Learning Wi-Fi Performance

Technical Report

Julien Herzen
EPFL
Switzerland
julien.herzen@epfl.ch

Henrik Lundgren
Technicolor
France
henrik.lundgren@technicolor.com

Nidhi Hegde
Alcatel-Lucent Bell Labs
France
nidhi.hegde@alcatel-lucent.com

***Abstract*—Accurate prediction of wireless network performance is important when performing link adaptation or resource allocation. However, the complexity of interference interactions at MAC and PHY layers, as well as the vast variety of possible wireless configurations make it notoriously hard to design explicit performance models.**

In this paper, we advocate an approach of "learning by observation" that can remove the need for designing explicit and complex performance models. We use machine learning techniques to learn implicit performance models, from a limited number of real-world measurements. These models do not require to know the internal mechanics of interfering Wi-Fi links. Yet, our results show that they improve accuracy by at least 49% compared to measurement-seeded models based on SINR. To demonstrate that learned models can be useful in practice, we build a new algorithm that uses such a model as an oracle to jointly allocate spectrum and transmit power. Our algorithm is utility-optimal, distributed, and it produces efficient allocations that significantly improve performance and fairness.

I. Introduction

We have witnessed a rapid adoption of Wi-Fi technology for home, enterprise and hotspot wireless networks. The result is often dense deployments of interfering Wi-Fi links that contend for a limited amount of spectrum. At the same time, these networks are under an ever increasing pressure to deliver a higher performance. Recent and ongoing IEEE amendments, such as 802.11n and 802.11ac, address this demand by including techniques such as wider channel bandwidths and faster modulation schemes. However, these enhancements put even more stress on the scarce spectrum resource and are sensitive to the operating conditions to deliver the effective performance improvements. Wider channels increase the spectrum usage and can create harmful interference. Higher modulation schemes require a higher SNR and less interference to correctly decode the transmissions. It is therefore increasingly important to carefully allocate resources such as spectrum and transmit power.

Efficient resource allocation requires realistic models. However, 802.11 networks – and especially those using newer amendments with variable bandwidth – are notoriously hard to model. They exhibit several performance intricacies due to complex interactions between the MAC and PHY layers, which manifest themselves in frequency, spatial and time domains. For example, using a wide channel bandwidth creates interference in frequency domain, but using a narrow bandwidth increases packet transmission times, which can create more interference in time domain (due to the rate anomaly problem of MAC layers based on CSMA/CA [10]). In addition, for a fixed transmit power, a narrow bandwidth packs more Watts per Hertz, which improves the transmission range [6], but also increases interference in spatial domain.

The work presented in this paper was supported (in part) by the Swiss National Science Foundation under grant number 200021-146423. Part of this work was carried out while the authors were working at Technicolor.

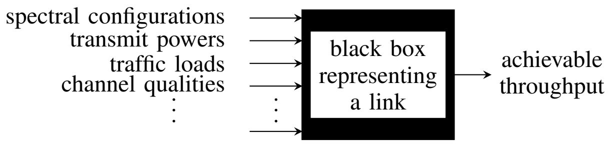

Figure 1: Black box representation of a link. It takes various configuration and topological features related to a given link and its neighbors as inputs, and it outputs a throughput.

Existing performance models for 802.11 networks, such as the Bianchi model [1], usually adopt *explicit* and *bottom-up* approaches; they model the actual mechanics of the protocol (for example, the CSMA/CA procedure of the MAC layer in [1]) in order to compute throughput figures. Unfortunately, these models do not capture heterogeneous PHY layer configurations, such as variable channel widths. In contrast, textbook models based on the SINR (signal to interference-plus-noise ratio) can be used to capture some of the phenomena occurring at the PHY layer. However, in turn, these models do not take the MAC layer into account and, as we will observe, they do not capture the actual performance of interfering links when CSMA/CA is employed.

In this paper we argue that, as far as quantitative performance predictions are concerned, it can be more efficient to learn *implicit* and *top-down* models directly from a set of observed measurements. We treat Wi-Fi links as black boxes with potentially unknown internal mechanics (see Figure 1). Such a black box takes some parameters as inputs (such as the spectral configurations of a link and its neighbors, as well as topological features such as current measurements of channel qualities), and it outputs a throughput value. Our goal is to find any function providing an accurate mapping between (potentially never-observed) inputs and outputs. In particular, we do not attempt to seed a pre-existing model (such as SINR-based or Markov-based) with measurements. Rather, we show that in some cases it can be more efficient to *learn the model itself* from a limited set of measurements.

Constructing useful black boxes is difficult for two main reasons. First, they must capture a fair level of complexity; the cross-layer relationships between the various input parameters and the obtained throughput are usually complex, multi-modal, nonlinear and noisy. Second, it is infeasible to simply measure the link performance for each possible combination of inputs.

Instead of conducting exhaustive measurements, we observe that a statistical representation of these black boxes can be learned by observing a limited number of input/output combinations. Using supervised machine learning techniques, it is possible to generalize the observations made on this limited subset of measurements, while still capturing the complex relationships between the inputs. We build such implicit models using real-world measurements and we test them systematically, by asking them to predict the throughput for links and configurations that have never been observed during the initial measurement phase. We observe that our learned black boxes improve prediction accuracy over models based on the SINR, which is usually the preferred metric for allocating resources such as spectrum or transmit power.

Finally, we demonstrate the usefulness of this "learning by observation" approach, by using one such black box as an oracle for allocating spectrum and transmit power in a dynamic fashion. In particular, we design and implement a complete, utility-optimal algorithm for the joint allocation of spectrum and transmit power. Our algorithm does not rely on a central controller, and requires only local collaboration between neighboring access points (APs). Yet, it converges to a global, network-wide solution of the utility maximization problem. We observe on a testbed deployment that it reacts well to various optimization objectives, such as maximizing throughput and/or fairness. In this context, our black box oracle is instrumental for capturing the intricate interference patterns and finding efficient configurations. To the best of our knowledge, it is the first implementation of a utility-optimal algorithm for spectrum allocation.

The paper is organized as follows. In Section II, we motivate our approach through a few illustrative examples. In Section III, we present our method to learn black box performance models. We evaluate the accuracy and generalization of our models in Section IV. We present our algorithm for spectrum and transmit power allocation in Section V and evaluate its performance in Section VI. We discuss the limitations of our models in Section VII. Finally, we present related work in Section VIII and give some concluding remarks in Section IX.

II. Motivation

In this section, we first detail some of the complexities inherent to the problem of allocating variable-width spectrum chunks to wireless nodes. In particular, we show why the performance achieved by a link depends in a highly complex fashion on spectrum configurations adopted by this link and its neighbors. Second, we give an example where SINR-based models – the prevailing class of models for adapting PHY layer parameters – fail to capture the actual performance of 802.11 networks.

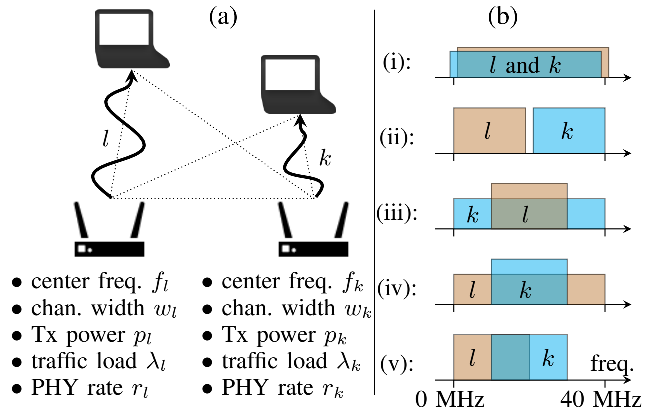

Figure 2: Two interfering links (a), and some possible spectral configurations (b). We propose a method to learn models that can predict the performance of the links in such arbitrary conditions.

A. An Introductory Two-link Example

Consider a simple setup with two Wi-Fi links l and k (which can be composed of two access points and two clients), shown in Figure 2(a). The two access points have arbitrary traffic loads (e.g., due to arbitrary exogenous arrivals). Consider now the problem of allocating a combination of channel center frequencies and channel widths to these two links. Assuming that the two possible channel widths are 20 MHz and 40 MHz (as in 802.11n), and that there is 40 MHz of total spectrum available, we list some possible allocations on Figure 2(b) (see [27] for an extensive treatment of such an example). Efficiently selecting configurations would clearly benefit from a model which could, for each possible combination of configuration, predict the throughput achievable by each link. A model explicitly designed for this task should take at least the following qualitative aspects into account:

- If the links are sufficiently far apart, the spectrum can be re-used, and both links can use a bandwidth of 40 MHz (allocation (i)).
- If the two links are close to each other, allocation (i) results in transmission arbitration in time-domain. Instead, it might be more efficient to opt for allocation (ii) in order to use orthogonal bands and reduce the time spent in backoff. In this case, however, some interference may still be observed due to power leakage creating adjacent-channel interference.
- If the links are far apart but link l has a poor channel quality, it can be beneficial to use allocation (iii). This is due to that, for a given transmit power, a narrow bandwidth packs more Watts per Hertz, which effectively increases the SNR and the transmission range [6]. Yet, using a narrow bandwidth also has the effect of increasing the time required to transmit a packet, which exacerbates the rate anomaly suffered by 802.11 [10], and can potentially decrease the overall efficiency.

- If, for instance, link k has a low traffic load, it does not need much spectrum and does not create much contention. In that case, allocation (iv) may be efficient.
- Finally, depending on the traffic loads, using allocation (v) with partially overlapping channels has the potential to create more efficient spectral re-use patterns [22].

Clearly, as also noted in [27], performance depends in a highly complex way on the actual topology, channel qualities, spectral configurations, etc. In particular, it is especially hard to predict in quantitative terms how a given configuration will perform. The complexity is further exacerbated if the nodes can adapt their transmit powers; although this feature can potentially improve spectral re-use [5], it is rarely used in practice as its impact is difficult to predict [21].

Due to the difficulty of predicting performance in the presence of complex interference patterns, the vast majority of works proposing models or optimizations for the PHY layer (e.g., [21], [22], [27]) are reduced to using SINR-based models. However, SINR models are not meant to capture 802.11 performance and, as we will see now, they can fail to capture important performance patterns.

B. An Example where SINR Models are Inappropriate

We now consider a real example from our testbed, again with two interfering links l and k. In this case, both links use 20 MHz of spectrum with the same center frequency (i.e., we consider a simpler setup with no spectral separation). The two links send saturated UDP traffic with packets of 1500 B, and they both use 802.11n in the 5.8 GHz band (void of external interference), using the same 2×2 MIMO configuration. Link l has a fixed transmit power set to 12 dBm, and k varies its transmit power from 3 dBm to 21 dBm. We measure the throughput obtained by l for two different pairs of links (l, k) on an indoor testbed (we give more details on our testbed and experimental setup in Section IV-A). For comparison, we also compute the information-theoretic capacity c_l of link l as

$$c_l = \text{constant} \cdot \log_2 (1 + \text{SINR}_l), \tag{1}$$

where the constant factor accounts for the bandwidth and MIMO configuration, and SINR_l denotes the SINR of link l. On such a two-link setup, the SINR is given by

$$\text{SINR}_l = \frac{P_{l \leftarrow l}}{N_0 + P_{l \leftarrow k}}, \tag{2}$$

where $P_{l \leftarrow l}$ (resp. $P_{l \leftarrow k}$) denotes the received power at the receiver of l (as measured by our NICs) from the transmitter of l (resp. from the transmitter of k), and N_0 is the background noise (also reported by our NICs).

We show both the measured throughput and the theoretic capacity for the two link pairs on Figure 3. The (schematized) topologies are shown at the top of the figure. For the first link pair, the throughput obtained by l decreases by about 50% when k increases its transmit power. This is due to an increased likelihood of collision at l's receiver and carrier-sensing activation at l's transmitter, as k increases its effective interference range. This qualitative trend is captured by the

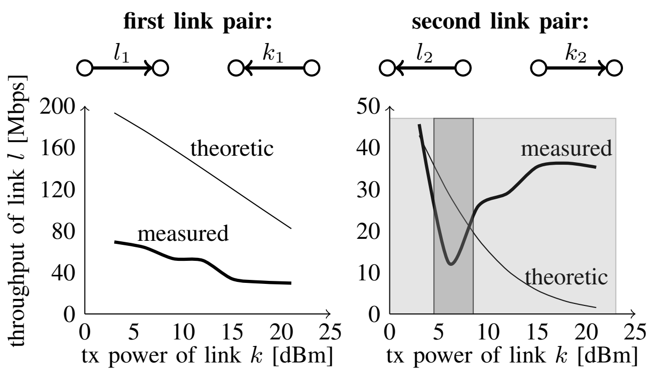

Figure 3: Measured throughput and theoretical capacity of l, when k varies its transmit power. The results are shown for two different pairs of links (l_1, k_1) and (l_2, k_2) from our testbed.

theoretical capacity, which decreases when $P_{l \leftarrow k}$ increases. However, in this case, the magnitude of the theoretical capacity is much higher than the actual throughput of the link.

The situation is different (and more surprising at first sight) for the second link pair. Here, we can decompose the measured performance in three distinct regimes (represented by three shaded regions in the figure). When k's transmit power is low, the links are nearly independent and l suffers little interference from k. When k's transmit power grows to intermediate values, k starts interfering with l. In this case, l carrier-senses k, and interference mitigation is done in time-domain via CSMA/CA. However, a closer inspection of packets reveals that k *itself* does not have a good channel quality (as it uses only an intermediate transmit power), which forces it to use relatively robust (and slow) modulations. As a result, in this intermediate regime, k consumes a significant portion of the time to transmit its packets, which reduces l's throughput (due to the rate-anomaly). Finally, when k uses a large transmit power, it also uses faster modulations, which has the apparently paradoxical effect of increasing l's throughput.

In this second example, the information-theoretic formulation for the capacity does not capture all these "802.11-specific" cross-layer and multi-modal effects. Instead, it shows a monotonic dependency on transmit power, because it treats the case of Gaussian channels subject to *constant* and *white noise* interference, with no intent of modeling 802.11 networks. In fact, in the cases where a time-sharing scheme such as CSMA/CA is employed, links often have the opportunity to transmit alone on the channel, thus without observing any interference at all during their transmission[1].

From these two simple examples, we observe that the theoretical capacity (i) might have a significantly different magnitude than the observed throughput, and (ii) might not capture the non-monotonic, multi-modal complex behaviors due to cross-layer interactions occurring when links are interfering (importantly, note that this observation holds for *any* model that is monotonic in the SINR).

Despite these problems – and despite the fact that SINR

[1]This is also the reason the actual throughput might be largely above the predicted capacity.

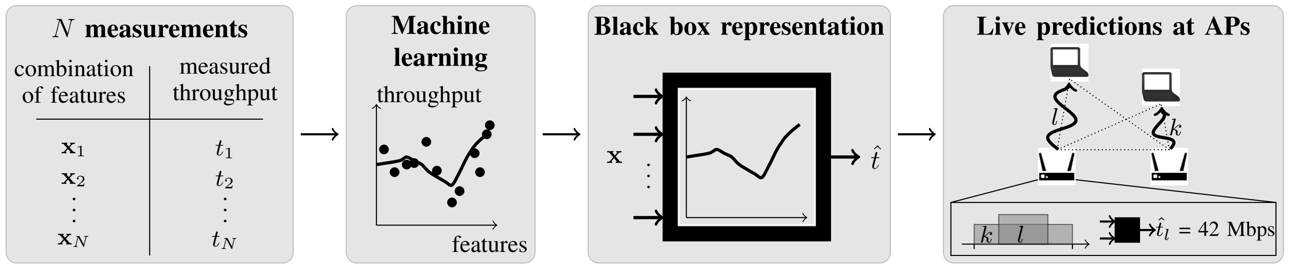

Figure 4: Overview of our learning setup.

models are usually not considered strong predictors of wireless performance – these models are still the models of choice for allocating resources at the PHY layer, due to their generality: By adapting judiciously the power values in the SINR Equation (2), it is possible to use variable transmit powers (as we just did), but also partially overlapping channels [22] and variable bandwidths [27] as inputs of SINR models. In addition, a large body of theoretic literature on *optimal* resource allocation also relies on SINR models in various contexts [3], [9], [14], [16], [18], [21], [25]. By contrast, MAC layer models such as Bianchi's are often accurate with homogeneous PHY configurations, but cannot be used to capture such heterogeneous PHY configurations.

III. Learning Performance Models

A. Approach

In the previous section, we observed that out-of-the-box SINR-based models might produce capacity predictions with a significantly different magnitude than the observed throughput. Therefore, a natural step to improve the accuracy is to seed (or *fit*) some parameters in SINR-based models (for instance, a factor controlling the magnitude of the prediction) to the observations of actual measurements. The approach of seeding a model with measurements has been taken in [20], [27], [28] and others (see Section VIII for a discussion).

In this paper, we also rely on an initial measurement phase. However, contrary to prior approaches, we do not try to fit or to seed a previously existing model (such as, for instance, SINR-based or Markov-based). Instead, we directly learn the model *itself* from the data, using supervised machine learning. Our overall approach is summarized in Figure 4, and it consists of four main steps:

1) **Measurement phase:** This phase consists in performing N short-duration controlled experiments. Considering again the black box representation of Figure 1 (although generalized for more than two links), each experiment consists in measuring the throughput of a given link l, for one particular combination of inputs (which we call *features*). This phase is relatively short; we observe in Section IV-D that it is possible to "learn" our entire indoor testbed with reasonable accuracy in less than 6 hours.
2) **Learning phase:** Once the measurements are obtained, this phase consists in finding a mathematical function that maps the features to observed throughputs. The function should be d-dimensional if there are d features, and it should approximate the throughput well on the measured data points. However, to be useful, it must not overfit existing measurements, which are intrinsically noisy. Instead, it should generalize to unseen combinations of input features (which can potentially relate to unseen nodes and links). Supervised machine learning provides us with precisely the tools to handle this challenge.
3) **Black box representation:** Once a function has been found which is both accurate and generalizable, we can discard the measurements and use the function itself to obtain throughput predictions.
4) **Usage in resource allocation:** This step is the operational phase. If we target distributed resource allocation, the black box can be used as an oracle by the access points themselves. For instance, in the two-links example of Figure 2, if l collaborates with k to acquire some of the features at a certain time instant, it can produce throughput predictions for various configurations (and thus choose efficient configurations without probing).

Importantly, we observe in Section IV-C that learned models continue to be useful in new or unseen environments, and that the training procedure does not need to be repeated when new wireless links come and go. We detail our procedure in the remainder of this section.

B. Feature Selection

Consider a link l, for which we want to predict saturated throughput (i.e., under saturated traffic load[2]) for arbitrary spectrum and transmit power configurations, given a set $\mathcal{N}_l$ of K neighboring links with arbitrary conditions, configurations and traffic demands. Such a scenario is shown in Figure 5 for $K = 2$. The features must include factors that impact the performance and are measurable by the transmitter of l and its immediate neighbors. We selected the following list of features, because it is known that they all have an immediate impact on performance [5], [6], [10], [22]:

- The power received by each node of l from every transmitting node, and the power received by every other node, from the transmitter of l. These quantities are denoted $P_1, \ldots, P_{11}$ in Figure 5 (assuming downlink traffic, from the APs to their clients). They depend on the transmit powers and the various channel gains, and they

[2]We target saturated throughput because it is the maximum achievable throughput in a given configuration. In particular, we assume that if throughput t is achievable, then any throughput $t' < t$ is also achievable.

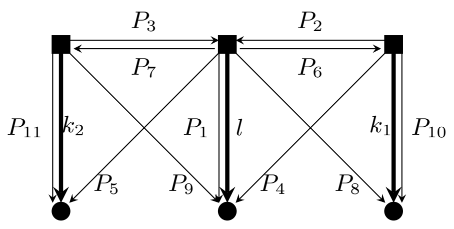

Figure 5: Throughput prediction setting with a link l and two neighboring links $\mathcal{N}_l = \{k_1, k_2\}$. We wish to predict the throughput that l could potentially obtain, given the various received powers, as well as the physical rates, channel widths, center frequencies, and traffic loads of k_1 and k_2.

can be easily measured online by commodity hardware using RSSI (received signal strength indicator). There are $5K + 1$ such power quantities in general.

- The channel widths used by l and by the links in $\mathcal{N}_l$. There are $K + 1$ such values.
- The spectral separations between the center frequency used by l, and the center frequencies used by all the other links in $\mathcal{N}_l$. There are K such values.
- The K average traffic loads of the links in $\mathcal{N}_l$.
- The physical rates (determined from the MCS index of 802.11n) used on each link in $\mathcal{N}_l$. These are used as features, because it is known that the modulation and coding employed for transmission have a strong effect on the performance of neighboring links [10]. There are again K such values.

Adding up the above-mentioned features, we have access to $d := 9K + 2$ quantities to estimate the throughput that l can obtain in the presence of K interferers. Note that this list of features is not an exhaustive list of the factors affecting performance that can be known or measured by the APs. For instance, we could make it more complete by including the packet sizes, higher order statistics to describe the traffic loads of interferers (instead of the mean only), or more detailed PHY layer information (e.g., to capture multipath effects or frequency-selective fading[3]). Including more features could further increase the predictive power and generality of the learned models. However, the features selected here already allow us to build useful models, while having the advantage of being simple and easy to acquire with commodity hardware.

C. Measurement Phase

The initial measurement phase consists of N measurements with different combinations of features. Some of the features can be directly controlled (namely, the channel widths, spectral separations and traffic loads) and others cannot (the received powers depend both on the transmit powers and channel gains, and the physical rates depend on the auto-rate mechanism used by the APs). Each of the N measurements consists of two sub-experiments. We first perform an experiment during which l is silent, in order to obtain a corresponding vector $\mathbf{x} \in \mathbb{R}^d$ of features (some of which are controlled, others are measured). We then repeat the experiment with l sending saturated traffic, and measure its throughput t_l. Our goal is to expose the learning procedure to as wide a variety of situations as possible. To this end, we apply the following sampling procedure for each of the N data points.

We start by selecting a link l uniformly at random among all the links formed by all the nodes of the network. We then sample K random interfering links, where K itself is randomly drawn between 0 and max_K, and max_K denotes a fixed upper bound on K. For l and the K links in $\mathcal{N}_l$, we sample transmit powers and spectral configurations uniformly at random from the set of configurations that do produce some interference (i.e., such that each link in $\mathcal{N}_l$ uses a band at least adjacent or partially overlapping with l). Finally, for each link k in $\mathcal{N}_l$, we sample a traffic load in the interval $(0, h(w_k)/K]$, where $h(w_k)$ is a value representing the maximum throughput achievable on an isolated link using bandwidth w_k. We take $h(20\text{ MHz}) = 80$ Mbps and $h(40\text{ MHz}) = 130$ Mbps in our training procedure, in line with the maximum achievable throughput of our 802.11n cards. Our goal is to predict performance for *arbitrary* interfering loads, and sampling the loads in this way allows us to expose the learning procedure to different environments with both light and heavy contention. In particular, we measured that the offered loads of the nodes in $\mathcal{N}_l$ was above capacity (i.e., saturated) in about 54% of the experiments (mainly due to inter-neighbors interference). The remaining experiments consist of non-saturated conditions.

Once the configurations have been chosen, we perform the first experiment with only the K interfering links active. During this experiment, we measure the average physical rates used by each of the K links in $\mathcal{N}_l$, and we group all the above-mentioned features in a vector $\mathbf{x}_i$. In order to vary K between 0 and max_K but keep features vectors of fixed dimension d, we append $9(max\_K - K)$ default "flag" values to $\mathbf{x}_i$, using -110 dBm for all the power values, and setting all the remaining features to zero[4]. We then perform the second experiment in the same conditions, but with link l sending saturated traffic, and we measure its achieved throughput. Each of the two sub-experiments constituting each of the N data points needs only to last a few seconds (in order to measure average physical rates and throughput), and the whole procedure is easily automated.

D. Learning

Let us write $\{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\} \subset \mathbb{R}^d \times \mathbb{R}$ for our set of measurements. Our goal is now to find a function $f : \mathbb{R}^d \to \mathbb{R}$ that maps $\mathbf{x}_i$ to a value close to t_i for each measurement i. This is an instance of a regression problem, and we learn the function f directly from the observed data.

Several techniques exist for learning functions from data. We do not go into the details, rather we present the overall characteristics of the different approaches that we considered

[3] [12] shows that considering channel measurements at the OFDM subcarrier level provides substantially more information on channel quality than crude RSSI. Unfortunately, such measurements are not available on our wireless cards.

[4]The current number of interfering links K is thus an implicit feature, encoded by the presence/absence of flag values.

in this paper, and we refer the reader to reference textbooks for details (see e.g., [2], [13]). We consider the following regression techniques.

Regression tree: This technique fits a binary tree to the data. Each feature vector corresponds to a path in the tree (from the root to a leaf), and each leaf corresponds to a (discretized) throughput value. The resulting model is elegant, because it yields predictions that can be evaluated by a sequence of "if-else" clauses on the features[5]. However, fitting an optimal tree is a NP-hard problem, and the obtained trees are usually sub-optimal. It also produces hard decision thresholds, which can affect generalization and accuracy.

Gradient Boosted Regression Trees (GBRT): This technique combines the predictions of M regression trees. Given a feature vector $\mathbf{x}$, the throughput is predicted as

$$\hat{t} = f(\mathbf{x}) = \sum_{m=1}^{M} \pi_m h_m(\mathbf{x}).$$

In the above expression, $h_m(\mathbf{x})$ denotes the prediction of the m-th tree, and the π_m's are the weighting coefficients (learned with gradient boosting [13]). We obtain the number of trees M as well as their depth by cross-validation. Using several trees has the potential to largely improve the predictive power compared to a single tree, however as we will see, it might still be subject to potential overfitting.

Support Vector Regression (SVR): For a feature vector $\mathbf{x}$, this method outputs a predicted throughput given by

$$\hat{t} = f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b,$$

where the α_i's and b are the fitted parameters (obtained by solving a convex minimization problem that accounts both for regression error and overfitting). The function $k(\cdot,\cdot)$ is a so-called *kernel function*. We generate our own SVR models using a common kernel function known as the radial basis function (RBF), specified by $k(\mathbf{x}_i, \mathbf{x}) = \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2)$, where γ is a parameter obtained by cross-validation. Usually, most of the α_i's are equal to zero, so SVR requires only a fraction of the initial measurements to be stored. Furthermore, this technique has a high descriptive power, and it can efficiently prevent overfitting.

SINR-based model: As a comparison to pure machine-learning techniques, we also fit SINR-based models to our measurements. In particular, we consider the following variant to Equation (1) for computing the theoretical capacity c_l of link l:

$$c_l = \Gamma \cdot w_l \cdot \log(1 + \text{SINR}_l), \tag{3}$$

where Γ is a constant that is fitted to measurements (using minimization of least square error), in order to correct for the magnitude problem mentioned in Section II. In addition, we also use the approach proposed in [22] in order to account for partially overlapping channels; namely, we scale each power

[5]For instance, on a simplistic tree of depth 2, a regression path could look like: "if received power $\leq X$ and frequency offset $> Y$, then predict Z".

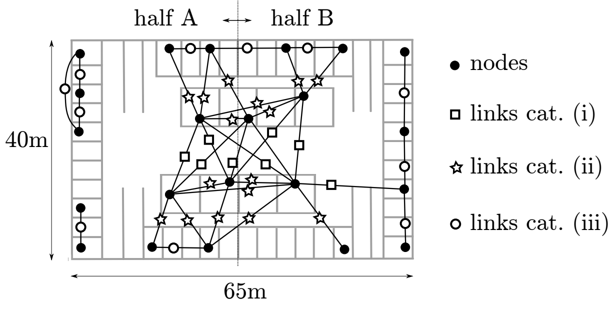

Figure 6: Layout of our 22-nodes wireless testbed. We also show the different link categories and the two halves of the testbed used in the experiments of Section IV-C2.

value appearing in the SINR Equation (2) by an appropriate value that accounts for the spectral overlap (assuming perfect bandpass filters). To the best of our knowledge, such models are the only existing models that can produce performance predictions at the same level of generality as our learned models (i.e., taking into account arbitrary spectral configurations with variable widths and variable transmit powers). In the next section, we evaluate the accuracy and generalizability of each class of models.

IV. Accuracy of Performance Predictions

In this section, we evaluate the accuracy and generalization of the different learning strategies in various conditions.

A. Experimental Setup and Methodology

Experimental Setup. We use a testbed of 22 nodes spread over an entire floor of an office building (see Figure 6). The nodes are Alix 2D2 boards, equipped with Atheros AR9220 wireless adapters. They run the OpenWrt 10.03 Linux distribution with the open source ath9k wireless drivers, and they use the default Minstrel autorate algorithm. We employ 20 and 40 MHz channel widths with 802.11n, 2×2 MIMO, and 10 different transmit power values in the set $\{3\text{dBm}, 5\text{dBm}, \ldots, 21\text{dBm}\}$. We use the 5.735-5.835 GHz band, which comprises a total of 100 MHz of spectrum.

Methodology. Our objective is to test the models with unknown combinations of features. As such, we only predict throughputs for data points that do not appear in the N measurements used for learning (or training). To this end, we always split our total set of measurements into a *training set* and a *test set*. The training set consists in the actual N measurements used for learning the models and their parameters, whereas the test set is used only once, for measuring the final accuracy.

We gathered a trace of 8900 measurements[6], with $max\_K = 3$. This set is voluntarily larger than what is actually needed, in order to allow us to test the effect of the number of measurements N on the models quality.

To evaluate the goodness of the regressions for the various models, we use the *coefficient of determination*[7] R^2. If we

[6]Our dataset is publicly available: http://www.hrzn.ch/data/lw-data.zip
[7]http://en.wikipedia.org/wiki/Coefficient_of_determination

have a test set with n throughput measurements $t_1, \dots, t_n$ and a given model predicts the throughputs $\hat{t}_1, \dots, \hat{t}_n$, then the coefficient of determination is given by

$$R^2 = 1 - \frac{\sum_i (t_i - \hat{t}_i)^2}{\sum_i (t_i - \bar{t})^2},$$

where $\bar{t}$ is the average throughput, given by $\bar{t} = \frac{1}{n}\sum_i t_i$. Concretely, the R^2-score quantifies how well a predictor does, compared to the simplest baseline strategy, which always predicts the mean throughput. It is equal to 1 if there is a perfect match between predicted and measured throughputs. Whereas, it can be negative if a strategy would do better by predicting the mean throughput. In addition to the R^2-score, we also compute the root mean squared error (RMSE), defined as $\text{RMSE} = \sqrt{\frac{1}{n}\sum_i (t_i - \hat{t}_i)^2}$. We used the Python machine learning package `scikit-learn` [24] to learn the various models.

B. Prediction Accuracy

In order to compare the accuracy of the different classes of models, we perform 50 consecutive splits of our measurements in training and test sets (50-fold cross validation). For each split, we evaluate the R^2-score and RMSE, and we show the average and standard deviations in Figure 7(a) for each class of model. In addition, we also show the detailed distribution of prediction errors in Figure 7(b) for models based on SVR and GBRT.

It appears clearly that the learned models, in particular the ones based on SVR and GBRT, perform significantly better than the SINR-based models. In terms of R^2-score, learned SVR and GBRT models improve the prediction accuracy by 54% and 71%, respectively, compared to SINR models (which, we recall, are the only known class of models capturing phenomena such as overlapping channels). In terms of error distribution, 90% of the errors made by learned models are between -25 Mbps and 25 Mbps, whereas 90% of the errors made by SINR-based models are between -35 Mbps and 36 Mbps. The fact that learned models are more accurate is remarkable; it demonstrates that, as far as performance prediction is concerned, learning abstract models coming from the machine learning domain can be much more efficient than trying to fit (or seed) pre-existing specialized models.

In order to visualize the actual predictions in detail, we also show a scatter plot of the predicted throughputs, against the actual measured throughputs, in Figure 8. We show both the predictions obtained by the SINR model and the learned SVR model (for reasons that will be clear soon, we show the results for SVR instead of GBRT, even though GBRT performs better in this particular setting). On these plots, the closer the points are to the diagonal, the better the prediction accuracy. Clearly, SVR models perform much better and produce fewer outlying predictions than SINR models.

Note that obtaining *perfect* predictions is impossible here, considering the fact that both the measured features and the throughput are highly noisy variables, measured with commodity hardware. To illustrate this, we examine in more detail

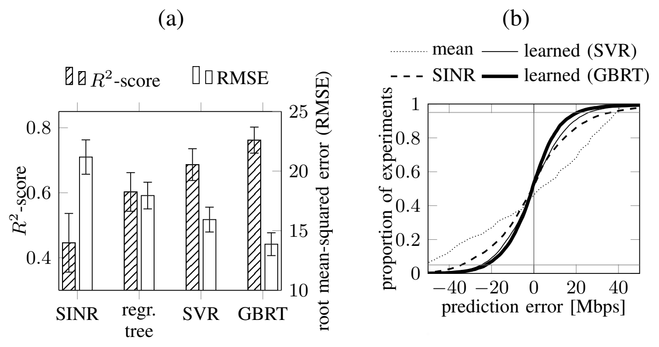

Figure 7: Summary of prediction performance for various models (a) and empirical CDF of prediction errors (b). The "mean" model in plot (b) represents the errors obtained by a baseline predictor that always predicts the mean throughput of the training set.

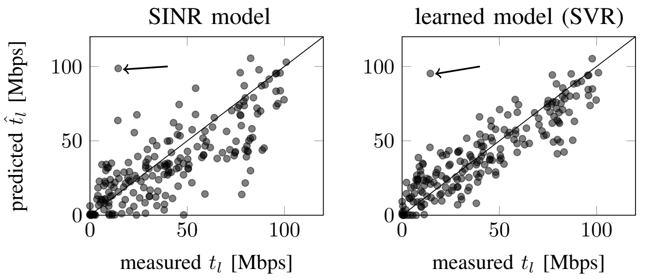

Figure 8: Predicted versus measured throughput, for SINR and a learned model, on a test set of 200 points.

the features corresponding to the worst prediction obtained by both models (shown by an arrow on the plots – incidentally, this is the same point for both models). This point corresponds to a link l subject to no (controlled) interference (i.e., $K = 0$), with an apparently good channel quality (the measured RSSI is -59 dBm), and using a bandwidth of 40 MHz, supposedly yielding the largest capacity. Yet, despite these features, the measured throughput was low. We can only speculate about the causes for this discrepancy; It may have been an especially unfavorable conjunction of high noise, both in the measurements of channel quality (which might have been worse than measured) and/or obtained throughput (which might have been temporarily altered by software factors). In any case, this example, although relatively extreme, illustrates the limits of throughput predictability with imperfect information.

C. Generalization

Due to the split between training set and test set, the previous results address cases where throughputs predictions are produced for unseen combinations of *features*. We now attempt to push our models further, by making them predict throughputs for unknown *links*, potentially belonging to different environments.

1) Predictions for Unknown Links: For each possible link l, we remove both l and its reverse link (obtained by inverting the transmitter and the receiver of l) from the training set. We then produce throughput predictions for each data point that contains l (or its reverse link), and show the results in Figure 9.

Compared with Figure 7(a), some models (especially the

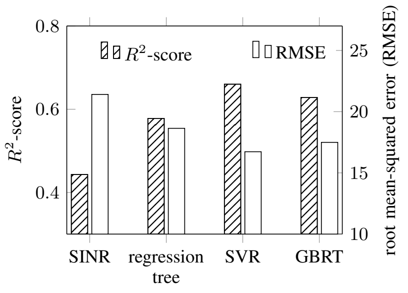

Figure 9: Prediction accuracy on links that have never been observed during the learning phase.

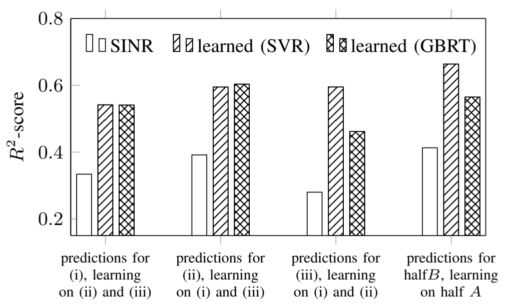

Figure 10: Prediction accuracy for never-observed groups of links. Even in the difficult cases, learned models largely outperform SINR models.

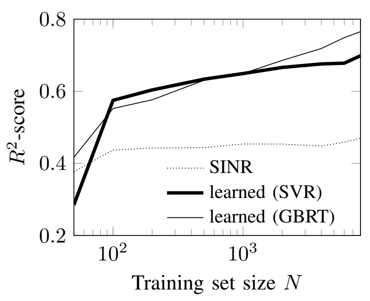

Figure 11: Prediction accuracy as a function of number of measurements N (note the logarithmic scale of the x-axis).

ones based on regression trees) see their accuracy slightly decreased. However, the models learned with SVR still perform remarkably well; in terms of R^2-score, their accuracy is reduced by less than 4%, and they still improve the accuracy by 49% compared to SINR-based models.

2) Different Environments: We now manually divide the links present in our trace in three distinct categories, depending on the type of attenuation that they experience. The categories are shown in Figure 6, and they correspond to the following link division: (i) links that traverse mostly empty space, (ii) links that traverse sparsely spaced walls and (iii) links that traverse densely spaced walls.

For each category, we remove all the links (and their reverse) belonging to this category from the training set. We then build the test set so as to predict throughput for links belonging *only* to this category. The goal of this experiment is to test prediction accuracy in the worst possible conditions: each model is learned on links that operate in conditions radically different than the conditions prevailing during the actual predictions. In addition to the three link categories (i)-(iii), we also split our testbed in two halves (also shown in Figure 6). Here too, we test the prediction accuracy when learning the models on the first half A of the testbed, and testing them on the second half B. The resulting accuracies are shown in Figure 10.

Even in these difficult cases, the learned models based on SVR show a graceful degradation and keep a relatively high accuracy (with R^2-scores always larger than 0.54). When producing predictions for the half B with models learned on the half A, models based on SVR even obtain similar accuracies as when learning using the full testbed. This allows us to draw some conclusions on the extent to which our method generalizes. Even when learning models on a different part of the testbed, or using radically different links, abstract models based on machine learning still have far more predictive power than measurement-seeded models based on SINR.

D. How Much Learning is Needed?

Finally, we measure the accuracy as a function of the training set size N. For different values of N, we learn models using N experiments sampled at random from our complete experiment trace. We then predict the throughput for all the other experiments, and measure the R^2-score. The results are shown in Figure 11. Using $N = 100$ training experiments is enough to obtain better accuracy than SINR models, and $N = 1000$ experiments already yield good predictive accuracies. If each experiment consists in 10 seconds of measurements (which is the duration that we employed), this means that an efficient performance model for an entire building-scale network such as ours can be learned in less than 6 hours.

V. Spectrum Allocation

We now present our algorithm for the joint allocation of spectrum and transmit power, which uses a learned model in order to predict performance in various configurations.

A. Model and Objective

Consider a set of L links, and a set $\mathcal{A}$ of access points (APs). Each link l is composed of one transmitter and one receiver, which operate on the same frequency band (i.e., using the same channel center frequency and width). In this paper we focus on the downlink, in line with asymmetric domestic Internet connections, where downlink traffic is largely dominating [29]. As such, we will use the terms transmitter and AP interchangeably. We write $l \in A$ if AP A the transmitter of link l. Let $\mathcal{F}, \mathcal{W}$ and $\mathcal{P}$ denote the finite sets of available center frequencies, channel widths and transmit powers, respectively. $\mathcal{C} := \{\mathcal{F}\times\mathcal{W}\times\mathcal{P}\}$ is thus the set of possible configurations. For link l, we denote by $f_l \in \mathcal{F}$ its center frequency, $w_l \in \mathcal{W}$ its channel width, and $p_l \in \mathcal{P}$ its transmit power. We impose that all the links sharing a common AP use the same spectrum configuration, and we define $\mathcal{S} \subseteq \mathcal{C}^{|\mathcal{A}|}$ the corresponding set of feasible configurations that satisfy this constraint. We write $c_A \in \mathcal{C}$ for the spectrum configuration of AP A, and more generally $c_{\mathcal{D}} \in \mathcal{C}^{|\mathcal{D}|}$ for the joint configuration of a set $\mathcal{D} \subseteq \mathcal{A}$ of APs. Finally, two APs are *neighbors* if they are close enough to interfere, and we write $\mathcal{N}_A$ the set of neighbors of AP A.

Our goal is to optimize jointly – and in a distributed way – the efficiency and fairness of such a network. To this end, we maximize the sum of link utility functions (see Section VI-A for examples of utility functions that achieve various efficiency/fairness tradeoffs):

$$\max_{\mathcal{S}} \quad \sum_l U_l(x_l), \tag{4}$$

where $U_l : \mathbb{R} \rightarrow \mathbb{R}$ is the (arbitrary[8]) utility function attached to link l, and x_l is the achievable (application layer) throughput of link l. The sum is taken over all the L links, and the complex dependencies on the spectral configurations are captured through the x_l variables.

B. Algorithm Description

The problem of finding *exact* solutions of Problem (4) is NP-hard (it is a generalization of the channel assignment problem, which itself solves a NP-hard graph coloring problem). Therefore, we use an iterative search method based on Gibbs sampling, in order to devise a practical algorithm that still provides asymptotic convergence guarantees. This tool has been used for different resource allocation problems (see e.g. [3], [18]). However, to the best of our knowledge, our algorithm is the first to optimize the channel center-frequencies, channel widths and transmit power.

Our distributed algorithm is shown in Algorithm 1. Each AP keeps an independent Poisson clock of rate λ. At each tick of the clock, the algorithm is run. We now describe this algorithm from the point of view of AP A. AP A first estimates its potential throughput, using the black box built in Section III, for each configuration $c_A \in \mathcal{C}$, using some information about its neighbors. This information consists of the neighboring configurations $c_{\mathcal{N}_A}$, the physical rates currently used by the neighbors, their traffic loads, as well as the channel gains. Such information needs to be exchanged on demand (but only once per algorithm iteration) among neighboring APs.

AP A then calculates $U^A(c_A)$, the sum of utilities of all links served by A, based on the estimated throughputs. It then sends a query request (line 11) to all its neighbors for their own estimated utilities for each configuration c_A (used by A). Let us now consider how this query is treated when AP A itself receives such a query, as shown starting in line 17. When AP A receives a query message from a neighbor B, it estimates the throughput that would be obtained by each of its links, for each possible configuration $c_B \in \mathcal{C}$ of B. AP A then computes $U^A(c_B)$ as the sum of utilities of all links $l \in A$ for each $c_B \in \mathcal{C}$, and sends these values to B.

When AP A receives replies from its neighbors, these values are stored in a matrix $\boldsymbol{U}_A^{\text{nb}}$, where $\boldsymbol{U}_A^{\text{nb}}(B, c_A)$ denotes the estimated utility of neighbor B when A uses configuration c_A. AP A then uses these values to compute $U_A^{\text{nb}}(c_A)$, the sum of neighbors' utilities when A uses configuration c_A, and $\widetilde{U}^A(c_A)$, the total estimate of the neighborhood utility, including A. This value is then used in line 16 to draw a random configuration according to the Gibbs distribution. Such a random sampling procedure converges to states that are arbitrarily close to the global optimum of Problem (4), as explained in the next section.

1) Configuration Sampling: We only give a brief justification of the asymptotic optimality of our scheme. A rigorous proof of optimality and convergence would follow an analysis similar to Borst et al. [3]. Denote $C_n \in \mathcal{S}$ the global state of the network at the n-th iteration of the algorithm. Each AP independently selects a (potentially new) configuration at instants of a Poisson clock, based only on the current configuration. C_n is thus a Markov chain, whose transitions probabilities are specified by the Gibbs distribution in line 16, which depends only on the sum of utilities in a given neighborhood. Using a reversibility argument, it can be shown (see [4]) that these transition probabilities are such that C_n converges in distribution, at geometric speed, to the stationary distribution with measure π given by $\pi(C) \propto \exp\left(\frac{\sum_l U_l(x_l)}{T}\right)$, where $C \in \mathcal{S}$ denotes a global state of configurations. This distribution is parametrized by T. For sufficiently low T, it assigns arbitrarily large probabilities to the states that are global optima of Problem (4).

Algorithm 1: Resource allocation at AP A

```
1  Initialization:
2  Set the temperature $T > 0$
3  Start with a random generalized configuration $c_A$
4  Start a Poisson clock with rate $\lambda$
5  Upon a tick of the Poisson clock:
6  for each $c_A \in \mathcal{C}$ do
7      for each link $l \in A$ do
8          estimate potential throughput $\hat{x}_l(c_A \cup c_{\mathcal{N}_A})$
9      compute $U^A(c_A) = \sum_{l \in A} U_l\left(\hat{x}_l(c_A \cup c_{\mathcal{N}_A})\right)$
10 for each neighbor $B \in \mathcal{N}_A$ do
11     send a query_u message to $B$
12 Upon reception of a reply_u message from $B$, fill matrix
   $\boldsymbol{U}_A^{\text{nb}}$, where $\boldsymbol{U}_A^{\text{nb}}(B, c_A)$ is the estimate utility of neighbor $B$
   when $A$ uses configuration $c_A$
13 for each $c_A \in \mathcal{C}$ do
14     compute $U_A^{\text{nb}}(c_A) = \sum_{B \in \mathcal{N}_A} \boldsymbol{U}_A^{\text{nb}}(B, c_A)$
15     compute $\widetilde{U}^A(c_A) = U^A(c_A) + U_A^{\text{nb}}(c_A)$
16 draw a new configuration $c_A$ at random, with probability
       $p(c_A) = \frac{\exp\left(\widetilde{U}^A(c_A)/T\right)}{\sum_{c'_A \in \mathcal{C}} \exp\left(\widetilde{U}^A(c'_A)/T\right)}$
17 Upon reception of a query_u message from a neighbor $B$:
18 for each $c_B \in \mathcal{C}$ do
19     for each link $l \in A$ do
20         estimate potential throughput $\hat{x}_l(c_A \cup c_B \cup c_{\mathcal{N}_A \setminus B})$
21     compute $U^A(c_B) = \sum_{l \in A} U_l\left(\hat{x}_l(c_A \cup c_B \cup c_{\mathcal{N}_A \setminus B})\right)$
22 send a reply_u message that contains the set
   $\{U^A(c_B) \ \forall \ c_B\}$ to $B$
```

VI. Algorithm Evaluation

A. Experimental Settings

Implementation. We implemented our algorithm in C++ using *Click* [19] in user space. We also implemented a distributed neighbor discovery mechanism, which consists in a simple rendez-vous protocol. The APs periodically switch to a pre-determined 20 MHz channel (to have the largest possible communication range), and send a broadcast frame (using the largest possible transmit power) that contains their

[8]In particular, the utility functions are not required to be concave.

public (wired) IP address. The neighboring APs that overhear this address then use their wired connection for the actual collaboration. In particular, the APs communicate the various channel qualities that they measure from the neighboring APs, as well as from their own and neighboring clients. They also inform their neighbors about their own traffic loads (which they can easily measure themselves). The performance predictions at lines 8 and 20 of Algorithm 1 are obtained with a black box model learned with SVR. Note that the algorithm deals with random configurations, channel gains and AP-clients combinations that have in general never been observed during the learning procedure.

Experimental Methodology. Unless otherwise stated, we use the following methodology and experimental settings to evaluate our algorithm. We randomly select between 8 and 10 AP-client pairs among the 22 nodes of our testbed. Each pair starts in a random configuration of channel, width and transmit power. We conduct our throughput measurements with saturated UDP traffic generated by `iperf`, with 1500 B packet size. After 600 seconds, we start our algorithm at each AP for 3000 seconds. Unless otherwise stated, the results shown on the plots are the averages and standard deviations obtained over 10 such runs. For the algorithm execution we set temperature $T = 0.01$ and the average wake-up time $\lambda = 600$ seconds (meaning that each AP "reevaluates" its spectral configuration every 10 minutes on average). In addition, we use the following three utility functions $U_l(x_l)$ in our study:

- $U_l(x_l) = x_l$. When all links use this utility function, the optimization problem (4) consists in maximizing the sum of throughputs, irrespective of other considerations such as fairness. We denote this utility function U^{thr}.
- $U_l(x_l) = \log(1 + x_l)$. Using this function is equivalent to maximizing proportional fairness; it provides a trade-off between efficiency and fairness by allocating more resources to links with larger potential throughput. We denote this utility function U^{prop}.
- $U_l(x_l; \alpha) = (1-\alpha)^{-1} x_l^{1-\alpha}$. This is the α-fairness utility function defined in [23], and parameterized by $\alpha \in \mathbb{R}$. The special case $\alpha = 1$ is separately defined as being equal to U^{prop}. Taking $\alpha \to \infty$ yields allocations that are max-min fair, and $1 < \alpha < \infty$ represents a compromise between proportional fairness and max-min fairness. We denote this utility function U^{α}.

Finally, we benchmark our algorithm against the one proposed in [18]. This algorithm uses a Gibbs sampler to find configurations of channel center frequencies that minimize the overall interference. We augment it to sample bandwidths and transmit powers as follows, for a fair comparison. We modulate the power received by a node a from a node b by (i) the transmit power used by b and (ii) the overlap between a's receive spectrum mask and b's transmit spectrum mask (see [22]), assuming perfect band-pass filters. We run the algorithm [18] (with our augmented metric) offline, using the whole testbed channel gains matrix in input, for 1000 iterations. The resulting allocations are denoted K+, and are run for 1000 seconds in our testbed. This is repeated 10 times to obtain confidence intervals.

B. Results

We investigate several aspects of our algorithm. We first evaluate its overall convergence and performance. Next, we test the influence of the optimization criteria by studying the performance, in terms of throughput and fairness, using various utility functions. Throughout this process, we observe in particular that our learned model is instrumental for achieving the performance gains. Finally, we examine in detail the spectrum allocations found by our algorithm, and we observe that they directly depend on the traffic loads and on the chosen utility functions.

Convergence and Performance. Figure 12 shows the temporal evolution of the average total sum of throughputs, when all links use U^{thr}, as well as the average sum for K+. K+'s single criterion to reduce interference does not result in significant throughput gain over random configurations in this case. This is because the K+ algorithm favors narrow channel widths and lower transmit powers in the case of channel overlap, which may not be beneficial in general. In contrast, our algorithm converges relatively fast (the mean wake-up period of 10 minutes means that APs wake-up for the first time after 1200 seconds on average). The final total throughputs are about 30% higher than those obtained with K+ or when random configurations are used[9].

Comparison of Utility Functions. We now compare the performance for various utility functions. We conduct experiments where all the links use U^{thr}, U^{prop}, or U^{α} with $\alpha = 4$. Figure 13 shows the steady-state throughput and Jain's fairness index, obtained by computing $\left(\sum_l x_l\right)^2 / (L \cdot \sum_l x_l^2)$, where x_l is the throughput obtained by link l.

As we have mentioned in Section VI-A, utility functions determine a trade-off between throughput and fairness. Quite remarkably, the practical results obtained on the testbed reflect well the objectives of the various utility functions: U^{thr} provides the greatest throughput, while both U^{prop} and U^{α} improve fairness. Furthermore, in line with theoretical expectations, U^{α} provides slightly better fairness and lower throughput than U^{prop}. To the best of our knowledge, this is the first observation that the framework of utility maximization can be used with spectrum assignment to achieve various optimization objectives in a real testbed. Overall, compared to random configurations and the K+ algorithm, it appears that all utility functions perform well both in terms of throughput and fairness.

Gains of the Learned Model over SINR-based Model. The key element that allows our algorithm to jointly optimize the network over all three parameters is our learned black box model. In Section IV, we compared its accuracy with that of measurement-seeded SINR models given by Equation (3). We now examine how this improved accuracy translates to the

[9]We observed similar convergence for TCP traffic with about 20% throughput gain.

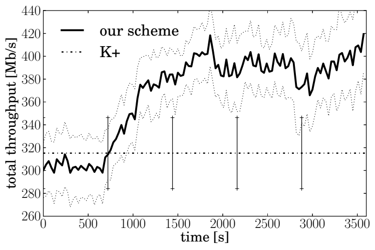

Figure 12: Sum of throughputs as a function of time. The first algorithm iteration happens after 1200 seconds on average. "K+" denotes the throughput obtained with the augmented algorithm of [18]

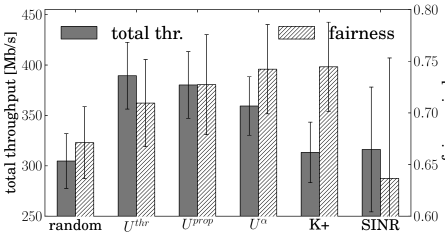

Figure 13: Throughput and fairness for different utility functions. We also show the results for random configurations, the "K+" algorithm, and our algorithm with U^{thr}, but using the measurement-seeded SINR model (3) instead of a learned model.

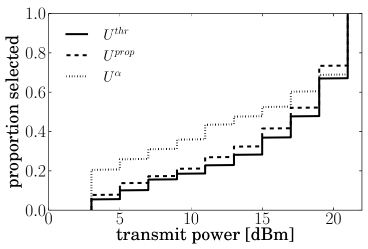

Figure 14: CDF of transmit powers sampled with different utilities and sampling policies. Fair policies sample lower transmit powers.

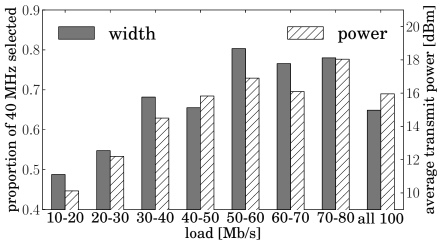

Figure 15: Proportion of time a bandwidth of 40 MHz is sampled, and average sampled transmit power, as a function of the AP traffic load. Efficient load balancing is achieved, as heavily loaded APs sample wider bandwidths and larger transmit powers.

enhanced performance of our algorithm. To this end, we also plot the throughput and fairness for U^{thr} obtained with the measurement-seeded SINR model in Figure 13 (labeled SINR). In this case, both the throughput and fairness significantly decrease (and become more variable), compared to the case where the algorithm uses the learned model.

Selected Configurations. We now investigate how the algorithm selects its spectral and power configurations, in different traffic load and utility functions scenarios. Figure 14 shows the distribution of the transmit powers selected by the algorithm (over all nodes and all experiments), for the three utility functions. It appears that fairer policies tend to use low transmit power for a higher fraction of time (see U^{α} vs U^{thr}). In turn, this means that the aggressiveness of the configurations (here in spatial domain), can be directly controlled by the utility functions.

We now study the impact of traffic loads, as it is taken into account by our models. To this end, we perform experiments where each link has a traffic load randomly chosen between 10 and 80 Mb/s. In these experiments, we use the utility functions $U_l(x_l) = \min\{x_l/load_l, 1\}$, where $load_l$ is the load of link l. This function is equivalent to U^{thr} when all links have a load of 100 Mb/s (which corresponds to a saturated situation in our case), and it is maximized as long as all links obtain a throughput that satisfy their demand[10]. Figure 15 shows the proportion of time (over all nodes and all experiments) that the algorithm selects a 40 MHz channel width (on the left y-axis), and the average transmit power in dBm (right y-axis), as a function of traffic load at the AP that makes these choices.

[10]Note that in this experiment, the links use different utility functions that depend on their (random) traffic demand.

We observe an elegant load-balancing pattern, as APs with heavier loads use more spectrum and larger transmit powers. This is a desirable feature (see e.g., [11]), and it directly relies on the ability of our learned models to suitably capture the effect of variable traffic loads. Interestingly, note that, when *all* APs generate 100 Mb/s of load (labeled "all 100" in Figure 15), the APs lower their resource consumption compared to cases with heterogeneous loads. This is because, in these cases, heavily-loaded APs compete with other heavily-loaded APs, and they naturally collaborate to share spectrum equitably. We thus deduce that the various utilities, fairness objectives and traffic loads directly impact the spectrum allocation patterns. In particular, both transmit power and channel widths are used to load-balance the spectrum usage as a function of the optimization criteria.

VII. Limitations and Discussion

We have evaluated our learned models in static conditions, a setting for which throughput prediction is somewhat easier (compared to say, high mobility with fast fading, short channel coherence times, etc). This is because, in this paper we deliberately restrict ourselves to using features easily accessible on commodity hardware (e.g., RSSI measurements). Such features are only meaningful on relatively coarse timescales (typically seconds) and cannot capture such fast-changing phenomena. In this sense, our black boxes suffer the same timescale limitations as any model (including SINR) using similar measurements. We leave for future work the evaluation of a similar learning framework using features operating at shorter timescales.

Importantly, using features operating at relatively coarse timescales already allows our learned models to be useful in practice. In Section V, we considered a setting where the global spectrum consumption are re-evaluated every few minutes by the APs[11]. Such global, relatively slow-varying spectrum allocation complements well (and provides more spectrum to) existing PHY techniques operating at fast timescales, such as interference cancellation and alignment.

[11]At faster timescales, the overhead of switching to different spectrum bands on commodity hardware would exceed the benefits of employing efficient spectrum allocations.

VIII. Related Work

Performance Models. Several papers propose measurement-based approaches to model performance and interference in 802.11 networks. In particular, [17], [20], [26], [28] propose to conduct initial measurement campaigns (where the number of measurements is typically a function of the number of nodes present in the network), in order to fit various performance models. [28] fits a model based on the SINR in order to estimate the packet loss probability, whereas [17], [26] and [20] use measurements-based Markov chain models to predict the capacity and/or interference of 802.11 networks. All of the above models are agnostic to the spectral configurations of the nodes, and they are designed to work when the links operate with a fixed channel width. In this paper, we also use an initial measurement phase. However, the crucial difference with our approach is that we are not constrained to any particular model, but rather employ supervised machine learning to learn *any* suitable model that captures both PHY and MAC layer complexities together.

[12] observes that measurements at the OFDM subcarrier level largely improves the accuracy of performance prediction. Unfortunately, the method does not take interference into account, and it cannot be used to make performance predictions when several links operate at the same time.

Finally, a few papers propose to use machine learning techniques in the context of wireless networks. [8] discusses the use of k-NN for link adaptation and [7] proposes an architecture for cognitive radios with learning abilities. However, these works do not attempt to predict performance. To the best of our knowledge, ours is the first work using machine learning to predict actual Wi-Fi performance.

Resource Allocation. Some recent works consider simultaneous channel center frequency and width allocation for 802.11 networks. [27] runs the spectrum allocation jointly with scheduling decisions at a central controller, and [15] proposes a distributed algorithm for the joint allocation of center frequencies and bandwidths. None of these algorithms considers the transmit power, and they do not adapt to various utility functions. Our learned models predict achievable throughputs, which can be directly plugged into the utility maximization framework. This removes the need to use indirect optimization objectives (such as minimization of interference, which often does not coincide with performance maximization [15]).

The theoretical works that are the closest to ours are [3], [25], [14]. These papers propose optimal algorithms for channel and/or power selection, but do not consider channel width. They are also based on Gibbs sampling or equivalent MCMC methods, but they have not been implemented in real networks. To the best of our knowledge, our work is the first to show that utility maximization can be performed in a spectrum assignment context, using a real testbed.

IX. Conclusions

We investigated and validated a new approach for predicting the performance of Wi-Fi networks. Rather than manually fitting complex models to capture complex dependencies, we showed that it is possible to directly learn the models *themselves*, from a limited set of observed measurements. This approach bypasses the usual modeling process, which requires both deep knowledge and tedious analysis, and yet often yields models that are either too restricted or too inaccurate. We observed that abstract black box models built using supervised machine learning techniques – without any deep knowledge of the complex interference dynamics of 802.11 networks – can largely outperform the dominant class of SINR-based models. Further, we have shown that these models still work when they have to predict performance for links that have never been observed during the learning phase.

We have used one such model as an oracle in a new distributed utility-optimal resource allocation algorithm. We observed that our algorithm adapts well to various optimization criteria, and that our learned model is instrumental for achieving good performance.

References

[1] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 2000.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[3] S. Borst, M. Markakis, and I. Saniee. Distributed power allocation and user assignment in OFDMA cellular networks. In *Allerton*, 2011.

[4] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag New York Inc., corr. edition, Feb. 2001.

[5] I. Broustis, J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos. Implications of power control in wireless networks: a quantitative study. In *PAM*, 2007.

[6] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. In *ACM SIGCOMM*, 2008.

[7] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea. Applications of machine learning to cognitive radio networks. *IEEE Wireless Communications*, 2007.

[8] R. C. Daniels and R. Heath. An online learning framework for link adaptation in wireless networks. In *Information Theory and Applications Workshop, 2009*.

[9] R. Etkin, A. Parekh, and D. Tse. Spectrum sharing for unlicensed bands. *Selected Areas in Communications, IEEE Journal on*, 25(3):517–528, 2007.

[10] M. H. Franck, F. Rousseau, G. Berger-sabbatel, and A. Duda. Performance anomaly of 802.11b. In *IEEE INFOCOM*, 2003.

[11] R. Gummadi and H. Balakrishnan. Wireless networks should spread spectrum based on demands. In *ACM HotNets*, 2008.

[12] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM Sigcomm*, 2010.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2nd edition, 2008.

[14] N. Hegde and A. Proutiere. Optimal decentralized spectrum sharing: A simulation approach. In *Proc. CISS*, Mar. 2012.

[15] J. Herzen, R. Merz, and P. Thiran. Distributed spectrum assignment for home wlans. In *IEEE INFOCOM*, 2013.

[16] J. Huang, R. A. Berry, and M. L. Honig. Auction-based spectrum sharing. *Mobile Networks and Applications*, 11(3):405–418, 2006.

[17] A. Kashyap, S. Ganguly, and S. R. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *ACM MobiCom*, 2007.

[18] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-based self organization of interfering 802.11 wireless access networks. In *IEEE INFOCOM*, 2007.

[19] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems, 2000.*

[20] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner. Predictable performance optimization for wireless networks. In *ACM SIGCOMM*, 2008.

[21] V. Mhatre, K. Papagiannaki, and F. Baccelli. Interference mitigation through power control in high density 802.11 WLANs. In *IEEE INFOCOM*, 2007.

[22] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh. Partially overlapped channels not considered harmful. In *ACM Sigmetrics*, 2006.

[23] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking (ToN)*, 8(5):556–567, 2000.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, and al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] L. P. Qian, Y. J. Zhang, and M. Chiang. Globally optimal distributed power control for nonconcave utility maximization. *CoRR*, abs/1101.0204, 2011.

[26] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan. A general model of wireless interference. In *ACM MobiCom*, 2007.

[27] S. Rayanchu, V. Shrivastava, S. Banerjee, and R. Chandra. FLUID: Improving throughputs in enterprise wireless lans through flexible channelization. In *ACM MobiCom*, 2011.

[28] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *ACM SIGCOMM*, 2006.

[29] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband internet performance: a view from the gateway. In *ACM SIGCOMM*, 2011.