IEEE Transactions on Cognitive Communications And Networking, Vol. 4, No.2, June 2018

Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks

Shanxing Wang, Hanpeng Liu, Pedro Henrique Gomes, and Bhaskar Krishnamachari

Presenters:

Abhinav Gupta, Aditya Gulati

Introduction

- In dynamic multichannel access problem, the motive is to select the best channel available to transmit the data successfully.
- The main challenge here is multiple correlated channels follow an unknown Markov model.
- The final objective boils down to maximize the expected number (long-term) of successful transmissions.

Introduction

- We consider a system with N channels (each channel having two possible states : good or bad).
- We have a single user who selects one channel at each time slot to transmit a packet. (If selected channel is good : Success and vice-versa).
- The user is only able to sense the selected channel hence making it a Partially Observable Markov Decision Process (POMDP).

Motivation for Dynamic Multichannel access

- Dynamic spectrum access is one of the keys to improve the spectrum utilization in wireless networks and complementing the need for capacity.
- In cognitive radio, secondary users may search and use idle channels that are not being used by primary users.
- In practice external interference can cause the channels in wireless networks to be highly correlated.

Problem Formulation

- We have a dynamic multichannel access problem where there is a single user choosing one out of N channels.
- At the beginning of each time slot, a user selects one channel to sense and transmit a packet. (+1/-1 reward/penalty).
- We find a sensing policy between belief vector and sensing action at each time slot to maximize the expected reward.

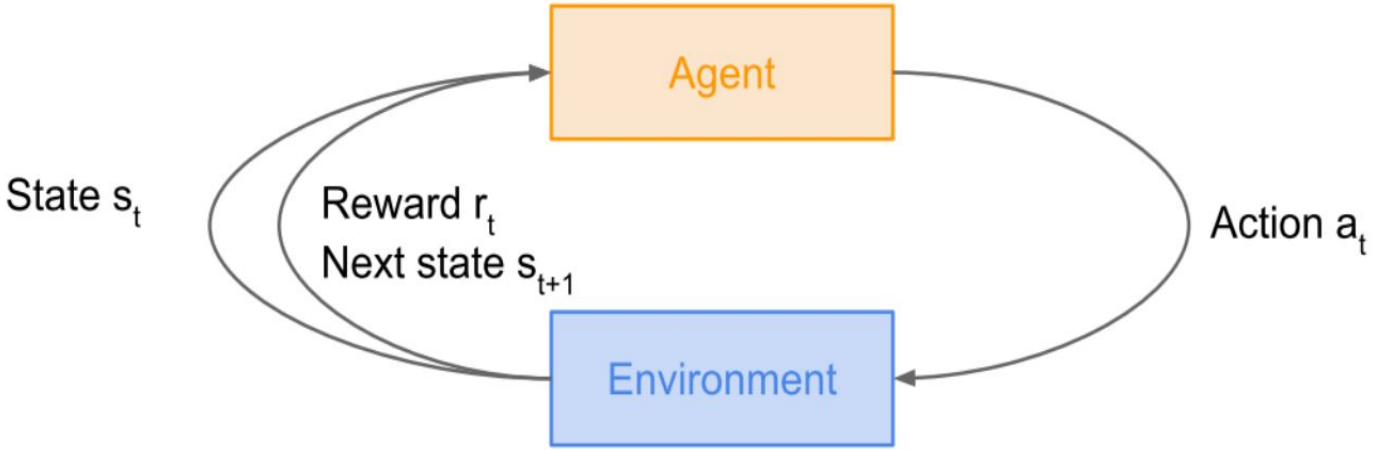
Conditions on channels

- When channels are i.i.d
  - Myopic policy is optimal under certain conditions
- When channels are independent but may follow different distributions
  - RMAB can be applied
- When channels have identical distributions but are not independent
  - Whittle Index Policy can be applied
- When channels are not independent nor are identically distributed
  - Following slides showcase a solution

Reinforcement Learning

- Deals with problems involving an agent interacting with an environment, which provides numeric reward signals, our goal is to learn how to take actions in order to maximize the reward

Reinforcement Learning



Source: CS231N course by Fei-Fei Li, Justin Johnson, Serena Yeung

Reinforcement Learning

- Mathematical formulation of the RL problem
- **Markov property**: Current state completely characterises the state of the world

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

Reinforcement Learning

- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t=0 until done:
  - Agent selects action a_t
  - Environment samples reward $r_t \sim R(. | s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P(. | s_t, a_t)$
  - Agent receives reward r_t and next state s_{t+1}

- A policy **π** is a function from S to A that specifies what action to take in each state
- **Objective**: find policy **π*** that maximizes cumulative discounted reward: $\sum_{t \geq 0} \gamma^t r_t$

$$\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi\right] \quad \text{with} \quad s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$$

Reinforcement Learning

Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$

How good is a state?

The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?

The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Reinforcement Learning

The optimal Q-value function Q* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Q* satisfies the following **Bellman equation**:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s, a\right]$$

Intuition: if the optimal state-action values for the next time-step Q*(s',a') are known, then the optimal strategy is to take the action that maximizes the expected value of $r + \gamma Q^*(s',a')$

The optimal policy **π*** corresponds to taking the best action in any state as specified by Q*

Reinforcement Learning

Value iteration algorithm: Use Bellman equation as an iterative update

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s,a\right]$$

Q_i will converge to Q* as i -> infinity

Q Learning

- When we use a function approximator to estimate the action-value function
- It is model-free method that can learn the policy directly via online learning

$$Q(s, a; \theta) \approx Q^*(s, a)$$

function parameters (weights)

If the function approximator is a deep neural network => **deep q-learning**!

Q-Learning Update

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s,a \right]$$

Forward Pass

Loss function: $L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s,a;\theta_i))^2 \right]$

where $y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a \right]$

Iteratively try to make the Q-value close to the target value (y_i) it should have, if Q-function corresponds to optimal Q* (and optimal policy π^*)

Backward Pass

Gradient update (with respect to Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot);s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)) \nabla_{\theta_i} Q(s,a;\theta_i) \right]$$

Source: CS231N course by Fei-Fei Li, Justin Johnson, Serena Yeung

Deep Reinforcement Learning

- Deep Q learning can be used as a way to model learning in an unknown environment.
- We integrate Q learning with Deep neural networks (hence Deep Q Learning), where we use a DNN with states as input and Q values as output to learn the policies.
- The main achievement here is it learns optimal policy of large systems without any a-priori knowledge of system dynamics.

Experience Replay

Learning from batches of consecutive samples is problematic:

- Samples are correlated => inefficient learning
- Current Q-network parameters determines next training samples (e.g. if maximizing action is to move left, training samples will be dominated by samples from left-hand size) => can lead to bad feedback loops

Address these problems using **experience replay**

- Continually update a **replay memory** table of transitions (s_t, a_t, r_t, s_{t+1}) as game (experience) episodes are played
- Train Q-network on random minibatches of transitions from the replay memory, instead of consecutive samples

Each transition can also contribute to multiple weight updates
=> greater data efficiency

Multichannel Access + Reinforcement Learning

Our objective is to find a sensing policy π∗ that maximizes the expected accumulated discounted reward over infinite time

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

We can solve this using an augmented MDP instead via value iteration, however the dimension of the the belief vector is exponentially large in the number of channels.

Motivation Towards RL

- Can Solve above using Myopic/Whittle Index Strategies.
- POMDP is bottleneck in large state space.
- No performance guarantees on above.
- Hence we need a model free approach which can learn the model directly from past instances. (Q- Learning)

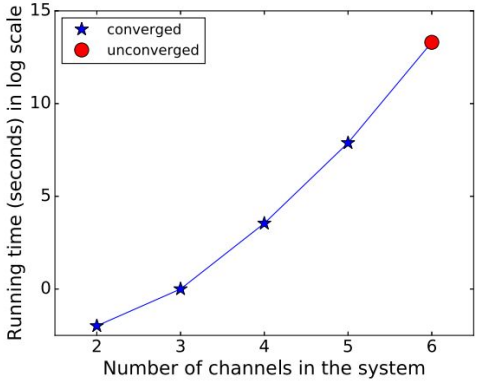


Fig. 1. Running time (seconds) in log scale of the POMDP solver as we vary the number of channels in the system.

Deep Q Learning Application+Motivation

- In Q Learning the state space grows exponentially. Hence DQN is used as a function approximator.
- DQN uses Deep Neural Network with states as input and estimated Q values as output to efficiently learn policies for high-dimensional, large state-space problems.
- The aim becomes to minimize the MSE between Q value and predicted from previous weight.

Evaluation

Optimal Policy Baseline

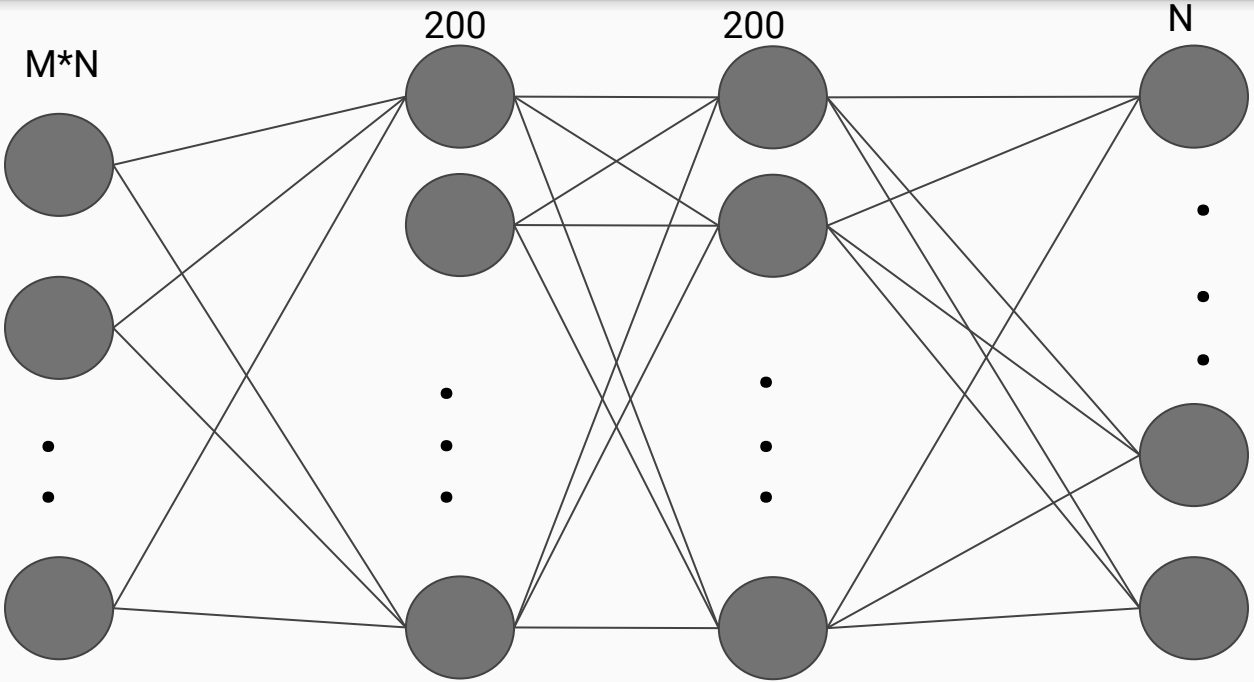
Algorithm 1 Optimal Policy

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 Architecture

Inputs are the combinations of previous actions and observations over previous M time slots

M=N(in paper)



i^{th} output represents the Q value of the state if i^{th} channel is selected

DQN Specifications

- Experience Replay discussed earlier is used to break correlations among data samples which makes training stable and convergent.
- A minibatch is randomly chosen from replay memory to compute the loss function and Adam algorithm is used to update the weights.

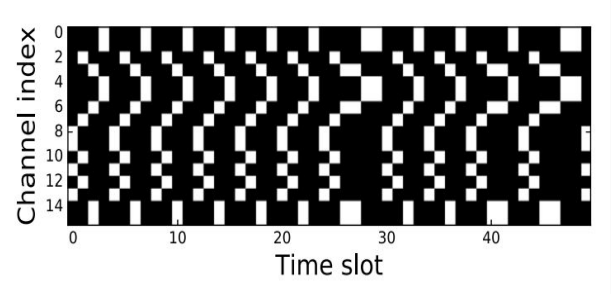


TABLE I
LIST OF DQN HYPERPARAMETERS

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

Experimental Results

- In the first graph, channels are activated in a round robin fashion.
- In the second graph, channels are activated in an arbitrary manner.
- DQN outperforms Whittel index heuristic and performs close to optimal, implies it is able to learn the underlying correlation among channels.

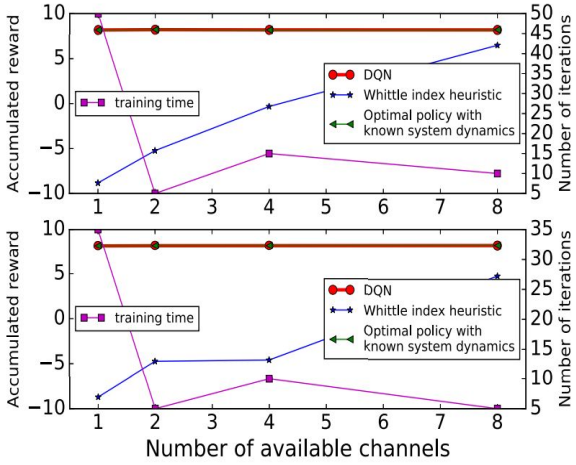
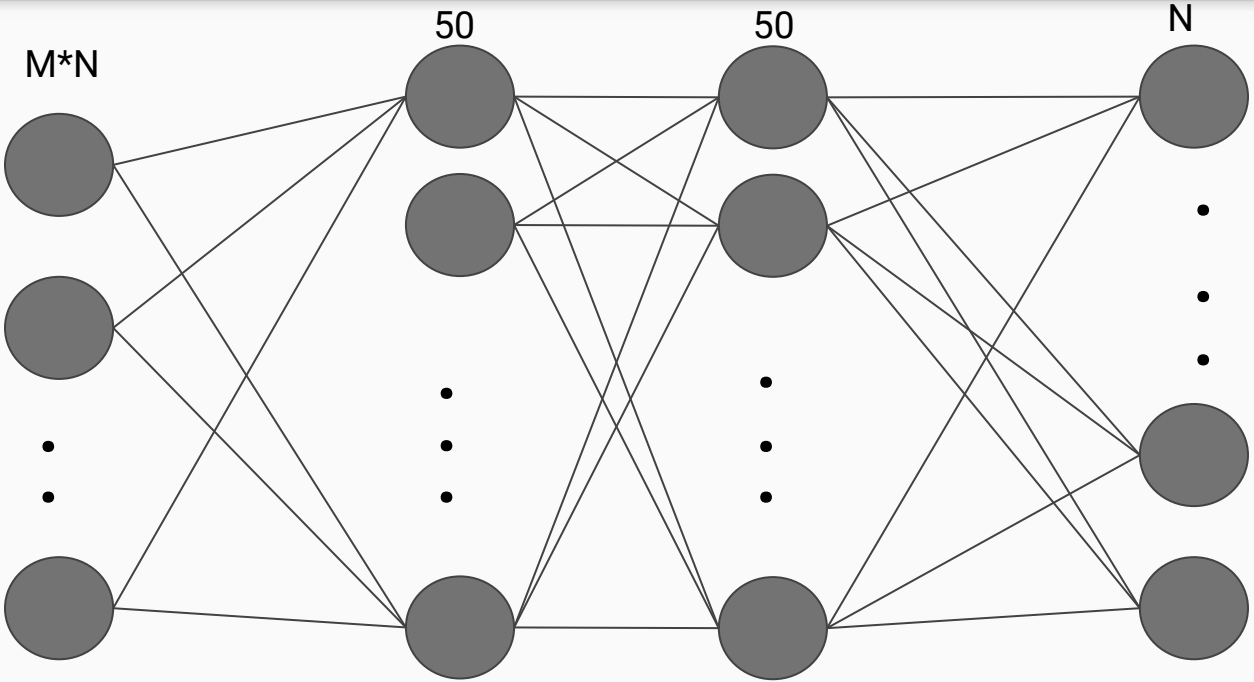


Fig. 3. Average discounted reward as we increase the number of good channels in the multiple good channels situation.

DQN architecture with Realistic simulations

Inputs are the combinations of previous actions and observations over previous M time slots

M=N(in paper)



i^{th} output represents the Q value of the state if i^{th} channel is selected

Evaluation with Realistic simulations

- Only 3 channels are independent, remaining are exactly identical or opposite to one of these independent channels.
- Since the optimal policy is computationally prohibitive, Myopic policy is used as a genie(knowing the system statistics a-priori).

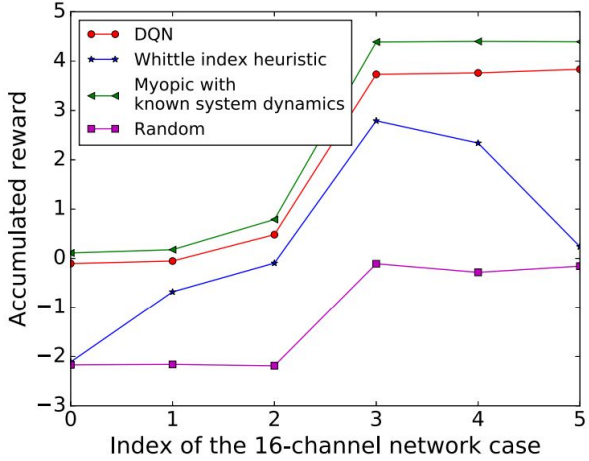


Fig. 4. Average discounted reward for 6 different cases. Each case considers a different set of correlated channels.

Evaluation with Real Data Trace

- Data collected using TelosB nodes, two motes were spaced at a distance of 20 meters, one acting as sender and the other as receiver
- Sender transmits one packet on all 16 channels, and the synchronized transmitter records the successful and failed attempts
- The only interference comes from surrounding WiFi networks that show high dynamic variability

Evaluation with Real Data Trace

- 8 channels were chosen that show significant WiFi interference
- Myopic policy is not considered as finding the transmission matrix of the entire system is computationally expensive
- Average accumulated discounted reward from the policies were:

0.947(DQN), 0.767(Whittle Index) and -2.170(Random policy)

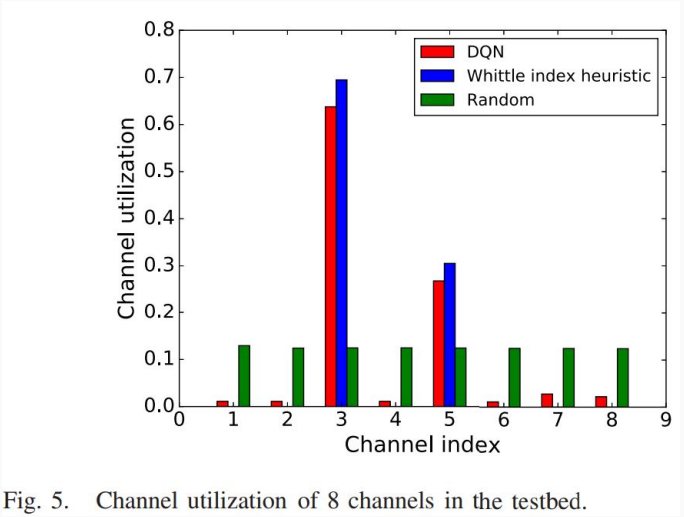


Fig. 5. Channel utilization of 8 channels in the testbed.

Evaluation in Time varying environments

- To make DQN more applicable in realistic, dynamic situations, DQN can check whether the policy already learned is still performing well
- If the performance of the current policy degrades below a certain threshold, DQN relearns

Algorithm 2 Adaptive DQN

```
First train DQN to find a good policy to operate with
for n = 1, 2, ... do
    At the beginning of period n
    Evaluate the accumulated reward of the current policy
    if The reward is reduced by a given threshold[4] then
        Re-train the DQN to find a new good policy
    else
        Keep using the current policy
```

Results

- DQN is able to relearn the new policy even after the environment changes(a different fixed-pattern channel switching case is followed)
- New DQN follows Algorithm 2

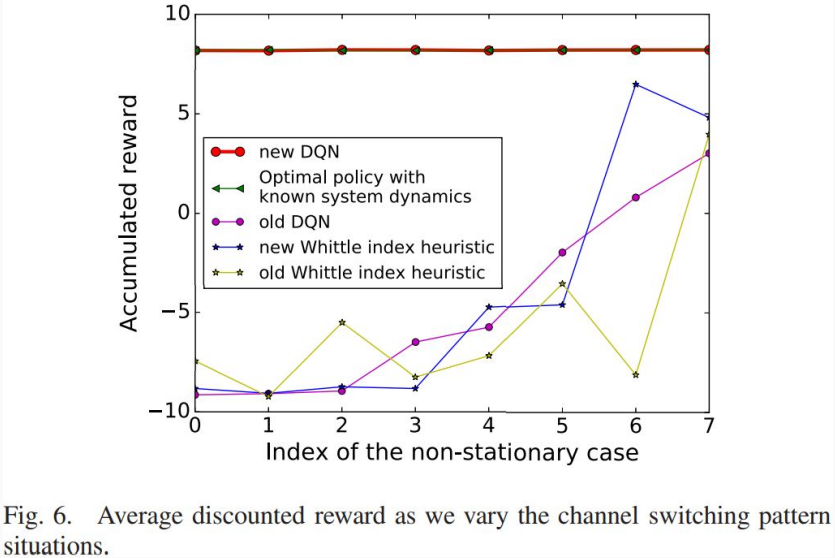


Fig. 6. Average discounted reward as we vary the channel switching pattern situations.

Results

- For first 10 iterations 8 channels were maintained to be good at each time slot
- At iteration 11, the environment changes to having only 1 channel good at each time slot
- Whittle Index (even manually trained to detect the change) does not perform well as it cannot make use of the correlation among channels

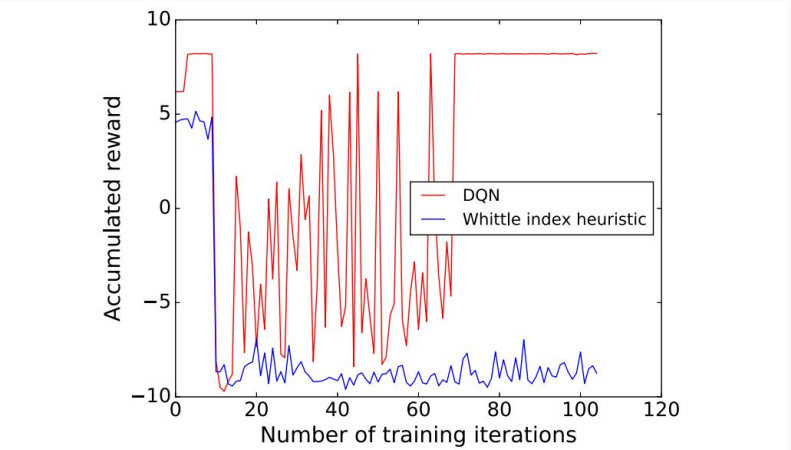


Fig. 7. Average discounted reward in real time during training in unknown fixed-pattern channel switching.

Conclusion

- We have discussed the dynamic multichannel access problem in a practical scenario.
- In the scenario where Channels are correlated and system statistics is unknown.
- Applied end-to-end DQN which utilizes historical observations and actions to find policy.
- Through simulations we show DQN is able to achieve same optimal performance even when system dynamics is unknown.
- Additionally, designed an adaptive DQN which can detect system changes and re-learn in non-stationary dynamic environment.