SpecSense: Crowdsensing for Efficient Querying of Spectrum Occupancy

Ayon Chakraborty†, Md. Shaifur Rahman†, Himanshu Gupta and Samir R. Das
Computer Science Department, Stony Brook University, Stony Brook, NY 11794, U.S.A.
{aychakrabort, mdsrahman, hgupta, samir}@cs.stonybrook.edu

***Abstract*—We describe an end-to-end platform called `SpecSense` to support large scale spectrum monitoring. `SpecSense` crowdsources spectrum monitoring to low-cost, low-power commodity SDR/embedded platforms and provides necessary analytics support in a central spectrum server. In this work, we describe `SpecSense` and address specific challenges related to accurately estimate spectrum occupancy on demand with low overhead. To address the accuracy question, we augment state-of-the-art spatial interpolation techniques to accommodate scenarios where RF propagation characteristics change across space. To address the overhead question, we solve the sensor selection problem to select the minimum number of spectrum sensors that can best estimate the spectrum at the requested locations.**

I. Introduction

There is a growing realization that the RF spectrum must be treated as an important natural resource that is in limited supply. This stems from unbridled growth of mobile data and the need to support emerging applications such as M2M communications, telemedicine, autonomous cars/drones and mobile virtual/augmented reality. Policy makers have been promoting new forms of spectrum sharing models to replace the regimented spectrum allocation models in existence today [3], [6]. The goal is to improve spectrum utilization as a means to alleviate any future spectrum crunch.

Just like any other resource with mismatched demand and supply, all steps towards better utilization from both technical and policy angles have increased the need for large scale spectrum monitoring [15], [25]. Large scale spectrum monitoring serves two primary purposes: (i) it can aid effective spectrum sharing technologies by identifying spectrum opportunities [3], [13]; (ii) it can act as a vehicle for deeper understanding of spectrum use across time and space [25]. The latter serves as a driver for future policy and technical directions. Alongside, indirect needs are also growing such as spectrum patrolling to detect unauthorized spectrum use [18], [37]. While the need for RF spectrum monitoring is not a new realization by itself, current approaches suffer from two significant limitations:

1) *Lack of scalability:* Most exiting approaches on RF spectrum monitoring [4], [20], [25] do not scale. For example, approaches such as Specnet [20] or Microsoft spectrum observatory [4] require use of relatively expensive spectrum sensing stations that are networked with data delivered to central, perhaps cloud-based platforms.[1] Alternatives such as Vscope [36] can provide fine-grain data across space, but are limited to contracted vehicular platforms.
2) *Lack of application support:* There is a growing body of work on applications that can benefit from spectrum awareness, such as spectrum opportunity detection, shared spectrum models [6], [12], [13], RF-based localization [22], transmitter identification [18], [37], spectrum analytics [33], etc. However, there is virtually no attempt in the community to couple such applications to large-scale spectrum monitoring. This is a missed opportunity as applications are important drivers for large-scale deployments.

In the `SpecSense` project, we are creating an infrastructure that integrates scalable RF spectrum monitoring with application support, and build an end-to-end system with *monitoring data and connected apps* working seamlessly at scale. The idea is to develop (1) an effective *crowdsourcing mechanism* using low-cost, low-power sensors [26], [28] to support distributed fine grain spectrum sensing, along with (2) necessary support for spectrum-aware applications. There is a growing evidence that incentive mechanisms can be developed to enable crowdsourcing measurements of this kind once data is considered valuable. A good example is FlightAware [2], a popular flight tracking company, which uses low-cost commodity RF sensor hardware (similar platforms as in this work) to capture ADS-B signals emitted from airplanes flying overhead. The power of FlightAware comes from crowdsourcing with 5000+ sensing sites worldwide, collating such data on the backend to track flights.

Fig. 1 provides an an overview of `SpecSense`. In `SpecSense`, the spectrum-aware apps are the drivers and spectrum data is collected and processed as needed by such apps. Supporting such on-demand mechanism is critical as large-scale spectrum data with fine time/frequency precision is unrealistic to collect and maintain.

Contributions: A large number of interesting technical challenges arise in `SpecSense`. For brevity we will focus our attention to only a few of them in this paper. We first show how crowdsourcing with large number of inexpensive RF sensors can be beneficial (Section II). In `SpecSense`, many applications have the canonical need to estimate the RF power

†The first two authors are co-primary student authors.

[1]This limitation is clearly evidenced by very limited actual deployment of Microsoft's observatory (only 11 stations in the US) even after several years of effort and all software tools made available.

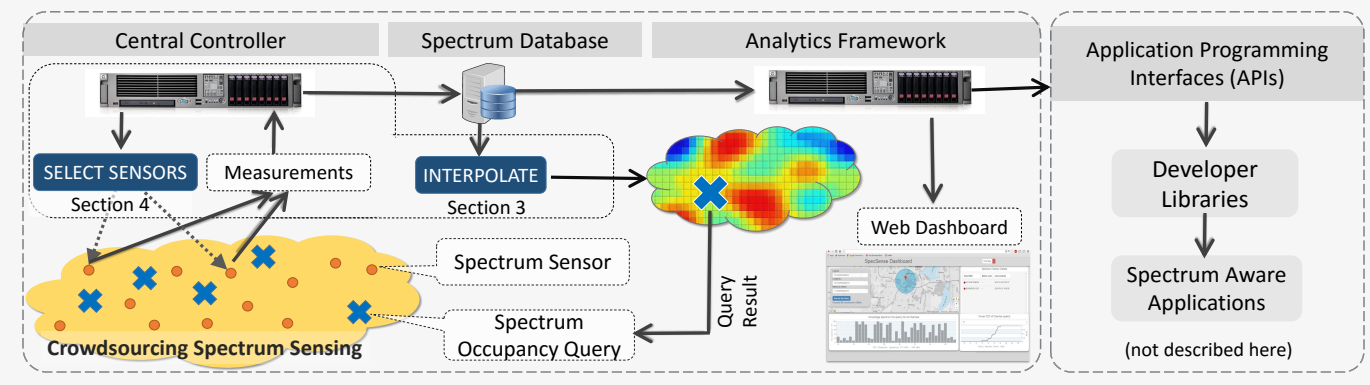


Fig. 1: End-to-end overview of SpecSense, with pointers to section numbers where the relevant component have been described.

within a specific frequency range at certain locations. We will address two challenges that arise to efficiently answer such "spectrum occupancy" queries.

(i) *Accurate spatial interpolation of RF signals:* To accurately answer spectrum occupancy queries, we need to develop effective techniques for spatial interpolation of RF signals. Thus, we extend the state-of-the-art spatial interpolation techniques to our specific context of RF signals (Section III).

(ii) *Optimizing sensor selection:* Sensors in SpecSense may run on battery-driven embedded platforms; thus, optimizing their battery drain and/or backhaul data communication is important. Many sensors depending on their location and/or capability may provide little benefit to the application's need of estimating spectrum occupancy. Thus, we develop techniques to select the best set of sensors under a given budget to serve a given set of spectrum occupancy queries (Section IV).

A systems level performance evaluation is presented in Section V in the prototype SpecSense system.

II. Power of Inexpensive Sensors

SpecSense's reliance on crowdsourcing means that the spectrum sensors cannot be large, power hungry, or expensive. In case of spectrum monitoring, a question arises whether inexpensive sensors with higher noise can be effective at all, compared to, say, expensive and more accurate lab-grade sensors. To investigate this issue, we have run a measurement study that shows that the inaccuracy of individual sensors can be countered by simply having more numerous sensors.

Our measurements scan part of the UHF TV spectrum (channel 21–51) and log: (i) the power spectral density (i.e., power for each frequency bin) for each 6 MHz TV channel, and (ii) the location coordinates of the measurement point. We use three different software-defined radio (SDR) platforms simultaneously on a moving vehicle to do the same measurements. One of them, ThinkRF WSA5000 [8], is a wide-band spectrum analyzer providing industry-standard sensitivity, tuning range, instantaneous bandwidth and scan rate. It serves as the reference platform as an example of a laboratory-grade, well-provisioned spectrum sensor with a high sensitivity. The two other platforms are BladeRF [1] and RTL-SDR dongle [14], [26]. These are USB-powered devices with far limited capability. Their cost and power consumption are 1 order (BladeRF) to 2 orders (RTL-SDR) of magnitude lower than ThinkRF.

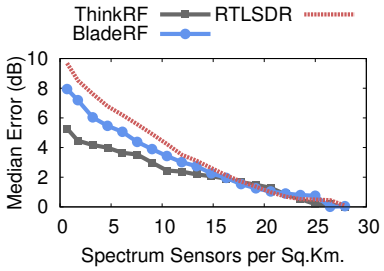


Fig. 2: Error Performance vs. deployment density for spectrum sensors of different types.

For each platform, we collected 4K+ measurements over different locations over a ≈150 mile drive. Using the geo-tagged spectrum measurements, we evaluated the total power in each of the 30 TV channels as sensed by these sensors at various locations. Then, we create a spectrum occupancy map by estimating the channel power at any given location in the given region using spatial interpolation (IDW; see Section III). We assume that measurements from the reference device (ThinkRF) provide the ground truth.

Using this data set, we can now sub-sample the measurement points randomly to create scenarios with less dense measurement data is available. See Fig. 2 for a range of such sub-sampling cases and the corresponding median errors (in dB) from the ground truth.[2] Clearly, all devices have decreasing error with higher density. While ThinkRF clearly has much lower error for a given density, the same error performance is possible at a somewhat higher density using the other sensors. For example, the median error for ThinkRF at 5 sensors/sq. km. is similar to the median error for the RTL-SDR dongle at about 12 sensors/sq. km, roughly a 2.5× higher density. BladeRF performs in between. Also, note that

[2]The errors are computed only at locations where ground truth measurements are available.

at high density of deployment (beyond 15 sensors per sq. km.) all sensors perform similarly regardless of capability. At this density the interpolation error dominates.

Clearly, the tradeoff is in favor of inexpensive sensors, as higher density still results in lower total cost and power with the added advantage of ubiquitous operation. There are several theoretical results in this space that also makes similar observations albeit with respect to specific signals and sensing models (e.g., [19]). `SpecSense` does not rely on specific sensors, but our current design uses above-mentioned commodity USB-powered SDRs with mobile phone or commodity embedded platforms.

III. Spatial Interpolation of RF Signals

Spatial interpolation of RF signal is the key enabler for many applications of `SpecSense`. The confluence of highly granular data gathering ability in `SpecSense` and the need for accurately answering the spectrum occupancy queries by applications require revisiting of spatial interpolation techniques. While prior work (e.g., [34]) has indeed considered spatial interpolation of spectrum data, they either used synthetic data or had access to much less granular data than what `SpecSense` is able to gather. Thus, some of the inefficiencies have not been exposed. Our specific contribution here is showcasing improved spatial interpolation of RF signals by data collected via `SpecSense`. Specifically, we extend the well-known *Ordinary Kriging (OK)* interpolation technique by (i) "detrending" the signal by averaging the path-loss exponent, and (ii) partitioning the given spatial region based on path-loss characteristics.

A. Basics of Spatial Interpolation of RF Signals

Many interpolation techniques have been explored for RF signals in recent works [10], [11], [29], [34], heavily borrowing on huge literature available in the general area of geospatial interpolation [24]. In a recent paper of interest [23], the authors have presented a performance evaluation of various techniques over signal strength data collected using commodity smartphones, and have observed that Ordinary Kriging (OK), Universal Kriging (UK) and Inverse Distance Weighting (IDW) methods perform better than other techniques. Also, [24], [30] note that the performance of UK deteriorates significantly with decrease in density of the observation data points. Because of this concern, we consider only IDW and OK in evaluating `SpecSense`. We describe these techniques next.

Inverse Distance Weighting (IDW): This is a straightforward interpolation technique that estimates the value at the location s_0 to be a weighted average of known values at nearby locations, weighted by inverse of their (powers of) distances from s_0.

Ordinary Kriging (OK): Like IDW, Ordinary Kriging also defines the predicted value as a linear combination of the known neighboring values, but unlike IDW, the weights are computed by minimizing the prediction variance (under certain assumptions). The main advantage of Kriging over other spatial interpolation techniques is that it considers the structure of the spatial correlation (deduced through *semivariograms*), and thus, yielding more reliable predictions. We start with some basic definitions.

Let $s \in \mathbb{R}^d$ be a generic location in a d-dimensional Euclidean space and $\{Z(s), s \in \mathbb{R}^d\}$ be a spatial random function (rf), with Z denoting the attribute/signal of interest. We assume that $Z(s)$ is continuous, i.e., the attribute Z can be observed at any point of the domain.

Semivariogram; Second-order Stationary. *Semivariogram* for a pair of locations (s_i, s_j) is denoted as $\gamma(s_i, s_j)$ and is defined as half of the variance of the difference between the field values at these locations.

A random function $\{Z(s), s \in \mathbb{R}^d\}$ is said to be *second-order stationary*, if the following two conditions hold: (i) the expectation $E[Z(s)]$ is a constant, and (ii) The semivariogram at a pair of locations s and $s+h$ depends only on the vector h (called the *lag*), i.e., $\gamma(s, s+h) = \gamma(h)$ for all s and h.

OK System of Equations. Let $\{Z(s)\}$ be a second-order stationary random function. Given observation values $Z(s_1), Z(s_2), \ldots, Z(s_n)$ at n locations, we wish to find the estimate $\widehat{Z}(s_0)$ of the value $Z(s_0)$ at location s_0. In particular, we seek a linear function predictor $\widehat{Z}(s_0) = \sum_{i=1}^{n} \lambda_i Z(s_i)$ that minimizes $V[\widehat{Z}(s_0) - Z(s_0)]$, the variance of the prediction error, where λ_i are the weights to be derived. With some arithmetic manipulation and writing $\gamma_{ij} = \gamma(s_i, s_j)$, the goal reduces to:

$$\text{Minimize } \; 2\sum_{i=1}^{n} \lambda_i \gamma_{i0} - \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j \gamma_{ij} \;\; \text{subject to } \sum_{i=1}^{n} \lambda_i = 1$$

The above is solved using the Lagrange multiplier method with a multiplier α, and results in the following system of Ordinary Kriging equations:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \alpha \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1n} & 1 \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2n} & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \gamma_{n1} & \gamma_{n2} & \cdots & \gamma_{nn} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma_{10} \\ \gamma_{20} \\ \vdots \\ \gamma_{n0} \\ 1 \end{bmatrix} \quad (1)$$

Using the OK interpolation technique involves: (i) determining the semivariogram function to estimate γ_{ij} values (as discussed below), (ii) using the above system of equations to predict the value at new locations.

Determining the semivariogram function: To use the above system of equations (Eqn. (1), we need to compute γ_{ij} values from the observation data. The simplest estimator for $\gamma(h)$ (and thus, for any γ_{ij} due to second-order stationary property) is [16]:

$$\gamma(h) = \frac{1}{2|N(h)|} \sum_{(s_i, s_j) \in N(h)} (Z(s_i) - Z(s_j))^2,$$

where $N(h)$ are the pairs of observations such that $h - \epsilon < |s_i - s_j| < h + \epsilon$ with ϵ being a small tolerance parameter. Now, to make the semivariogram function continuous and negative-semidefinite, a parametric semivariogram model is usually

fitted to the above estimated model. We use the commonly used exponential model with two parameters C and a:

$$\gamma(h) = C\left(1 - e^{-(h/a)}\right).$$

See Fig. 4 for an example.

B. Improving OK by Detrending and Partitioning

In this section, we introduce two techniques to improve OK in our context of RF signal interpolation: (i) detrending the observation data based on average path-loss exponent, and (ii) partitioning the given spatial region based on path-loss characteristics.

Detrending the observation data: Recall that the rf $\{Z(s)\}$ must have a constant mean to derive the OK system of equations. To ensure this condition, we decompose the received signal (in dBm) at s_i into two parts:

$$Z(s_i) = L(s_i) + \delta(s_i),$$

where $L(s_i)$ is estimated independently as shown below and $\delta(s_i)$ is the component estimated by the OK technique. Here, we implicitly assume the log-normal path-loss model, where the received power at a distance of d from the transmitter with a transmitter power P is given by:

$$P - 10\alpha \log_{10}(d) + N(0, \sigma^2) \tag{2}$$

where $N(0, \sigma^2)$ reflects the attenuation due to flat fading (e.g., shadowing) and is represented by a Gaussian (or normal) random variable with zero mean and variance σ^2, α is the path-loss exponent, and all power values are expressed in dBm. Thus, we let $L(s_i)$ represent the $P - 10\alpha \log_{10}(d)$ component of the received power and estimate it as described below, and use OK to estimate $\delta(s_i)$ which has a constant mean of zero.

Estimating $L(s_i)$ Values. We estimate $L(s_i)$ component of the received power as follows. The basic idea is to estimate α as the mean of the perceived α across all observation points.

1. First, we *assume* that the transmitter is located at the observation point s_t that has the maximum signal strength.
2. Then, we compute the "perceived" path-loss exponent α_i at observation point s_i to be: $\dfrac{Z(s_t) - Z(s_i)}{10 \log_{10}(d_i)}$ where $Z(s_t)$ and $Z(s_i)$ are the observed values at the locations s_i and the assumed transmitter location s_t respectively, and d_i is the distance between s_i from s_t.
3. Now, we compute the mean $\bar{\alpha}$ of the perceived α_i's, and use it to compute the $L(s_i)$ values at the observation locations *as well as the new location* s_0 by:

$$L(s_i) = Z(s_t) - 10\bar{\alpha} \log_{10} d_i.$$

Predicting Value at New Location s_0. Once $L(s_i)$ component has been estimated as above, we can compute $\delta(s_i) = Z(s_i) - L(s_i)$ at the observation locations (but not s_0). Finally, we use the OK technique to estimate the $\delta(s_0)$ value at the new location s_0 using the $\delta(s_i)$ values at the observation locations.

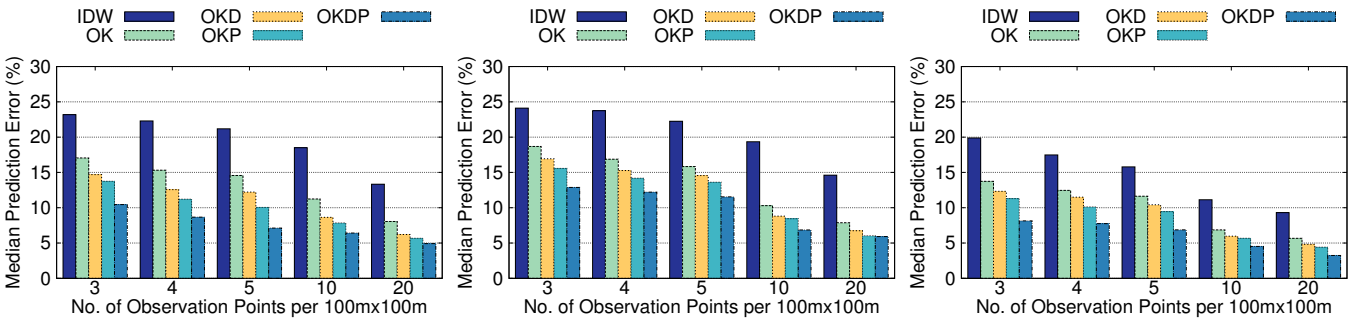
OK with Partitioning: In our context of predicting RF signal values, Ordinary Kriging technique can be further improved by partitioning the region of interest into subregions with different path-loss characteristics and then applying the OK technique independently for each subregion. The intuition behind such an approach is as follows. Recall that one of the conditions required for optimality of OK technique is that the semivariogram at a pair of locations s_i and $s_i + h$ depends only on h. However, in our context of RF signal, such a condition doesn't really hold – partly because of the different path-loss characteristics in different parts of the given region, and our proposed approach of partitioning the region into subregions is to mitigate its impact.

In particular, in the proposed partitioning approach, we first partition the given region into subregions based on the average path-loss characteristics, as described below. Then, we construct semivariogram function curves for each subregion independently (Fig. 4). Finally, to predict the value at a new location, we first determine the subregion it belongs to, then use neighboring observation locations from the same subregion and the corresponding semivariogram function to predict its value.

We note that the above partitioning technique will result in lesser observation points in each subregion, but our experiments suggest that the disadvantage due to lesser observation points is more than offset by more accurate semivariograms due to partitioning.

Partitioning the Region into Subregions. One simple approach to partitioning a region into subregions is just use the terrain information/characteristics; e.g., indoor and outdoor regions can be considered two subregions, as they are likely to have different path-loss characteristics. However, a terrain-based approach is not always feasible due to lack of sufficient information about terrain characteristics. Thus, we use the following approach to partition the given region: First, we compute the Voronoi diagram [27] over the observation points, and assign the Voronoi region of an observation point as its initial subregion. Then, we iteratively merge two *adjacent* subregions into (bigger) subregions, based on the "merging condition" defined below. This merging of adjacent subregions is repeated until no two adjacent subregions satisfy the merging-condition. Eventually, the above process results in partitioning of the original region into contiguous subregions.

Merging Condition. First, as done before to estimate $L(s_i)$ values, we compute the *perceived* path-loss exponent α_i at each observation point s_i (by assuming the transmitter location to be at the point with maximum signal strength). Then, we iteratively merge two adjacent subregions, if the corresponding two sets of perceived path-loss exponents are "similar;" in particular, if X and Y are the 25-75 percentile ranges of the $alpha_i$ values in the two sets, then the two subregions are merged if the *overlap* between the X and Y ranges is more than 50% of the smaller of the X and Y ranges. If there are multiple pairs of subregions that can be merged, then we pick the pair with the largest overlap between their exponent sets. Note that the 25-75 percentile overlap condition minimizes the impact of any outliers.



(a) Cellular Dataset (b) USRP Dataset (c) Synthetic Dataset

Fig. 3: Median prediction errors for various interpolation techniques for the three datasets.

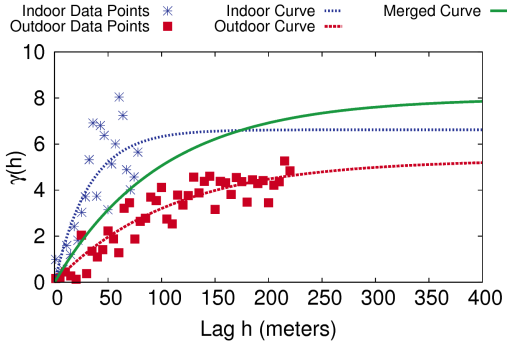


Fig. 4: Fitted curves of the semivariograms after partitioning into subregions.

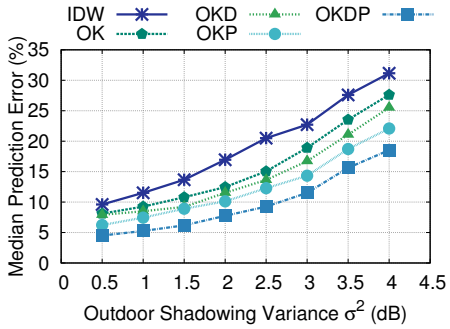


Fig. 5: Median prediction errors for varying shadowing variance (indoor's is $2\sigma^2$) for the synthetic dataset.

C. Performance of Interpolation Techniques

To showcase the improvements due to proposed detrending and partitioning, we present a comparison of proposed OK-based techniques, viz., Ordinary Kriging (OK), OK with detrending (OKD), OK with partitioning (OKP), and OK with detrending and partitioning (OKDP). We use IDW as a baseline. For evaluation, multiple datasets are used described in the following.

Datasets: We use three datasets: two of these are real dataset collected using SpecSense using RTL-SDR with Nexus 5 and Samsung Galaxy S4 phones, while the third dataset is synthetic.

- *Cellular Data.* This data is the total signal strength collected in AT&T's LTE downlink (751MHz) in a 2 MHz bandwidth. Samples were collected in 1500 locations spanning indoors and outdoors covering a ≈15K sq m area.
- *USRP Data.* For this data set, a USRP transmits software-synthesized DTV signal in an otherwise free (DTV channel 26) using a transmit power of -10dBm. The tramsmitter is located indoors, but data is collected both indoors and outdoors at 500 locations in a ≈5K sq m area of campus.
- *Synthetic Data.* Real data does not provide any way to control the shadowing variance. To study the impact on different variances, we also consider a simulated environment with an area of 2 km × 2 km, with two indoor subareas (with roughly 40% area) and a transmitter placed in the center. Using the log-normal path-loss model (Eqn. 2), we compute the signal strength at various indoor and outdoor locations. We use different path-loss exponents and variance (σ) of the shadowing component for indoor and outdoor subregions. In all, 1000 random locations were used.

Performance Comparison: For each experiment and dataset, we randomly chose a certain number of observation points depending on the desired density. Using these observation points, we predict the value at each of the remaining points, compute the prediction error $\frac{|\widehat{Z}(s_0) - Z(s_0)|}{Z(s_0)} \times 100\%$ for each point s_0, where $Z(s_0)$ and $\widehat{Z}(s_0)$ are the true and predicted values respectively. When comparing various approaches, we use the same set of observation (and thus, testing) points. For a given setting, we conduct 20 experiments and plot the median prediction error P_{err} value across all experiments and testing points. The density of observation points is 4 per 100m × 100m in Figs. 5-7. For each of the data sets, our partitioning algorithm yielded two partitions closely matching the indoor and outdoor partitioning of the region. See Fig. 4 for the variograms of the subregions created. Figs. 3 and 5 plot median predication errors for varying densities of observation points and variance of shadowing attenuation, respectively, across all datasets. In both the plots, we observe that, for all datasets, the median prediction errors of the techniques decreases in the following order: IDW, OK, OKD, OKP, OKDP. In particular, the improvement of OKDP over OK is significant — reducing the prediction error by one-third to half. Also, as expected, the prediction error decreases with increase in the density of observation points and increases with the increase in the variance. Finally, Figs. 6 and 7 plot the reduction in % of observation points needed compared to IDW to achieve at most 5% prediction error, for different

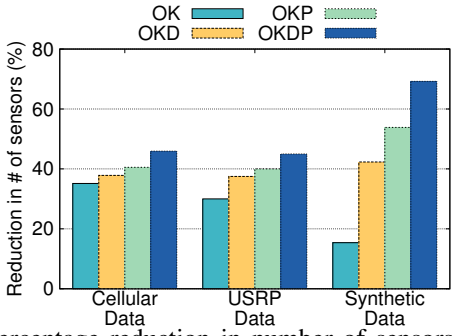


Fig. 6: Percentage reduction in number of sensors compared to IDW.

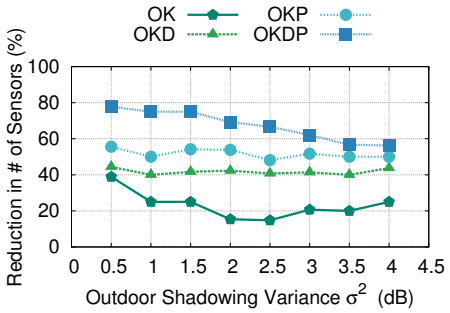


Fig. 7: Percentage reduction in number of sensors compared to IDW, to achieve 5% prediction error for varying shadowing variance (indoor's is $2\sigma^2$), for the synthetic dataset.

datasets and variances respectively. We observe the same order of relative performance of different techniques as above.

IV. Optimized Selection of Sensors

During `SpecSense` operation, there are three main events: (i) availability (arrival) of new spectrum sensors, (ii) spectrum-occupancy query requests by user apps, (iii) actually sensing/measurement of the spectrum signal by *some* of the spectrum sensors (when instructed by the system, in response to the queries [5]). The first two steps are user-driven, while the third step involves an optimization problem. In particular, we address the problem of selecting a minimum number of sensors to report signal at their locations in order to best predict the values at the given query locations. This saves the battery power and backhaul communication costs for the sensors that contribute little to answering spectrum queries. The above optimization problem may be executed at the arrival of each new query, or at regular intervals for the set of queries arriving in the previous interval. In the latter approach, the interval may depend upon certain factors such as the arrival frequency of queries, tolerable prediction delay, etc. Our below problem formulation assumes a set of queries as an input, but can also be used for a single query. We start with the formulation of the above sensor selection problem, and then present our proposed heuristic followed by a performance evaluation.

Selection of Sensors (SOS) Problem: Given a set of sensor locations S, a set of query location Q, and a constraint m, the SOS problem is to select a set of at most m sensors $S_Q \subseteq S$ that minimizes the total prediction error (based on the OK interpolation technique) for the given queries. The total prediction error for the queries Q from a selected set of sensors S_Q is given by

$$\sum_{q \in Q} (Z(q) - \widehat{Z}(q|S_Q)) \tag{3}$$

where $Z(q)$ is the true value at location q and $\widehat{Z}(q|S_Q)$ is the predicted value from observations S_Q using the OK interpolation technique.

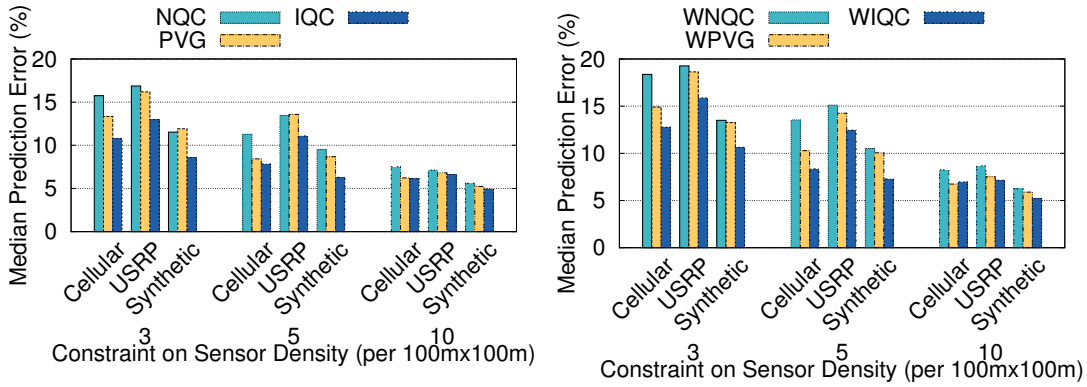
In general, the above "sensor selection" problem has been studied before in similar contexts [17], [21]. Most recently, [35] considers the above problem with the different objective of minimizing the OK prediction variance, which is an *indirect* measure of prediction error; [35] *incorrectly* claims that a straightforward greedy approach for the problem will be approximate.[3] Instead, we directly focus on minimizing the prediction error. In our evaluation (see below), we show that our proposed heuristic outperforms the greedy heuristic that targets minimization of prediction variance.

The above SOS problem can be easily shown to be NP-hard by a reduction from the set cover problem. Thus, we focus on designing efficient heuristics. We note that the straightforward greedy approach that iteratively picks the sensor that minimizes the total prediction error, is infeasible since the prediction error can't be computed due to unknown true values at the queries. Thus, we instead propose a greedy heuristic based on "coverage" of queries by sensors.

Proposed Heuristic: Iterative Query Cover (IQC): Our proposed heuristic, viz., iterative query cover (IQC), is based on the intuition that the prediction error of a particular query is minimized by maximizing the number of neighboring observation points. To define neighbors of a query location, we assume a given/known "correlation range" r, which defines the maximum distance of spatial correlation and can be easily deduced from historical semivariogram curve [24]. Based on the above intuition, the IQC heuristic works in a sequence of rounds, wherein in each round it selects sensors to ensure at least one neighboring sensor for each query. To maximize the number of neighbors for each query, IQC tries to maximize the number of rounds by minimizing the number of sensor selected in each round. More formally, let S_Q be the set of sensors already selected in previous rounds. Then, in the current round, we essentially run a greedy set cover heuristic to cover all (or as many as possible) queries in Q using a minimal number of sensors from $S - S_Q$. The set of sensors selected in this round are added to the maintained solution set S_Q. Then, we go to the next round. The heuristic stops when $|S_Q|$ becomes m (which could happen in the middle of a round). See Algorithm 1.

Nearest-Query Cover (NQC) Heuristic: For comparison purposes, we also explore another greedy heuristic NQC — which is based on the thesis that the prediction error is minimized by selecting sensors closest to the queries. Like

[3]Essentially, their claim that the OK prediction variance function is "submodular" is true only in the *special* case when there are no "suppressor" variables [17], [21].



(a) Unweighted Algorithms (b) Weighted Algorithms

Fig. 8: Median prediction errors of various heuristics for varying constraint (number of sensors selected per 100m $\times$ 100m), for different datasets.

Algorithm 1 Iterative Query Cover (IQC) Heuristic

1: **Input:** Set of sensors S, set of queries Q, correlation range r, and constraint $m \leq |S|$
2: **Output:** Set of selected sensors S_Q, $|S_Q| \leq m$.
3: $S_Q \leftarrow \{\}, Q_r \leftarrow Q$
4: /* Q_r is the set of uncovered queries in the current round */
5: **while** $|S_Q| < m$ **do**
6: **for all** $s \in (S - S_Q)$ **do**
7: $C_s \leftarrow \{q \in Q_r \mid ||q - s|| \leq r\}$
8: **end for**
9: $s \leftarrow \arg\max_{\forall s \in (S - S_Q)} |C_s|$
10: $S_Q \leftarrow S_Q \cup \{s\}$
11: $Q_r \leftarrow Q_r - \{C_s\}$
12: **if** $Q_r = \emptyset$ **then** /* start a new round */
13: $Q_r \leftarrow Q$
14: **end if**
15: **end while**

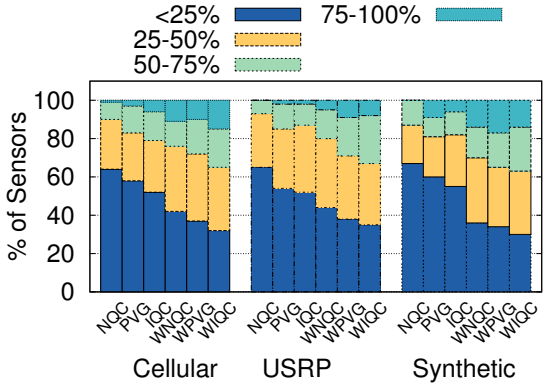


Fig. 9: Residual battery levels for various heuristics and datasets, for a fixed constraint of 4 sensors per 100m $\times$ 100m.

IQC, the NQC heuristic also executes in rounds. As before, let S_Q be the set of sensors already selected in previous rounds. Then, in the current round, for each query q (that has a sensor within r in $(S - S_Q)$), we select the sensor closest to q in $(S - S_Q)$. As before, we update S_Q, proceed to the next round, and continue till $|S_Q|$ becomes m.

Weighted SOS Problem: Portable spectrum sensors have limited energy and it is more desirable to choose those sensors that have more battery power remaining. We incorporate this prioritization by assigning weights to the sensors in the SOS problem, and appropriately generalizing the problem formulation and the heuristics as follows. Using the same notation as before, the weighted-SOS (WSOS) problem is to find a set of m sensors $|S_Q| \subseteq S$ that minimizes two objectives, viz., the total prediction error as well as the total cost $\sum_{s \in S_Q} 1/b(s)$ of the selected sensors where $b(s)$ is the remaining battery level at a sensor s.[4]

Our heuristics IQC and NQC heuristics can be easily generalized to the above weighted SOS problem as follows. For IQC, the only change is that we replace Line 9 of Algorithm 1 by

$$s \leftarrow \arg\max_{\forall s \in (S - S_Q)} |C_s| b(s).$$

Similarly, for NQC, within each round, for each query q, instead of picking the nearest sensor, we pick the sensor s whose $d_{sq}/b(s)$ is the smallest where d_{sq} is the distance of s from q.

Performance Comparison: We now evaluate our proposed heuristics. In addition to the above IQC and NQC heuristics and their weighted versions, we also consider the *Prediction-Variance Greedy (PVG)* heuristic that, in each iteration, selects the sensor that maximizes the reduction in OK prediction variance which can be computed from the OK system of equations [24]. For each experiment, we randomly pick half of the given points as candidate sensor locations, and choose a certain number of query points from the remaining points. We initially assign 100% battery energy to each sensor, and deduct a small amount (0.5%) of battery energy for each sensing measurement. For a given constraint m, we run 10,000 experiments and in each experiment, randomly pick 5 to m

[4]We could have focused on minimizing the linear combination of these two objectives, but that requires associating arbitrary weights with the two objectives. An alternate formulation could be to use cost as the constraint, but that precludes a fair comparison of the weighted and unweighted versions of the problem.

(randomly picked number) number of queries. Fig. 8 plots the median prediction errors for various heuristics. We make two observations: (i) First, IQC outperforms PVG as well as NQC in weighted as well as unweighted versions. The relative performance of IQC and NQC confirms the intuition that a query is better interpolated by more neighboring points in its range rather than a smaller number of closer neighbors. (ii) Second, the weighted versions perform only slightly worse than their unweighted versions. We also compare the distribution of remaining battery levels in the sensors after all the experiments, for each constraint m. See Fig. 9. We observe that, as expected, the weighted versions have fewer sensors with low battery levels than their unweighted versions, which confirms the effectiveness of weighted heuristics. As before, IQC outperforms NQC as well as PVG in all settings.

V. System Level Evaluation

We now describe the overall operation of the `SpecSense` system, and present key performance results of the end-to-end system.

A. The SpecSense System

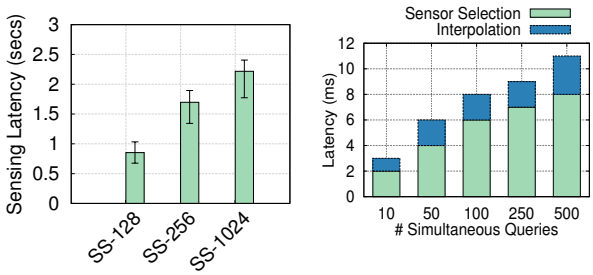
As shown in Fig. 1, the `SpecSense` system primarily consists of four high level components: (a) sensing and/or query clients, (b) central controller, (c) spectrum database, and (d) analytics framework on top of the database. At this time, the sensing clients consist of (i) a USB-powered RF sensor (e.g., RTL-SDR [7], BladeRF [1]) with (ii) an ARM-based processor board (Raspberry Pi) or smartphone acting as the host. `SpecSense` is not tied to these sensors; they are used for their low cost, low power and off-the-shelf availability.

In response to the spectrum occupancy queries, the central controller runs the sensor selection algorithm to allocate spectrum sensing tasks to individual clients that are registered with the controller. The clients respond back with the power spectral density for the requested channel. The controller passes on such data with time and location stamps to the interpolation algorithm which generates answers to the spectrum occupancy queries, communicates them to the query clients and also stores them in the spectrum database. A web-based dashboard is also implemented that visually shows channel availability, query results, and the location/type of sensors on a map.

B. Latency Measurements

We benchmark `SpecSense` to measure different components of latency for spectrum occupancy queries. All measurements are done with a set of WiFi connected Samsung Galaxy S4 phones and RTL-SDR dongles as the clients sensing the UHF TVWS bands and the controller, database and analytics platform running on a server-class computer in the lab. There are two significant components of the latency – sensing latency and query latency.

Sensing Latency: We define sensing latency as the total time elapsed between the controller issuing a scan instruction for a sensor and the time when the results are available back at the controller. It involves the communication delay between the controller and the sensor, delay to invoke the driver, gather the IQ samples from the device, and computational latency to compute the FFT over the samples. We benchmark the sensing latency in `SpecSense` for different FFT sizes. See Fig. 10a. The latency is $\approx$ 800ms–2.3 secs, of which communication latency is in the order of 100s of ms[5] and the time required to fetch the IQ samples from the device are few ms to 10s of ms. The FFT computation time depends on the FFT size and is the dominant part of the latency. However, this can be potentially be outsourced to an FPGA or ASIC on the phone in future designs to bring it down to negligible levels.



(a) Sensing Latency (b) Query Latency

Fig. 10: Latency benchmarks for the `SpecSense` system. SS-128, SS-256 and SS-1024 respectively uses 128, 256 and 1024 point FFT.

Query Latency: In a dynamic spectrum access scenario, spectrum occupancy queries need to be served in semi real-time. This also implies that both sensor selection and interpolation algorithms need to be implemented and executed efficiently.

The current deployment spans across our university campus ($\approx$ 6 sq. km) with $\approx$10 mobile spectrum sensor nodes. Due to such limited hardware we moved the sensors across the area to mimic a huge number of *virtual* sensors at different locations. This assumes that the spectrum map does not change over time which is indeed the case for TV channels we are using as test cases. In all, we mimic $\approx$500 *virtual* sensors in the entire area.

We run spectrum occupancy queries at random locations within the area. The controller is queried with the coordinates of a location with no sensor to estimate the power at that location for a given channel. The involves two steps, viz., sensor selection (Section IV) and interpolation (Section III). Fig. 10b shows that a 50$\times$ increase in the number of simultaneous queries causes the latency to increase by only 3$\times$, which clearly indicates the scalability of `SpecSense`. Second, for a large number of queries, the total computation time is only within 10s of milliseconds where as the network latency involving the query client and the controller can be 100s of milliseconds.

Overall, this measurements establish that FFT computation on the sensor is still the dominant part of the overall end-to-end latency. Optimizing this to run directly on hardware has

[5]The amount of data to be communicated is modest and depends on FFT size. This is roughly in the order of a few 10KBs. Compare this with a sample Google search that consumes about 40 KB.

the potential of achieving sub-seconds latency for spectrum occupancy queries in SpecSense.

VI. Related Work

Large-scale spectrum Monitoring: While spectrum monitoring itself is not a new idea and substantial literature exists [4], [20], [25], [36], they all use lab-grade expensive sensors. Recent work has also demonstrated spectrum sensing in mobile, low-power commodity platforms [14], [26], [28]. Mobile-based crowdsourcing applications for spectrum monitoring have been presented in [9], [31], [32]. While all these ideas subscribe to the general philosophy of large-scale spectrum monitoring, they all stop short of building any significant support infrastructure for enabling apps to use spectrum data.
Radio Environment Mapping: A large number of recent works [9], [10], [23], [29], [34], [35] have addressed mapping techniques from sparse sensor measurements (often using Ordinary Kriging) using extensive wardriving or crowdsourcing. It is unclear however a realistic spectrum sensing and query infrastructure can address the algorithmic/computation issues used in such techniques, particularly when spectrum queries need to be served quickly or near realtime. We improve upon existing interpolation techniques and address scalability issues.

VII. Conclusions

In this paper, we described SpecSense — an enabling platform that supports crowdsourced spectrum sensing and spectrum-aware apps. We specifically concentrated on two related problems related to efficient answering of spectrum occupancy queries – effective spatial interpolation of RF signals and optimized selection of sensors. Our overall results indicate that in practical deployments, on-demand answering of spectrum occupancy queries can be effectively done with only a modest latency overhead.

Acknowledgement

This work is partially supported by NSF grants AST-1443951 and CNS-1642965.

References

[1] Blade-RF. http://nuand.com/.
[2] FlightFeeder for Android, FlightAware. http://flightaware.com/adsb/android/.
[3] LTE-U Wi-Fi Coexistence in unlicensed spectrum. http://bit.ly/1LQblKu.
[4] Microsoft Spectrum Observatory project. https://observatory.microsoftspectrum.com/.
[5] Protocol to Access White-Space (PAWS) Databases: Use Cases and Requirements. https://datatracker.ietf.org/doc/rfc6953/.
[6] Realizing the Full Potential of Government-Held Spectrum to Spur Economic Growth. http://1.usa.gov/22i7oJE.
[7] RTL-SDR Dongle. http://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/.
[8] ThinkRF Spectrum Analyzer. thinkrf.com/.
[9] A. Achtzehn, J. Riihihjärvi, I. A. Barriía Castillo, M. Petrova, and P. Mähönen. CrowdREM: Harnessing the power of the mobile crowd for flexible wireless network monitoring. In *Proc. ACM HotMobile*, 2015.
[10] A. Achtzehn, J. Riihijarvi, and P. Mahonen. Improving accuracy for TVWS geolocation databases: Results from measurement-driven estimation approaches. In *Proc. IEEE DySPAN*, 2014.
[11] A. Achtzehn, J. Riihijarvi, G. M. Vargas, M. Petrova, and P. Mahonen. Improving coverage prediction for primary multi-transmitter networks operating in the TV whitespaces. In *Proc. IEEE SECON Conference*, 2012.
[12] M. Buddhikot. Towards a virtual cellular network with variable grade spectrum: challenges and opportunities. In *Proc. ACM MobiCom*. ACM, 2013.
[13] M. Buddhikot, C. Kim, and J. Ryoo. Design and implementation of an end-to-end architecture for 3.5 GHz shared spectrum. In *Proc. IEEE DySPAN*, 2015.
[14] A. Chakraborty and S. R. Das. Radio environment mapping with mobile devices in the TV white space. In *Proc. ACM MobiCom*, 2013.
[15] A. Chakraborty and S. R. Das. Measurement-augmented spectrum databases for white space spectrum. In *Proc. ACM CoNEXT*, pages 67–74. ACM, 2014.
[16] N. A. C. Cressie. *Statistics for spatial data*. J. Wiley & Sons, 1993.
[17] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *Proc. ACM STOC*, 2008.
[18] A. Dutta and M. Chiang. "See Something, Say Something" Crowdsourced Enforcement of Spectrum Policies. *IEEE Transactions on Wireless Communications*, 2016.
[19] P. Ishwar, A. Kumar, and K. Ramachandran. On distributed sampling in dense sensor networks: a bit conservation principle. *Proc. IPSN*, 2003.
[20] A. Iyer, K. K. Chintalapudi, V. Navda, R. Ramjee, V. Padmanabhan, and C. Murthy. SpecNet: Spectrum sensing sans frontieres. In *Proc. NSDI*, 2011.
[21] A. Krause, B. Mcmahan, C. Guestrin, and A. Gupta. Selecting observations against adversarial objectives. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Proc. NIPS*. 2008.
[22] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.
[23] M. Molinari, M. Fida, M. K. Marina, and A. Pescapè. Spatial interpolation based cellular coverage prediction with crowdsourced measurements. In *Proc. SIGCOMM Workshop on C2B(I)D*, 2015.
[24] J.-M. Montero and G. Fernandez-Aviles. *Spatial and Spatio-Temporal Geostatistical Modeling and Kriging*. John Wiley & Sons, 2015.
[25] J. Naganawa, H. Kim, S. Saruwatari, H. Onaga, and H. Morikawa. Distributed spectrum sensing utilizing heterogeneous wireless devices and measurement equipment. In *Proc. IEEE DySPAN*, 2011.
[26] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng. Towards commoditized real-time spectrum monitoring. In *Proc. ACM HotWireless*, 2014.
[27] A. Okabe. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons, 2000.
[28] D. Pfammatter, D. Giustiniano, and V. Lenders. A software-defined sensor architecture for large-scale wideband spectrum monitoring. In *Proc. IEEE IPSN*, 2015.
[29] C. Phillips, M. Ton, D. Sicker, and D. Grunwald. Practical radio environment mapping with geostatistics. In *Proc. IEEE DySPAN*, 2012.
[30] M. Sherman. *Spatial Statistics and Spatio-temporal Data: Covariance Functions and Directional Properties*. John Wiley & Sons, 2011.
[31] J. Shi, Z. Guan, C. Qiao, T. Melodia, D. Koutsonikolas, and G. Challen. Crowdsourcing access network spectrum allocation using smartphones. In *Proc. ACM HotNets*, 2014.
[32] D.-H. Shin, S. He, and J. Zhang. Joint sensing task and subband allocation for large-scale spectrum profiling. In *Proc. IEEE INFOCOM*, 2015.
[33] S. Yin, D. Chen, Q. Zhang, M. Liu, and S. Li. Mining spectrum usage data: a large-scale spectrum measurement study. *IEEE Transactions on Mobile Computing*, 11(6):1033–1046, 2012.
[34] X. Ying, C. W. Kim, and S. Roy. Revisiting TV coverage estimation with measurement-based statistical interpolation. In *Proc. IEEE COMSNETS*, 2015.
[35] X. Ying, S. Roy, and R. Poovendran. Incentivizing crowdsourcing for radio environment mapping with statistical interpolation. In *Proc. IEEE DySPAN 2015*, 2015.
[36] T. Zhang and S. Banerjee. A Vehicle-based Measurement Framework for Enhancing Whitespace Spectrum Databases. In *Proc. ACM MobiCom*, 2014.
[37] M. Zheleva, R. Chandra, A. Chowdhery, A. Kapoor, and P. Garnett. Txminer: Identifying transmitters in real-world spectrum measurements. In *Proc. IEEE DySPAN*, 2015.