Online Distributed Sensor Selection

Daniel Golovin
Caltech

Matthew Faulkner
Caltech

Andreas Krause
Caltech

ABSTRACT

A key problem in sensor networks is to decide which sensors to query when, in order to obtain the most useful information (e.g., for performing accurate prediction), subject to constraints (e.g., on power and bandwidth). In many applications the utility function is not known a priori, must be learned from data and can even change over time. Furthermore, for large sensor networks solving a centralized optimization problem to select sensors is not feasible, and thus we seek a fully distributed solution. In this paper, we present *Distributed Online Greedy* (DOG), an efficient, distributed algorithm for repeatedly selecting sensors online, only receiving feedback about the utility of the selected sensors. We prove very strong theoretical no-regret guarantees that apply whenever the (unknown) utility function satisfies a natural diminishing returns property called *submodularity*. Our algorithm has extremely low communication requirements, and scales well to large sensor deployments. We extend DOG to allow observation-dependent sensor selection. We empirically demonstrate the effectiveness of our algorithm on several real-world sensing tasks.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; G.3 [**Probability and Statistics**]: Experimental Design; I.2.6 [**AI**]: Learning

General Terms

Algorithms, Measurement

Keywords

Sensor networks, approximation algorithms, distributed multiarmed bandit algorithms

IPSN'10, ???

1. INTRODUCTION

A key challenge in deploying sensor networks for real-world applications such as environmental monitoring [17], building automation [22] and others is to decide when to activate the sensors in order to obtain the most useful information from the network (e.g., accurate predictions at unobserved locations) and to minimize power consumption. This sensor selection problem has received considerable attention [1, 29, 9], and algorithms with performance guarantees have been developed [1, 15]. However, many of the existing approaches make simplifying assumptions. First of call, many approaches assume (1) that the sensors can perfectly observe a particular sensing region, and nothing outside the region [1]. This assumption does not allow us to model settings where multiple noisy sensors can help each other obtain better predictions. There are also approaches that base their notion of utility on more detailed models, such as improvement in prediction accuracy w.r.t. some statistical model [9] or detection performance [16]. However, most of these approaches make two crucial assumptions: (2) The model, upon which the optimization is based is known in advance (e.g., based on domain knowledge or data from a pilot deployment) and (3), the solution of the sensor selection optimization is obtained in a centralized fashion (i.e., some centralized processor selects the sensors which obtain highest utility w.r.t. to the model). We are not aware of any approach that simultaneously addresses the three main challenges (1), (2) and (3) above and still provides theoretical guarantees.

In this paper, we develop an efficient algorithm, called *Distributed Online Greedy* (DOG), which addresses these three central challenges. Prior work has shown that many sensing tasks satisfy an intuitive diminishing returns property, submodularity, which states that activating a new sensor helps more if few sensors have been selected so far, and less if many sensors have already been selected. Our algorithm applies to any setting where the true objective is submodular [20], thus capturing a variety of realistic sensor models. Secondly, our algorithm does not require the model to be specified in advance: It learns to optimize the objective function in an online manner. Lastly, the algorithm is distributed; the sensors decide whether to activate themselves

based on local information. We analyze our algorithm in the no-regret model, proving convergence properties similar to the best bounds for any centralized solution.

A bandit approach toward sensor selection. At the heart of our approach is a novel distributed algorithm for multi-armed bandit problems: In the classical multi-armed bandit [21] setting, we picture a slot machine with multiple arms, where each arm generates a random payoff with unknown mean. Our goal is to devise a strategy for pulling arms to maximize the total reward accrued. The difference between the optimal payoff and the obtained payoff is called the regret. There are algorithms for which the average per-round regret is $\mathcal{O}(\sqrt{n \log n}/\sqrt{T})$ where n is the number of arms, and T the number of rounds. Suppose we would like to, at every time, select k sensors. The sensor selection problem can then be cast as a multi-armed bandit problem, where there is one arm for each possible set of k sensors, and the payoff is the accrued utility for the selected set. However, since there is a large number of possible sets of sensor activations, the number n of arms is exponentially large. Thus existing regret bounds are $\mathcal{O}(n^{k/2}\sqrt{\log n}/\sqrt{T})$, i.e., exponential in k. However, since the utility function is submodular, the payoffs of these arms are correlated. Recent results [25] show that this correlation due to submodularity can be exploited by reducing the n^k-armed bandit problem to k n-armed bandit problems, with only a bounded loss in performance. However, the existing no-regret algorithms for bandit optimization are centralized in nature: A centralized processor maintains a set of "weights" associated with each arm, which depend on the performance of the arms in past rounds. Arms are then pulled at random by interpreting the weights as probabilities. The key question in distributed online submodular sensing is thus how to perform this sampling in a distributed fashion. In this paper, we develop a scheme where each sensor maintains their own weights, and activates itself *independently from all other sensors* purely depending on this weight.

Observation specific selection. Another key question not addressed in centralized sensor selection is the fact that sensors have more information about whether they should become activated. In many applications, obtaining sensor measurements is much less costly than transmitting the measurements across the network. For example, when using cell phones in participatory sensing [5], it is inexpensive to obtain measurements (such as temparature, GPS readings, etc.) on a regular basis, but expensive to constantly communicate the measurements over the network. In such a setting, one would like to activate sensors depending on their observations. We extend our algorithm to allow for observation specific activation, and analyze the tradeoff between power consumption and obtained utility.

Our main contributions.

- Distributed EXP3, a novel distributed implementation of the classic multiarmed bandit algorithm.
- Distributed Online Greedy (DOG) and LAZYDOG, novel algorithms for distributed online sensor selection, which applies in many settings, only requiring the utility function to be submodular.
- OD-DOG, an extension of DOG to allow for observation-dependent selection.
- We analyze our algorithm in the no-regret model and prove that it attains the optimal regret bounds attainable by any efficient centralized algorithm.
- We evaluate our approach on several real-world sensing tasks.

2. THE SENSOR SELECTION PROBLEM

We now formalize the sensor selection problem. Suppose a network of sensors has been deployed at a set of locations V with the task of monitoring some phenomenon (e.g., measure temperature in a building). Typically, constraints such as limited bandwidth for communication, or limited battery power require us to select a subset $\mathcal{A}$ of these sensors for activation, according to some utility function. The activated sensors then send their data to a server (base station). We will first review the traditional, offline setting where the utility function is specified in advance. We will then address the more challenging setting where the utility function needs to be learnt from data in an online manner.

2.1 The Offline Sensor Selection Problem

In the offline sensor selection problem, one specifies a sensing quality objective function $f(\mathcal{A})$, and chooses a set to maximize this function subject to some constraints, e.g., on the number of activated sensors. One possible choice for the sensing quality is based on prediction accuracy (we will discuss other possible choices later on). In many applications, measurements are correlated across space, which allows us to make predictions at the unobserved locations. For example, prior work [9] has considered the setting where a random variable $\mathcal{X}_s$ is associated with every location $s \in V$, and a joint probability distribution $P(\mathcal{X}_V)$ models the correlation between sensor values. Here, $\mathcal{X}_V = [\mathcal{X}_1, \ldots, \mathcal{X}_n]$ is the random vector over all measurements. If some measurements $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$ are obtained at a subset of locations, then the conditional distribution $P(\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A}) = \mathbf{x}_\mathcal{A}$ allows to make predictions at the unobserved locations, e.g., by predicting $\mathbb{E}[\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}]$. Furthermore, this conditional distribution quantifies the *uncertainty* in the prediction: Intuitively, we would like to select sensors that minimize the predictive uncertainty. One way to quantify the predictive uncertainty is the mean squared prediction error,

$$\mathrm{MSE}(\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathbf{x}_\mathcal{A}) = \frac{1}{n} \sum_{s \in V\setminus\mathcal{A}} \mathbb{E}[(\mathcal{X}_s - \mathbb{E}[\mathcal{X}_s \mid \mathbf{x}_\mathcal{A}])^2 \mid \mathbf{x}_\mathcal{A}].$$

In general, the measurements $\mathbf{x}_\mathcal{A}$ that sensors $\mathcal{A}$ will make is not known in advance. Thus, we can base our optimization on the *expected mean squared prediction error*,

$$\mathrm{EMSE}(\mathcal{A}) = \int dp(\mathbf{x}_\mathcal{A})\, \mathrm{MSE}(\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathbf{x}_\mathcal{A}).$$

Equivalently, we can minimize the *reduction* in mean squared prediction error,

$$f_{\mathrm{EMSE}}(\mathcal{A}) = \mathrm{EMSE}(\emptyset) - \mathrm{EMSE}(\mathcal{A}).$$

By definition, $f_{\mathrm{EMSE}}(\emptyset) = 0$, i.e., no sensors obtain no utility. Furthermore, f_{EMSE} is monotonic: if $\mathcal{A} \subseteq \mathcal{B} \subseteq V$, then $f_{\mathrm{EMSE}}(\mathcal{A}) \leq f_{\mathrm{EMSE}}(\mathcal{B})$, i.e., adding more sensors always helps. That means, f_{EMSE} is maximized by the set of all sensors V. However, in practice, we would like to only select a small set of, e.g., at most k sensors due to bandwidth and power constraints:

$$\mathcal{A}^* = \arg\max_{\mathcal{A}} f_{\mathrm{EMSE}}(\mathcal{A}) \text{ s.t. } |\mathcal{A}| \leq k.$$

Unfortunately, this optimization problem is NP-hard, so we cannot expect to efficiently find the optimal solution. Fortunately, it can be shown [8] that in many settings[1], the function f_{EMSE} satisfies an intuitive diminishing returns property called submodularity. A set function $f : 2^V \to \mathbb{R}$ is called *submodular* [20] if, for all $\mathcal{A} \subseteq \mathcal{B} \subseteq V$ and $s \in V \setminus \mathcal{B}$ it holds that $f(\mathcal{A} \cup \{s\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{s\}) - f(\mathcal{B})$. Many other natural objective functions for sensor selection satisfy submodularity as well [18]. For example, the *sensing region* model where $f_{REG}(\mathcal{A})$ is the total area covered by all sensors $\mathcal{A}$ is submodular. The *detection* model where $f_{DET}(\mathcal{A})$ counts the expected number of targets detected by sensors $\mathcal{A}$ is submodular as well.

A fundamental result by Nemhauser et al.[20] states that for monotonic submodular functions, a simple greedy algorithm, which starts with the empty set $\mathcal{A}_0 = \emptyset$ and iteratively adds the element

$$s_k = \arg\max_{s \in V\setminus\mathcal{A}_{k-1}} f(\mathcal{A}_{k-1} \cup \{s\});\ \ \mathcal{A}_k = \mathcal{A}_{k-1} \cup \{s_k\}$$

obtains a near-optimal solution: For the set $\mathcal{A}_k$ it holds that $f(\mathcal{A}_k) \geq (1-1/e) \max_{|\mathcal{A}|\leq k} f(\mathcal{A})$, i.e., the greedy solution obtains at least a constant fraction of $(1 - 1/e) \approx 63\%$ of the optimal value.

One fundamental problem with this offline approach is that it requires the function f to be specified in advance, i.e., before running the greedy algorithm. For the function f_{EMSE}, this means that the probabilistic model $P(\mathcal{X}_V)$ needs to be known in advance. While for some applications some prior data, e.g., from pilot deployments may be accessible, very often no such prior data is available. This leads to a "chicken-and-egg" problem, where sensors need to be activated to collect data in order to learn a model, but also the model is required to inform the sensor selection. This is akin to the "exploration–exploitation tradeoff" in reinforcement learning [2], where an agent needs to decide whether to explore and gather information about effectiveness of an action, or to exploit, i.e., choose actions known to be effective. In the following, we devise an online monitoring scheme based on this analogy.

[1]For Gaussian models and conditional suppressorfreeness [8]

2.2 The Online Sensor Selection Problem

We now consider the more challenging problem where the objective function is not specified in advance, and needs to be learnt during the monitoring task. We assume that we intend to monitor the environment for a number T of time steps (rounds). In each round t, a set S_t of sensors is selected, and these sensors transmit their measurements to a server (base station). The server then determines a sensing quality of $f_t(S_t)$ quantifying the utility obtained from the resulting analysis. For example, if our goal is spatial prediction, the server would build a model based on the previously collected sensor data, pick a random sensor s, make prediction for the variable $\mathcal{X}_s$, and then compare the prediction μ_s with the sensor reading x_s. The error $f_t = \sigma_s^2 - (\mu_s - x_s)^2$ is an unbiased estimate of the reduction in EMSE. In the following analysis, we will only assume that the objective functions f_t [2] are bounded (w.l.o.g., take values in $[0, 1]$), are monotone and submodular. Our goal is to maximize the total reward obtained by the system over T rounds, $\sum_{t=1}^{T} f_t(S_t)$.

Our goal is to develop a protocol for selecting the sets S_t of sensors at each round, such that, after a small number of rounds, the average performance of our online algorithm converges to the same performance of the offline strategy (which knows the objective functions). We thus compare our protocol against all strategies that can select a fixed set of k sensors for use in all of the rounds; the best such strategy obtains reward $\max_{S\subseteq V:|S|\leq k} \sum_{t=1}^{T} f_t(S)$. The difference between this quantity and what our protocol obtains is known as its *regret*, and an algorithm is said to be *no-regret* if the limit of its average regret over all rounds tends to zero. When $k = 1$, our problem is simply the well-studied *multi-armed bandit* (MAB) problem, for which many no-regret algorithms are known (see e.g., the survey of [12]). For general k, because the average of a set of submodular functions remains submodular, we can apply the result of Nemhauser *et al.* [20] (*c.f.*, Sec. 2.1) to prove that a simple greedy algorithm obtains a $(1-1/e)$ approximation to the optimal offline solution. Moreover, Feige [11] showed that this is optimal in the sense that obtaining a $(1 - 1/e + \epsilon)$ approximation for any $\epsilon > 0$ is NP-hard. These facts suggest that we cannot expect any efficient online algorithm to converge to a solution better than $(1 - 1/e) \max_{S\subseteq V:|S|\leq k} \sum_{t=1}^{T} f_t(S)$. We therefore define the $(1 - 1/e)$-regret of a sequence of (possibly

[2]We assume f_t is given as a *value oracle*, i.e., given a query $S \subseteq V$ we have a black box computing $f_t(S)$.

random) sets $\{S_t\}_{t=1}^T$ as

$$R_T := (1-1/e) \cdot \max_{S \subseteq V : |S| \leq k} \sum_{t=1}^{T} f_t(S) \; - \; \sum_{t=1}^{T} \mathbb{E}\left[f_t(S_t)\right]$$

where the expectation is taken over the distribution for each S_t. We say an online algorithm producing a sequence of sets has *no-*$(1-1/e)$*-regret* if $\lim_{T\to\infty} \frac{R_T}{T} = 0$.

3. OVERVIEW OF OUR APPROACH

In the following Sections, we will develop our Distributed Online Greedy (DOG) algorithm for distributed online selection of sensors. In Sec. 4, we will first discuss how the greedy algorithm for a known objective function can be extended to an online algorithm with no-$(1-1/e)$-regret. The key step is to replace each greedy selection step with a multi-armed bandit algorithm. In Sec. 5, we will then show how the online algorithm from Sec. 4 can be implemented efficiently in a distributed manner. A primary consideration in devising the protocol is the cost of sending messages between sensors and the base station. We consider two cost models.

- In the *broadcast* model, each sensor can broadcast a message to all other sensors at unit cost.
- In the *point-to-point* model, messages can only be between two sensors or a sensor and the base station, and each message has unit cost.

In Sec. 5 we first consider the broadcast model, and then show how it can be extended to the point-to-point model in Sec. 6. In Sec. 7, we will then extend our algorithm to allow for observation-specific selection.

4. CENTRALIZED ALGORITHM FOR ONLINE SENSOR SELECTION

We will first develop a centralized algorithm for online sensor selection which is guaranteed to achieve no $(1-1/e)$-regret. Our strategy will be to convert the greedy algorithm for centralized sensor selection for known submodular functions as introduced in Sec. 2.1 into an online algorithm. As we will show, the key problem will be to develop an algorithm for selecting a *single* sensor, which we consider in Sec. 4.1. In Sec. 4.2 we will then present our algorithm for selecting multiple sensors, which relies on the single sensor case as a subroutine.

4.1 Centralized Online Single Sensor Selection

Let us first consider the case where $k = 1$, i.e., we would like to select one sensor at each round. This simpler problem can be interpreted as an instance of the multi-armed bandit problem (as introduced in Sec. 2.2), where we have one arm for each possible sensor. If we allow centralized solutions, we can use no-regret algorithms such as the EXP3 algorithm [2] to obtain a no-regret sensor selection. EXP3 works as follows: It is parameterized by a *learning rate* η, and an *exploration probability* γ. It maintains a set of weights w_s, one for each arm (sensor) s, initialized to 1. At every round t, it will select an arm with probability

$$p_s = (1-\gamma)\frac{w_s}{\sum_{s'} w_{s'}} + \frac{\gamma}{n},$$

i.e., with probability γ it explores, picking an arm uniformly at random, and with probability $(1-\gamma)$ it exploits, picking an arm s with probability proportional to its weight w_s. Once an arm s has been selected, a feedback $r = f_t(\{s\})$ is obtained, and the weight w_s is updated to

$$w_s \leftarrow w_s \exp(\eta r / p_s).$$

Auer et al. [2] showed that with appropriately chosen learning rate η and exploration probability γ it holds that the cumulative regret R_T of EXP3 is $\mathcal{O}(\sqrt{nT})$, i.e., the average regret R_T/T converges to zero.

4.2 Centralized Selection of Multiple Sensors

In principle, we could interpret the sensor selection problem as a $\binom{n}{k}$-armed bandit problem, and apply existing no-regret algorithms such as EXP3. Unfortunately, this approach does not scale, since the number of arms grows exponentially with k. However, in contrast to the traditional multi-armed bandit problem, where the arms are assumed to have independent payoffs, in the sensor selection case, the utility function is submodular and thus the payoffs are correlated across different sets. Recently, Streeter and Golovin showed how this submodularity can be exploited, and developed a no-$(1-1/e)$-regret algorithm for online maximization of submodular functions [25]. The key idea behind their algorithm, OG_{unit}, is to turn the offline greedy algorithm into an online algorithm by replacing the greedy selection of the element s_k that maximizes the benefit $s_k = \arg\max_s f(\{s_1, ..., s_{k-1}\} \cup \{s\})$ by a bandit algorithm. As shown in the pseudocode below, OG_{UNIT} maintains k bandit algorithm, one for each sensor to be selected. At each round t, k sensors are selected according to the choices of the k bandit algorithms $\mathcal{E}_i$. Once the elements have been selected, the i-th bandit algorithm $\mathcal{E}_i$ receives as feedback the incremental benefit $f_t(s_1, \ldots, s_i) - f_t(s_1, \ldots, s_{i-1})$, i.e., how much additional utility is obtained by adding sensor s_i to the set of already selected sensors.

Algorithm OG_{UNIT} from [25]:
Initialize k multiarmed bandit algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$, each with action set V.
For each round $t \in [T]$
  For each stage $i \in [k]$ in parallel
    $\mathcal{E}_i$ selects an action v_i^t
  For each $i \in [k]$ in parallel
    feedback $f_t(\{v_j^t : j \leq i\}) - f_t(\{v_j^t : j < i\})$ to $\mathcal{E}_i$.
  Output $S_t = \{a_1^t, a_2^t, \ldots, a_k^t\}$.

In [24] it is shown that OG_{UNIT} has a $\left(1 - \frac{1}{e}\right)$-regret bound of $\mathcal{O}(kR)$ in this feedback model assuming each $\mathcal{E}_i$ has expected regret at most R.

Unfortunately, EXP3 (and in fact all multiarmed bandit algorithms with no-regret guarantees for non-stochastic reward functions) require sampling from some distribution with weights associated with the sensors. If n is small, we could simply store these weights on the server, and run the bandit algorithms $\mathcal{E}_i$ there. However, this solution does not scale to large numbers of sensors. Thus the key problem for online sensor selection is to develop a multi-armed bandit algorithm which implements *distributed sampling* across the network, with minimal overhead of communication. In addition, the algorithm needs to be able to maintain the distributions (the weights) associated with each $\mathcal{E}_i$ in a distributed fashion. In the following, we will develop DOG, an efficient distributed algorithm that address these key challenges.

5. DISTRIBUTED ALGORITHM FOR ONLINE SENSOR SELECTION

We will now develop DOG, an efficient algorithm for distributed online sensor selection. For now we make the following assumptions:

1. Each sensor $v \in V$ is able to compute its contribution to the utility $f_t(S \cup \{v\}) - f_t(S)$, where S are a subset of sensors that have already been selected.

2. Each sensor can broadcast to all other sensors.

3. The sensors have calibrated clocks and unique, linearly ordered identifiers.

These assumptions are reasonable in many applications: (1) In target detection, for example, the objective function $f_t(S)$ counts the number of targets detected by the sensors S. Once previously selected sensors have broadcasted which targets they detected, the new sensor s can determine how many additional targets have been detected. Similarly, in statistical estimation, one sensor (or a small number of sensors) randomly activates each round and broadcasts its value. After sensors S have been selected and announced their measurements, the new sensor s can then compute the improvement in prediction accuracy over the previously collected data. (2) The assumption that broadcasts are possible may be realistic for dense deployments and fairly long range transmissions. In Sec. 6 we will show how assumptions (1) and (2) can be relaxed.

As we have seen in Sec. 4, the key insight in developing a centralized algorithm for online selection is to replace the greedy selection of the sensor which maximally improves the total utility over the set of previously selected sensors by a bandit algorithm. Thus, a natural approach for developing a distributed algorithm for sensor selection is to first consider the single sensor case.

5.1 Distributed Selection of a Single Sensor

The key challenge in developing a distributed version of EXP3 when sensors can broadcast to all other sensors is to find a way to sample exactly one element from a probability distribution p over sensors in a distributed manner. We measure the cost of the sampling procedure in terms of the number of broadcast messages.

A naive distributed sampling scheme. A naive distributed algorithm would be to let each sensor keep track of all activation probabilities p. Then, one sensor (e.g., with the lowest identifier) would broadcast a single random number u uniformly distributed in $[0, 1]$, and the sensor v for which $\sum_{i=1}^{v-1} p_i \leq u < \sum_{i=1}^{v} p_i$ would activate. However, for large sensor network deployments, this algorithm would require each sensor to store a large amount of global information (all activation probabilities p). Instead, each sensor v could store only their own probability mass p_v; the sensors would then, in order of their identifiers, broadcast their probabilities p_v, and stop once the sum of the probabilities exceeds u. This approach only requires a constant amount of local information, but requires an impractical $\Theta(n)$ messages to be sent, and sent sequentially over $\Theta(n)$ time steps.

Distributed multinomial sampling. In this section we present a protocol that requires only $\mathcal{O}(1)$ messages in expectation, and only a constant amount of local information.

For a sampling procedure with input distribution p, we let $\hat{p}$ denote the resulting distribution, where in all cases at most one sensor is selected, and nothing is selected with probability $1 - \sum_v \hat{p}_v$. A simple approach towards distributed sampling would be to activate each sensor $v \in V$ *independently* from each other with probability p_v. While in expectation, exactly one sensor is selected, with probability $\prod_v (1 - p_v) > 0$ no sensor is selected; also since sensors are selected independently, there is a nonzero probability that more than one sensor is selected. Using a synchronized clock, the sensors could determine if no sensor is selected. In this case, they could simply repeat the selection procedure until at least one sensor is selected. However, in the event that more than one sensor is selected, we have to pick exactly one from the remaining sensors. One naive approach would be to pick the sensor uniformly at random. This uniform sampling among the active sensors can be implemented using few messages. The following protocol implements this two-stage selection procedure:

The Simple Protocol:
For each sensor v in parallel
  Sample $X_v \sim \text{Bernoulli}(p_v)$.
  If $(X_v = 1)$, X_v activates.
All active sensors S coordinate to select a single sensor uniformly at random from S.

It is not hard to show that for all sensors v,

$$\hat{p}_v = p_v \cdot \mathbb{E}\left[\frac{1}{|S|} \;\middle|\; v \in S\right] \geq p_v / \mathbb{E}\left[|S| \mid v \in S\right] \geq p_v/2$$

by appealing to Jensen's inequality. Since $\hat{p}_v \leq p_v$, we find that this simple protocol maintains a ratio $r_v := \hat{p}_v/p_v \in [\frac{1}{2}, 1]$. Unfortunately, this analysis is tight, as can be seen from an example with two sensors with $p_1 = \varepsilon, p_2 = 1 - \varepsilon$.

To improve upon the simple protocol, first consider running it on an example with $p_1 = p_2 = \cdots = p_n = 1/n$. Since the protocol behaves exactly the same under permutations of sensor labels, by symmetry we have $\hat{p}_1 = \hat{p}_2 = \cdots = \hat{p}_n$, and thus $r_i = r_j$ for all i, j. Now consider an input distribution p where there exists integers N and $k_1, k_2, \ldots, k_n$ such that $p_v = k_v/N$ for all v. Replace each v with k_v fictitious sensors, each with probability mass $1/N$, and each with a label indicating v. Run the simple protocol with the fictitious sensors, selecting a fictitious sensor v', and then actually select the sensor indicated by the label of v'. By symmetry this process selects each fictitious sensor with probability $(1 - \beta)/N$, where β is the probability that nothing at all is selected, and thus the process selects sensor v with probability $k_v(1 - \beta)/N = (1 - \beta)p_v$ (since at most one fictitious sensor is ever selected).

We may thus consider the following improved protocol which incorporates the above idea, simulating this modification to the protocol exactly when $p_v = k_v/N$ for all v.

The Improved Protocol(N):
For each sensor v in parallel
  Sample $X_v \sim \text{Binomial}(1/N, \lceil N \cdot p_v \rceil)$.
  If $(X_v \geq 1)$, then activate sensor v.
From the active sensors S, select sensor v with probability $X_v / \sum_{v' \in S} X_{v'}$.

This protocol ensures the ratios $r_v := \hat{p}_v/p_v$ are the same for all sensors, provided each p_v is a multiple of $1/N$. Assuming the probabilities are rational, there will be a sufficiently large N to satisfy this condition. To reduce $\beta := \mathbf{Pr}[S = \emptyset]$ in the simple protocol, we may sample each X_v from $\text{Bernoulli}(\alpha \cdot p_v)$ for any $\alpha \in [1, n]$. The symmetry argument remains unchanged. This in turn suggests sampling X_v from $\text{Binomial}(\alpha/N, \lceil N \cdot p_v \rceil)$ in the improved protocol. Taking the limit as $N \to \infty$, the binomial distribution becomes Poisson, and we obtain the desired protocol.

The Limit Protocol(α):
Same as the improved protocol, except each sensor v samples $X_v \sim \text{Poisson}(\alpha p_v)$

Straight-forward calculation shows that

$$\mathbf{Pr}[S = \emptyset] = \prod_v \exp\{-\alpha \cdot p_v\} = \exp\{-\sum_v \alpha \cdot p_v\} = e^{-\alpha}$$

Let C be the number of messages. Then

$$\mathbb{E}[C] = \sum_v \mathbf{Pr}[X_v \geq 1] = \sum_v (1 - e^{-\alpha p_v}) \leq \sum_v \alpha p_v = \alpha$$

Here we have used linearity of expectation, and $1 + x \leq e^x$ for all $x \in \mathbb{R}$. In summary, we have the following result about our Limit Protocol:

PROPOSITION 5.1. *Fix any fixed p and $\alpha > 0$. The Limit Protocol always selects at most one sensor, ensures*

$$\forall v : \mathbf{Pr}[v \textit{ selected}] = (1 - e^{-\alpha})p_v$$

and requires no more than α messages in expectation.

In order to ensure that exactly one sensor is selected, whenever $S = \emptyset$ we can simply rerun the protocol with fresh random seeds as many times as needed until S is non-empty. Using $\alpha = 1$, this modification will require only $\mathcal{O}(1)$ messages in expectation and at most $\mathcal{O}(\log n)$ messages with high probability in the broadcast model. We can combine this protocol with EXP3 to get the following result.

THEOREM 1. *In the broadcast model, running* EXP3 *using the Limit Protocol with $\alpha = 1$, and rerunning the protocol whenever nothing is selected, yields the same regret bound as standard* EXP3*, namely $\mathcal{O}(\sqrt{\mathsf{OPT} n \log n})$, where OPT is the total reward of the best action. Furthermore, in each round at most $e/(e-1) + 2 \approx 3.582$ messages are broadcast in expectation.*

Proofs of our theoretical results can be found in Appendix **??**

5.2 The Distributed Online Greedy Algorithm

We now use our single sensor selection algorithm to develop our main algorithm, the Distributed Online Greedy algorithm (DOG). It is based on the distributed implementation of EXP3 using the Limit Protocol. Suppose we would like to select k sensors at each round t. Each sensor v maintains k weights $w_{v,1}, \ldots, w_{v,k}$ and normalizing constants $Z_{v,1}, \ldots, Z_{v,k}$. The algorithm proceeds in k stages, synchronized using the common clock. In stage i, a single sensor is selected using the Limit Protocol applied to the distribution $(1 - \gamma)w_{v,i}/Z_{v,i} + \gamma/n$. Suppose sensors $S = \{v_1, \ldots, v_{i-1}\}$ have been selected in stages 1 through $i - 1$. The sensor v selected at stage i then computes its local rewards $\pi_{v,i}$ using the utility function $f_t(S \cup \{v_i\}) - f_t(S)$. It then computes its new weight

$$w'_{v,i} = w_{v,i} \exp(\eta \pi_{v,i}/p_{v,i}),$$

and broadcasts the difference between its new and old weights $\Delta_{v,i} = w'_{v,i} - w_{v,i}$. All sensors then update their i^{th} normalizers using

$$Z_{v,i} \leftarrow Z_{v,i} + \Delta_{v,i}.$$

Fig. 1 presents the pseudo-code of the DOG algorithm. Thus given Theorem 12 of [24] we have the following result about the DOG algorithm:

THEOREM 2. *The* DOG *algorithm selects, at each round t a set $S_t \subseteq V$ of k sensors such that*

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} f_t(S_t)\right] \geq \frac{1-\frac{1}{e}}{T}\max_{|S|\leq k}\sum_{t=1}^{T} f_t(S) - O\left(k\sqrt{\frac{n\log n}{T}}\right).$$

In expectation, only $\mathcal{O}(k)$ messages are exchanged each round.

6. THE POINT-TO-POINT MODEL

In some applications, the assumption that sensors can broadcast messages to all sensors may be unrealistic. Furthermore, in some applications sensors may not be able to compute the marginal benefits $f_t(S \cup \{s\}) - f_t(S)$ (since this calculation may be computationally complex). In this section, we analyze LAZYDOG, a variant of our DOG algorithm, which replace the above assumptions by the assumption that there is a dedicated base station[3] available which computes utilities and which can send non-broadcast messages to individual sensors.

We make the following assumptions:

1. Every sensor stores its probability mass p_v with it, and can send messages to and receive messages from the base station.

2. The base station is able, after receiving messages from a set S of sensors, to compute the utility $f_t(S)$ and send this utility back to the active sensors.

We do not assume that the base station has access to all weights of the sensors – we will only require the base station to have $\mathcal{O}(k + \log n)$ memory. In the fully distributed algorithm DOG that relies on broadcasts, it is easy for the sensors to maintain their normalizers $Z_{v,i}$, since they receive information about rewards from all selected sensors. The key challenge when removing the broadcast assumption is to maintain the normalizers in an appropriate manner.

6.1 Lazy renormalization and Distributed EXP3

EXP3 (and all multiarmed bandit algorithms with no-regret guarantees against arbitrary reward functions) must maintain a distribution over actions, and update this distribution in response to feedback about the environment. In EXP3, each sensor v requires only $w_v(t)$ and a normalizer $Z(t) := \sum_{v'} w_{v'}(t)$ to compute $p_v(t)$[4]. The former changes only when v is selected. In the broadcast model the latter can simply be broadcast at the end of each round, however things are not so easy in the point-to-point model. Nevertheless, in this section we show that for EXP3 (and many algorithms like it), we can perform the necessary renormalization in a lazy manner, while keeping the amount of communication quite low.

Our lazy renormalization scheme for EXP3 works as follows. Each sensor v maintains its weight $w_v(t)$ and an estimate $Z_v(t)$ for $Z(t) := \sum_{v'} w_{v'}(t)$, Initially, $w_v(0) = 1$ and $Z_v(0) = n$ for all v. The central server stores $Z(t)$. Let $\rho(x, y) := (1-\gamma)\frac{x}{y} + \frac{\gamma}{n}$. Each sensor then proceeds with the sampling procedure of section 5.1 as though its probability mass in round t was $q_v = \rho(w_v(t), Z_v(t))$ instead of its true value of $\rho(w_v(t), Z(t))$. Moreover, v's estimate $Z_v(t)$ is only updated on rounds when it communicates with the server under these circumstances. This allows the estimated probabilities of all of the sensors to sum to more than one, but has the benefit of significantly reducing the communication cost in the point-to-point model under certain assumptions. We call the result *Distributed EXP3*, give its pseudocode for round t in Fig. 2.

Since the sensors underestimate their normalizers, they may activate more frequently than in the broadcast model. Fortunately, the amount of "overactivation" remains bounded. We prove Theorem 3 and Corollary 6.1 in Appendix B.

THEOREM 3. *The expected number of sensor activations in any round of our Distributed* EXP3 *algorithm is at most $\alpha\,(1 + (e-1)/\alpha)$, and the expected number of messages is at most twice this.*

Unfortunately, there is still an $e^{-\alpha}$ probability of nothing being selected. To address this, we can set $\alpha = \ln n$, and in the rare case that nothing is selected, transmit a message to each of the n sensors to rerun the protocol.

COROLLARY 6.1. *There is a distributed implementation of* EXP3 *that always selects a sensor in each round such that the expected number of sensor activations in any round is at most $\ln n + \mathcal{O}(1)$ and the expected number of messages is at most $2\ln n + \mathcal{O}(1)$.*

6.2 LazyDOG

Once we have the distributed EXP3 variant described above, we can use it for the bandit subroutines in the OG_{UNIT} algorithm (*c.f.* Sec. 4.2). We call the result the LAZYDOG algorithm, due to its use of lazy renormalization. It has the same performance guarantees with respect to $\sum_t f_t(S_t)$ as DOG, works in the point-to-point communication model, and requires few messages or sensor activations. Corollary 6.1 immediately implies the following result.

COROLLARY 6.2. *The expected number of sensors that activate each round in our* LAZYDOG *algorithm is at most $k\ln n + \mathcal{O}(k)$, and and the expected number of messages is at most $2k\ln n + \mathcal{O}(k)$.*

[3]Though the existence of such a base station means the protocol is not completely distributed, it is realistic in sensor network applications where the sensor data needs to be accumulated somewhere for analysis.

[4]We let $x(t)$ denote the value of variable x at the start of round t, to ease analysis. We don't actually need to store the value of the variables at each time step.

Algorithm: Distributed Online Greedy (DOG) (described in the broadcast model)

Input: $k \in \mathbb{N}$, a set V, and $\alpha, \gamma, \eta \in \mathbb{R}_{>0}$. Reasonable defaults are any $\alpha \in [1, \ln |V|]$, and $\gamma = \eta = \min\left(1, (|V| \ln |V|/g)^{1/2}\right)$, where g is a guess for the maximum cumulative reward of any single sensor [2].

Initialize $w_{v,i} \leftarrow 1$ and $Z_{v,i} \leftarrow |V|$ for all $v \in V$, $i \in [k]$. Let $\rho(x, y) := (1-\gamma)\frac{x}{y} + \frac{\gamma}{|V|}$.

for each round $t = 1, 2, 3, \ldots$ **do**
  Initialize $S_{v,t} \leftarrow \emptyset$ for each v in parallel.
  for each stage $i \in [k]$ **do**
    for each sensor $v \in V$ in parallel **do**
      repeat
        Sample $X_v \sim \text{Poisson}(\alpha \cdot \rho(w_{v,i}, Z_{v,i}))$.
        if $(X_v \geq 1)$ **then**
          Broadcast $\langle \text{sampled } X_v, \text{id}(v) \rangle$; Receive messages from sensors S. (Include $v \in S$ for convenience).
          if $\text{id}(v) = \min_{v' \in S} \text{id}(v')$ **then**
            Select exactly one element v_{it} from S such that each v' is selected with probability $X_{v'} / \sum_{u \in S} X_u$.
            Broadcast $\langle \text{select } \text{id}(v_{it}) \rangle$.
          Receive message $\langle \text{select } \text{id}(v_{it}) \rangle$.
          if $\text{id}(v) = \text{id}(v_{it})$ **then**
            Observe $f_t(S_{v,t} + v)$; $\pi \leftarrow f_t(S_{v,t} + v) - f_t(S_{v,t})$; $\Delta_v \leftarrow w_{v,i}(\exp\{\eta \cdot \pi / \rho(w_{v,i}, Z_{v,i})\} - 1)$;
            $Z_{v,i} \leftarrow Z_{v,i} + \Delta_v$; $w_v \leftarrow w_v + \Delta_v$; Broadcast $\langle \text{weight update } \Delta_v, \text{id}(v) \rangle$.
        if receive message $\langle \text{weight update } \Delta, \text{id}(v_{it}) \rangle$ **then** $S_{v,t} \leftarrow S_{v,t} + v_{it}$; $Z_{v,i} \leftarrow Z_{v,i} + \Delta$
      until v receives a message of type $\langle \text{select } \text{id} \rangle$;

Output: At the end of each round t each sensor has an identical local copy $S_{v,t}$ of the selected set S_t.

Figure 1: The Distributed Online Greedy Algorithm

7. OBSERVATION-DEPENDENT SAMPLING

Theorem 2 states that DOG is guaranteed to do nearly as well as the offline greedy algorithm run on an instance with objective function $f_\Sigma := \sum_t f_t$. Thus the reward of DOG is asymptotically good on average. In many applications, however, we would like to perform well on rounds with "atypical" objective functions. For example, in an outbreak detection application as we discuss in Sec. 8, we would like to get very good data on rounds with significant events, even if the nearest sensors typically report "boring" readings that contribute very little to the objective function. For now, suppose that we are only running a single MAB instance to select a single sensor in each round. If we have access to a value oracle for f_t on round t, then we can perform well on atypical rounds at the cost of some additional communication by having each sensor v take a local reading of its environment and estimate its payoff $\bar{\pi} = f_t(\{v\})$ if selected. This value, which serves as a measure of how interesting its input is, can then be used to decide whether to boost v's probability for reporting its sensor reading to the server. In the simplest case, we can imagine each v has a threshold τ_v such that v activates with probability 1 if $\bar{\pi} \geq \tau_v$, and with its normal probability otherwise. In the case where we select $k > 1$ sensors in each round, each sensor can have a threshold for each of the k stages, where in each stage it computes $\bar{\pi} = f_t(S \cup \{v\}) - f_t(S)$ where S is the set of currently selected sensors. Since the activation probability only goes up, we can retain the performance guarantees of DOG if we are careful to adjust the feedback properly.

Ideally, the sensors should learn what their thresholds τ_v should be. We treat the selection of τ_v in each round as an online decision problem that each v must play. We construct a particular game that the sensors play, where the strategies are the thresholds (suitable discretized), there is an *activation cost* c_v that v pays if $\bar{\pi}_v \geq \tau_v$, and the value of of the information obtained is split between the sensors that activate. Let $\pi_v = f_t(S \cup \{v\}) - f_t(S)$ be the marginal benefit of selecting v given that sensor set S has already been selected. Let A be the set of sensor that activate in the current iteration of the game, and let $\max(\pi_{(A \setminus v)}) := \max(\pi_{v'} : v' \in A \setminus \{v\})$. The particular reward function ψ we chose for each sensor v for each iteration of the game is

$$\psi_v(\tau) = \begin{cases} c_v - \max\left(\pi_v - \max\left(\pi_{(A \setminus v)}\right), 0\right) & \text{if } \bar{\pi} < \tau \\ \max\left(\pi_v - \max\left(\pi_{(A \setminus v)}\right), 0\right) - c_v & \text{if } \bar{\pi} \geq \tau \end{cases}$$

based on empirical performance.

In the broadcast model where each sensor can compute its marginal benefit, we can use any standard no-regret algorithm for combining expert advice, such as *Randomized Weighted Majority* (WMR) [19], to play this game and obtain no regret guarantees[5] for selecting τ_v. In our context

[5]We leave it as an open problem to determine if the outcome is close to optimal when all sensors play low regret strategies (i.e., is the *price of total anarchy* [4] small in any variant of this game with

Algorithm: Distributed EXP3 (executing on round t)

Input: Parameters $\alpha, \eta, \gamma \in \mathbb{R}_{>0}$, sensor set V.
Let $\rho(x, y) := (1-\gamma)\frac{x}{y} + \frac{\gamma}{|V|}$.
Sensors (or Actions/Arms, equivalently):
foreach sensor v in parallel **do**
  Sample r_v uniformly at random from $[0, 1]$.
  if $(r_v \geq 1 - \alpha \cdot \rho(w_v(t), Z_v(t))$ **then**
    Send $\langle r_v, w_v(t)\rangle$ to the server.
    Receive message $\langle Z, w\rangle$ from server.
    $Z_v(t+1) \leftarrow Z$; $w_v(t+1) \leftarrow w$.
  else $Z_v(t+1) \leftarrow Z_v(t)$; $w_v(t+1) \leftarrow w_v(t)$.
Server:
Receive messages from a set S of sensors.
if $S = \emptyset$ **then** Select nothing and wait for next round.
else **foreach** sensor $v \in S$ **do**
  $Y_v \leftarrow \min\{x : \mathbf{Pr}[X \leq x] \geq r_v\}$, where
  $X \sim \text{Poisson}(\alpha \cdot \rho(w_v(t), Z(t)))$.
  Select v with probability $Y_v / \sum_{v' \in S} Y_{v'}$.
  Observe the payoff π for the selected sensor v^*;
  $w_{v^*}(t+1) \leftarrow w_{v^*}(t) \cdot \exp\{\eta\pi/\rho(w_{v^*}(t), Z(t))\}$;
  $Z(t+1) \leftarrow Z(t) + w_{v^*}(t+1) - w_{v^*}(t)$;
  for each $v \in S \setminus v^*$ **do** $w_v(t+1) \leftarrow w_v(t)$;
  for each $v \in S$ **do** Send $\langle Z(t+1), w_v(t+1)\rangle$ to v.

Figure 2: Distributed EXP3: the limit protocol(α) with lazy renormalization, applied to EXP3

a sensor using WMR simply maintains weights $w(\tau_i) = \exp(\eta \cdot \psi_{\text{total}}(\tau_i))$ for each possible threshold τ_i, where $\eta > 0$ is a learning parameter, and $\psi_{\text{total}}(\tau_i)$ is the total cumulative reward for playing τ_i in every round so far. On each step each threshold is picked with probability proportional to its weight. In the more restricted point-to-point model, we can use a modification of WMR that feeds back unbiased estimates for $\psi_t(\tau_i)$, the payoff to the sensor for using a threshold of τ_i in round t, and thus obtains reasonably good estimates of $\psi_{\text{total}}(\tau_i)$ after many rounds. We give psuedocode in Fig. 3. In it, we assume that an activated sensor can compute the reward of playing any threshold.

We incorporate these ideas into the DOG algorithm, to obtain what we call the *Observation-Dependent Distributed Online Greedy* algorithm (OD-DOG). In the extreme case that $c_v = 0$ for all v the sensors will soon set their thresholds so low that each sensor activates in each round. In this case OD-DOG will exactly simulate the offline greedy algorithm run on each round. In other words, if we let $G(f)$ be the result of running the simple offline greedy algorithm on the problem

$$\arg\max\{f(S) : S \subset V,\ |S| \leq k\}$$

then OD-DOG will obtain a value of $\sum_t f_t(G(f_t))$; in contrast, DOG gets roughly $\sum_t f_t(G(\sum_t f_t))$, which may be

a reasonable way of splitting the value from the information?)

Algorithm: Modified WMR (point-to-point setting)

Input: parameter $\eta > 0$, threshold set $\{\tau_i : i \in [m]\}$
Initialize $w(\tau_i) \leftarrow 1$ for all $i \in [m]$.
for each round $t = 1, 2, \ldots$ **do**
  Select τ_i with probability $w(\tau_i) / \sum_{j=1}^{m} w(\tau_j)$.
  if sensor activates **then**
    Let $\psi(\tau_i)$ be the reward for playing τ_i in this round of the game. Let $q(\tau_i)$ be the total probability of activation conditioned on τ_i being selected (including the activation probability that does not depend on local observations.)
    for each threshold τ_i **do**
      $w(\tau_i) \leftarrow w(\tau_i) \exp(\eta\psi(\tau_i)/q(\tau_i))$.

Figure 3: Selecting activation thresholds for a sensor

significantly smaller. Note that Feige's result [11] implies that the former value is the best we can hope for from efficient algorithms (assuming P $\neq$ NP). Of course, querying each sensor in each round is impractical when querying sensors is expensive. In the other extreme case where $c_v = \infty$ for all v, OD-DOG will simulate DOG after a brief learning phase. In general, by adjusting the activation costs c_v we can smoothly trade off the cost of sensor communication with the value of the resulting data.

8. EXPERIMENTS

In this section, we evaluate our DOG algorithm on several real-world sensing problems.

8.1 Data sets

Temperature data. In our first data set, we analyze temperature measurements from the network of 46 sensors deployed at Intel Research Berkeley. Our training data consisted of samples collected at 30 sec. intervals on 3 consecutive days (starting Feb. 28th 2004), the testing data consisted of the corresponding samples on the two following days. The objective functions used for this application are based on the expected reduction in mean squared prediction error f_{EMSE}, as introduced in Sec. 2.

Precipitation data. Our second data set consists of precipitation data collected during the years 1949 - 1994 in the states of Washington and Oregon [27]. Overall 167 regions of equal area, approximately 50 km apart, reported the daily precipitation. To ensure the data could be reasonably modeled using a Gaussian process we preprocessing as described in [17]. As objective functions we again use the expected reduction in mean squared prediction error f_{EMSE}.

Water network monitoring. Our third data set is based on the application of monitoring for outbreak detection. Con-

sider a city water distribution networks for, delivering water to households via a system of pipes, pumps and junctions. Accidental or malicious intrusions can cause contaminants to spread over the network, and we want to install sensors to detect these contaminations as quickly as possible. In August 2006, the Battle of Water Sensor Networks (BWSN) [10] was organized as an international challenge to find the best sensor placements for a real (but anonymized) metropolitan water distribution network, consisting of 12,527 nodes. In this challenge, a set of intrusion scenarios is specified, and for each scenario a realistic simulator provided by the EPA is used to simulate the spread of the contaminant for a 48 hour period. An intrusion is considered detected when one selected node shows positive contaminant concentration. The goal of BWSN was to minimize impact measures, such as the expected population affected, which is calculated using a realistic disease model. For a security-critical sensing task such as protecting drinking water from contamination, it is important to develop sensor selection schemes that maximize detection performance even in adversarial environments (i.e., where an adversary picks the contamination strategy knowing our network deployment and selection algorithm). The algorithms developed in this paper apply to such adversarial settings. We reproduce the experimental setup detailed in [16]. For each contamination event i, we define a separate submodular objective function $f_i(S)$ that measures the expected population protected when detecting the contamination from sensors S. In [16], Krause et al. showed that the functions $f_i(A)$ are monotonic submodular functions.

8.2 Convergence experiments

In our first set of experiments, we analyzed the convergence of our DOG algorithm. For both the temperature [T] and precipitation [R] data sets, we first run the offline greedy algorithm using the f_{EMSE} objective function to pick $k = 5$ sensors. We compare its performance to the DOG algorithm, where we feed back the same objective function at every round. We use an exploration probability $\gamma = 0.01$ and a learning rate inversely proportional to the maximum achievable reward $f_{\mathrm{EMSE}}(V)$. Fig. 4(a) presents the results for the temperature data set. Note that even after only a small number of rounds (≈ 100), the algorithm obtains 95% of the performance of the offline algorithm. After about 13,000 iterations, the algorithm obtains 99% of the offline performance, which is the best that can be expected with a .01 exploration probability. Fig. 4(b) show the same experiment on the precipitation data set. In this more complex problem, after 100 iterations, 76% of the offline performance is obtained, which increases to 87% after 500,000 iterations.

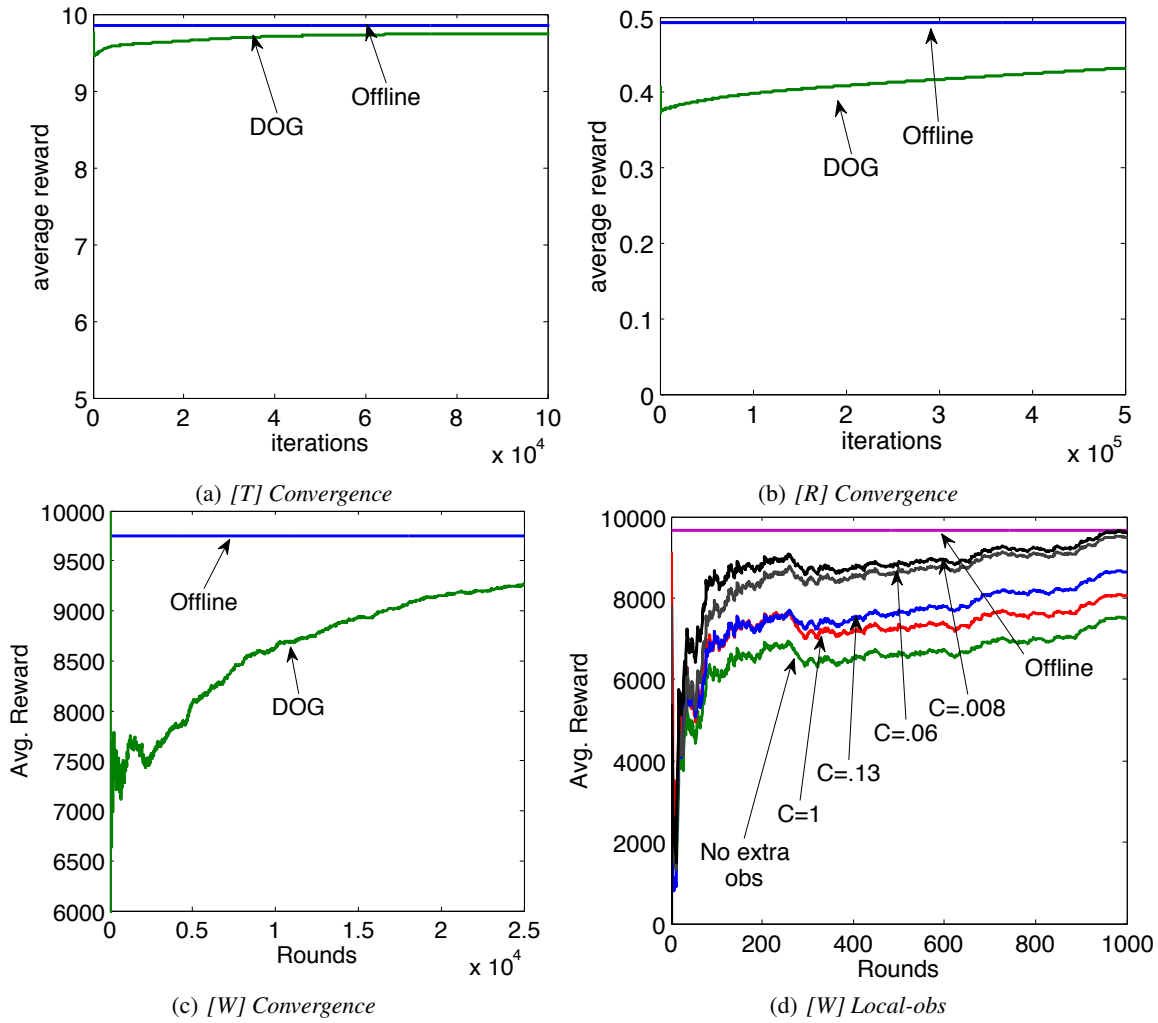
We also perform experiments on the water distribution monitoring data set [W]. We use our algorithms to select an active subset of $k = 3$ from a placement of 20 sensors. At each round, a different objective function F_i is selected, one for each contamination event. We first run the offline greedy algorithm on the average objective $f_{\Sigma}(S) = \sum_i f_i(S)$. We compare the offline result with our DOG algorithm. Fig. 4(c) presents the results of this experiment. Note that in contrast to the results in Fig. 4(a) and Fig. 4(b), where we always feed back the same objective function, in the varying objective function case, there is more variance in the performance. However, even in spite of these varying objectives, the average performance still converges to the value of the offline solution that optimizes the selection given knowledge of all objective functions.

8.3 Observation dependent activation

We also run experiments on our OD-DOG algorithm with observation specific sensor activations. We choose different values for the activation cost c_v, which we vary as multiples of the total achievable reward. Fig. 4(d) presents the rewards obtained over varying numbers of rounds for the different values of c_v. The activation cost c_v lets us smoothly trade off the average number of sensors activating each round and the average obtained reward. Fig. 4(d) presents the performance of different observation-dependent selection strategies for varying activation costs. Note that for low activation costs, the performance quickly (within 1000 iterations) converges to the performance of the offline solution. In contrast, when not considering local observations, the same performance level is not reached even for 25,000 iterations. Even under the lowest activation costs in our experiments, the number of extra activations per stage in the OD-DOG algorithm is at most 5.8. These results indicate that observation specific activation can lead to drastically improved performance at small additional activation cost.

9. RELATED WORK

Sensor Selection. The problem of deciding when to selectively turn on sensors in sensor networks in order to conserve power was first discussed by [23] and [29]. Many approaches for optimizing sensor placements and selection assume that sensors have a fixed region [14, 13, 3]. These regions are usually convex or even circular. Furthermore, it is assumed that everything within this region can be perfectly observed, and everything outside cannot be measured by the sensors. For complex applications such as environmental monitoring however, such assumptions are unrealistic, and the direct optimization of prediction accuracy is desired. The problem of selecting observations for monitoring spatial phenomena has been investigated extensively in geostatistics (*c.f.*, [7] for an overview), and more generally (Bayesian) experimental design (*c.f.*, [6]). Several approaches have been proposed to activate sensors in order to minimize uncertainty [29] or prediction error [9]. However, these approaches do not have performance guarantees. Submodularity has been used to analyze algorithms for placing [17] or selecting [28] a fixed set of sensors . These approaches however assume that the model is known in advance.



(a) *[T] Convergence* (b) *[R] Convergence*

(c) *[W] Convergence* (d) *[W] Local-obs*

Figure 4: Experimental results on [T] Temperature data, [R] precipitation data and [W] water distribution network data.

Submodular optimization. The problem of centralized maximization of a submodular function has been studied by [20], who proved that the greedy algorithm gives a factor $(1-1/e)$ approximation. Since then, several algorithms have been developed for solving submodular maximization problems subject to more complex constraints (see [26] for an overview). Streeter and Golovin developed an algorithm for online optimization of submodular functions, which we build on in this paper [25].

10. CONCLUSIONS

In this paper, we considered the problem of repeatedly selecting subsets S_t from a large set of deployed sensors, in order to maximize a sequence of submodular utility functions $f_1, \ldots, f_T$. We developed an efficient Distributed Online Greedy algorithm DOG, which we proved to obtain no $(1 - 1/e)$-regret, essentially the best possible performance obtainable unless $P = NP$. Our algorithm is fully distributed, requiring only a small number of messages to be exchanged at each round with high probability. We analyze our algorithm both in the broadcast model (where sensors can broadcast messages to all other sensors), and in the point-to-point model, where a separate base station is responsible for computing utilities of selected sets of sensors. Our LAZYDOG algorithm for the latter model uses lazy renormalization in order to reduce the number of messages required from $\Theta(n)$ to $\mathcal{O}(k \log n)$, and the server memory required from $\Theta(n)$ to $\mathcal{O}(k + \log n)$, where k is the desired number of sensors to be selected. In addition, we developed OD-DOG, an extension of DOG that allows observation-dependent sensor selection. We empirically demonstrate the effectiveness of our algorithms on three real-world sensing tasks, demonstrating the ability of our DOG algorithm to converge to a comparable performance of the offline algorithm. In addition, our results with the OD-DOG algorithm indicate that a small number of extra sensor activations can lead to drastically improved convergence. We believe that our results provide an interesting step towards a principled study of distributed active learning and information gathering approaches.

11. REFERENCES

[1] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IPSN*, 2004.

[2] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[3] X. Bai, S. Kumar, Z. Yun, D. Xuan, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *ACM MobiHoc*, Florence, Italy, 2006.

[4] Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. Regret minimization and the price of total anarchy. In *STOC '08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 373–382, 2008.

[5] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *World Sensor Web Workshop, ACM Sensys*, 2006.

[6] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Stat. Sci.*, 10(3):273–304, Aug. 1995.

[7] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

[8] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *STOC*, 2008.

[9] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.

[10] A. Ostfeld et al. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *To appear in J. Wat. Res. Plan. Mgmt.*, 2008.

[11] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[12] Dean P. Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, October 1999.

[13] H. H. Gonzalez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th ACM Symp. Comp. Geom.*, pages 232–240, 2001.

[14] D. S. Hochbaum and W. Maas. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985.

[15] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in Artificial Intelligence (UAI)*, 2005.

[16] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *J. Wat. Res. Plan. Mgmt.*, 136(6), 2008.

[17] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *JMLR*, 2007.

[18] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI Nectar track*, 2007.

[19] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[20] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

[21] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.

[22] V. Singhvi, A. Krause, C. Guestrin, J. Garrett, and H.S. Matthews. Intelligent light control using sensor networks. In *SenSys*, 2005.

[23] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *ICC*, 2001.

[24] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. Technical Report CMU-CS-07-171, Carnegie Mellon University, 2007.

[25] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584, 2008.

[26] Jan Vondrák. *Submodularity in Combinatorial Optimization*. PhD thesis, Charles University, Prague, Czech Republic, 2007.

[27] M. Widmann and C. S. Bretherton. 50 km resolution daily precipitation for the pacific northwest. http://www.jisao.washington.edu/data_sets/widmann/, May 1999.

[28] J.L. Williams, J.W. Fisher III, and A.S. Willsky. Performance guarantees for information theoretic active inference. In *AISTATS*, 2007.

[29] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing*, 19(2):61–72, 2002.

APPENDIX

A. RESULTS IN THE BROADCAST MODEL

PROOF OF THEOREM 1. To prove the regret bounds, note that in every round the distribution over sensor selections in the variant of EXP3 we describe (that uses the distributed multinomial sampling scheme and repeatedly reruns the protocol in order to always select some sensor in each round)

is precisely the same as the original EXP3. Thus the regret bounds for EXP3 [2] carry over unchanged. We next bound the number of broadcasts. Fix a round, and let S set of sensors that activate in that round. The total number of broadcasts is then $|S|+2$; using their calibrated clocks, each sensor (re)samples $X_v \sim \text{Poisson}(\alpha p_v)$ and activates if $X_v \geq 1$. If no sensors activate before a specified timeout period, the default behavior is to rerun the sampling step. Eventually $|S| \geq 1$ sensors activate in the same period. A distinguished sensor in S then determines the selected sensor v, broadcasts $\text{id}(v)$, and v broadcasts its observed reward. We prove $\mathbb{E}[|S|] \leq \alpha/(1-e^{-\alpha})$ in Proposition A.1. When $\alpha = 1$, this gives us the claimed bound on the number of broadcasts. □

PROPOSITION A.1. *Rerunning the limit protocol until an element is selected results in at most $\alpha/(1-e^{-\alpha})$ elements being activated in expectation. Moreover, this value is tight.*

PROOF. Let $X_v \sim \text{Bernoulli}(\alpha \cdot p_v)$ be the indicator random variable for the activation of v, and let $X := \sum_v X_v$. The expected number of sensor activations is then

$$\mathbb{E}[X \mid X \geq 1] = \mathbb{E}[X] / \mathbf{Pr}[X \geq 1].$$

In the limit as $\max_v p_v$ tends to zero, X converges to a Poisson random variable with mean α. In this case, $\frac{\mathbb{E}[X]}{\mathbf{Pr}[X \geq 1]} = \alpha/(1-e^{-\alpha})$ To see that this is an upper bound, consider an arbitrary distribution p on the sensors, and fix some v with $x := p_v > 0$. We claim that replacing v with two sensors v_1 and v_2 with positive probability mass x_1 and x_2 with $x = x_1 + x_2$ can only serve to increase the expected number of sensor activations, because $\mathbb{E}[X]$ is unchanged, and $\mathbf{Pr}[X \geq 1]$ decreases. The latter is true essentially because $\mathbf{Pr}[\exists i \in \{1,2\} : v_i \text{ activates}] = 1 - (1-x_1)(1-x_2) = x - x_1 x_2 < x$. To complete the proof, notice that repeating this process with $v = \arg\max(p_v)$ and $x_i = x/2$ ensures X converges to a Poisson variable with mean α, while only increasing $\mathbb{E}[X \mid X \geq 1]$. □

B. RESULTS IN THE POINT-TO-POINT MODEL

In this section we will bound the communication overhead of using lazy renormalization for any MAB algorithm satisfying certain assumptions enumerated below, and then show how these bounds apply to EXP3.

Fix an action v and a multiarmed bandit algorithm. Let $p_v(t) \in [0,1]$ be the random variable denoting the probability the algorithm assigns to v on round t. The value of $p_v(t)$ depends on the random choices made by the algorithm and the payoffs observed by it on previous rounds. We assume the following about each $p_v(t)$.

1. $p_v(t)$ can be computed from local information v possesses and global information the server has.
2. There exists an $\epsilon > 0$ such that $p_v(t) \geq \epsilon$ for all t.
3. $p_v(t) < p_v(t+1)$ implies v was selected in round t.
4. There exists $\hat{\epsilon} > 0$ such that $p_v(t+1) \geq p_v(t)/(1+\hat{\epsilon})$ for all t.

Many MAB algorithms satisfy these conditions. For example, all MAB algorithms with non-trivial no-regret guarantees against adversarial payoff functions must continually explore all their options, which effectively mandates $p_v(t) \geq \epsilon$ for some $\epsilon > 0$. In Lemma 1 we prove that EXP3 does so with $\epsilon = \gamma/n$ and $\hat{\epsilon} = (e-1)\frac{\gamma}{n}$, assuming payoffs in $[0,1]$. In this case, Theorem 4 bounds the expected increase in sensor communications due to lazy renormalization by a factor of $1 + \frac{e-1}{\alpha}$.

THEOREM 4. *Fix a multiarmed bandit instance with possibly adversarial payoff functions, and a MAB algorithm satisfying the above assumptions on its distribution over actions $\{p_v(t)\}_{v \in V}$. Let $q_v(t)$ be the corresponding random estimates for $p_v(t)$ maintained under lazy renormalization with oversampling parameter α. Then for all v and t,*

$$\mathbb{E}[q_v(t)/p_v(t)] \leq 1 + \frac{\hat{\epsilon}}{\alpha\epsilon}$$

and

$$\mathbb{E}[q_v(t)] \leq \left(1 + \frac{\hat{\epsilon}}{\alpha\epsilon}\right) \mathbb{E}[p_v(t)].$$

PROOF. Fix v, and let $p(t) := p_v(t)$, $q(t) := q_v(t)$. We begin by bounding $\mathbf{Pr}[q(t) \geq \lambda p(t)]$ for $\lambda \geq 1$. Let t_0 be the most recent round in which $q(t_0) = p(t_0)$. We assume $q(0) = p(0)$, so t_0 exists. Then $q(t) = p(t_0) \geq \lambda p(t)$ implies $p(t_0)/p(t) \geq \lambda$. By assumption $p(t')/p(t'+1) \leq (1+\hat{\epsilon})$ for all t', so $p(t_0)/p(t) \leq (1+\hat{\epsilon})^{t-t_0}$. Thus $\lambda \leq (1+\hat{\epsilon})^{t-t_0}$ and $t - t_0 \geq \ln(\lambda)/\ln(1+\hat{\epsilon})$. Define $t(\lambda) := \ln(\lambda)/\ln(1+\hat{\epsilon})$.

By definition of t_0, there were no activations under lazy renormalization in rounds t_0 through $t-1$ inclusive, which occurs with probability $\prod_{t'=t_0}^{t-1}(1-\alpha q(t')) = (1-\alpha q(t))^{t-t_0} \leq (1-\alpha q(t))^{\lceil t(\lambda) \rceil}$, where α is the oversampling parameter in the protocol. We now bound $\mathbb{E}[q(t)/p(t) \mid q(t)]$. Recall that $\mathbb{E}[X] = \int_{x=0}^{\infty} \mathbf{Pr}[X \geq x]\, dx$ for any non-negative random variable X. It will also be convenient to define $\omega := \ln(1/(1-\alpha q(t)))/\ln(1+\hat{\epsilon})$ and assume for now that $\omega > 1$. Conditioning on $q(t)$, we see that

$$\begin{aligned}
\mathbb{E}[q(t)/p(t) \mid q(t)] &= \int_{\lambda=0}^{\infty} \mathbf{Pr}[q(t) \geq \lambda p(t)]\, d\lambda \\
&= 1 + \int_{\lambda=1}^{\infty} \mathbf{Pr}[q(t) \geq \lambda p(t)]\, d\lambda \\
&\leq 1 + \int_{\lambda=1}^{\infty} (1-\alpha q(t))^{t(\lambda)} d\lambda \\
&= 1 + \int_{\lambda=1}^{\infty} \lambda^{\ln(1-\alpha q(t))/\ln(1+\hat{\epsilon})} d\lambda \\
&= 1 + \int_{\lambda=1}^{\infty} \lambda^{-\omega} d\lambda \\
&= 1 + \tfrac{1}{\omega-1}
\end{aligned}$$

Using $\ln\left(\frac{1}{1-x}\right) \geq x$ for all $x < 1$ and $\ln(1+x) \leq x$ for all $x > -1$, we can show that $\omega \geq \alpha q(t)/\hat{\epsilon}$ so $1 + \frac{1}{\omega-1} \leq$

$\alpha q(t)/(\alpha q(t) - \hat{\epsilon})$. Thus, if $\alpha q(t) > \hat{\epsilon}$ then $\omega > 1$ and we obtain $\mathbb{E}[q(t)/p(t) \mid q(t)] \leq \alpha q(t)/(\alpha q(t) - \hat{\epsilon})$.

If $q(t) >> \hat{\epsilon}$, this gives a good bound. If $q(t)$ is small, we rely on the assumption that $p(t) \geq \epsilon$ for all t to get a trivial bound of $q(t)/p(t) \leq q(t)/\epsilon$. We thus conclude

$$\mathbb{E}[q(t)/p(t) \mid q(t)] \leq \min\left(\alpha q(t)/(\alpha q(t) - \hat{\epsilon}),\ q(t)/\epsilon\right). \tag{B.1}$$

Setting $q(t) = (\hat{\epsilon}/\alpha + \epsilon)$ to maximize this quantity yields an unconditional bound of $\mathbb{E}[q(t)/p(t)] \leq 1 + \hat{\epsilon}/\alpha\epsilon$.

To bound $\mathbb{E}[q(t)]$ in terms of $\mathbb{E}[p(t)]$, note that for all q

$$\begin{aligned} q/\mathbb{E}[p(t) \mid q(t) = q] &\leq \mathbb{E}[q(t)/p(t) \mid q(t) = q] \\ &\leq 1 + \hat{\epsilon}/\alpha\epsilon \end{aligned}$$

where the first line is by Jensen's inequality, and the second is by equation B.1. Thus $q \leq (1 + \hat{\epsilon}/\alpha\epsilon)\,\mathbb{E}[p(t) \mid q(t) = q]$ for all q. Taking the expectation with respect to q then proves $\mathbb{E}[q_v(t)] \leq \left(1 + \frac{\hat{\epsilon}}{\alpha\epsilon}\right)\mathbb{E}[p_v(t)]$ as claimed. □

LEMMA 1. EXP3 *with $\eta = \gamma/n$ satisfies the conditions of Theorem 4 with $\epsilon = \gamma/n$ and $\hat{\epsilon} = (e-1)\frac{\gamma}{n}$.*

PROOF. The former equality is an easy observation. To prove the latter equality, fix a round t and a selected action v. Let $w_v(t)$ be the weight of v in round t, and $W(t)$ be the total weight of all actions in round t. Let π be the payoff to v in round t. Given the update rule $w_v(t+1) = w_v(t)\exp\left(\frac{\gamma}{n}\frac{\pi(v,t)}{p_v(t)}\right)$, only the probabilities of the other actions will be decreased. It is not hard to see that they will be decreased by a multiplicative factor of at most $W(t)/W(t+1)$, no matter what the learning parameter γ is. By the update rule,

$$W(t+1) = W(t) + w_v(t)\left(\exp\left(\frac{\gamma}{n}\frac{\pi}{p_v(t)}\right) - 1\right).$$

Let $p := p_v(t)$ and $x := \frac{\gamma}{n}\pi$. Dividing the above equation by $W(t)$, we get

$$\begin{aligned} \frac{W(t+1)}{W(t)} &= 1 + p\left(\exp\left(x/p\right) - 1\right) && \text{(B.2)} \\ &\leq 1 + p\left(x/p + (e-2)(x/p)^2\right) && \text{(B.3)} \\ &\leq 1 + x + (e-2)x^2/p && \text{(B.4)} \end{aligned}$$

where in the second line we have used $e^x \leq 1 + x + (e-2)x^2$ for $x \in [0,1]$. Note $\pi \leq 1$ implies $x \leq \gamma/n \leq p$, so $\frac{W(t+1)}{W(t)} \leq 1 + (e-1)x \leq 1 + (e-1)\frac{\gamma}{n}$. It follows that setting $\hat{\epsilon} = (e-1)\frac{\gamma}{n}$ is sufficient to ensure $p_v(t+1) \geq p_v(t)/(1+\hat{\epsilon})$ for all t. □

We now prove Theorem 3 and Corollary 6.1.

PROOF OF THEOREM 3.. We prove in Lemma 1 that EXP3 satisfies the conditions of Theorem 4 with $\epsilon = \gamma/n$ and $\hat{\epsilon} = (e-1)\frac{\gamma}{n}$. Thus by Theorem 4

$$\begin{aligned} \mathbb{E}\left[\sum_v q_v(t)\right] &\leq (1 + (e-1)/\alpha)\,\mathbb{E}\left[\sum_v p_v(t)\right] \\ &= (1 + (e-1)/\alpha) \end{aligned}$$

because $\sum_v p_v(t) = 1$. Each sensor v activates with probability $\alpha q_v(t)$, so the expected number of activations is

$$\mathbb{E}\left[\alpha\sum_v q_v(t)\right] \leq \alpha\left(1 + (e-1)/\alpha\right).$$

Also, it is not too difficult to show that lazy renormalization selects each sensor v with probability $(1 - e^{-\alpha})p_v(t)$, because of the way the random bits r_v are shared in order to implement a coupled distribution for sensor activation and selection.

As for the number of messages, note that each message involves a sensor as sender or receiver, and by inspection the protocol only involves two messages per activated node. □

PROOF OF COROLLARY 6.1.. Use the distributed EXP3 protocol with lazy renormalization with $\alpha = \ln n$. We have already established that the probability of nothing being selected is $e^{-\alpha}$ or $1/n$ in this case. If nothing is selected, send out n messages, one to each sensor, to rerun the protocol. The expected number of messages sent to initiate additional runs of the protocol is $\sum_{x=1}^{\infty} nx/n^x = (1 - 1/n)^{-2} = 1 + \mathcal{O}(1/n)$. Let X be the number of sensor activations. As in the proof of Proposition A.1, if Y is the expected number of sensor activations without rerunning the protocol when nothing is selected, then $\mathbb{E}[X] = \mathbb{E}[Y]/\mathbf{Pr}[Y \geq 1]$. By Theorem 3 $\mathbb{E}[Y] \leq \alpha(1 + (e-1)/\alpha)$. Since $\mathbf{Pr}[Y \geq 1] = 1 - e^{-\alpha}$, we conclude

$$\mathbb{E}[X] \leq \ln n + (e-1) + \mathcal{O}\left(\frac{\ln n}{n}\right).$$

As for the number of messages, note that other than messages sent to initiate additional runs of the protocol, there are only two messages per activated node. □