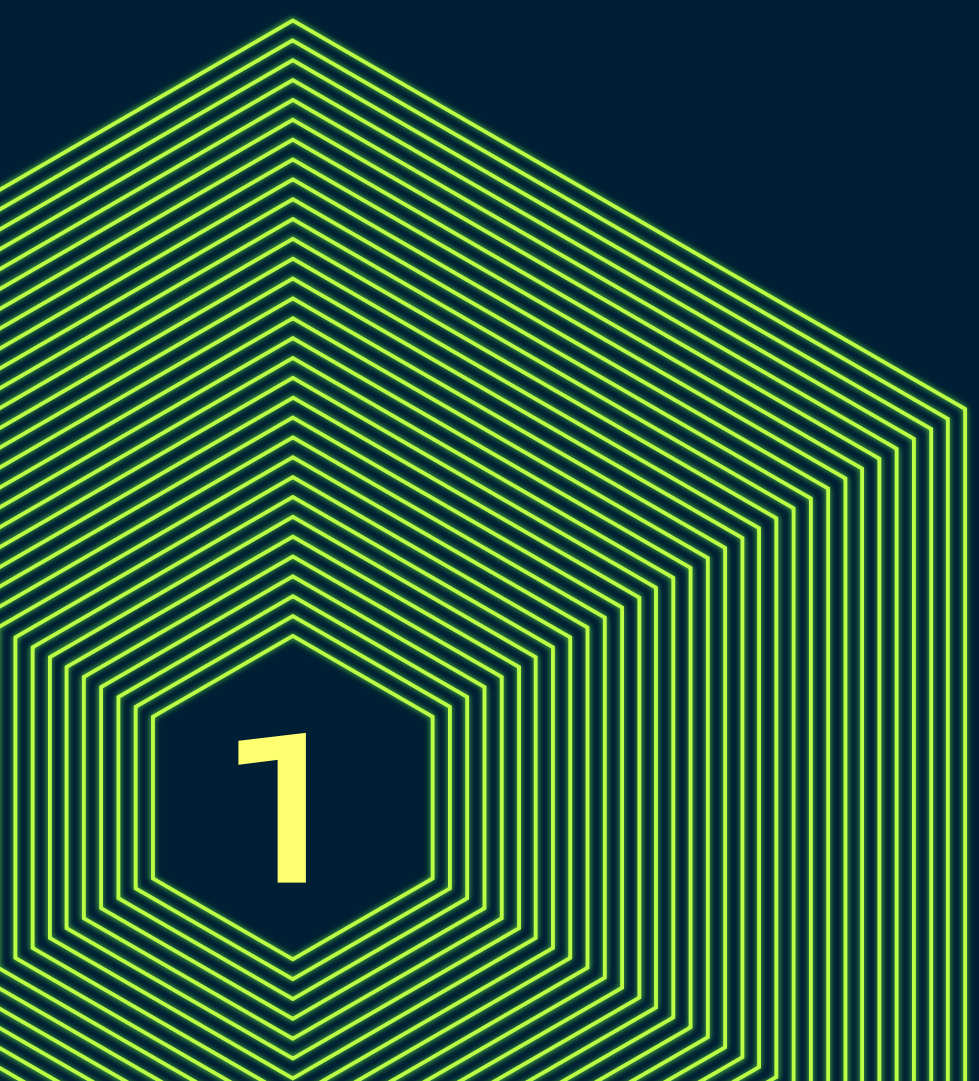


Evaluation

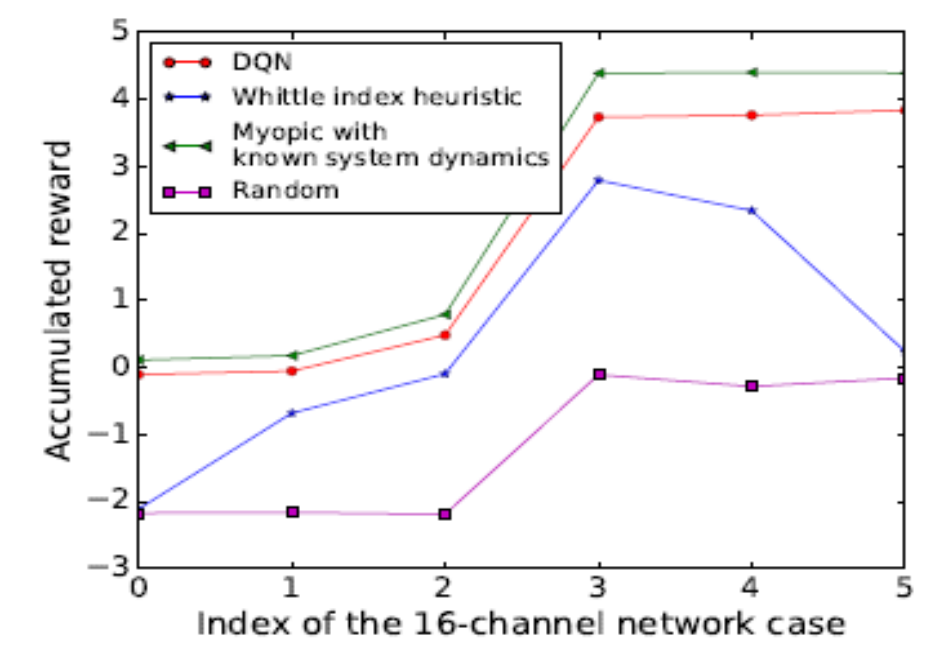
In following graphs DQN's performance evaluated VS other solutions represented earlier



EXPERIMENT AND EVALUATION OF DQN FOR MORE COMPLEX SITUATIONS



Perfectly correlated scenario



Average discounted reward for 6 different cases. Each case considers a different set of correlated channels

Real data trace

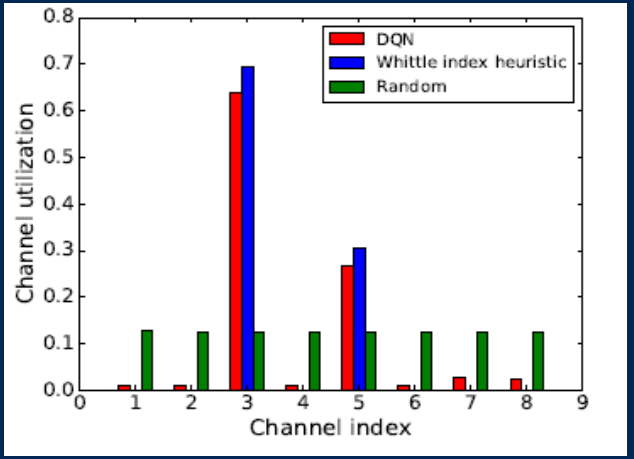
Procedure

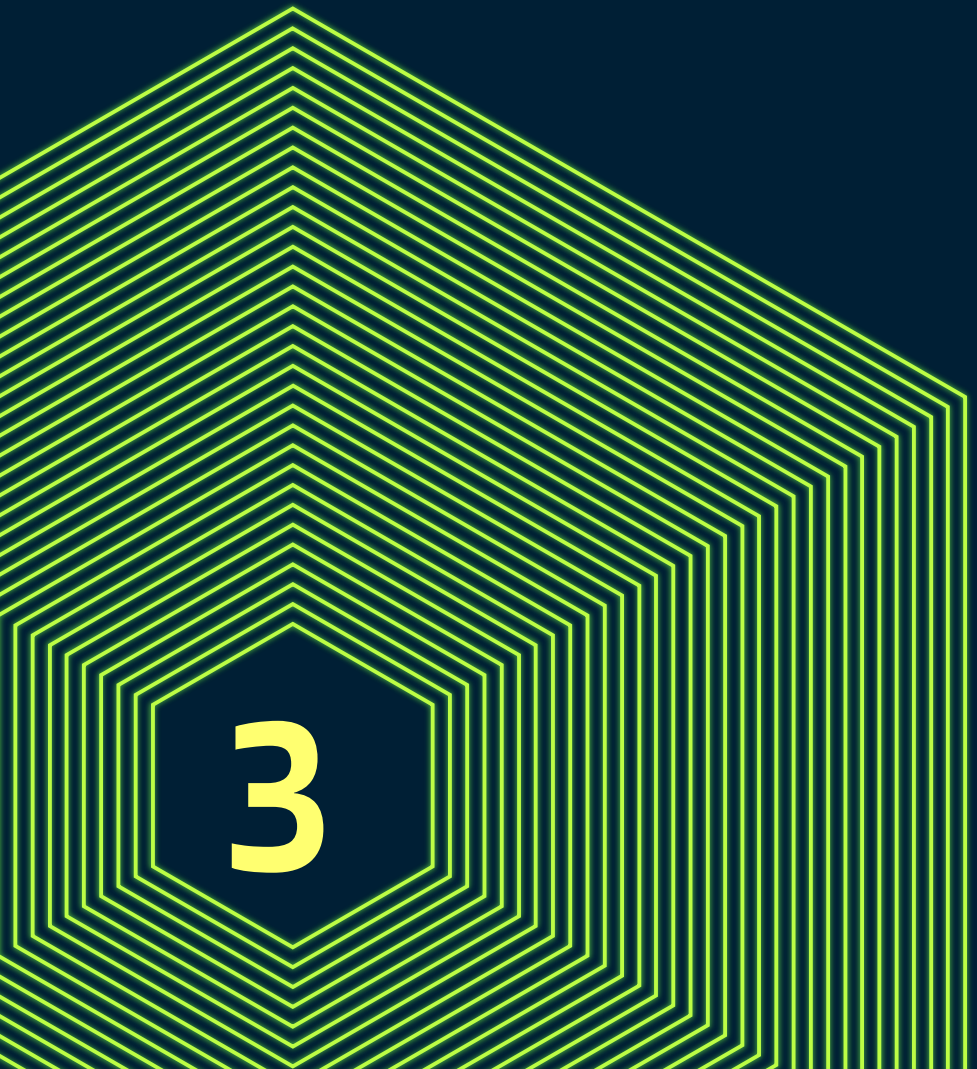
- The testbed is composed of TelosB nodes with IEEE 802.15.4 radio
- a pair of motes distanced approximately 20 meters to be transmitter/receiver
- transmits one packet rapidly to each one of the 16 available channels
- synchronized receiver records the successful and failed attempts
- only interference coming from surrounding WiFi networks
- ignore 8 good channels that result in remaining 8 channels that show significant WiFi

Results

The average accumulated discounted reward from each policy is listed in descending order:

- 0.947 (DQN),
- 0.767 (Whittle Index)
- -2.170 (Random Policy).





Practical Issues

Challenge: sender receiver synchronization

One approach is to run same structured DQNs at the sender and the receiver separately

How to do this?

USE ACK/NAK !!!

By ACK/NAK final learned policy is guaranteed to be the same, So WHAT ELSE?

The channel mismatch problem can still happen when an ACK/NAK is lost, which not only causes loss of communication, but also results in different learned DQN models at the sender and receiver that give different channel selection policies.

What Can we do for resynchronization?

One possible approach is to find a way to let the sender and the receiver be aware of the time when a channel mismatch happens, and try to recover in time.

Step-by-step approach:

- Once mismatch happens, sender stops updating it's DQN
- till recovery, transmits data from one single good channel
- Along with the data messages, the sender also sends the timestamp when the channel mismatch was perceived.

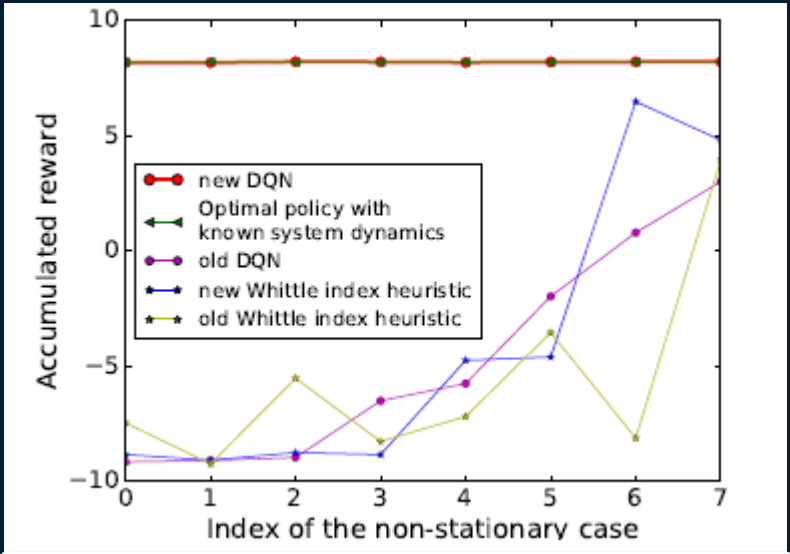
The sender keeps sending this channel mismatch time information until an ACK being received

ADAPTIVE DQN FOR UNKNOWN, TIME-VARYING ENVIRONMENTS

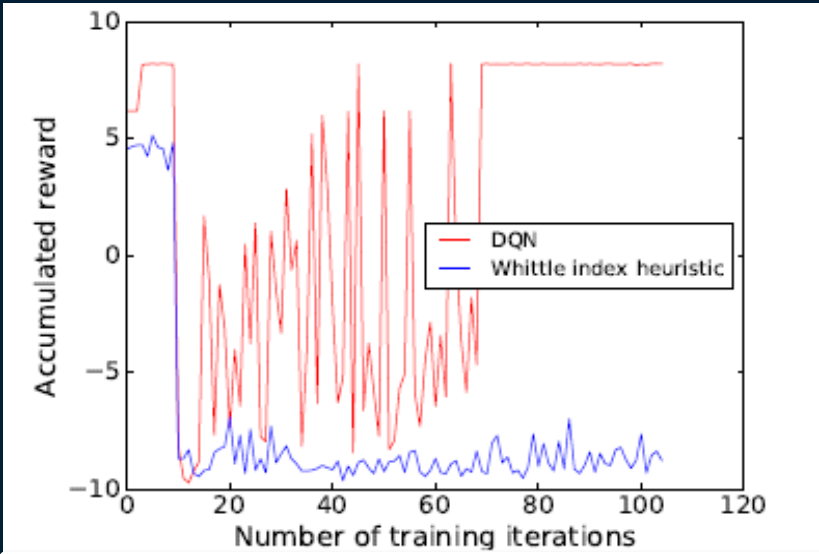
Algorithm 2 Adaptive DQN

1: First train DQN to find a good policy to operate with
2: **for** $n = 1, 2, \ldots$ **do**
3: At the beginning of period n
4: Evaluate the accumulated reward of the current policy
5: **if** The reward is reduced by a given threshold[4] **then**
6: Re-train the DQN to find a new good policy
7: **else**
8: Keep using the current policy

Evaluate The Enhancement



Average discounted reward as we vary the channel switch



Average discounted reward in real time during training in unknown fixed-pattern channel switching

Thanks for reading!

Any Question?
Contact me via My email:
Hamidreza.khoyi99@gmail.com