# FIT: On-the-Fly, In-Situ Training for SNR-Based Rate Selection

Hui Liu, Jialin He, Sabyasachi Gupta, Dinesh Rajan, *Senior Member, IEEE*, and Joseph Camp, *Member, IEEE*

***Abstract*—Existing rate adaptation protocols have advocated training to establish the relationship between channel conditions and the optimum modulation and coding scheme. However, innate with in-field operation is encountering scenarios that the rate adaptation mechanism has not yet encountered. Frequently, protocols are optimally tuned for indoor environments but, when taken outdoors, perform poorly. Namely, the decision structure formed by offline training, lacks the ability to adapt to a new situation *on the fly*. The changing wireless environment calls for a rate adaption scheme that can quickly infer the channel type and adjust accordingly. Typical SNR-based rate adaptation schemes do not capture the nuance of the performance variable in different channel types. In this paper, we propose a novel scheme that allow SNR-based rate selection algorithms to be trained online *in the environment* in which they are operating. Inspired by the idea that, to do well, an athlete must train for the type of athletic event and environment in which they are competing, we propose FIT, an on-the-fly, in-situ training mechanism for SNR-based protocols. To do so, we first propose the FIT framework which addresses the challenges of making rate decisions with unpredictable fluctuation and lack of repeatability of real wireless channels. To distinguish between channel types in the training, we then characterize wireless channels according to the link-layer performance and introduce a novel, computationally-efficient, channel performance manifold matching technique to infer the channel type given a sequence of throughput measurements for various link-level parameters. To evaluate our methods, we implement rate selection which uses FIT for training alongside channel performance manifold matching. We then perform extensive experiments on emulated and in-field wireless channels to evaluate the online learning process, showing that the rate decision structure can be updated as channel conditions change using existing traffic flows. The experiments are performed over multiple frequency bands. The proposed FIT framework can achieve large throughput gains compared to traditional SNR-based protocols (8X) and offline-training-based methods (1.3X), particularly in a dynamic wireless propagation environments that lack appropriate training.**

***Index Terms*—Rate Adaptation, Online Training, In-situ Measurements, Context Awareness, Channel Performance Manifold Matching**

This work was supported in part by NSF grants CNS-1526269 and CNS-1823304 and Air Force Office of Scientific Research grant FA9550-19-1-0375. This work was presented in part at IEEE WCNC, New Orleans, USA, March 2015 [1] and Wiopt, Tempe, USA, May 2016 [2].(Corresponding author: Sabyasachi Gupta.)

Hui Liu is with Michigan State University, MI 48824 (e-mail: huil@smu.edu).

Jialin He is with Microsoft, Redmond, WA 98052 (e-mail: jhe@smu.edu).

S. Gupta, J. Camp, and D. Rajan are with the Department Electrical and Computer Engineering, Southern Methodist University, Dallas, TX 75275 USA (e-mail: sabyasachig@smu.edu; rajand@smu.edu; camp@smu.edu).

## I. INTRODUCTION

IEEE 802.11 networks typically use significantly larger packets and resulting channel access times than cellular networks to overcome the overhead of transmitting the packet header. To overcome channel fading in this context requires some form of feedback to the transmitter to alert the need for rate changes as the channel quality fluctuates. Hence, a number of rate adaptation approaches have been proposed to improve the spectral efficiency when the channel quality improves and simply remain connected when channel quality degrades. One class of rate adaptation mechanism used in practice is loss-based and suffers from poor rate decisions due to a confusion between collision-based and channel-based losses [3]–[7]. A second type of rate adaptation uses channel quality directly (i.e., SNR-based protocols) and can distinguish between such losses. However, to perform optimally, SNR-based schemes need to be trained in each environment in which it is used [8]–[12]. If the training is done offline, even excessive levels of training may not cover all scenarios. Further, even with a comprehensive rate decision structure for all contexts, the training still may not be similar across device types (i.e., the decision structure is likely hardware dependent).

Rate adaptation protocols are just beginning to use context information, considering vehicular or pedestrian speed, or the direction of motion [13]–[15], and/or channel type information. Machine learning techniques can be used to extract relationships between the context information, the settings of the transmitter and receiver, and the resulting wireless performance [16]–[19]. The established rate decision structure can suggest the potential optimal settings for the transmitter-receiver pair according to the channel condition efficiently without adding overhead. Moreover, even with such a comprehensive decision structure, the rate adaptation mechanism still has the challenge of distinguishing between environments to leverage the training. To recognize these different contexts in which training has occurred, a mechanism might isolate training to given geographical regions and/or times of day. However, such a splitting of training data would require an extensive database (and extensive training).

To address the aforementioned challenges, in this paper, we propose on-the-fly, in-situ training (FIT) for SNR-based rate selection protocols. FIT can leverage an existing decision structure to look up the optimal transmitter-receiver settings and simultaneously update the decision structure using the measured performance of existing data transmissions as training for the immediately-upcoming rate decisions. In essence, FIT leverages the no-overhead advantage of loss-based rate adaptation protocols to adapt the SNR-based rate adaptation protocol. We implement the FIT training mechanism, and SNR-based protocol structure on a hardware platform to experimentally evaluate our approach with emulated and in-field training. The proposed framework is the first framework for SNR-based rate adaptation that trains in an online manner, immediately using in-field data. The main contributions of this paper are as follows:

- We propose FIT, an online classification-based rate adaptation framework. We utilize online machine learning techniques to make quick and reliable decisions about the modulation and coding scheme. The decision structure for rate adaptation is adjusted to the dynamic fluctuations of wireless channels efficiently when new context information is collected using wireless performance information from existing data transmissions (i.e., no additional overhead packets to train for rate decisions are needed). To do so, the training process has been modified from optimal transmission mode prediction to quantized throughput prediction to efficiently use training data and due to the inability of the channel to stay the same while progressing through all transmission schemes.
- The wireless performance changes with the context setting such as indoor, outdoor, or vehicular environments. In order to address the potentially infinite different scenarios that might be encountered, we define the notion of channel type. Channel type (details in Section III) is used to conveniently categorize the channels based on their measured link-layer performance rather than low-level channel tap-delay profile/Doppler information. We introduce a computationally-efficient, channel performance manifold matching framework to infer channel type given a sequence of throughput measurements for various link-level parameters. This proposed method succinctly captures the relationships between the performance of the various modulation and coding schemes without any additional overhead on data transmission.
- Based on the channel type inference, we decipher which online decision structure to contribute to in the knowledge of the optimal modulation and coding scheme to use in that particular context. Such an approach removes confusion of the decision structure that occurs when training the same structure across diverse wireless environments. This approach also allows previously encountered channels to pick up the training where it left off and form new decision structures where there is a lack of training.
- We implement the proposed FIT framework on an off-the-shelf platform and test its performance to demonstrate the impact of in-situ training updates on rate selection performance. FIT improves the throughput of rate adaptation by up to $822.08\%$ over a purely SNR-based method (i.e., one that does not use context information) and by $38.64\%$ over an offline-based method.

The remainder of this paper is organized as follows. In Section II, we first discuss the system model and offline training method for rate selection using decision tree. Then, we propose an online learning framework for rate adaptation and data collection procedure for the online training. In Section III, we propose a channel performance manifold matching algorithm to classify an unknown channel into one of the channel models in the training set. In Section IV, we evaluate the performance of the online training for rate selection and channel performance manifold matching strategy with experiments on emulated and in-the-field channels. Related work is presented in Section V, and the paper concludes in Section VI.

## II. System Model

The objective of this paper is to develop a framework for online learning in adaptive wireless protocol. First, we introduce the notations used. Let set $\mathcal{A} = \{A_1, A_2, \ldots, A_L\}$ represent the various attributes or contextual information that are available in a particular system. For instance, SNR, velocity, and channel type are considered as the attributes in our offline training schemes as we will discuss later. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$ represent the set of transmission modes (i.e., modulation and coding schemes) available in a given system. For instance, for the Ubiquiti XR2, a mini-PCI WLAN card [20] employed in our experimental section for radio communication, $\mathcal{M}_{Ubiquiti}$ represents the supported 802.11a/g-compliant coding and modulation pairs.

In this paper, the optimization metric of interest is the measured throughput $G$. We use the notation $G(m_i)$ to denote the throughput of the $i$-th transmission mode. This throughput is calculated as

$$G(m_i) = (1 - PER(m_i)) \cdot R_{th}(m_i) \cdot \frac{l_{payload}(m_i)}{l_{packet}(m_i)} \tag{1}$$

where $PER(m_i)$ is the measured packet error rate, $R_{th}(m_i)$ is the physical layer data rate, and $l_{payload}(m_i)$ and $l_{packet}(m_i)$ are the size of the payload and packet, respectively.

In traditional SNR-based adaptation, the objective is to select the optimal transmission mode, $m^*$ as

$$m^* = arg\max_{m_i} G(m_i), \text{given} A_1 \equiv SNR \tag{2}$$

This problem is typically solved using prior training to generate a look-up table from the set of attributes to the optimal mode selected. In case the received SNR is the sole attribute, the look-up table is efficiently represented by a set of thresholds on the SNR where each mode is optimal.

The attributes we consider are received SNR, node velocity and channel type (which is formally defined in Section III). In short, two different channel conditions are said to belong to the same channel type if their effective performance (as measured by the throughput) of the various modes exhibit similar behavior for all values of the various attributes.

We propose two different approaches to solve the new optimization problem. In the first method, we use offline training to generate an appropriate mapping from the set of attributes to the optimal mode selected. Two different flavors of this method, which differ in their computational complexity and amount of training required, are discussed. In the second method, we use online, in-situ training to continually refine the mapping from attributes to the optimal mode choice. In

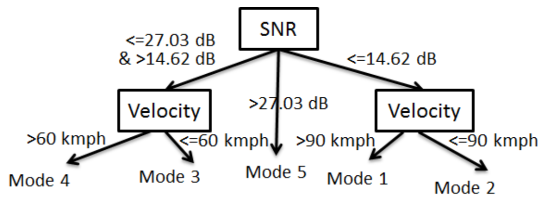


Fig. 1. A decision tree for rate adaptation.

the sequel, we describe the details of the proposed offline and online training methods.

## *A. Offline Training for Attribute Based Rate Selection*

Adaptive learning algorithms have been widely investigated in the literature [21]–[23]. Among classical learning methods, the decision tree is popular due to its low complexity and high accuracy of prediction. Such algorithms typically use training data to derive an empirical relationship between the desired output (class) and the inputs (attributes). The resulting relationship is represented using a decision tree as shown in Figure 1. Once the tree-based classifier is generated, adapting its structure requires a repetition of the training process with a newer or enlarged training set.

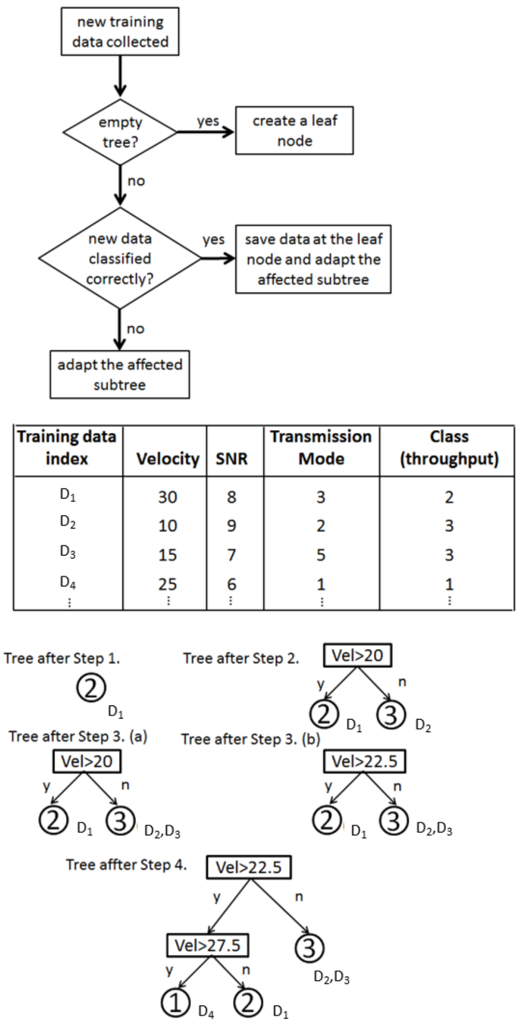
The offline learning algorithm that we use is the popular C4.5 algorithm that is based on the entropy gain ratio. During the offline training phase, the C4.5 algorithm is used to partition the labeled training instances based on the information entropy metric, until all instances have been assigned to the leaf node of the tree [24]. Each leaf node has a specified class label. The ability of the decision tree to handle multiple classes and missing values makes it suitable for our application. During the adaptation phase, this tree serves as the look-up table to decide the optimal transmission mode for each set of attributes. For instance, when the received SNR is between (15 dB, 32 dB] and the velocity is higher than $45$ kmph, Mode 3 is suggested as the optimal mode. We now describe two different approaches to generate this decision tree.

**Training With Exhaustive Information.** In this case, we consider that the training set includes the knowledge of the optimum transmission mode to use for each combination of attribute values. For instance, the training data could include $X$ records of $\mathcal{A} = \{SNR, \text{velocity}, \text{channel-type}\}$ values and the optimal transmission mode in each of these cases.

**Training with Partial Information.** In this case, the training set includes knowledge of the throughput (or any desired performance metric) for only a random subset of modes for each combination of the attributes. Since there are $|\mathcal{M}|$ modes available in the system, we restrict the amount of training data for each combination of the attributes to no greater than $L|\mathcal{M}|$. One key advantage of using this method is that it lends itself to an immediate online extension.

## *B. In-Situ Training*

We develop an online version of the decision tree based learning algorithm to adapt the classifier to changing environments. Our algorithm is based on the incremental learning algorithm in [23] and is able to efficiently update the existing tree with new training data. Since only one rate can be used at a given time, the optimal transmission mode is not available on-the-fly. Consequently, we alter the input and the output of the training so that the algorithm now takes the current transmission mode as an input and the quantized version of the throughput as the output (instead of the optimal mode).



| Training data index | Velocity | SNR | Transmission Mode | Class (throughput) |
|---|---|---|---|---|
| $D_1$ | 30 | 8 | 3 | 2 |
| $D_2$ | 10 | 9 | 2 | 3 |
| $D_3$ | 15 | 7 | 5 | 3 |
| $D_4$ | 25 | 6 | 1 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Fig. 2. The flow chart of learning a new data.

The classifier resulting from our algorithm is represented as a binary tree, which means every decision node can lead to two outcomes. Each branch can be further divided into sub-trees. The general procedure and example of incorporating new data into an existing tree is shown in Fig. 2. The merit of efficiently updating only the sub-tree would become more clear when the decision structure grows with more training data. As shown in the example in Fig. 2, a test condition based on one of the attributes of the training data is saved at each decision node. Data $D_1$ is learned when the tree is empty. Therefore, the tree is represented as a leaf node, which indicates the class of data $D_1$. If new training data is ready when the tree is not empty, the data is first classified following branches in the existing tree-like classifier. If the existing tree is not able to classify the data correctly, the current leaf node will be converted into a decision node using the test condition based on the attribute selection metric. In the example, the only leaf node formed in Step 1, is converted into a decision node to classify data $D_1$

and data $D_2$ in Step 2. If the classifier can predict the class label with its attributes correctly, the data will be added to the set of retained data related to the leaf node. Further, the test condition as well as the structure of the sub-tree involved in the classification of the new data, will be adjusted based on the changed data set using Quinlan's information gain ratio as the metric [24]. In Step 3 of the example, data $D_3$ is added and the test condition is adjusted.

If the value for one of the attributes is missing, the data will be passed down the branch that does not satisfy the test condition. If any operations change a data group, the algorithm will check if test condition based on the data group is the optimal condition. The order for checking the optimality is from the root node to its sub-trees. The updating process can be paused and resumed, and the tree-like classifier can be saved at any time.

### C. Training Data Collection

Let $\mathcal{R} = \{r_1, r_2, \dots, r_R\}$, $\mathcal{V} = \{v_1, v_2, \dots, v_V\}$, and $\mathcal{S} = \{s_1, s_2, \dots, s_S\}$ represent, respectively, the sets of channel types, velocities, and SNR values in the training set. In the training phase, we measure the throughput $G_{m,s,v,r}$ that the system can achieve using mode $m$ in the channel type $r$ with the velocity $v$ and SNR $s$. For each model $r$, the $V \cdot S \cdot M$ throughput values constitute the performance data $p_r$. A complete training set $\mathcal{P} = \{p_1, p_2, \dots, p_R\}$ contains $R \cdot V \cdot S \cdot M$ throughput values, which describes the system performance in these $R$ channels.

The decision structure in the offline method models the relationship between the contextual information and the optimum mode. To derive the decision structure, the offline-training-based method requires the knowledge of the best mode $m^*$ in the training data which can be obtained as,

$$m^* = arg \max_{m_i} G(m_i)$$
$$\text{given} A_1 \equiv SNR, A_2 \equiv \text{velocity}, A_3 \equiv \text{ch-type} \quad (3)$$

One major challenge for in-situ training is that the optimal mode is unknown in practical situation. Typically, only the performance of one or at best a subset of modes is available in each dynamically changing context. Our solution is to directly learn the relationship between the context information and the throughput of each mode. Each training data is just the raw measured data which is represented as,

$$\{SNR, velocity, channel\ type, m_c, G(m_c)\} \quad (4)$$

where $m_c$ is the mode used in current scenario. The accumulative training process will result in a trained decision structure, with which the throughput for all modulation and coding schemes under a certain scenario can be predicted.

### D. In-Situ-Training for SNR-based Rate Selection

Based on our solutions, we propose, FIT, our in-situ training framework for SNR-based rate selection. Instead of directly deriving the function of link adaptation by learning algorithm as

$$f_{\text{offline}} : \{s_i, v_i, r_i\} \to m^*, \quad (5)$$

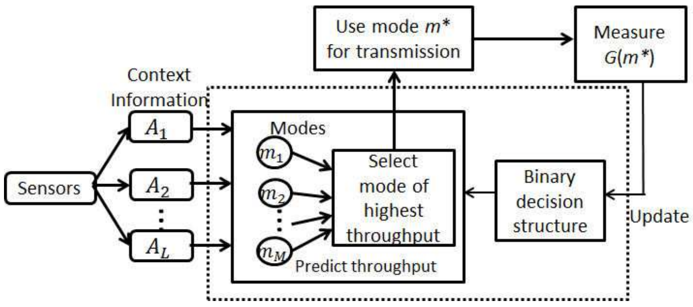


Fig. 3. FIT framework.

first, we derive the function below using online learning algorithm as

$$f_{\text{online}} : \{s_i, v_i, r_i, m_i\} \to G(m_i), \quad (6)$$

Then, given the collected context information, (6) is able to predict the throughput for all the modes. Subsequently, the best mode $m^*$ is selected by comparing the predicted throughput values of all modes.

The framework of FIT is depicted in Fig. 3. With the collected context information $\{A_1, A_2, \dots, A_L\}$, the decision tree predicts all the throughput values $\{G(m_1), G(m_2), \dots, G(m_M)\}$ corresponding to the transmission modes. After comparing those values, the mode $m^*$ with the highest throughput is employed by the transmitter/receiver. Finally, the binary decision tree partially updates the decision structure with the new training data, $\{s_i, v_i, r_i, m^*, G(m^*)\}$.

For the proposed FIT, we add the mode index as one attribute and directly predict the throughput. This strategy makes the in-situ training data collection possible and also substantially distinguishes our proposed framework from the existing link adaptation framework based on offline training. Compared to the existing offline methods, the advantages of FIT are clearly evident. First, FIT can save computational resources both for the initial training data collection and subsequent periodic retraining. Further, through incrementally learning from the newly collected data, our proposed framework adapts to the dynamic environment.

We observe that, given a particular environment, all training data points based on the context information, SNR, velocity, and mode are equally important to model the throughput performance. Hence, the proposed FIT based rate decision structure equally weights data points for each observed environment. In the next section, we propose a mechanism, which is useful to detect a new environment, such that a new rate decision structure is constructed based on new data points in that environment.

## III. Channel Performance Manifold Matching

As noted in Section II, the proposed FIT framework requires knowledge of the channel type to distinguish various environments as well as knowledge of SNR and velocity for the rate decision training. Channel type is a simpler way to represent the main characteristics of the infinitely varied propagation environment, such as fading, multipath, and doppler. In a wireless system, SNR can be measured by the channel estimation module. Additionally, many mobile devices can now obtain velocity using an on-board GPS module. However, the channel type information cannot be directly obtained. In this section, we introduce a channel performance manifold matching algorithm that leverages the geometrical structure of the performance curves and classifies an unknown channel into one of the channel models in the training set. The algorithm also provides a confidence level for this classification, and if a minimum desired confidence level is not obtained, the algorithm flags the current measurement location as a new channel type that has not been previously encountered. Subsequently, an additional decision tree or sub-tree for the new channel type is trained.

## A. Proposed Algorithm

In practical system, depending on available memory, only the most frequently used channel types can be stored. Let $\mathcal{C}$ denote the countably infinite set of all possible channel realizations in the real world. Let $\mathcal{P}$ contain the performance data $p_{c_i}$ for each $c_i \in \mathcal{C}$. Each $p_{c_i}$ is a vector of size $|V \cdot S \cdot M|$ that represents the transceiver's performance using different transmission parameters in channel $c_i$. Let $\mathcal{T}$ represent the set of all possible tree structures introduced in Section II. Each $t \in \mathcal{T}$ can be regarded as the outcome of the following mappings: $c \mapsto p \mapsto t$. If $p_{c_i}$ and $p_{c_j}$ $(i \neq j)$ always have identical transmission modes that achieve maximum throughput, then they are mapped onto the same tree structure. Our channel performance manifold matching algorithm is designed to solve the problem of classifying a random channel $c$ into the subset $C_k \subset \mathcal{C}$, within which all the channels are mapped to the same sub-tree structure $t_k$. All such channels $C_k$ are defined as belonging to a particular channel type.

We use the throughput as the performance metric and consider the same training set as described in Section II. The channel performance manifold matching relates the throughput of different transmission modes to the context information. Meanwhile, it also searches for the throughput of all the channel models in the training set, which have the same context information. Finally, it calculates the *similarity* of the geometric shape of the throughput points measured in a test case with those of different channel models in the training set. The algorithm selects the most similar model as the channel model for the current test channel and uses this result along with other context information for optimal mode selection using the decision tree.

Using the same training set $\mathcal{P}$ as in Section II, we now use $N_I(N_I \geq 2)$ throughput values measured in the real channel, whose channel type is unknown, and their related context information to classify the channel into one of the $R$ channels. For each $i(i = 1, \ldots, N_I)$, $\hat{G}(m_i)$ denotes the measured throughput, using mode $\hat{m}_i$, and $\hat{s}_i, \hat{v}_i$ represent the corresponding SNR and velocity, respectively. Then, we search the training set and find the throughput of all the channel models with the same context information and transmission mode. In other words, we look for $G_{r,\hat{v}_i,\hat{s}_i,\hat{m}_i}$ of all $r \in R$ channel models. It is possible that $\hat{v}_i$ is not in $\mathcal{V}$ (same for $\hat{s}_i$). In this case, a linearly interpolated throughput $\tilde{G}_{r,\hat{v}_i,\hat{s}_i,\hat{m}_i}$ is calculated based on the throughput of the nearest velocity and SNR values.

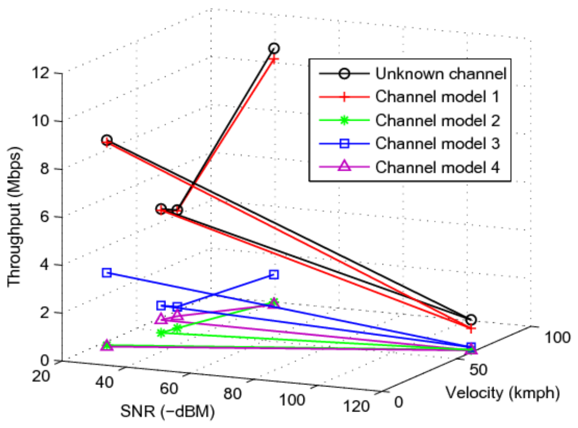


Fig. 4. Plot of the throughput versus SNR and Velocity for different channels.

The training set for a particular channel ($p_r$) is a family of 3-D meshes, where each mesh represents the throughput of a certain transmission mode with respect to SNR and velocity. With interpolation, the meshes can be filled and turned into surfaces. By labeling all the $\tilde{G}_{r,\hat{v}_i,\hat{s}_i,\hat{m}_i}$, we can draw $R$ curves as shown in Fig. 4, indicating the throughput variation trend in the order of $i$ from 1 to $N_I$. Similarly, we plot the throughput for the unknown test channel. The training set vector and the measured vector can be expressed as

$$\overline{TSV}_m = \Big((\hat{s}_{i+1}, \hat{v}_{i+1}, \tilde{G}_{r,\hat{v}_{i+1},\hat{s}_{i+1},\hat{m}_{i+1}}), (\hat{s}_i, \hat{v}_i, \tilde{G}_{r,\hat{v}_i,\hat{s}_i,\hat{m}_i})\Big)$$
$$\overline{MV}_m = \Big((\hat{s}_{i+1}, \hat{v}_{i+1}, \hat{G}_{r,\hat{v}_{i+1},\hat{s}_{i+1},\hat{m}_{i+1}}), (\hat{s}_i, \hat{v}_i, \hat{G}_{r,\hat{v}_i,\hat{s}_i,\hat{m}_i})\Big),$$

respectively, and let $\gamma_{r,i}$ be the angle between $\overline{TSV}_m$ and $\overline{MV}_m$. The similarity index $\gamma_r$ is defined as

$$\gamma_r = \sum_{i=1}^{N_I-1} \gamma_{r,i} \tag{7}$$

There are numerous metrics that can be used to measure such similarities. However, the proposed metric is simple and demonstrates good performance. The channel performance manifold matching algorithm computes the channel model $\gamma_{r_{j_0}}$ that most likely has the same performance as the unknown channel. Formally, we have

$$j_0 = \arg_j \min \gamma_{r_j} \tag{8}$$

Subsequently, the decision tree for channel model $r_{j_0}$ can be used for the unknown channel.

Fig. 4 is a representative 3-D plot for $R = 4$ and $N = 5$. We can see that the throughput of the channel model 1 follows the trend of that of the unknown channel, which results in the minimum $\gamma_r$ compared with other channel models. When $N_I$ grows, more features are provided by $\{\hat{G}_i\}$ and the resulting matching is more accurate. Thus, there is a natural trade off between computation complexity and estimation accuracy, which is quantified in Fig. 5 and discussed in the next Section.

TABLE I
CHANNEL PERFORMANCE MANIFOLD MATCHING IN KNOWN CHANNELS

| $N_I$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Accuracy | 74.8% | 81.9% | 87.5% | 90% | 91.7% | 94.3% |

## B. Algorithm evaluation

Using a channel emulator, we can reproduce the same channel in multiple tests, providing a useful method to verify the channel performance manifold matching algorithm. We have implemented two tests to validate our channel performance manifold matching algorithm. In the first test, we select $N_I$ sets of $(r_i, v_i, s_i, m_i)$ where $r_i = r_0$, for some fixed $r_0 \in \mathcal{R}$ for all $i = 1, \ldots, N_I$, $m_i \in \mathcal{M}$ but $v_i \notin \mathcal{V}$, $s_i \notin \mathcal{S}$. We measure the throughput $G_i = G_{r_i,v_i,s_i,m_i}$, and using $\{G_i\}$ and the corresponding $\{v_i\}$, $\{s_i\}$ , $\{m_0\}$ as the input to the algorithm, and check if the inferred value equals $r_0$.

The second verification test is to generate a new channel $c_{new} \notin \mathcal{R}$ with a randomly generated power-delay profile, select $N$ sets of $(c_{new}, s_i, v_i, m_i)$, measure the resulting throughput $\{\eta_i\}$, and check if the algorithm can choose the model $r_{fit} \in \mathcal{R}$ which is most similar to $c_{new}$. As stated in Section II, we seek to select the transmission mode that maximizes throughput. Therefore, the definition of $r_{fit}$ is the channel model that, once fed into the decision tree along with the context information, outputs the transmission mode $m^* \in \mathcal{M}$ that achieves the maximum throughput in $c_{new}$. Next, we will introduce the proposed method to determine $r_{fit}$ and then discuss the evaluation results.

To determine $r_{fit}$, we first find $m^*$ by measuring the throughput of all the $m_k \in \mathcal{M}$ in the unknown channel with the same context information. In other words, $m^* = \arg\max_{m_k} G_{c_{new},v_i,s_i,m_k}, k = 1, \ldots, R$. Then, we consider $R$ scenarios such that $r_{fit} = r_j, j = 1, \ldots, R$. For each scenario, we input $(r_j, v_i, s_i)$ into the decision tree and check if the output transmission mode $m_j$ equals $m^*$. We repeat the procedure by choosing different $v$ and $s$ and find the $m_j$ which has the most matching instances. We will call this scheme the *exhaustive search* algorithm and compare it with the performance of the channel performance manifold matching algorithm.

We use the same training set and the decision tree in Section II where four channel models are involved, *i.e.*, $R = 4$ in both evaluations. In the first type of evaluation, for the exhaustive search algorithm, we choose $N_I = 16$ records to investigate the effectiveness of the algorithm. The matching accuracy on average is 40%. Since our manifold matching algorithm can work with $N_I(N_I \geq 2)$ records and it has better performance with more records, we have done a progressive evaluation while increasing the record number. Table I shows the result of the evaluation. The superiority of the proposed channel performance manifold matching algorithm is clearly evident from the high accuracy of 74.8% with only 2 test points and over 94% accuracy with just 7 test points.

In the second evaluation, among 24 results that are given by the exhaustive search algorithm, 66.7% select channel model 3 in $\mathcal{R}$ as $r_{fit}$. For our manifold matching algorithm, we have done the progressive evaluation as shown in Fig. 5. The confidence of selecting channel model 3 is the highest out of all channels with just 2 records and a monotonically increasing confidence with additional records. On the other hand, for the other channel models the confidence is low with 7 records.

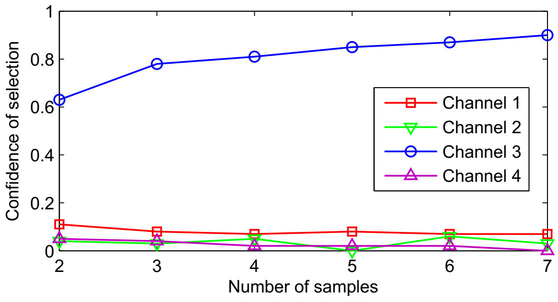


Fig. 5. Confidence of the Channel Performance Manifold Matching with Various Number of Samples

Our experimental evaluation demonstrates several tests where the proposed channel performance manifold matching method leads to significant increase in throughput. It should also be noted that incorrect channel matching does not necessarily mean lower system throughput. Classifying an unknown channel into other channel models than $r_{fit}$ may not result in the highest throughput, but feeding this result into the decision tree may still increase the throughput over an purely SNR-based rate adaptation scheme.

## C. Complexity Analysis

Here, we provide the worst case computational complexity of the proposed channel performance manifold matching mechanism integrated with the decision tree structure. In big-O notation, $\mathcal{O}(\cdot)$ is used to evaluate the worst-case time complexity of the proposed strategy [25]. Let the channel performance manifold matching algorithm detect a total of $R$ different channel types during the overall procedure, and the training data at each channel type $r \in R$, be $\mathcal{D}_r$. Therefore, the total number of training data points is $\mathcal{D} = \sum_{r \in R} \mathcal{D}_r$.

The computation requirement for classifying an unknown channel with $N_I$ data points to one of the $R$ channels is on the order of $N_I R$, which can be explained as follows: For each channel $r \in R$, calculation of $\gamma_{r,i}$ for all $i \in N_I$, as well as the calculation of $\gamma_r$, has time complexity $\mathcal{O}(N_I)$, and this process is repeated for all $r \in R$ channels. The channel performance manifold matching mechanism runs a total of $\frac{D}{N_I}$ times and therefore requires $\mathcal{O}(DR)$ time. The time complexity of each decision tree $r \in R$ is $\mathcal{O}(\mathcal{D}_r \log \mathcal{D}_r)$ [23], and the overall time complexity of computing all the decision trees for $R$ channels is $\mathcal{O}\left(\max_{r \in R}(\mathcal{D}_r \log \mathcal{D}_r)\right)$. Therefore, the time complexity of the overall procedure is $\mathcal{O}(DR) + \mathcal{O}\left(\max_{r \in R}(\mathcal{D}_r \log \mathcal{D}_r)\right)$.

## IV. Experimental Results

In this section, we evaluate the performance of FIT with experiments on emulated and in-the-field channels. Through experiments, we demonstrate the following key results regarding FIT: *(i)* it uses fewer data points to perform almost as well as methods that use extensive offline training, *(ii)* it can incrementally update the decision structure effectively and efficiently, and *(iii)* it outperforms both the static offline-training-based and SNR-based methods. To experimentally show these results, we use an off-the-shelf, 802.11-based Gateworks 2358 platform and Ubiquiti XR-2 radios as wireless nodes in our experiments.

## *A. Comparison of Different Classification-based Link Adaptation Frameworks with Emulated Data*

To collect data for analyzing and comparing the performance of link adaptation frameworks, we use repeatable and controllable emulated channels. We generate emulated channels using the Azimuth ACE-MX channel emulator to which we connect our transmit and receive nodes.We compare the link adaptation frameworks in four channel models specified by International Telecommunication Union (ITU). We use Channels 1, 2, 3 and 4 to represent the ITU Pedestrian A and B and ITU Vehicular A and B channel models, respectively [26]. Channel 1 represents a channel that lacks a strong multipath component (*e.g.*, an environment that lacks tall buildings or structures). Channel 2 represents an environment where there are some multipath components. Channel 3 has multiple strong multipath reflections (*e.g.*, a downtown environment with a line-of-sight component), while Channel 4 has a multipath component that is stronger than the line-of-sight component.

For simplicity, we assume that the SNR is available at the transmitter; in practice this information could be obtained via a feedback link from the receiver or directly measured at the transmitter using the channel reciprocity property. A computer captures the SNR and throughput from the transmitter and the velocity from the channel emulator. The throughput is calculated using (1) based on the received ACKs at the transmitter side to find the PER. Note that different context information from the transmitter side or the receiver side can also be used in the proposed framework depending on the users' need.

The emulator allows us to test the performance of all available modes under each scenario (the combination of one SNR and one velocity) to determine the best mode. This exhaustive measurement technique allows calculation of benchmarks for the performance of any adaptive scheme. Consequently, we use prediction accuracy, throughput improvement over a purely-SNR based rate adaptation method and the throughput gap from maximum achievable performance as the performance metrics.

Our hardware supports the following 8 transmission rates for link adaptation: 6, 9, 12, 18, 24, 36, 48 and 54 Mbps. In each type of channel, we collect the training data under 40 different scenarios, which are combinations of 8 attenuation values: 0, 6, 12, 18, 24, 30, 36, 42 dB and 5 velocities: 0, 30, 60, 90, 120 kmph. The testing data (i.e., the data used to measure effectiveness of the training) is collected with random velocities and attenuation values that are different from the training data.

Under each scenario, we consider the channel type, SNR, velocity, and throughput for all 8 modes as the raw data. We generate the training data in two ways based on collected raw data. First, we determine the best mode for each training data point and prepare the data in structure 1 as shown in Table II to input to the learning algorithm. Data structure 1 is desirable for training the decision structure for offline-training-based link adaptation. Then, to evaluate the proposed structure of the training data, we prepare the data as structure 2 shown in Table II.

TABLE II
UTILIZED CONTEXT INFORMATION AND DATA STRUCTURE

| | Context information | Class |
|---|---|---|
| Data structure 1 | channel type, velocity, SNR | best mode |
| Data structure 2 | channel type, velocity, SNR, mode index | throughput |

To test the feasibility of the proposed structure of the training data, we train the classifiers with the C4.5 algorithm [27]–[29]. We compare and evaluate the link adaptation frameworks with different structures of the training data in 4 different channels. We also evaluate the frameworks in the situation when combining the 4 training sets and the 4 testing sets for the 4 channels as a pair of training and testing sets. Channel 1 and channel 2 have similar properties, but channel 2 has a larger propagation delay. Channel 4 is similar to channel 3, but has a larger propagation delay.

We investigate the accuracy of the rate prediction and the throughput gap from the maximum achievable throughput. The results are shown in Table III. We compare their performances with an SNR-only method, which chooses the rate based on only SNR information. We then show the throughput improvement over the SNR-only method in the $3^{rd}$ and $6^{th}$ columns of Table III.

From Table III, we first notice that for channel 1, the two link adaptation frameworks based on different structures of training data both perform comparably to the SNR-only method, with a small gap to the maximum achievable throughput performance. The SNR-only method achieves a smaller gap than the classification-only methods in this channel. Since channel 1 is a slow-fading channel with a small propagation delay, a purely-SNR based method performs effectively in this situation. While the propagation environment grows to be more complicated, the performance of the SNR-based method degrades. Thus, from channel 2 to channel 4, the classification-based methods gain significant improvement over the SNR-based method. For example, the offline-training-based framework with training data in structure 1 and in structure 2 obtain 1675% and 1506% relative improvement in channel 2, respectively

Without the best mode in the training data (i.e., with data structure 2), the offline-training-based framework obtains comparable throughput with the framework using data structure 1. Furthermore, the framework using the training data in structure 2 still obtains higher throughput over the SNR-only method. These results demonstrate that data structure 2 can maintain the high accuracy achieved by data structure 1 while also being more suited for online training.

TABLE III
PERFORMANCE OF LINK ADAPTATION

| Channel | Using Knowledge of Best Mode Only | | | Using Quantized Throughput | | |
|---|---|---|---|---|---|---|
| | Accuracy of Rate Prediction | Throughput Improvement over SNR-only Scheme | Gap from Maximum Achievable Throughput | Accuracy of Rate Prediction | Throughput Improvement over SNR-only Scheme | Gap from Maximum Achievable Throughput |
| 1 | 53.85% | -3.81% | 9.74% | 61.54% | -3.29% | 9.25% |
| 2 | 84.62% | 1675.27% | 8.46% | 69.23% | 1506.22% | 17.18% |
| 3 | 64.62% | 623.90% | 0.80% | 57.69% | 527.74% | 14.00% |
| 4 | 46.15% | 671.91% | 18.70% | 50% | 707.38% | 14.96% |
| All 4 | 67.31% | 73.58% | 9.08% | 58.65% | 64.83% | 13.66% |

## *B. Investigation of the Granularity of Quantized Throughput and the Performance of Adaptive Tree*

In order to treat the continuous values of throughput as the class label in the proposed framework, we quantize the throughput into $N_Q$ discrete values. In Table III, we set $N_Q$ to 1000. With a small $N_Q$, the granularity of the quantization is large, and the classifier can not distinguish between throughput values close to one another. While a large $N_Q$ can improve the resolution, it leads to a complicated structure of the classifier, which can deteriorate the efficiency of the in-situ adaptation process. Thus, to investigate the effect of the granularity of this quantization process on the performance of the proposed framework, we test it with different values of $N_Q$. To test the performance of the adaptive-learning algorithm in parallel, we employ it to derive the decision structure in this subsection.

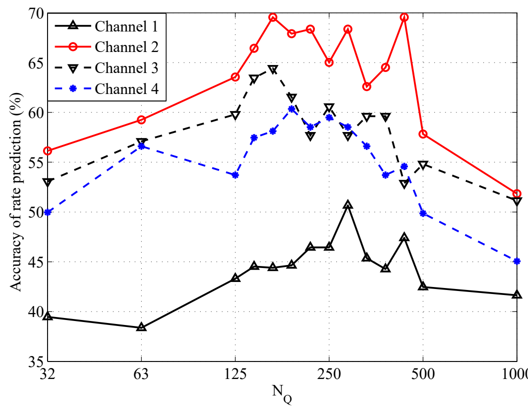


Fig. 6. Accuracy of rate prediction with various $N_Q$.

We vary $N_Q$ over the values in the set $\{32, 63, 125, 150, 175, 200, 225, 250, 300, 350, 400, 450, 500, 1000\}$ and do experiments on 4 frequency bands: 700 MHz, 900 MHz, 2.4 GHz and 5.8 GHz. The accuracy of rate prediction and the throughput performance are averaged across different bands for each channel, as shown in Fig. 6 and Table IV respectively. It can be observed that:

- The classification-based rate adaptation framework with the proposed training data structure outperforms SNR-only method with varying $N_Q$ in most cases. On average, the classification-based framework with the proposed data structure obtains $147.59\%$ relative improvement in throughput.
- As $N_Q$ increases, the accuracy of rate prediction first increases and then decreases. For small values of $N_Q$, increasing the granularity of the quantization can help the decision tree to distinguish between roughly equivalent throughput values. Increasing $N_Q$ beyond a certain value leads to a reduction in accuracy of rate prediction. As mentioned above, our proposed framework will generate a binary decision tree. Thus, too many class labels would results in a complex structure of the binary decision tree.
- To achieve the best performance, we should find a trade-off between the granularity of the quantization and the complexity of the decision tree.
- A higher improvement in accuracy does not strictly result in corresponding improvement in throughput. Since the throughput of each mode as well as the maximum achievable throughput is different between different scenarios, the throughput improvement of each data point varies.
- In simplistic settings, *e.g.*, for Channel 1, which is a slow-fading channel with a small propagation delay, FIT may encounter marginal performance deterioration compared to the SNR-only method. This is because performance of the SNR-only method is close to the maximum achievable throughput for such a channel, and FIT requires training before it performs close to the maximum achievable throughput. For all other channel types, the SNR-only method is significantly further away from the maximum achievable throughput, and our proposed FIT framework performs close to the maximum achievable throughput.

## *C. Evaluation of FIT with Emulated Data*

In this subsection, we test the performance of FIT based on emulated data. To prepare the training and testing sets, we model the natural changing process of the context information. We gradually decrease the SNR by increasing the attenuation and randomly vary the velocity between 0 and 120 kmph. The training data set contains 40 data points with the lowest velocities. The testing data set contains 26 data points with the highest velocities. We prepare the training and testing set as data structure 2 in Table II , which means there are $40 * 8 = 320$ and $26 * 8 = 208$ data points in the generated training set and testing set, respectively.

We evaluate and compare the offline-based method and FIT in two kinds of trials. In the first trial, we train the classifier with the training set and test the performance of the link adaptation framework with the testing set. In the second trial, we initially train the classifier using the training set. During testing phase, for each 8 testing points from one raw data in the testing set we first test them and then use them as newly

TABLE IV
THROUGHPUT PERFORMANCE COMPARED TO SNR-ONLY METHOD AND THE MAXIMUM ACHIEVABLE THROUGHPUT WITH VARIOUS $N_Q$. NOTE THAT IN THE TABLE, "I" REPRESENTS IMPROVEMENT OVER SNR-ONLY METHOD AND "G" DENOTES GAP FROM MAXIMUM ACHIEVABLE THROUGHPUT.

| $N_Q$ | 32 | 63 | 125 | 250 | 500 | 1000 |
|---|---|---|---|---|---|---|
| Channel 1 | I: -16.81%<br>G: 25.79% | I: -18.34%<br>G: 27.22% | I: -21.15%<br>G: 29.68% | I: -13.85%<br>G: 23.25% | I: -12.99%<br>G: 22.35% | I:-17.20%<br>G: 25.91% |
| Channel 2 | I: 433.00%<br>G: 22.18% | I: 429.66%<br>G: 22.83% | I: 460.74%<br>G: 17.41% | I: 434.02%<br>G: 19.15% | I: 407.35%<br>G: 27.76% | I: 415.04%<br>G: 28.22% |
| Channel 3 | I: 171.61%<br>G: 22.23% | I: 139.79%<br>G: 27.80% | I: 167.59%<br>G: 23.92% | I: 160.80%<br>G: 26.40% | I: 157.42%<br>G: 29.68% | I: 164.66%<br>G: 22.63% |
| Channel 4 | I: 171.25%<br>G: 28.47% | I: 177.48%<br>G: 24.54% | I: 163.88%<br>G: 33.03% | I: 167.97%<br>G: 31.94% | I: 161.29%<br>G: 35.06% | I: 159.59%<br>G: 32.80% |

collected data to incrementally train the existing decision tree. In these experiments, the testing data is learned as the newly collected data from the regular communication.

In practice, the propagation property of wireless channels would change due to the variation of the environment, e.g. seasonal variations, weather variations. To evaluate the case of changing wireless channels, we implement two kinds of trials: (1) training and testing in the same channel type; (2) training and testing in different channel types. We collect training set in channel type 3 and 4 testing sets in 4 different channel types.

The results are shown in Table V. With the offline-based method, the link adaptation framework is embedded in the classifier derived from the training set and tested on the testing set. FIT updates the link adaptation decision structure during the testing phase after testing each 8 testing points from one raw data. It can be observed that the accuracy of rate selection of the proposed FIT is higher compared to the offline-based method, particularly, when training and testing in different channel types. The proposed framework improves the accuracy of rate prediction by up to 65.38%.

The FIT and offline schemes achieve significant throughput improvement (upto 592% and 822%, respectively) compared to the SNR-only method. This result suggests that considering all context information (*e.g.*, velocity, SNR, and channel type) for rate adaptation can make significant improvement compared to the rate adaptation based only on SNR information, signaling the SNR-to-rate relationship does not strictly hold due to the latency in the feedback mechanism. Context information helps to anticipate changes made to channel quality, even when the feedback duration forces a mismatch in the SNR reported and that actually experienced when the data packet is received. Compared to the offline-based method, FIT improves the throughput performance by 33% when training and testing in the same channel and by up to 28% when training in channel 3 and testing in channel 2. These results demonstrate the necessity and effectiveness of in-situ training.

## D. Evaluation of FIT with In-Field Training

In order to evaluate FIT with in-field data, we set up a measurement system with two wireless nodes communicating over the air. One of the nodes is located on a car and the other is set on top of a building in town. We collect the throughput data with context information. Based on the location and surroundings where the data is collected, we divide the region around the building into several sub-regions and assume different sub-regions represent different channel types. To verity the performance of FIT, we first pick the tuple of channel type, SNR, and velocity values as the testing scenarios and interpolate the throughput values for a certain mode based on the records collected in the same channel type that have similar SNR and velocity values. With the interpolated throughput values of different modes in one scenario, we can determine the transmission mode that can provide the best performance. In parallel, we feed the context information of the scenario into the decision tree updated by in-situ training to evaluate if the output agrees with the best mode we previously inferred.

We collect the source data in data structure 2 directly for training at sub-region 1 and the source data for testing set at sub-region 2. The two sub-regions are close to one another. When the decision structure is not adaptive to the testing data, the accuracy of rate prediction is 20%. The throughput improvement over SNR-only method and gap from maximum achievable throughput is $-33.86\%$ and 46.88%, respectively. When we embed the online adaptive function and the decision structure is adaptive to each testing data after testing it (i.e., the testing data is then used as training), the accuracy is 60%, an increase of 200% compared to the non-adaptive framework. The throughput improvement and gap from the maximum are 6.64% and 14.35%, respectively.

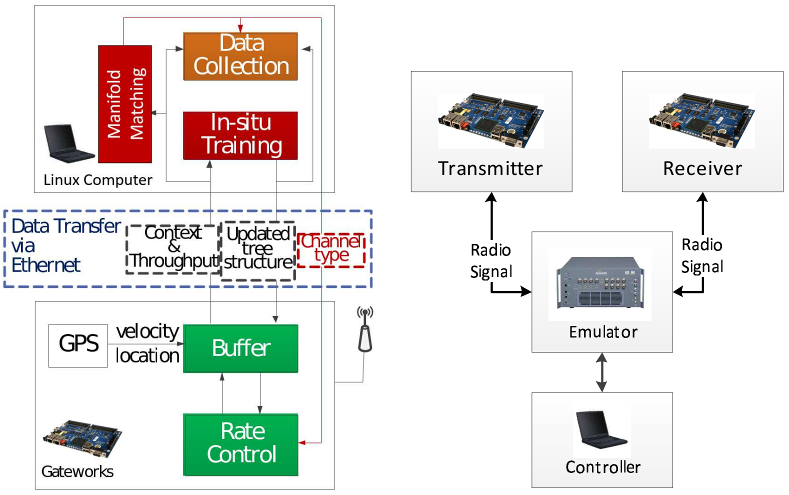
## E. FIT Implementation and Performance

*1) Experiment Setup:* A Linux computer acts as the backhaul of the FIT system, which runs the in-situ training and the channel performance manifold matching based on the received training data. We build a special version of the Atheros Linux wireless driver - *ath5k*, in which the rate control module has been redesigned to act as an agent between the backhaul and wireless transmission module. The driver is designed to pass the performance data to the backhaul and to update the tree structure along with the inferred channel type generated by the backhaul. We burn the modified driver onto the Gateworks 2358 board, essentially changing the functionality of the kernel space. By comparison, the applications running in the user space are impervious to this change. We use *iperf* as the tool to measure the throughput. The node with this special version of wireless driver is used as the transmitter, and the receiver node is unchanged.

Fig. 7(a) shows the key elements of the transmitter mentioned above. Our system setup is shown in Fig. 7(b). Note that the computer in Fig. 7(b) only functions as a controller of the

TABLE V
PERFORMANCE OF FIT AND OFFLINE-BASED METHOD ON EMULATED DATA. NOTE THAT IN THE TABLE, "A" REPRESENTS THE ACCURACY OF RATE PREDICTION, "I" REPRESENTS IMPROVEMENT OVER SNR-ONLY METHOD AND "G" DENOTES GAP FROM MAXIMUM ACHIEVABLE THROUGHPUT.

| Channel | | Testing: 1 | | Testing: 2 | | Testing: 3 | | Testing: 4 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Offline | FIT | Offline | FIT | Offline | FIT | Offline | FIT |
| Training: 3 | A | 11.54% | 46.15% | 19.23% | 46.15% | 38.46% | 65.38% | 15.79% | 47.37% |
| | I | -31.16% | -27.94% | 313.59% | 428.81% | 592.96% | 822.08% | 538.39% | 677.97% |
| | G | 36.64% | 33.43 | 46.06% | 31.03% | 33.92% | 12.07% | 33.86% | 20.48% |



(a) Transmitter (b) System

Fig. 7. Experiment Setup.

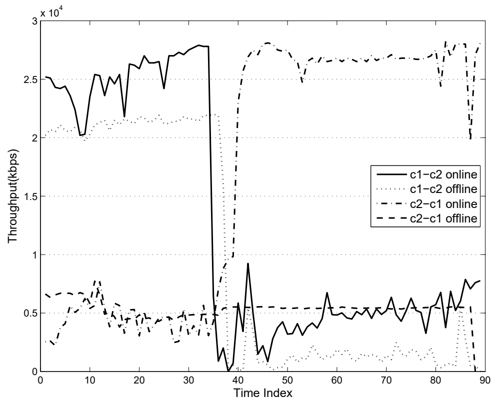


Fig. 8. Performance with channel transitions.

channel emulator to ensure the consistency of each experiment and it does not exchange any information with the wireless nodes.

*2) Unintentional Channel Transition:* As discussed in Section III, the training sets for different channel types represent their various characteristics. In other words, if the system is only trained with the training set from one channel type, without in-situ training, the accuracy of the rate prediction will be degraded when the system encounters a different channel type. In contrast, if in-situ training is employed in such a case, the system will adapt to the varying channel type and get better performance.

To show the performance improvement of the in-situ training over the offline training, we have created several scenarios on the channel emulator which involve channel type transitions. By using the channel emulator, we expose both in-situ training and online training in the same channel-type transitions to show that the former has a higher adaptability to the environment variation than the latter.

In this particular experiment (Unintentional Channel Transition), for each channel transition, the system is only trained with the training set of the first channel type so that it knows nothing about its performance in the second channel type.

Fig. 8 plots the throughput (in $i$ second increments) obtained using *iperf* in one transition using both training methods. The transition happens at approximately 35 second. It can be seen that the offline training can (at most) achieve similar performance as the in-situ training before the transition. However, its performance falls below that of the scheme with in-situ training after the transition. Table VI shows that in-situ training tends to provide throughput gains over offline training. Once a channel transition occurs, none of the methods ensure finding the optimal rate decision initially. Therefore, it is possible that the offline scheme achieves higher throughput initially. However, once a sufficient amount of training data in the new channel is received, FIT achieves higher throughput performance. Such a phenomenon can be observed for the transition from Channel 1 to 2. In the case of the Channel 3 to 4 transition, we expect to observe a higher throughput gain for FIT over the offline method if the experiment is continued for a longer duration and more time is given for the rewards of the training to be leveraged.

*3) Awareness of the Channel Transition:* In this experiment, we continue to compare the performance difference between in-situ and offline training with the channel performance manifold matching. With the channel performance manifold matching, the system is able to detect the channel transitions and adapt. To demonstrate the performance gain brought by

TABLE VI
THROUGHPUT COMPARISON FOR DIFFERENT CHANNEL TRANSITIONS

| Transition | Offline | In-situ | Improvement |
|---|---|---|---|
| Channel 1 to Channel 2 | 8245 | 9452 | 14.6% |
| Channel 2 to Channel 1 | 8992 | 13664 | 52.0% |
| Channel 3 to Channel 4 | 7736 | 7220 | -6.7% |
| Channel 4 to Channel 3 | 6274 | 7932 | 26.4% |

channel performance manifold matching, we design two other strategies. With the same channel transition as the last experiment, we first assume that a genie-based channel inference exists in which a genie can notify the system with the change in real time. The other strategy is the in-situ training with the aforementioned channel performance manifold matching.

The impact of Table VII is two-fold. First, each scheme achieves throughput gains from in-situ training because in-situ training would adjust the tree structure in real time so that the decision tree for rate prediction is according to the current realization of the channel. Second, channel performance manifold matching can have a positive impact in terms of throughput improvement. Theoretically, the throughput achieved by the genie-based strategy is the outer bound of that of the channel performance manifold matching because the former can always infer the channel type perfectly. From the table, we can see that the throughput achieved by the channel performance manifold matching is close to the genie-based case, and it is higher than the strategy of not known channel transitions.

## V. Related Work

Many works have considered rate adaptation via loss statistics [4]–[7], [30] or SNR-based probing [8], [9], [11], [12]. Moreover, other works have used a notion of context information to aid rate selection with knowledge of vehicular speed, pedestrian speed, or the direction of motion [13]–[15], [31] and/or channel type information. For the former, historical data is used with a lack of overhead needed, but in the absence of channel quality or context information. For work related to the latter mechanisms, training is offline in nature and is subject to poor performance in scenarios distinct from their training [10]. In contrast to both types, we allow training on-the-fly in the environment in which the rate adaptation mechanism will be operating with knowledge of the channel type, quality, and context information.

Machine learning algorithms have been widely used to optimize wireless system performance. In particular, the resulting decision structure has been used in cognitive radios for dynamic spectrum access and capacity maximization in cognitive radios [17], [32]. Conversely, genetic algorithms have been used to optimize physical layer parameters [33]. In [34], a deep learning technique is proposed for routing computation and traffic control in a software-defined communication system. To solve the power allocation problem for the downlink transmission in a spectrum sharing multi-tier 5G environment, an online learning based approach is employed in [35]. In [36], deep learning was adopted for direction-of-arrival estimation and channel estimation in multi-antenna systems, which depends upon a large amount of offline training. The works which most closely resemble our works are [16], [37], [38]. In [16], [37], the authors put constraints on the wireless performance metric to achieve performance optimization rather than seeking the optimal realization directly. Moreover, [37] does not implement a decision structure which can be updated on-the-fly in the field on newly-observed performance data. These limitations highlight the need for more effective online learning algorithms for rate adaptation. In [38], a rate adaptation strategy based on a deep learning architecture is proposed for unmanned aerial vehicle (UAV) network, which can adapt in a new environment. However, unlike our proposed scheme, offline learning with a large amount of training data is required to create the decision structure for the rate adaptation. Furthermore, in [38], the online learning is employed each time a new environment is detected. Our channel performance manifold matching algorithm categorize the wide expanse of an infinite number of possible environments into a countable set of channel types. Then, the online learning mechanism is activated only if a new channel type is encountered, and therefore has a lower computational requirement.

## VI. Conclusion

In this paper, we discuss the two challenges posed on online adaptive link adaptation due to the fluctuating wireless channel. We present our solutions to these challenges by modifying the data structure for training and applying the online adaptive infrastructure. Based on our solutions, we propose an online adaptive link adaptation framework which can conserve the manpower and resources for preparing infrastructure offline. A novel channel performance manifold matching strategy is introduced which synergistically integrate with the online link adaptation framework to achieve efficient link adaptation with low computational requirement. The proposed strategy improves the accuracy of rate prediction by up to 65.38%. Compared to the SNR-only method and offline-based method, our proposed framework achieves 822.08% and 33% improvement, respectively. The experiments show that the adaptive framework can adapt the decision structure to the changing propagation environment and improve its performance on rate prediction.

We considered a Ubiquiti family of radios to evaluate performance of our proposed channel performance manifold matching integrated with a rate decision structure. As discussed in this work, an online mechanism is inherently running on the device and training on the fly when new channels are experienced, or an insufficient level of training is observed. We have also observed through other measurements via crowdsourcing that users and thereby devices have a bimodal or trimodal location distribution [39]. We show in this paper, extremely high levels of gain can be experienced in less than a minute after transitioning to a new location. Considering all of these factors, we provide a framework for any type of device to spend an infinitesimal amount of time making suboptimal decisions (while still sending data) in exchange for a very large majority of time that near optimality will be achieved. Our proposed framework is flexible to support additional attributes in the future, if necessary.

Lastly, we have used two decision tree algorithms from the ID3/C4.5 family, C4.5 and ITI [23], since these are

TABLE VII
THROUGHPUT'S COMPARISON OF DIFFERENT STRATEGIES

| Unintentional | | | Manifold Matching | | | Genie | | |
|---|---|---|---|---|---|---|---|---|
| Offline | In-situ | Impr | Offline | In-situ | Impr | Offline | In-situ | Impr |
| 8992 | 13664 | 52.0% | 9254 | 15293 | 65.3% | 9023 | 15976 | 77.1% |

well-established algorithms, provide high training accuracy, and have been applied to many research fields [40]–[43]. Furthermore, the C4.5 algorithm has been applied to many research fields. Across all these applications where a time-varying environment is experienced, our channel performance manifold matching algorithm could be used. In other words, detecting the environment for C4.5 based decision trees has applicability beyond just the field of wireless communications and networking. Our proposed framework can be extended with other incremental decision tree based algorithms if the proper conditions hold. Namely, these conditions have to do with the ability of the decision tree to handle multiple classes and missing values.

## REFERENCES

[1] H. Liu, J. He, O. Altintas, R. Vuyyuru, J. Camp, and D. Rajan, "FIT: On-the-fly, in-situ training with sensor data for snr-based rate selection," in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, Mar. 2015, pp. 1201–1206.

[2] J. He, H. Liu, P. Cui, J. Landon, D. Rajan, and J. Camp, "Geometry-based channel recognition for context-aware applications," in *Proc. Int. Symp. Modeling Opt. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2016, pp. 1–6.

[3] J. Choi et. al., "Adaptive optimization of rate adaptation algorithms in multi-rate wlans," in *Proc. of ICNP*, 2007, pp. 144–153.

[4] A. Kamerman et. al., "WaveLAN II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, pp. 118–133, Summer 1997.

[5] J. C. Bicket et. al., "Bit-rate selection in wireless networks," M.S. Thesis, MIT, February 2005.

[6] S. Wong et. al., "Robust rate adaptation for 802.11 wireless networks," in *Proc. of ACM MobiCom*, 2006.

[7] I. Pefkianakis, S. H. Y. Wong, H. Yang, S. Lee, and S. Lu, "Toward history-aware robust 802.11 rate adaptation," *IEEE Trans. Mobile Comput.*, vol. 12, no. 3, pp. 502–515, Mar. 2013.

[8] G. Holland et. al., "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proc. of ACM MobiCom*, July 2001.

[9] B. Sadeghi et. al., "Opportunistic media access for multirate ad hoc networks," in *Proc. of ACM MobiCom*, Sept. 2002.

[10] J. Camp et. al., "Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1949–1962, Dec. 2010.

[11] W. Gong, H. Liu, J. Liu, X. Fan, K. Liu, Q. Ma, and X. Ji, "Channel-aware rate adaptation for backscatter networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 751–764, April 2018.

[12] L. Wang, X. Qi, J. Xiao, K. Wu, M. Hamdi, and Q. Zhang, "Exploring smart pilot for wireless rate adaptation," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 4571–4582, Jul. 2016.

[13] L. Ravindranath et. al., "Improving wireless network performance using sensor hints," in *Proc. of USENIX NSDI*, San Jose, California, Apr. 2010.

[14] P. Shankar et. al., "CARS: Context-aware rate selection for vehicular networks," in *Proc. of IEEE ICNP*, Breckenridge, CO, Oct. 2008.

[15] W. Gong, S. Chen, J. Liu, and Z. Wang, "Mobirate: Mobility-aware rate adaptation using phy information for backscatter networks," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 1259–1267.

[16] R. Daniels, et. al., "Online adaptive modulation and coding with support vector machines," in *Proc. of IEEE EW*, 2010.

[17] S. Haykin, et. al., "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, 2005.

[18] K. Tsagkaris, et. al., "Neural network-based learning schemes for cognitive radio systems," *Computer communications*, vol. 31, no. 14, pp. 3394–3404, 2008.

[19] S. Yun, et. al., "Multiclass support vector machines for adaptation in mimo-ofdm wireless systems," in *Allerton*, 2009, pp. 1145–1152.

[20] Ubiquiti xtremerange2 datasheet. [Online]. Available: https://dl.ubnt.com/xr2_datasheet.pdf

[21] P. Auer, et. al., "Adaptive and self-confident on-line learning algorithms," *Journal of Computer and System Sciences*, vol. 64, no. 1, pp. 48–75, 2002.

[22] P. Utgoff, et. al., "Incremental induction of decision trees," *Machine Learning*, vol. 4, no. 2, pp. 161–186, 1989.

[23] ——, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, no. 1, pp. 5–44, 1997.

[24] J. R. Quinlan et. al., "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, March 1986.

[25] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley, 2011.

[26] *Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000*, ITU Recomm. M.1225, 1997.

[27] S. R. Safavian et. al., "A survey of decision tree classifier methodology," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 21, pp. 660–674, 1991.

[28] C. Yang et. al., "Application of decision tree technology for image classification using remote sensing data," *Agricultural Systems*, vol. 76, no. 3, pp. 1101–1117, 2003.

[29] V. Park et. al., "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. of INFOCOM*, vol. 3, 1997, pp. 1405–1413.

[30] "MadWifi Project," http://madwifi.sourceforge.net, September 2005.

[31] G. Judd et. al., "Efficient channel-aware rate adaptation in dynamic environments," in *ACM MobiSys*, June 2008.

[32] C. Clancy et. al., "Applications of machine learning to cognitive radio networks," *IEEE Journal on Wireless Communications*, vol. 14, no. 4, pp. 47–52, 2007.

[33] T. Rondeau et. al., "Cognitive radios with genetic algorithms: Intelligent control of software defined radios," in *SDR Forum*, 2004, pp. C3–C8.

[34] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[35] I. AlQerm and B. Shihada, "Energy-efficient power allocation in multitier 5G networks using enhanced online learning," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11 086–11 097, 2017.

[36] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8549–8560, 2018.

[37] R. Daniels et. al., "An online learning framework for link adaptation in wireless networks," in *Proc. of IEEE ITA Workshop*, 2009.

[38] S. He, W. Wang, H. Yang, Y. Cao, T. Jiang, and Q. Zhang, "State-aware rate adaptation for UAVs by incorporating on-board sensors," *IEEE Trans. Veh. Technol.*, to appear, 2019.

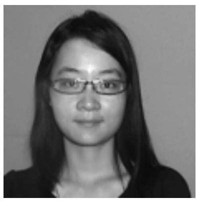
[39] J. Camp and R. Meikle, "A global measurement study of context-based propagation and user mobility," in *Proc. ACM MOBISYS (HotPlanet Workshop)*, Jun. 2012, p. 2126.

[40] K. Gokcesu, M. M. Neyshabouri, H. Gokcesu, and S. S. Kozat, "Sequential outlier detection based on incremental decision trees," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 993–1005, 2019.

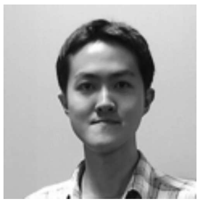
[41] R. Abdalkareem, S. Mujahid, and E. Shihab, "A machine learning approach to improve the detection of CI skip commits," *IEEE Trans. Softw. Eng.*, pp. 1–1, 2020.

[42] H. Ma, H. Xie, and D. Brown, "Eco-driving assistance system for a manual transmission bus based on machine learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 572–581, 2018.

[43] J. B. Zurn, X. Jiang, and Y. Motai, "Video-based tracking and incremental learning applied to rodent behavioral activity under near-infrared illumination," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 6, pp. 2804–2813, 2007.



**Hui Liu** received the Bachelor's and Master's degree in electrical and electronics engineering from the Beijing Institute of Technology, Beijing, China, and the Ph.D. degree in electrical engineering from Southern Methodist University, Dallas, TX, USA. She is currently with the Michigan State University, Michigan, USA.



**Jialin He** received the BS degree in electrical and information engineering and MS degree in signal and information processing from the Tianjin University, Tianjin China. He received the Ph.D. degree in electrical engineering from Southern Methodist University, Dallas, TX, USA. He is currently with Microsoft, Redmond, Washington.

**Sabyasachi Gupta** received the M.Tech degree from the National Institute of Technology (NIT), Durgapur, India and the Ph.D. degree from the Indian Institute of Technology Delhi, New Delhi, India. He has held research positions with the University of Pompeu Fabra, Barcelona, Spain and with the University of Alabama, Tuscaloosa, USA. He is currently a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, Southern Methodist University, Dallas, USA. He was the recipient of the Institute Gold Medal from NIT Durgapur in 2010. His research interests are in resource allocation for wireless networks, mobile-edge computing networks, investigating application of optimization technique and graph theory for wireless communication, routing algorithms.

**Dinesh Rajan** (Senior Member, IEEE) is currently Department Chair and Cecil and Ida Green Professor in the Electrical and Computer Engineering Department at Southern Methodist University. He received the B.Tech. degree in Electrical Engineering from Indian Institute of Technology (IIT), Madras. He received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from Rice University, Houston, Texas. He joined the Electrical Engineering Department at Southern Methodist University, Dallas, Texas in August 2002 as an Assistant Professor. His current research interests include communications theory, wireless networks, information theory and computational imaging. He received a NSF CAREER award for his work on applying information theory to the design of mobile wireless networks. He is also a recipient of the Golden Mustang Outstanding faculty award and the Senior Ford Research Fellowship from SMU.

**Joseph Camp** (Member, IEEE) received the B.S. (Hons.) degree in electrical and computer engineering from the UT-Austin, Austin, TX, USA, and the M.S. and Ph.D. degrees in electrical and computer engineering from Rice University, Houston, TX, USA. He is currently an Associate Professor of Electrical and Computer Engineering with Southern Methodist University, Dallas, TX, USA. He joined the SMU Faculty in 2009. His research team has performed more than 200 million in-field wireless measurements around the world via Android deployment and local characterization via drones, campus buses, vehicles, and buildings. His research interests are wireless communications and networking, crowdsourcing, and drones, specifically focused on the deployment, measurement, and analysis of large-scale systems and development of embedded protocols. He received the National Science Foundation CAREER Award in 2012 and the Golden Mustang Teaching Award in 2014.