Improving Robustness of a Cognitive Radio Engine for Stationary and Non-stationary Environments

Hamed Asadi, Haris Volos, Michael Marefat, and Tamal Bose
Dept. of Electrical and Computer Engr.
The University of Arizona
Tucson, AZ 85721-0104
{hasadi, hvolos, marefat, tbose}@arizona.edu

***Abstract*—A Cognitive Radio Engine (CE) is an intelligent agent which observes the radio environment and chooses the best communication settings that best meet the application's goal. In this process, providing reliable performance is one of the major tasks in designing CEs for wireless communication systems. The main purpose of this work is providing predictable performance and controlling the cost of intelligent algorithms based on the CE's experience and complexity analysis respectively. In this work, we extend our meta-CE design to control the cost of computations and provide more reliable performance for providing the minimum requirement of the radio applications in different scenarios. To achieve this, we use robust training algorithm in two different levels alongside of the individual CE algorithms. The RoTA, enables radio to guarantee some minimum output performance based on the learning stages. RoTA uses confidence interval approximation for standard normal distribution to calculate the lower and upper bounds of CE's expected performance to analyze the reliability of decisions. Moreover, in the case of non-stationary environments, RoTA is facilitated by forgetfulness factor to provide minimum performance guarantees. The second level of RoTA operates in meta-level to control the amount of computation complexity of intelligent algorithms in all levels with respect to the obtained performance and complexity analysis.**

I. Introduction

A radio has traditionally used a fixed set of communication methods selected by its operator. Today, however, a radio is expected not only to use numerous communication methods, but also to be able to select the method (from a plethora of available methods) that best meets its objective in the current operating environment. Moreover, a radio is expected to utilize the most appropriate channel among the available channels for transmitting opportunistically, which provides the best conditions for radio's objective.

Typically, the radio designer will analyze each communication method in terms of the desired objective under assumed channel models. However, if the channel models do not hold or the communication methods perform in a way not expected, the design becomes irrelevant. The same applies when the desired objective changes.

The question then arises: What if a radio could be designed in such a way that it could determine on its own the best communication method to use to meet its objectives? The scope of our work [1] [2] [3] [4] [5] is to propose methods that can make such a design possible. This work is based on the pioneering work of Mitola, who first described the cognitive radio (CR). Mitola's ideal CR is not only capable of optimizing its own wireless capabilities, but also of self-determining its goals by observing its human operator's behavior [6]. In this work, we are only interested in optimizing the wireless capabilities of the radio by observing the environment and its own performance.

In the last few years, communication engineers have borrowed ideas from the fields of machine learning and Artificial Intelligence (AI) in an effort to make CR possible [7]. The agent who implements the methods that enable the radio to have the desired functionality is referred to as a Cognitive Engine (CE). This paragraph provides a brief overview of the state-of the art in CE design related work. The works of Rieser, Rondeau, and Le [8] [9] attempt to approach Mitola's vision by proposing a CE that deals with the user, policy and radio domains. Their designs are similar and based on the Genetic Algorithm (GA), Case Based Reasoning (CBR), and multi objective optimization principles. The work of [10] designed a CBR based CE for IEEE 802.22 WRAN applications also looking into both the radio and policy domains. Other works are more focused only on the radio domain. The work in [11] and [12] applied a GA and particle swarm optimization respectively to multi-channel links. Finally, [13] and [14] applied an Artificial Neural Network (ANN) and predicate logic respectively for learning and optimization of a wireless link.

The primary focus of CE related work was the ability of the CE to find the optimal configuration. The effects of this process received minimal attention. There are two major effects: learning speed and performance during learning. The former has received some treatment in the literature [9], [11], [12]. However, we are not aware of other works addressing performance during learning, which has two main aspects: aggregate performance during learning and providing bounded performance during learning. In our previous work [2] [3] we proposed a method that maximizes the aggregate performance during learning by minimizing the learning costs. The latter method also provides two level of learning to speed up the training process. In [15] we have provided a method to stabilize the performance of a CE within an operating window. However, the proposed methods make no attempt to provide a minimum performance guarantee during their training process. This aspect may lead to suboptimal performance, but more importantly can cause outages, which

is unacceptable for sensitive applications where bounded performance is paramount. This issue will be more critical under non-stationary environments that channel conditions change before reaching the optimal performance.

For example, an ambulance attempts to use its radio during a critical operation, expecting the radio to adapt in order to provide the required link. In this scenario, learning speed is important but maintaining a required reliability is absolutely paramount. In other words, the ambulance would rather have a slowly improving connection than a connection that erratically fluctuates from no throughput to high throughput. The same applies to a mobile user that watches a live stream of his/her favorite show and prefers to have a slowly improving connection than a link that causes pauses at crucial moments. Our proposed training algorithm makes the link stable during learning even under high dynamic environments.

The main purpose of this work is providing predictable performance and controlling the cost of intelligent algorithms based on the CE's experience and complexity analysis respectively. In our previous publication, we have proposed a meta-CE that is able to evaluate various CE algorithms' performance automatically [5] [3]. A meta-CE is generally considered to be comprised by a set of CE algorithms and meta-cognition module that provides the meta-abilities of the CE. We showed that Meta-CE is able to provide much higher performance than each individual CE algorithm due to its flexibility and ultimately predictability of the Cognitive Radio Systems (CRS). We also have proposed a several knowledge indicators to estimate the amount of knowledge that is obtained by various CE algorithms [4].

Providing predictable performance at all times is of paramount importance in different CE techniques. In this work, we extend our meta-CE design by using Robust Training Algorithm (RoTA) to provide more reliable performance for providing the minimum requirement of the radio applications in different scenarios. To achieve this goal, we use RoTA in two different levels alongside of the individual CE algorithms. The first level of RoTA operates with individual CEs to control the exploration and exploitation rate. The RoTA, in this level, enables the Meta-CE to guarantee some minimum output performance based on the learning stages. RoTA in this level, also uses confidence interval approximation for standard normal distribution to calculate the lower and upper bounds of CE's expected performance to analyze the reliability of decisions. Moreover, we propose a modified RoTA to provide a minimum performance level under non-stationary environments. The second level of RoTA operates in meta-level to control the amount of computation complexity of intelligent algorithms in all levels with respect to the obtained performance and operating scenarios (channel conditions, operating objectives, radio applications, etc.).

This paper is organized as follows: Section II, formulates the problem and presents an overview of some cognitive engine (CE) algorithms. Section III, presents the RoTA in stationary environments along with some results. Section IV, presents the modified RoTA to operate in non-stationary environments. Section V, provides a new metacognitive engine (meta-CE) framework to govern the computation cost of CE algorithms. Finally, Section VI provides some concluding remarks.

II. Problem Formulation

A primary function of the CE is to learn the capabilities of the radio. This is generally done by trial and error. When the radio performs this function, it is said to be exploring. On the other hand, when the radio is choosing methods with the best-known performance, it is said to be exploiting. The exploration operation consumes valuable resources such as time and energy and might significantly impact the link performance (i.e., dropped packets). One option to avoid these negative effects during the radio's operation is to put the CE through prolonged training sessions covering most expected operating conditions. However, even if the CE is assumed to go through prolonged learning (exploration) sessions, it is practically impossible to expose it to all possible channel conditions a priori specifically in the case of opportunistic spectrum access (OSA) that there will be more than a channel available in each decision step. Consequently, it is reasonable to expect that the CE sooner or later will face unknown conditions. In such a case, if the radio is operating in a critical mission it may not have the luxury of time to learn what is best before operating; it has to establish a connection and learn at the same time. Optimally balancing exploration vs. exploitation ensures that the negative effects of learning will be kept to a minimum.

The general problem in a link adaptation CE is that there is a list with a large number of possible communication methods that can be used. Each potential method is a discrete combination of modulation, coding, antenna techniques, and other possible parameters defining the communication method to be used. In this list, some of the methods are eligible and the rests are ineligible. An eligible method is a method that meets minimum performance requirements of the operation objective, and an ineligible method fails to meet those requirements given the current environment. The minimum performance requirements are typically a given PSR or bit rate (bits/s/Hz). Also it can be some minimum objective level such as throughput, power consumption, etc., For example, if a method has a 90% PSR in the current environment but the minimum required PSR is 95%, then this method will be ineligible. The goal in a link adaptation CE is to find the eligible method with the highest performance metric. In most cases, such as when the goal is to maximize bandwidth efficiency, the maximum potential performance of each configuration is already known. Therefore, the list of configurations can be potentially sorted by how well each item serves the current goal as apriori knowledge. In this case, the problem becomes a search through a sorted list.

The problem of exploration vs. exploitation is classically studied by using the mathematical framework of the Multi-Armed Bandit (MAB) problem. We introduce the MAB problem and we define how the distribution parameters are estimated recursively to be used in RoTA.

A. Multi-Armed Bandit

The MAB problem gets its name from the slot machines (bandits) found in casinos. A typical slot machine has a

single arm that when pulled returns a reward with a certain probability. In the multi-armed bandit problem it is assumed that the player is faced with either multiple machines or a single machine with multiple arms and his goal is to get the maximum reward by using the machines. Generally it is assumed that the player has little or no information about the bandits and he has to decide between exploring for the most rewarding machines and using the machine that was found to yield the higher reward. Essentially this is an information acquisition problem and the player is always faced with the same options.

Adapting the description found in [16], let Y be the set of K slot machines (comm. options), and let R_y be a random variable that gives the amount of the reward returned (capacity) if we use the machine y. Also let μ_y be the unknown true mean of R_y and σ_y^2 be the variance. Finally, let $(\bar{\mu}_y, \hat{\sigma}_y^2, n)$ be the estimate of the mean and the variance of R_y after n iterations and a belief state $\pi_y(n)$ which represents our knowledge about the underlying reward distribution of random variable R_y at a time step n. The estimates $(\bar{\mu}_y, \hat{\sigma}_y^2, n)$ can be an example of a belief state.

Let $x_y^n = 1$ if the y_{th} machine is played at iteration n and R_y^n the reward returned on that round. Also let

$$N_y^n = \sum_{y=1}^{n} x_y^n \tag{1}$$

be the total number of times the yth machine was used.

In the MAB problem we are looking for a policy that maximizes the expected return $V(s)$:

$$V(s) = E_\pi \sum_{n=1}^{N} \gamma^n R^n \tag{2}$$

where N is the maximum number of plays (often assumed to be ∞), E_y is the expectation operator over the belief state π, γ is a discount factor $o < \gamma < 1$, and R_n the return at time n. The discount factor is used to ensure a finite return when $N \to \infty$. Another interpretation is to treat $1 - \gamma$ as the probability that the process is going to stop [17]. Therefore, the discount factor is a way to express our expectation on the duration of the optimization horizon. A low value discount factor discounts future returns with a higher rate. As result, when balancing exploration vs. exploitation the latter has a higher weight. On the other hand, a high valued discount factor ($\gamma \to 1$) will make future rewards more important and exploration will have a higher weight than the previous case.

Finding the policy that maximizes 2 is a K-dimensional problem. There are several methods to address this problem such as ϵ-greedy, Softmax [18], and Gittins index [19] strategies that are presented in our previous works [3] [4]. Briefly, the ϵ-greedy strategy randomly explores the different communications methods with probability ϵ and uses the communication method with the highest average throughput with probability $1 - \epsilon$. The softmax strategy is probability matching that probability of choosing each communication method reflects the idea that the expected reward of a given communication method should match its actual probability of being the optimal method. The Gittins strategy maximizes the total sum of rewards collected over a long-term horizon. The strategy is simply to use the method with the highest Gittins index, which is based on the reward statistics of each method and must be estimated only when those statistics change (i.e., only when each method is used).

III. Robust Training

Learning is done by sending training packets. A training packet is defined as a packet that we are uncertain about its PSR θ. It is also assumed that the training packets carry payload data. Therefore, if the packet is successfully delivered it is going to contribute to the communication links throughput R.

During training, packets of a confident estimate of θ (that we are sure about their PSR θ) are also sent. Sending these packets makes it possible to maintain, during training, a θ_y or R_y (the estimated PSR and throughput by using y_{th} communication method) close to the minimum required PSR θ_{min} or throughput R_{min} during training. These packets shall be subsequently referred to as offsetting packets. Therefore, in this case the effect on θ and link's throughput R from the training packets is counterbalanced by the offsetting packets.

Let θ_{ly}, R_{ly} and θ_{uy}, R_{uy} be the lower and upper bound of PSR θ_y and throughput R_y of y_{th} communication method respectively. θmin is the minimum required PSR. R_{min} minimum required throughput (bits/s). The lower and upper bound of PSR is estimated by approximating Beta distribution using a normal distribution with a mean $\hat{\theta} = \frac{\alpha}{m'}$ and variance $\sigma^2 = \frac{\hat{\theta}(1-\hat{\theta})}{m'}$. The resulting confidence interval is given by [20]:

$$\hat{\theta}_y - z_{\delta/2}\sigma < \theta_y < \hat{\theta}_y + z_{\delta/2}\sigma \tag{3}$$

where $z_{\delta/2}$ is the $1 - \delta/2$ percentile of the standard normal distribution, α is the number of successful packets (trials), β is the number of unsuccessful packets, and $m' = \alpha + \beta$ the total number of transmitted packets. This approximation is valid when $1 < \alpha < m'$ and when both $n\theta$ and $n(1-\theta)$ are greater than five.

To estimate the lower and upper bounds of each communication method's throughput, we used the T distribution for $n < 30$ and normal distribution for $n > 30$ as follows:

$$\bar{\mu}_y(n) - t_{\delta/2,n'-1}\frac{\hat{\sigma}_y}{\sqrt{n'}} < R_y < \bar{\mu}_y(n) + t_{\delta/2,n'-1}\frac{\hat{\sigma}_y}{\sqrt{n'}} \tag{4}$$

$$\bar{\mu}_y(n) - z_{\delta/2}\frac{\hat{\sigma}_y}{\sqrt{n'}} < R_y < \bar{\mu}_k(n) + z_{\delta/2}\frac{\hat{\sigma}_y}{\sqrt{n'}} \tag{5}$$

where $t_{\delta/2,n'-1}$ is the $1 - \delta/2$ percentile of the T distribution by $n' - 1$ degree of freedom, and n' is the number of trials that we tried y_{th} communication method.

Previously, we classified communication methods to eligible and ineligible methods. Hereafter, by classifying eligible methods to two subclasses, we will have following communication methods' classes:

1) Offsetting methods
2) Ineligible methods
3) Training methods

First, the offsetting methods are the communication methods that satisfy following conditions:

$$\theta_{ly}(n) > \theta_{min} \quad \text{and} \quad R_{ly}(n) > R_{min} \tag{6}$$

Therefore, offsetting methods are the communication methods that we are confident about their minimum performances. Second, the ineligible methods are the communication methods that will have following conditions:

$$\theta_{min} > \theta_{uy}(n) \quad \text{and} \quad R_{min} > R_{uy}(n) \tag{7}$$

Hence, we are confident that the ineligible communication methods cannot provide the minimum required performances. Third, the Training methods are the communication methods, whose performances we are still not confident about. The training methods needs more trials to be classified in one of the aforementioned classes. Therefore, the training methods can be classified based on equation 8:

$$\theta_{uy}(n) > \theta_{min} > \theta_{ly}(n) \quad \text{and} \quad R_{uy}(n) > R_{min} > R_{ly}(n) \tag{8}$$

To implement the RoTA, we need to define a window of packets that the algorithm tries to keep their average window performance equal or higher than minimum applications requirements (θ_{min}, R_{min}). N_ω is the training window length in packets, N_T expected number of training packets within the training window. $\vec{\alpha}$ and $\vec{\beta}$ is the vector of the last N_ω successes and failures respectively. Finally, α_ω and β_ω the sum of the last $N_\omega - 1$ successes and failures respectively.

A. The Robust Training Algorithm

The RoTA is very straightforward: based on the last N_w successes and failures, training is performed when it is expected that $\theta \geq \theta_{min}$ and $R \geq R_{min}$. If there are no offsetting methods available, then there is no other option than to enter a training loop until an offsetting method is found. The RoTA finishes when there are no more training methods. In that situation, all K methods have either been evaluated or found to perform less than the best known achieving method that meets the minimum performance requirements (θ_{min}, R_{min}).

The pseudo code of RoTA is provided below:

Input: K methods, θ_{min}, R_{min}, and N_w
Result: Runs until no training methods exist

1: Compile the offsetting and training lists
2: $\alpha_w \leftarrow 0, \beta_w \leftarrow 0, R_w \leftarrow 0$
3: $\alpha \leftarrow \{\}, \beta \leftarrow \{\}, R \leftarrow \{\}$
4: **while not** training list is empty **do**
5: **while** offsetting list is empty **do**
6: **for** $i = 1$ to N_w **do**
7: find the training method with $\max\{R_{ly}\}$
8: use the training method
9: **if** $(\theta_{uy} < \theta_{min}$ **and** $R_{uy} < R_{min})$ **or** $(\theta_{ly} \geq \theta_{min}$ **and** $R_{ly} \geq R_{min})$ **then**
10: update the offsetting, training and ineligible lists
11: **end if**
12: **end for**
13: **end while**
14: **if** empty training list **then**
15: break while
16: **end if**
17: **for** $i = 1$ to N_w **do**
18: **if** $\frac{R_w}{N_w + 1} > R_{min}$ **then**
19: randomly pick a training method
20: use the training method
21: update α, β, R, α_w, β_w, and R_w
22: **if** $(\theta_{uy} < \theta_{min}$ **and** $R_{uy} < R_{min})$ **or** $(\theta_{ly} \geq \theta_{min}$ **and** $R_{ly} \geq R_{min})$ **then**
23: update the offsetting, ineligible, and training lists
24: **end if**
25: **else**
26: pick a high θ_{ly} offsetting method
27: use the offsetting method
28: update α, β, R, α_w, β_w, and R_w
29: **if** $\theta_{ly} < \theta_{min}$ **or** $R_{ly} < R_{min}$ **then**
30: break for
31: **end if**
32: **end if**
33: **end for**
34: update the offsetting, ineligible, and training lists
35: **end while**

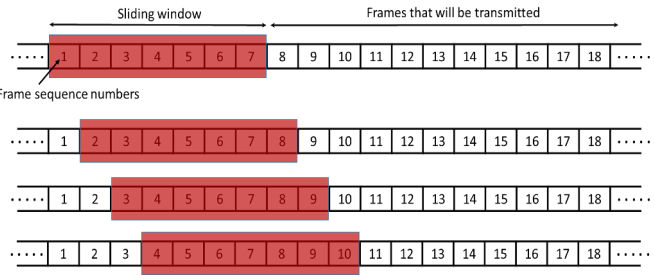

Fig. 1. RoTA Sliding Window

Some comments on the algorithm: first, the reason that operations in line 6 and 17 is limited to N_w is to avoid using one communication method that needs many trials to be classified as an ineligible or offsetting method. The break in the line 30 is for the case when an offsetting method is found to no longer satisfy minimum requirements θ_{min} and R_{min}. Figure 1 represents the main idea of RoTA. The RoTA tries to maintain the θ_{min} and R_{min} in the sliding window. For example, in line 18, $\frac{R_w}{N_w+1} > R_{min}$ looks ahead in the next step and estimates R assuming the next packet is dropped, if R_{min} is expected to be maintained, training is allowed because its potential negative effects are expected to be mitigated. Same in Figure 1, if the statement in line 18 is correct, it means for the next packet (packet number 8) the algorithm can use a training method because the dropped packet can be tolerated without violating minimum required throughput.

To facilitate training, an offsetting method with $R_{ly} \geq R_{min}$ is required. The greater the difference, the faster is the training rate, because more (potentially) failing packets can

be tolerated. The minimum training window that is expected to accomodate at least N_T training packets is estimated in [15].

B. Results

To evaluate the RoTA, we present a simple example. In this instance, we assume a 4 by 4 MIMO system with QPSK, 8PSK, 16, 32, 64, 128 and 256 QAM as a modulation type with eight error correction rates: 1, $\frac{7}{8}$, $\frac{3}{4}$, $\frac{2}{3}$, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{6}$ and $\frac{1}{8}$ and antenna techniques: VBLAST, STBC and MRC. For our channel scenarios, we consider an SNR in the range of 0-50 dB and the log_{10} of the eigen spread (κ) of the channel matrix in the range of 0-12. The CR also has 12 channels available with different SNR and bandwidth (either 1.25 or 2.5 MHz). At each time step, the CR can transmit over only one of the available channels.The minimum require PSR and throughput of the system is $\theta_{min} = 0.65$ and $R_{min} = 5(Mbits/sec)$. Thus, the CE algorithm has 1320 different actions (communication methods) to choose from, and the goal is finding the optimal action that provides the highest reward (here throughput) under noisy environment with respect of minimum required PSR θ_{min} and providing minimum required throughput R_{min} during operation.

Figure 2 illustrates the average results from a hundred different runs of proposed RoTA. In Figure 2 we can see the amount of achieved throughput and PSR, also the percent of remaining training methods vs. the total number of communication methods K, and also the instantaneous exploration rate of the algorithm. The exploration rate represents the percent that the algorithm randomly selects a communication method. Based on the result we can distinguish four distinct learning stages in the learning process of the proposed RoTA. The expectation and amount of robustness vary in different stages.

The first stage that we call it infant stage (highlighted by blue color), is the time that algorithm does not know any offsetting method. In this stage the algorithm has high exploration rates, which presents the amount of uncertainty in this stage. The algorithm cannot provide any performance guarantee at this stage. The second stage (highlighted by cyan color) is started by finding the first offsetting methods and will continue to stabilize the minimum required performance. During this learning stage, called childhood, the algorithm decreases the exploration rate and by using founded offsetting methods tries to guarantee the minimum required throughput and PSR. After stabilizing the performance, it's time to do more exploration. The third stage, called teenager (highlighted by red color), is the stage that the algorithm can transmit more training packets to classify the rest of the methods in training list. During this stage the algorithm grantees the minimum required performances. At the end of the teenager stage, the algorithm reaches near optimal performance. The fourth and last stage, called adult (highlighted by yellow color), is the stage that the algorithm obtained enough knowledge from previous stages. In this stage, algorithm not only provides the minimum required performance but also will find the optimal communication method with the highest throughput.

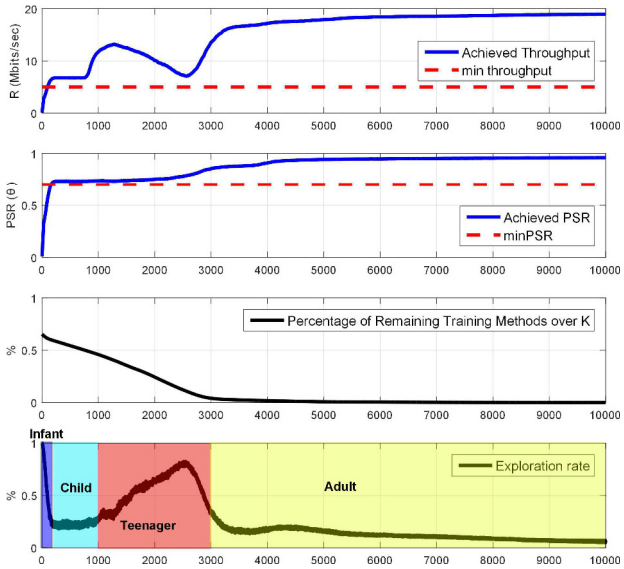

Fig. 2. RoTA Results

IV. Robust Training under Non-Stationary Conditions

The general problem that we address in this section is operating in non-stationary environments. Imagine a situation that is common in wireless communications, when the estimated channel conditions change rapidly. In such a case, the obtained knowledge by CE about the performance of communication methods will no longer be valid. For example, if the proposed RoTA figured out the optimal communication method(s) and classified all other methods as an ineligible or offsetting method, but the channel conditions change. In this situation, the knowledge obtained by RoTA does not provide the expected performance. Even worse is that the algorithm cannot even provide the minimum required PSR and throughput for the system. Furthermore, to provide the minimum required performances, the RoTA needs a many iterations to transfer its information to the new channel conditions. This happens because the obtained knowledge is the average of n trials, therefore to overcome the previous knowledge, the RoTA needs to try $m >> n$ trials to wipe out the outdated knowledge.

A. Forgetfulness Factor

To overcome the aforementioned problem, update rule of performance value estimation needs to change for communication methods. The main reason that the CE insists on its obtained knowledge is the number of trials (weights) that it has done to collect information. To decrease the weight of old information we suggest to use exponentially-wighted average method [21]. By using this method, RoTA tries to balance between the old and new information for each communication method. To estimate the mean and variance of communication methods' performances we use following equations:

$$\bar{\mu}_y(n) = \bar{\mu}_y(n-1) + \lambda[R_y(n) - \bar{\mu}_y(n-1)] \quad (9)$$

$$\hat{\sigma}_y^2(n) = (1-\lambda)(\hat{\sigma}_y^2(n-1) + \lambda(R_y(n) - \bar{\mu}_y(n-1))^2) \quad (10)$$

TABLE I
CHANNEL CONDITIONS CHANGING TIMES

Time Step								
1758	2790	3472	4133	5029	6039	7632	9066	9638

where λ is a positive parameter, $0 < \lambda < 1$, that can be either constant, or a function of time, $\lambda = \lambda_y(t) = \frac{1}{n}$. As n reaches ∞, by the law of large numbers $\bar{\mu}_y(n)$ converges to the actual mean that is suitable for stationary problems. Constant λ parameter method is called an exponentially-weighted average method [21] and is well-suited for non-stationary scenarios as the estimation of the means and variances never converge and continuously fluctuate in response of the environment changes. In the case of constant λ, the estimated values are more based on the current observed values. The equation 9 can be written in the form:

$$\bar{\mu}_y(n) = (1-\lambda)^n \bar{\mu}_y(0) + \sum_{i=1}^{n} \lambda(1-\lambda)^{n-i} R_y(n) \qquad (11)$$

From 11, the weight given to an observed value decreases exponentially as the age of observation increases. Since the λ parameter decays the old observations, we call it forgetfulness factor. The high valued forgetfulness factor $\lambda \to 1$ will make recent rewards more important and the algorithm forgets the previous values faster.

B. Results

To evaluate the effect of different values of forgetfulness factor on the performance of a RoTA, we use previous simulation's setup with some modifications. We assumed the CR has 5 channels available at each time step, therefore the CE has 550 various actions (communication methods) to choose from. The minimum required PSR and throughput of the system is $\theta_{min} = 0.6$ and $R_{min} = 8(Mbits/sec)$. For the non-stationary environment, we assume the channel conditions (SNR, eigen spread, and bandwidth) of all 5 available channels change at time steps mentioned in table I.

Figure 3 demonstrates PSR and throughput of the system with and without using forgetfulness factor. The plots on first row represents the RoTA without using forgetfulness factor, however the plots on second row shows the RoTA, which is facilitated by forgetfulness factor ($\lambda = 0.2$). As we can see, when RoTA uses forgetfulness factor, it's able to recover itself much faster.

Figure 4 shows the aggregated results of two aforementioned CEs. Plot (a) represents the amount of total data transfered with two CEs. The algorithm with forgetfulness factor transfers almost twice more than the stationary RoTA. Plot (b) shows the amount of data that we lost because of not meeting the minimum required performance by two CEs. Also, plot (c) represents the amount of time step that each CE was under the minimum required performances. As we can see, the CE with forgetfulness factor was not able to provide minimum performance just 4% of the total time steps, however the other CE could not meet the minimum performance more than 60% of the time steps.

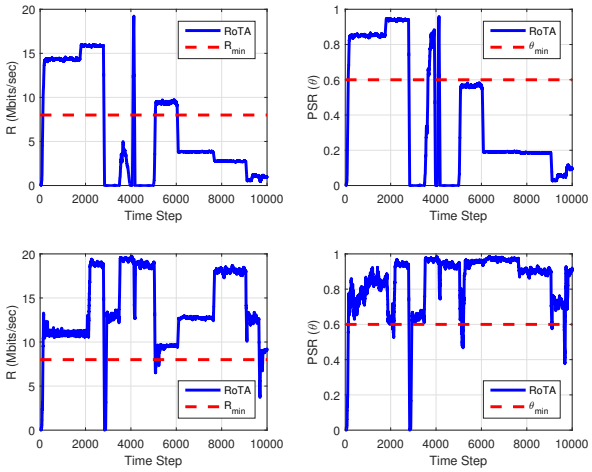

Fig. 3. RoTA in Non-stationary Conditions Results

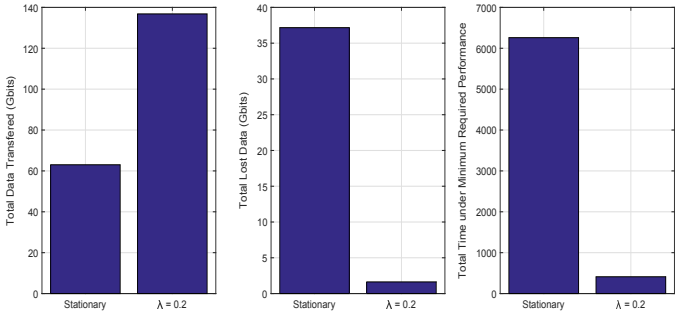

Fig. 4. Non-stationary Aggregated Results

To compare the effect of different values of λ we consider λ in the range of 0-0.95 with 0.05 steps, and for each λ we average the output of hundred iterations. Figure 5 represents the results for various forgetfulness factors. The figure shows the values between 0.1 and 0.3 provide better outputs. For instance, the CE with $\lambda = 0.25$ has the minimum lost data, however the CE with $\lambda = 0.1$ spends to recover the performance.

V. ROBUST TRAINING IN META LEVEL

The two proposed robust training algorithms operate in the level of individual CEs, and the goal was providing some guaranteed performances in stationary and non-stationary channel conditions. However, in this section, we want to use RoTA in the level of metacognition [3] [5]. One of the most criticism against using learning algorithms for wireless communications is the computation cost and complexity of these kinds of algorithms that will be added to the system. Although, the proposed learning algorithms based on the MAB are not computationally expensive, we propose a meta-CE framework to govern the cost of individual CEs.

Figure 6 illustrates the framework of a meta-CE, which employs various individual CE algorithms. The meta-CE is facilitated with a meta module to control the complexity of the learning algorithms by using a proposed robust training

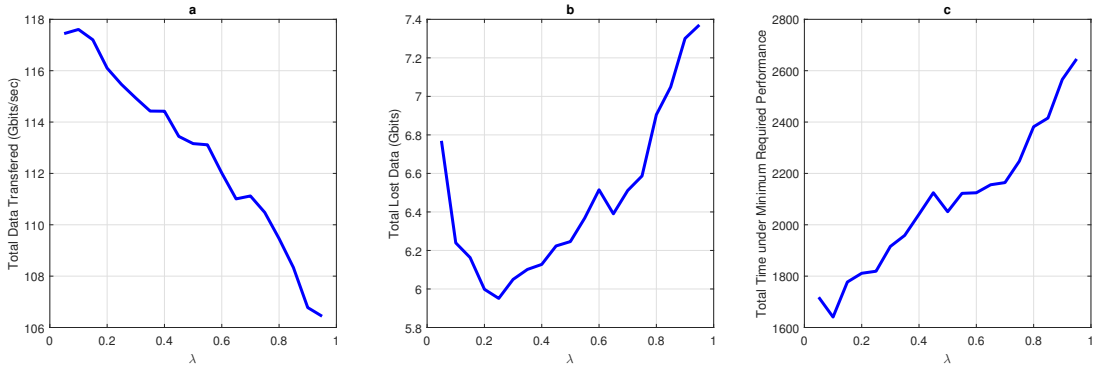

Fig. 5. Different Forgetfulness Factors

algorithm. The meta-CE keeps the computational cost of the CR in a certain level by turning down the operation of CE algorithms when there is not enough resources.

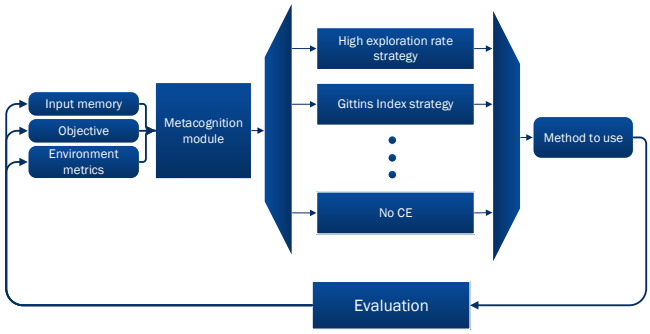

Fig. 6. RoTA Meta-CE Framework

The proposed framework is also able to be operated with two levels of RoTA as an individual CE to guarantee minimum required performance, and the other level as a metacognition module to maintain the computation cost of a CR.

VI. Conclusions

In this paper, we made three important contributions: first, we proposed the novel Robust Training Algorithm (RoTA) to guarantee a minimum performance. RoTA allows the CE to learn by combining training packets of unknown performance with offsetting packets of known performance. RoTA is valuable for applications that relatively smooth operating links are of paramount importance.

Secondly, we propose some tuning factor (forgetfulness factor) to make the RoTA appropriate for non-stationary environments. The forgetfulness factor helps RoTA to give higher weights to new observations and forget the outdated information faster.

Thirdly, we proposed a new meta-CE framework to control the computation cost of the individual CE algorithms. The meta-CE operates RoTA in two distinct level to maintain the complexity and minimum required performance of cognitive radio systems.

VII. Acknowledgements

This project was partially supported by the Broadband Wireless Access and Applications Center (BWAC); NSF Award No. 1265960.

References

[1] H. I. Volos and R. M. Buehrer, "On Balancing Exploration vs. Exploitation in a Cognitive Engine for Multi-Antenna Systems," in *Proceedings of the IEEE Global Telecommunications Conference*, Nov. 2009, pp. 1–6.

[2] ——, "Cognitive Engine Design for Link Adaptation: An Application to Multi-Antenna Systems," *IEEE Transactions on Wireless Communications*, vol. 9, no. 9, pp. 2902–2913, Sept. 2010.

[3] H. Asadi, H. Volos, M. Marefat, and T. Bose, "Learning Characterization Framework and Analysis for a Meta-Cognitive Radio Engine," in *Proceedings of SDRWInnComm 2014 Wireless Innovation Conference on Wireless Communications Technologies and Software Defined Radio*, Mar. 2014, pp. 132–139.

[4] ——, "On Quantifying the Experience Level of a Cognitive Engine," in *Proceedings of SDRWInnComm 2015 Wireless Innovation Conference on Wireless Communications Technologies and Software Defined Radio*, March 2015, pp. 9–15.

[5] ——, "Metacognitive radio engine design and standardization," *Selected Areas in Communications, IEEE Journal on*, vol. 33, no. 4, pp. 711–724, April 2015.

[6] J. Mitola, III, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," Ph.D. dissertation, The Royal Institute of Technology (KTH), Stockholm , Sweden, May 2000.

[7] A. He, K. K. Bae, T. R. Newman, J. Gaeddert, K. Kim, R. Menon, L. Morales-Tirado, J. J. Neel, Y. Zhao, J. H. Reed, and W. H. Tranter, "A Survey of Artificial Intelligence for Cognitive Radios," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 4, pp. 1578–1592, May 2010.

[8] C. J. Rieser, "Biologically Inspired Cognitive Radio Engine Model Utilizing Distributed Genetic Algorithms for Secure and Robust Wireless Communications and Networking," Ph.D. dissertation, Virginia Tech, 2004.

[9] T. W. Rondeau, "Application of Artificial Intelligence to Wireless Communications," Ph.D. dissertation, Virginia Tech, 2007.

[10] A. He, J. Gaeddert, K. Bae, T. R. Newman, J. H. Reed, L. Morales, and C. Park, "Development of a Case-Based Reasoning Cognitive Engine for IEEE 802.22 WRAN Applications," *ACM Mobile Computing and Communications Review*, vol. 13, no. 2, pp. 37–48, 2009.

[11] T. R. Newman, B. A. Barker, A. M. Wyglinski, A. Agah, J. B. Evans, and G. J. Minden, "Cognitive Engine Implementation for Wireless Multicarrier Transceivers," *Wiley Journal on Wireless Communications and Mobile Computing*, vol. 7, no. 9, pp. 1129–1142, 2007.

[12] Z. Zhao, S. Xu, S. Zheng, and J. Shang, "Cognitive Radio Adaptation Using Particle Swarm Optimization," *Wireless Communications and Mobile Computing*, vol. 9, no. 7, pp. 875–881, 2009.

[13] N. Baldo and M. Zorzi, "Learning and Adaptation in Cognitive Radios Using Neural Networks," in *5th IEEE Consumer Communications and Networking Conference*, Jan. 2008, pp. 998–1003.

[14] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea, "Applications of Machine Learning to Cognitive Radio Networks," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 47–52, August 2007.
[15] H. I. Volos and R. M. Buehrer, "Robust Training of a Link Adaptation Cognitive Engine," in *IEEE Military Communications Conference*, Oct. 31–Nov. 3. 2010, pp. 1318–1323.
[16] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2007.
[17] I. M. Sonin, "A generalized Gittins index for a Markov chain and its recursive calculation," *Statistics & Probability Letters*, vol. 78, no. 12, pp. 1526–1533, September 2008.
[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.
[19] J. C. Gittins, *Multi-Armed Bandit Allocation Indices*. Wiley, Chichester, NY, 1989.
[20] L. M. Leemis and K. S. Trivedi, "A comparison of approximate interval estimators for the bernoulli parameter," *The American Statistician*, vol. 50, no. 1, pp. 63–68, 1996.
[21] D. Koulouriotis and A. Xanthopoulos, "Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems," *Applied Mathematics and Computation*, vol. 196, no. 2, pp. 913 – 922, 2008.