Region Sampling: Continuous Adaptive Sampling on Sensor Networks

Song Lin #1, Benjamin Arai #2, Dimitrios Gunopulos #‡3, Gautam Das *4

#*Department of Computer Science and Engineering, University of California, Riverside*
‡*Department of Informatics, University of Athens*
**Department of Computer Science and Engineering, The University of Texas at Arlington*
1slin@cs.ucr.edu
2barai@cs.ucr.edu
3dg@cs.ucr.edu
4gdas@cse.uta.edu

***Abstract*— Satisfying energy constraints while meeting performance requirements is a primary concern when a sensor network is being deployed. Many recent proposed techniques offer error bounding solutions for aggregate approximation but cannot guarantee energy spending. Inversely, our goal is to bound the energy consumption while minimizing the approximation error. In this paper, we propose an online algorithm, Region Sampling, for computing approximate aggregates while satisfying a pre-defined energy budget. Our algorithm is distinguished by segmenting a sensor network into partitions of non-overlapping regions and performing sampling and local aggregation for each region. The sampling energy cost rate and sampling statistics are collected and analyzed to predict the optimal sampling plan. Comprehensive experiments on real-world data sets indicate that our approach is at a minimum of 10% more accurate compared with the previously proposed solutions.**

I. Introduction

Recent improvements in hardware technology have allowed for wide-scale distribution of sensor networks in a variety of settings. Inexpensive and small sensor devices have many resource limitations that introduce new challenges for data collection and aggregation in sensor networks. Among these limitations, energy is usually the primary concern when designing an in-network algorithm. This is because the sensor devices are either powered via a limited power source (such as batteries) where the total amount of expendable energy is limited, or renewable sources (such as solar energy) that allow continual use of energy but at a limited rate of consumption [1].

There are numerous applications in data-mining for efficient sampling techniques such as preserving data characteristics that would be otherwise unobtainable through traditional exact solutions. Distributed database sampling plays an important role in revealing patterns and correlations in streaming data. Traditional database research such as decision support and data analysis often requires the building of high-level structures such as mining models. Aggregation is useful for building such models that can be later used for various mining techniques (clustering, classification, etc.). In this paper, we are interested in the continuous approximation of aggregates for long-term monitoring in sensor networks that have a variety of real world applications such as monitoring air pollution [2], wild life habitats [3] and health conditions of residents [4]. In these applications, there is usually a query node in the network which continuously evaluates the aggregate query for all sensors. This node is used to detect events occurring within the monitored area and to notify the user to take actions accordingly.

One possible solution for this continuous aggregation problem is to apply in-network aggregation algorithms (such as TAG [5] or COUGAR [6]) to compute the exact result for a given query. In TAG, each node aggregates all the sensor data in a subtree and sends the aggregate results to its parent. As the energy resources for sensor devices are usually very limited, this approach may become infeasible for long-term applications.

Since the energy consumption of the sensor network is dominated by radio transmissions, there are two network traffic reduction approaches to address the problem. One is to collect the sensor data periodically instead of performing a query continuously. This approach has drawbacks in that important information may be missed or disregarded. Another solution to the energy limitation problem is data sampling that performs data collection and aggregation at every time moment, but only a small fraction of the sensor data is accessed. In this approach, there is usually a pre-defined energy budget which is defined as the total network energy cost that can be utilized for each round of data sampling. The size of the budget depends on the total energy capacity of the sensor network and how many times the user expects to issue a query over the network.

Example: Let us consider a sensor network with 6 sensors and a query node as shown in Figure 1. At each timestamp every sensor takes a reading and sends it back to the query node. Suppose at some timestamp we take a reading from each sensor to obtain the following values $\{2, 2, 4, 4, 8, 10\}$. If the energy budget of the network allows, we can obtain a value from every sensor to obtain an exact average value of 5. Now assume we do not have enough energy budget and we can only access at most 4 sensors. In order to meet the budget constraint we can randomly sample 4 values from the network (e.g. $\{S1, S2, S4, S6\}$) but this would give us an average value

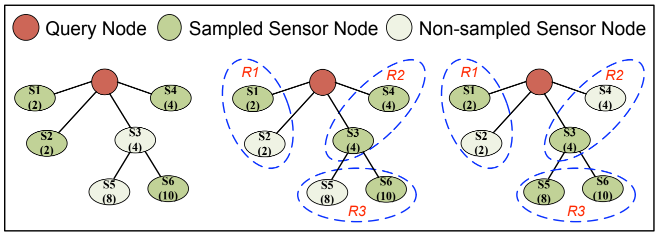

Fig. 1. Data sampling in sensor networks

of 4.5 and 10% relative error. We can improve our estimation if we realize that some sensors' readings are similar. Suppose we group the sensors together as follows $R1 = \{S1, S2\}$, $R2 = \{S3, S4\}$ and $R3 = \{S5, S6\}$, then randomly sampling from each group gives us samples $\{\{2\}, \{4, 4\}, \{10\}\}$. If we scale each respective group (i.e. we estimate each group as $R1$=\{2, 2\}, $R2$=\{4,4\}, $R3$=\{10,10\}), we can get a new average value of 5.33 reducing the relative error to 6.7%. Taking the example one step further, we can vary the allocation of energy between groups based upon the variance of sensor readings within each group. Since groups $R1$ and $R2$ contain a variance of 0 we need only a single reading from each of them (i.e. $S1$,$S3$). On the other hand, group $R3$ contains a higher variance so we allocate more energy budget which allows for 2 readings for group $R3$ (i.e. $S5, S6$). This gives us a final value of 5 reducing the approximation error to 0.

The distances of sensors from the query node play an important role in determining the energy cost of sending data back to the query node, thus introducing a large variance of sampling energy costs for different sensors. As shown in Figure 1, sensors $S5$ and $S6$ are 2 hops away from the query node and sensors $S1$, $S2$, $S3$ and $S4$ are able to communicate with the query node directly. Therefore, it requires more energy to sample a reading from sensors $S5$ and $S6$ than from other sensors. In order to increase the approximation accuracy, we may prefer to sample the sensors near the query node as we could obtain more samplings from them. But we may also prefer to sample from sensors far away from the query node if the data there are good sampling representatives. Considering both the variance of sampling costs and the variance of sampling goodness at different sensors, the challenge is the distribution of the pre-defined energy budget in such a way that the accuracy of the aggregate approximation over the sensor network is maximized.

Goal of the paper: We investigate the feasibility of continuous approximation of aggregation queries in sensor networks given an energy budget for query processing.

Our Approach: Our approach begins by segmenting the network into several non-overlapping regions and performs sampling within each region. Each region then computes sample statistics, and sends the results back to the query node. The query node combines the partial aggregates from each region to approximate the query result. The sample statistics along with the sampling energy cost for each region are updated at the query node to compute the optimal sampling plan for each region. As the regions offer different sampling statistics and sensor-sampling cost, different sample rates will be assigned to regions in order to maximize the approximation accuracy while constraining the query processing cost within a pre-defined budget. We summarize the contributions of this paper as follows:

- We propose the Bottom_up Partitioning algorithm to partition the sensor network into regions for efficient Region Sampling.
- We propose the Region Sampling algorithm to continuously approximate aggregate queries in sensor networks while constraining the network energy cost within a pre-defined energy budget.

The rest of this paper is organized as follows. We describe related sampling and approximation techniques for sensor networks in Section II and present our problem definition in Section III. We describe our framework in Section IV. We introduce our approach in Section V. We provide extensive experimental evaluations of our technique in Section VI, and finally, in Section VII, we conclude this paper.

II. Related Work

Energy efficiency for sensor networks has been an intense area of research in recent years [7], [8], [9], [10], [5], [11], [12], [13], [14], [15], [16], [17], [18]. Among these techniques, various data collection approaches have been proposed such as *event detection* [19], *in-network aggregation* [5], [6], *distributed compression* [13], *model-based* apporaches [17], [8], and *data sampling* [10], [18].

Several techniques have been proposed to approximate aggregate queries in sensor networks. One such approach is *approximation caching* [12], [16]. The idea is to let the query node store a cached value along with an update threshold for each sensor; when a sensor notices that its value has surpassed the threshold, it sends an update message to the query node. CONCH [16] also provides an efficient Spatio-Temporal Suppression technique in which nearby sensors with small difference in their values are also suppressed from updating. A major problem with *approximation caching* is the large energy overhead of the update message transmission when many sensor readings do not stay within the update threshold. In addition, the energy cost for query processing cannot be constrained, which limits its feasibility for energy constraining applications such as the *continuous query processing* in our problem setting.

Another solution for aggregate approximation is to use statistical model to capture the correlations of readings at different sensors. The *model-based* approach [8] samples a small fraction of the sensor data from the network and utilizes the correlation model to estimate the non-sampled sensor readings. The idea is to find the best set of sensors to sample, such that the overall approximation error is minimized. This best sampling set is computed by greedily selecting the sensor that maximizes the estimation confidence of the correlation model. A drawback of this approach is the large computational overhead and large energy consumption. This is because as the sensor data and their correlations evolve over time, the model must be updated frequently in order to accurately represent the correlation between the sensor data, thus limiting the practical employment of this approach.

Stratified Sampling [20] is an efficient method for sampling from a population. It groups members of the population into several relatively homogeneous subgroups and performs random sampling from each subgroup (stratum) independently. The global aggregate result is a weighted combination of all the partial aggregate results at each stratum. In *Stratified Sampling*, however, it is assumed that the sampling cost of retrieving a value is the same for all the strata, which is not always true in sensor networks. In addition, *Stratified Sampling* assumes uniform sampling within each stratum, which is usually difficult to be guaranteed in sensor network applications as sensors may fail periodically or the communication between sensors may be interrupted by obstacles. All these differences limit the practical application of *Stratified Sampling* in sensor networks.

Most likely the closest work to our own is Best-Effort Cache [12], which is a communication constrained data caching algorithm for synchronization between source data objects and cached copies. However, Best-Effort Cache assumes the sampling cost across all sensors is the same and its goal is to minimize the average update divergence of objects over time, which is different from our problem setting.

III. Problem Formulation

Let S denote a sensor network of n sensors. Each sensor generates a record r_i ($1 \leq i \leq n$) every ϵ seconds and the query node aggregates these records for query processing. Due to the energy limitation of the sensor network, the energy cost for each query processing is constrained by a budget B, which prevents the query node from collecting all the sensor readings and computing the exact aggregation result. Our problem can be described as follows:

Given a network of n sensors, with a communication cost matrix representing the communication distance between any two sensors in the network, and a user-defined energy budget B. Our goal is to partition the sensor network into k distinct regions and determine a sampling schedule that continuously approximates aggregate queries on the sensor network with minimal approximation error while constraining the energy cost for processing each query to the energy budget B.

In the following sections, we will show how to exploit the variance of readings and sampling cost of individual sensors to achieve a more accurate approximation. The basic idea is that regions of sensors that have large variance in sensor readings should be sampled more and the sensors with large sampling cost should be sampled less frequently.

IV. Foundations of our Approach

In this section, we present the principles behind our approach for online aggregate approximation in sensor networks. Specifically, we provide the energy cost model and error-bounded sampling theorems that drive our algorithm.

A. Query Cost Measures

Since the energy consumption of the sensor network is dominated by radio transmissions, we compute the total amount of energy spent on transmission as the primary measurement for the cost of a query. In our setting, a network of n sensors is segmented into k non-overlapping regions, with each region representing a fraction of the sensornet. More specifically, each region has a region-head which samples data records from sensors located in the region, aggregates these samples, and transmits the local aggregates to the query node. We represent the energy consumption from sampling and data transmission for a region C_i as

$$E_{total}(i) = E_{region}(i) + E_{network}(i) \quad (1)$$

where $E_{region}(i)$ is the energy cost to sample data records within the region and $E_{network}(i)$ is the cost to send aggregate results from the region-head to the query node.

According to [16], we can then represent $E_{region}(i)$ as

$$E_{region}(i) = e_{record} \times m_i \times l_i \quad (2)$$

where e_{record} is the average energy overhead of transmitting a sample record between nearby sensors in a hop, m_i is the number of records to sample in region C_i and l_i is the average path length from a sensor in region C_i to the region-head. Similarly, the cost of data transmission between the region-head and the query node can be represented as

$$E_{network}(i) = e_{meta} \times L_i \quad (3)$$

where $meta$ represents the data message transferred from the region-head to the query node, e_{meta} represents the energy cost to transmit $meta$ in a single hop and L_i is the number of hops from the region-head to the query node.

B. Sampling for Average Queries

Suppose there are n_i sensors in region C_i, we can estimate the average of the generated n_i records by sampling $\bar{m}_i$ ($\bar{m}_i < n_i$) records from it ($x'_1, x'_2, ..., x'_{\bar{m}_i}$). The average of the sampled records y'_i ($y'_i = \sum_{j=1}^{\bar{m}_i} x'_j / \bar{m}_i$) can be taken as an estimator of the exact average (y_i) of all the n_i records. We define the sampling error threshold Δ_i as the expected absolute difference between the estimate average and the exact average:

$$\Delta_i^2 = E[(y'_i - y_i)^2] \quad (4)$$

In this paper, we extend the *Cross-Validation* technique [21] to compute the relationship between the error threshold Δ_i and the sample size $\bar{m}_i$. The Cross Validation Error (CVE_i) for the average queries can be computed by randomly bi-partitioning the $\bar{m}_i$ samples into two halves S_1 and S_2 and compute

$$y'_{i1} = \frac{2}{\bar{m}_i} \times \sum_{s \in S_1} s, y'_{i2} = \frac{2}{\bar{m}_i} \times \sum_{s \in S_2} s, CVE_i = |y'_{i1} - y'_{i2}| \quad (5)$$

C. Sampling for Median Queries

Given a set of n_i numbers, the median query is to first sort the numbers increasingly and then finding the middle number of the sorted result. The approximation error for median query is the distance between the true rank of the approximated median and the rank of the exact median (1/2).

$$\Delta_i^2 = E\left[\left(\sum_{j\in C_i, j<med'} \frac{1}{n_i} - \frac{1}{2}\right)^2\right] \tag{6}$$

According to [22], we can also compute the Cross Validation Error (CVE_i) for median queries by bi-partitioning the $\bar{m}_i$ samples into two halves S_1 and S_2, find the median med_1 of S_1 and compute

$$CVE_i = \left|\sum_{s\in S_2, s<med_1} \frac{2}{\bar{m}_i} - \frac{1}{2}\right| \tag{7}$$

With current sample size $\bar{m}_i$ and current sampling statistics CVE_i (computed by (5) or (7)), we can approximate the aggregate more accurately if we sample more records from the region. In order to bound the approximation error by some threshold Δ_i, the number of records we need to sample from the region is represented as follows [22],

$$m_i = \frac{\bar{m}_i}{2} \times \frac{CVE_i^2}{\Delta_i^2} \tag{8}$$

V. Region Sampling Algorithm

In this section we develop an online Region Sampling algorithm, that continuously approximates aggregation queries in sensor networks. The Region Sampling algorithm segments the sensor network into several non-overlapping regions, and computes an optimal network-sampling plan to maximize the query approximation accuracy. Our algorithm works in three phases: 1) Region Construction, 2) Training, and 3) Online Approximation. The region construction phase partitions the network into several regions and selects a sensor in each region as region-head. In the training phase, the region-head collects training data from all the sensors in the region, computes the sampling statistics (i.e. *CVE*) and sends them to the query node. After the training phase, an online approximation process is called every time moment when the sensors generate new records. The query node computes the query result by combining the local aggregates from all the regions and updates the network sampling plan according to the sampling statistics and sampling cost rate in different regions.

A. Region Construction

Once all the sensor nodes have been deployed into the monitoring field, we partition the sensor-embedded area into k non-overlapping cells[1], and define each one of these cells as a region of the sensor network. We then use any energy-aware sensor network algorithm (e.g. the clustering algorithms [23], [24]) to construct an energy efficient network topology for communication between sensors within a region and communication between regions.

[1]Section VI-E.3 will describe a method to find the best value of k experimentally.

After the region construction, each region selects a region-head, which samples data records from the sensors within the newly created region, computes the in-region aggregate result from the collected data and sends the in-region aggregate result back to the query node. Normally the region-head incurs a greater energy cost than other sensor nodes, as it needs to communicate with other sensors more frequently during data sampling and result reporting to the query node. We therefore need to spread the energy expense over all the sensors in the network. We employ a dynamic scheme, where the region-head is randomly rotated among the sensors in the region. We assume that a new region-head is randomly selected within each region after a time interval τ. We also make the assumption (which follows established previous work [23]) that all the sensors in each region are within a few (typically 1-3) hops from each other, and therefore the cost of communicating between two sensors in the same region is constant.

It is easy to see that the partitioning of regions could affect the accuracy of the sampling algorithm greatly. A good partitioning scheme should group in the same region the sensors with positions close to each other and having readings similar to each other. In order to find the optimal partitioning of the monitoring area into k regions, we utilize the training data records collected from each sensor as prediction of the future sensor readings and try to find the optimal region partitioning accordingly. The region partitioning optimization problem can be defined as below.

Problem 1 *Given a network of N sensors, with sensor i having a training record r_i. Also given a distance matrix $D_{N\times N}$ with $D(i,j)$ be the distance from sensor i to sensor j. The goal is to segment the network of N sensors into k regions and apply our Region Sampling algorithm on the regions in such a way that the expected approximation error for the (average) query is minimized.*

$$Minimize \;\; F = E[(Y-Y')^2] = \sum_{1\le i\le k} \frac{n_i^2 S_i^2}{n^2 m_i}, \tag{9}$$

subject to

$$\sum_{1\le i\le k} (e_{record} \times m_i \times l_i + e_{meta} \times L_i) = B$$

where m_i is the sampling size we retrieve from region C_i and S_i^2 is the variance of training data in region C_i, l_i is the average distance between two sensors in region C_i and L_i is the average length from the region-head to the query node.

Problem 1 is a clustering problem and we will provide a heuristic approach to approximate its solution. We use a fast and approximate solution because, a) the exact solution requires large computational overhead at the query node (with time complexity exponential to the network size n); b) the exact partitioning solution is only optimal for training data, and may not be an optimal solution for new sensor data that generated on the fly. Therefore a fast and approximate solution is preferred in our case. Before presenting our approach, let us first consider the following sub problem.

Problem 1.1 *Suppose the regions $C_1, C_2, ...C_k$ have already been created, which segment the network of N sensors into k non-overlapping parts (with cardinality of n_i for region C_i). Each sensor has a training reading r_i. With energy budget B, we approximate the average value of all the sensors' training reading $Y = \sum_{1 \le i \le N} \frac{r_i}{N}$ by performing sampling from each region. The goal is to find the best sampling size for each region in such a way that the expected approximation error is minimized.*

$$Minimize \quad E[(Y - Y')^2] = \sum_{1 \le i \le k} \frac{n_i^2 S_i^2}{n^2 m_i}, \tag{10}$$

$$subject\ to \sum_{1 \le i \le k} (e_{record} \times m_i \times l_i + e_{meta} \times L_i) = B.$$

where the meanings of the notations are the same as in Problem 1. This problem is a constrained optimization problem and it has a close form solution [25] as follows,

$$m_i = \frac{\tau \sqrt{\lambda_i}}{\sqrt{\mu_i} \sum_{1 \le j \le k} (\sqrt{\lambda_j \mu_j})}. \tag{11}$$

where $\lambda_i = (n_i^2 S_i^2)/(n^2)$, $\mu_i = e_{record} \times l_i$ and $\tau = B - \sum_{1 \le i \le k} (e_{meta} \times L_i)$.

Considering Equation (11) and (9), we can rewrite the Optimal Region Partitioning problem as

$$Minimize \quad F = \frac{e_{record}}{n^2} \cdot \frac{\left(\sum_{1 \le i \le k} (n_i S_i \sqrt{l_i})\right)^2}{B - \sum_{1 \le j \le k} (e_{meta} \times L_j)}$$

In this *evaluation function* F, n_i is the number of sensors in region C_i, S_i^2 is the variance of sensor readings in region C_i, l_i is the average distance between two sensors in region C_i and L_i is the average length from the region-head of region C_i to the query node.

As shown in the *evaluation function*, the goal of the region partitioning is to minimize both the variance of sensor readings (S_i) and the average distance between sensors (l_i) within each region. Here we propose a heuristic approach, Bottom_up Partitioning, to approximate the optimal solution quickly. Similar to agglomerative hierarchical clustering [26], the idea is to initially segment the network of n sensors into n regions with each region having a sensor in it. Then $(n-k)$ merging operations of regions are performed. In each merging operation, we first find *Candidate Region Pair* by finding the regions that are next to each other in the network topology (i.e. there is at least one direct edge between a sensor in one region and another sensor in the other region). Then we merge the *Candidate Region Pair* into one larger region if the merging introduces the least evaluation cost F. After the $(n-k)$ merging operations, the resulted k regions are taken as an approximation of the optimal partitioning of the network and are therefore utilized for online query processing. The time complexity of this approach is $O(n^2)$ as at most $n-1$ region pairs (i.e. the number of edges in the network topology which is a Minimum-cost Spanning Tree) are evaluated in each round of merging operation and we perform $(n-k)$ merging operations totally.

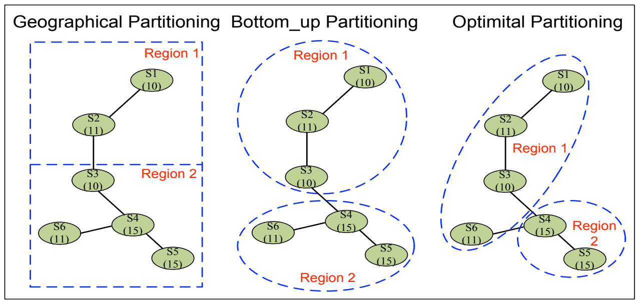

Fig. 2. Region partitioning approaches

Take Figure 2 as an example, Geographical Partitioning segments the area into even-sized geographical cells and assign each cell as a region. It is fast to compute but fails to provide good accuracy for data sampling. The Optimal Partitioning approach considers all the possible region configurations and is able to find the best one that achieves the best sampling accuracy. The computation for Optimal Partitioning, however, is very slow. Our Bottom_up Partitioning algorithm considers only nearby regions for merging (e.g. in the first round of region merging, only sensors $\{S1, S2\}$, $\{S2, S3\}$, $\{S3, S4\}$, $\{S4, S5\}$, $\{S4, S6\}$ are considered as *Candidate Region Pairs* for merging), which is both effective and efficient in computing a good partitioning of the network for data sampling.

B. The Training Phase

When a user initiates a continuous aggregate query, the training phase is triggered to create the initial sampling plan for the online query processing. It works as follows:

1) The region-head collects training data records from the sensors in the region.
2) The region-head computes and sends to the query node the sample statistics (*CVE*) of the training data for the corresponding region.
3) Once the query node receives the sample statistics from all the regions, it computes the initial sampling plan for the network.

After the region construction process, the region-head dispatches a (usually large) fraction of sensors in the region to measure their local environment and generate training data records. Then the region-head collects the training data and computes the *CVE* of them. The size and the *CVE* of the training data are the sample statistics that are transmitted to the query node.

In our settings, the total energy cost to process an aggregate query is upper bounded by B,

$$\sum_{1 \le i \le k} E_{total}(i) \le B \tag{12}$$

Our objective is to create a sampling plan that determines the best sample data size for each region such that the expected approximation error for the query is minimized. Our optimal sampling plan for average queries is constructed upon the following theorem,

Theorem 1: We segment the sensor network S (of totally n sensors) into k regions $C_1, C_2, ...C_k$ ($S = \bigcup_{1\leq i\leq k} C_i$). Each region $C_i (1 \leq i \leq k)$ has $n_i (1 \leq i \leq k)$ sensors and a sampling error threshold $\Delta_i (1 \leq i \leq k)$ for the average query. If we combine the samples from all the regions as the samples of S, the error threshold Δ for the average query approximation in S is

$$\Delta = \sqrt{\sum_{1\leq i\leq k} (n_i^2 \Delta_i^2) \ / \ n^2} \qquad (13)$$

Considering Equation (8) and (13), we can get $\Delta^2 = \sum_{1\leq i\leq k} \frac{\lambda_i}{m_i}$, where $\lambda_i = (n_i^2 \bar{m}_i CVE_i^2)/(2n^2)$.
Taking Equation (1), (2), (3) and (12), we can get

$$\sum_{1\leq i\leq k} \mu_i m_i - \tau = 0,$$

where $\mu_i = e_{record} \times l_i$ and $\tau = B - \sum_{1\leq i\leq k} (e_{meta} \times L_i)$.

Problem 2. *Given λ_i, μ_i and τ, the objective is to find the best value for valuable m_i such that the approximation error Δ is minimized.*

$$\textit{Minimize } \Delta^2 = \sum_{1\leq i\leq k} \frac{\lambda_i}{m_i}, \textit{ subject to } \sum_{1\leq i\leq k} \mu_i m_i - \tau = 0.$$

We can solve this constrained optimization problem by the Lagrange multipliers method [25]. The solution for **Problem 2** has a closed form as follows,

$$m_i = \frac{\tau\sqrt{\lambda_i}}{\sqrt{\mu_i}\sum_{1\leq j\leq k}(\sqrt{\lambda_j \mu_j})}. \qquad (14)$$

C. The Online Approximation Phase

After the training phase, the online approximation process is called every time moment to answer continuous queries over the sensor network. It works as follows:

1) The query node sends out the initial sampling plan to all region-heads.
2) The region-head samples data records in the region according to the sampling plan from the query node.
3) The region-head computes the in-region aggregate and the *CVE* of all samples in the region and then sends them back to the query node.
4) The query node combines the in-region aggregates and outputs the answer.
5) The query node utilizes the reported *CVE* from each region to set up the optimal sampling plan for the next time moment. At the next time moment, the query node sends out the new sampling plan to all the region-heads and continues by returning to step 2.

The Online Approximation Phase uses the sampling plan computed by the Training Phase as the initial sampling plan The sampling plan is then updated every time moment a new round of query execution is issued. The computation of the optimal sampling plan is the same as stated for the training phase. We can compute the optimal sampling plan for each round of query processing by inputting into the sampling plan computation procedure (as described in section V-B) the sample statistics (*CVE*, $\bar{m}$, etc) reported from the region-heads at current time moment.

D. Computational Complexity

Our Region Sampling algorithm described above is implemented at region-heads and the query node. For region-heads, this includes the computation of the *CVE* and the in-region aggregate from the samples (with time complexity $O(m)$, where m is the sample size of the region). The query node computes the query result by integrating all the in-region aggregates and utilizes equation (14) to compute the optimal sampling plan for the next round of query processing. This processing has the time complexity of $O(k)$ where k is the number of regions in the sensor network. Therefore the time complexity of our algorithm is linear to the sampling size for region-head and is linear to the region number for query node, which is pretty cheap and can be easily applied to current sensors with low computational capability.

E. Tolerance to Failures

In wireless sensor networks, the failures may occur so frequently that the reliability becomes a major concern in the designing of any query processing plan. Generally, there are two types of failures in sensor networks: node failures and link failures. The node failures are those failures due to the malfunction of some sensor device. For example, the radio of the sensor is out of order, or the battery of the sensor is depleted. On the other hand, link failures occur where there are some communication failures between two nearby sensors. Link failures happen due to environmental changes or emerging obstacles between the communicating sensors eventually causing loss of messages.

For node failures, we employ a standard heartbeat technique [27] as our failure tolerant strategies. The idea is to store in each sensor a neighbor list indicating the membership of neighboring adjacent nodes. At every time interval τ, every sensor node in the network sends a small heartbeat message to its neighbors. If all the links from some sensor fail, the query node considers the sensor as failed sensor and will revise the network topology by removing the failed node from the network. As the aggregation results from the region-heads are critical for the accuracy of the approximation, we utilize a monitor-and-backup mechanism to recover from region-head failures. The idea is to let the region-head randomly choose a sensor (or multiple sensors) in the region as the backup sensor and it sends a heart-beat message to the backup sensor in each round of query processing. If the backup sensor doesn't hear

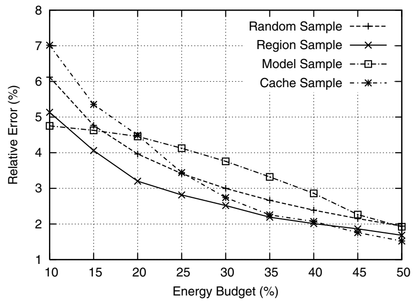

Fig. 3. Varying energy budget for average queries (Intel data).

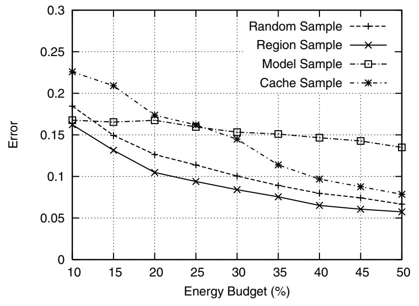

Fig. 4. Varying energy budget for median queries (Intel data).

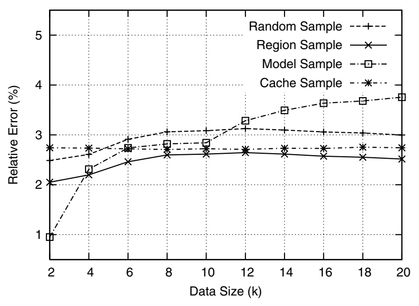

Fig. 5. Varying data size for average queries (B = 30%, Intel data).

from the region-head, it will nominate itself as the region-head and then perform data sampling and result reporting to the sink.

In order to deal with link failures, we use a link redundancy policy to improve reliability of data transmission. The idea is to construct multiple routing trees for reliable data transmission. In addition to the original routing topology, each node randomly adds an alternative path for routing to the region-head. Therefore, more alternative paths will provide more reliability against link failures. Such kind of link redundancy, which depends on the possibility that the sensor generates alternative paths, is effective in overcoming link failures, although it introduces communication overhead and extra energy cost for query processing.

VI. Experimental Evaluation

In this section, we present an extensive performance evaluation of the Region Sampling algorithm and four competing sampling techniques using two real world data sets. Our goal is to demonstrate that our Region Sampling algorithm can efficiently execute approximate queries with high accuracy with a given energy budget B.

A. Data Sets

We use two real world data sets for our experiments:

Intel: This data set is a trace of sensor readings collected from 54 sensors in the Intel Research lab between February and April, 2004.[2] The sensors collected timestamped humidity, temperature and voltage values in 31 second intervals. The data was collected using the TinyDB in-network query processing system, built on the TinyOS platform. We split the data set into non-overlapping training and test data sets (with 1/20 used for training), and create the initial query plan from the training data.

Weather: This data set consists of atmospheric data collected from 32 sites in the weather stations of Washington and Oregon. Each of the 32 sites maintains the average temperature on an hourly basis between 2003 and 2004. We use the data from the first 30 days as the training data set and the remaining as test traces.

[2]http://berkeley.intel-research.net/labdata/

B. Comparison with Techniques

We evaluate the performance of Region Sampling and four other query processing strategies as follows,

Random Sampling: This is a simple sampling strategy that samples data records from each sensor node with the same probability.

Model Sampling: The BBQ algorithm [8] constructs a statistical model based upon the correlation of data between sensors. It estimates readings at the non-sampled sensors according to the values at the sampled sensors.

Cache Sampling: The Best-Effort Cache algorithm [12] assigns to each object a priority according to its divergence from the cached copy. With sampling constraints, Cache Sampling only updates the objects with highest priorities.

Spatio-Temporal Sampling: This is the CONCH algorithm [16] that utilizes a spatio-temporal suppression technique to filter unnecessary messages in the network.

C. Performance Evaluation

In our experiments, the energy budget B (or cost C) is defined as the percentage of energy in relation to the total amount of energy required to sample every sensor in the network. In the following subsections, we evaluate and compare the performance of four energy bounding sampling techniques (i.e. Region Sampling, Random Sampling, Model Sampling and Cache Sampling) on the Intel dataset.[3]

1) Varying Energy Budget: We vary the energy budget B for data sampling in each round and compute the average sampling errors of the four sampling techniques. Figure 3 and 4 show that, for both average queries and median queries, all methods provide better approximation accuracy as the energy budget is increased. Region Sampling outperforms Random Sampling and Cache Sampling in most cases. In addition, when a large energy budget is given, Region Sampling outperforms Model Sampling with a smaller estimation error.

2) Varying Sensor Data Size: As the sensor data records are generated and sampled at each time moment, the generated data size at each sensor increases as the time progresses. In this subsection, we quantify the average approximation accuracies of different sampling techniques with varied sensor

[3]We don't compare with CONCH here as the energy cost C for query processing cannot be constrained in CONCH.

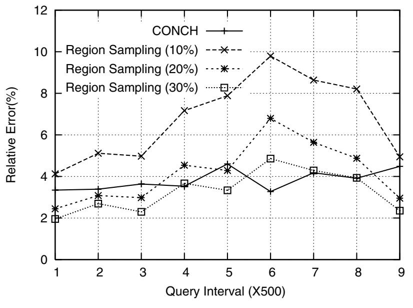

Fig. 6. Approximation errors at different time Intervals (Weather Data).

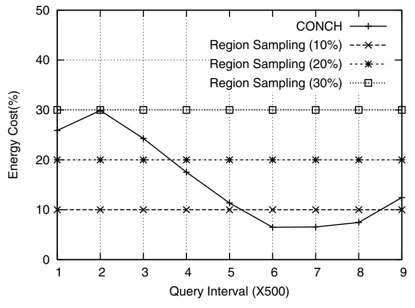

Fig. 7. Query energy costs at different time intervals (Weather Data).

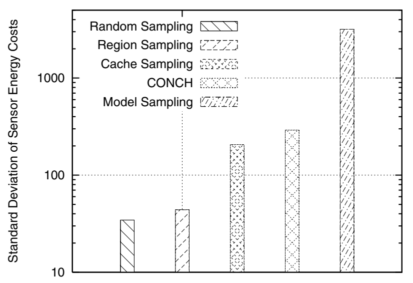

Fig. 8. Energy costs at different sensors (Weather Data).

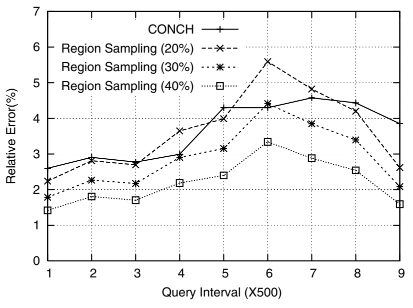

Fig. 9. Query approximation errors at different time intervals (Noisy Weather Data).

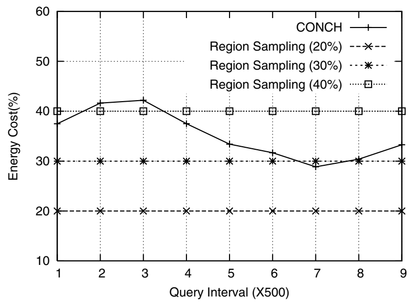

Fig. 10. Query energy costs at different time intervals (Noisy Weather Data).

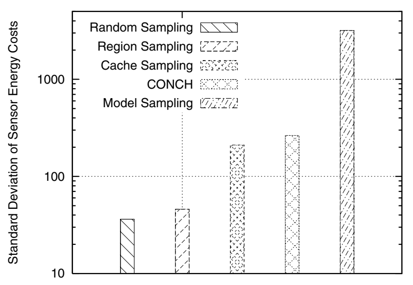

Fig. 11. Standard deviation of energy costs at different sensors (Noisy Weather Data).

data sizes (i.e., we compute the average approximate errors for the first 2000 tuples, the first 4000 tuples, etc). Figure 5 shows the approximation accuracies with different data sizes given the fixed network energy budget B as 30%. It can be concluded from these figures that Region Sampling always outperforms Random Sampling and Cache Sampling. Region Sampling outperforms Model Sampling when time progresses (i.e. when the data size is larger). In addition, the performance of Model Sampling degrades dramatically as time progresses. This is because the correlation model is computed once in the initialization phase and becomes stale as time progresses. Our approach avoids this issue by reevaluating the state of the regions for each round of query processing.

D. Energy Bounded Region Sampling

In the next experiment, we compare the performance of our Region Sampling algorithm with the error-bounding technique CONCH [16]. A very nice property of CONCH is that the approximation accuracy can be easily bounded given the update threshold. This fact is experimentally validated in Figure 6 and 9 where we evaluate the approximation error of CONCH (with a fixed update threshold) and three Region Sampling plans (with different energy budget rates). As shown in Figure 6 and 9, CONCH achieves a similar approximation error for different time intervals and always achieves an error of less than 5%, which is within the accuracy guaranteed by the update threshold.

The disadvantage of CONCH is that it cannot predict or constrain the energy cost it will spend in the network, which makes it unsuitable for the settings where the total energy cost for each query has to be constrained. As shown in Figure 7 and 10, the energy cost C (as defined in Section VI-C) of CONCH fluctuates greatly (i.e. 7%$\sim$30% in Figure 7 and 29%$\sim$42% in Figure 10) as time progresses making it infeasible to reliable bound the energy consumption. In contrast, Region Sampling plans always effectively constrain the energy costs within the pre-defined budgets.

Figures 6 and 7 suggest that with the same energy cost, CONCH is better than Region Sampling. This is because the Weather data set shows much stronger temporal correlation (that CONCH takes advantage of) than spatial correlation (that Region Sampling addresses). In those cases where sensor data have less temporal correlation (as in the case of Noisy weather data set which is generated by adding temporal noise into the Weather data set), Region Sampling outperforms CONCH in approximation accuracy. This fact is validated by Figure 9 and 10.

Finally we want to find how the sampling energy costs is distributed between the different sensors. With 32 weather sensors each having 4500 data records, we first fix the update threshold (5%) and run CONCH algorithm to find the total network energy cost C of CONCH (which cannot be predicted or bounded beforehand). We then use this cost C as the energy budget for other approaches and try to evaluate how the energy cost C are distributed at different sensors. It can be concluded from Figure 8 and 11 that Random Sampling is able to balance energy spending in the network in the best way as it always samples sensors randomly. Region Sampling, which increases the approximation accuracy of Random Sampling, is also able

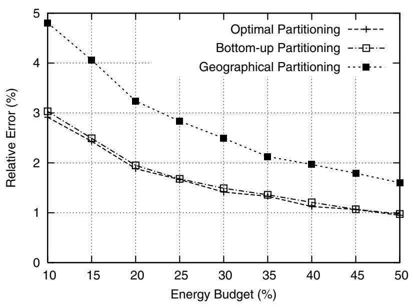

Fig. 12. Varying energy budget.

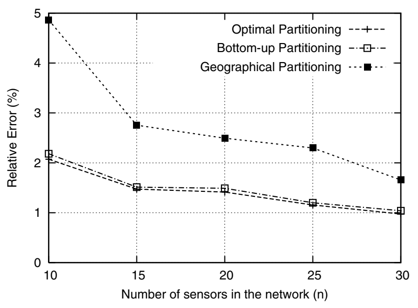

Fig. 13. Varying network size.

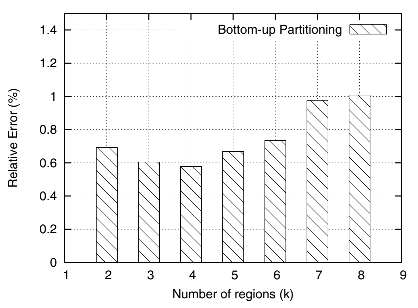

Fig. 14. Varying region number.

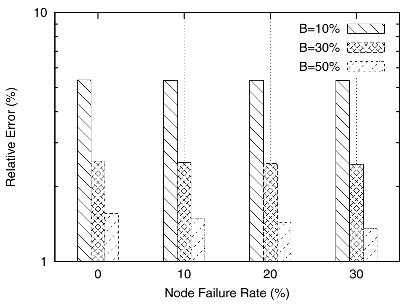

Fig. 15. Varying node failure rates and query energy budget B.

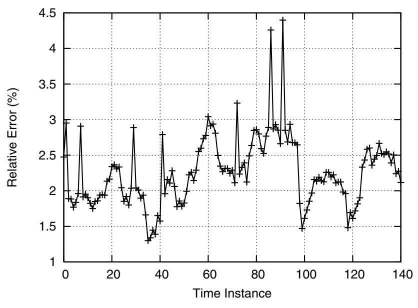

Fig. 16. Recovery from region-head failures.

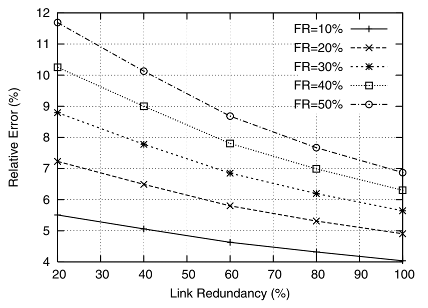

Fig. 17. Varying link redundancy and link failure rate (FR).

to balance the energy costs at different sensors very well. The Cache Sampling and CONCH approaches are not efficient in energy balancing as they only samples those sensors with high priorities or with new reading surpassing the threshold. Model Sampling, which always accesses the best sampling sensor set (computed by the correlation model in the initialization phase) and never samples other sensors, provides the worst energy balancing among different sensors.

E. Impact of Region Partitioning

As stated in Section V-A, the partitioning of regions affects the performance of our Region sampling algorithm. In this subsection, we compare the performance of three region partitioning algorithms, *Optimal Partitioning*, *Bottom_up Partitioning* and *Geographical Partitioning*, for average queries on the Intel data set with the same network settings. As we have described in Section V-A, *Optimal Partitioning* performs brute-force enumeration of all possible partitioning of regions and find the one with the smallest evaluation value. *Bottom_up Partitioning* keeps evaluating and merging small nearby sub-regions into a bigger region until the total region number (k) has been reached. *Geographical Partitioning* segments the network into k even-sized geographical cells and takes each cell as a region.

1) Varying Energy Budget: We vary the energy budget for data sampling and compute the average sampling errors of the three partitioning techniques. We can conclude from Figure VI-C.2 that, the sampling errors decrease for all three algorithms when the sampling budget increases, as more samples can be retrieved from the network for more accuracy approximation of the all the sensor data. *Optimal Partitioning* and *Bottom_up Partitioning* outperform *Geographical Partitioning* in all cases. This is due to the fact that they consider both network topology and data variance in the region construction process.

2) Varying Network Size: We also evaluate the performance of three partitioning algorithms with varied network size. Figure 13 shows the approximation errors of these techniques with a fixed networks energy budget of 30%. It can be concluded that the performance of *Bottom_up Partitioning* is close to that of *Optimal Partitioning* and they both greatly outperform *Geographical Partitioning*. This validates our remark that *Bottom_up Partitioning* is efficient for large sensor networks.

3) Varying Number of Regions: We evaluate the performance of *Bottom_up Partitioning* with varied number of regions k. Figure 14 shows that with increasing value of k, the approximation error decreases and then increases. This can be explained by considering both the in-region sampling budget E_{region} and the meta data overhead between region-heads and the query node $E_{network}$. With increased k, the average region size becomes smaller and the average distance between a sensor to its region-head is smaller. Therefore it is cheaper to sample a reading from a sensor and we can sample more readings to increase approximation accuracy given the same E_{region}. On the other hand, the overhead of meta data transmission $E_{network}$ increases with larger k. This will decrease the energy budget E_{region} that can be used for in-region sampling. The choice of best region number k has to deal with the tradeoff between the average in-region sensor distance and overhead of meta data transmission.

After evaluating the approximation accuracy of *Bottom_up Partitioning* with different values of k, we choose the best k that provides the minimal approximation error. Then we use the corresponding partitioning associated with the best k and apply the Region Sampling algorithm to approximate the aggregate queries on the fly.

F. Dealing with Failures

We evaluate our Region Sampling algorithm in case of both node failures and link failures. Figure 15 shows the approximation errors of average queries for varying node failure rates and different energy budgets. It can be concluded that with increasing node failure rate, the approximation error decreases under the same energy budget. This is because the number of "live" (non-failure) sensors decreases as failure rate increases, and on the other hand, the sample size does not change as the energy budget B is fixed. Therefore, the sample will include a larger fraction of the "live" sensors' data and approximate the query more accurately. Such effects of the node failures are more obvious when B is large, which is validated in Figure 15.

In addition to node failure, we also evaluate the performance of our region-head backup mechanism as described in Section V-E. We set the failure rate for the region-head as 5% and run our Region Sampling algorithm for 140 consecutive queries. As shown in Figure 16, whenever there is a region-head failure, the approximation error will increase sharply as the sink cannot receive the local aggregate result from the failed region-head. After that the impact of region-head failure is recovered effectively as the backup sensor would replace the failed sensor as a new region-head and the sampling task is performed efficiently.

Finally, Figure 17 shows that our link redundancy mechanism can deal with link failures efficiently. With increased link redundancy, we can recover from link failures more effectively. However, more link redundancy introduces more energy consumption (e.g. With 30% link redundancy, 30% more energy are required for data transmission) and thus the choice of the redundancy is a tradeoff between the energy consumption and the reliability of data transmission.

VII. Conclusions

We proposed Region Sampling, an efficient online sampling algorithm for those applications where the energy cost for query processing has to be constrained by a pre-defined budget. Unlike previous works, Region Sampling segments the sensor network into several non-overlapping regions and performs sampling and local aggregation in each region respectively. The sampling energy cost rate and statistical information from past queries are collected and analyzed to compute the optimal sampling plan for future queries. Comprehensive experiments on real world data sets show that Region Sampling is both efficient and practical for approximating continuous aggregate queries in sensor networks.

Acknowledgment

The work of D. Gunopulos was supported by grants NSF IIS-0534781 and Health-e-Child. The work of G. Das was supported by unrestricted gifts from Microsoft Research and start-up funds from the University of Texas, Arlington.

References

[1] F. Simjee, D. Sharma, and P. H. Chou, "Everlast: long-life, supercapacitor-operated wireless sensor node," in *SenSys*, 2005, p. 315.
[2] W. Tsujita, H. Ishida, and T. Moriizumi, "Dynamic gas sensor network for air pollution monitoring and its auto-calibration," in *IEEE Sensors*, 2004, pp. 56 – 59.
[3] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA*, 2002.
[4] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, and M. Sarrafzadeh, "Wireless sensor networks for health monitoring," in *Mobiquitous*, 2005, pp. 479–481.
[5] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.
[6] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Rec.*, 2002.
[7] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *ICDE*, 2004.
[8] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *VLDB*, 2004.
[9] A. Deshpandey, S. Nathz, P. B. Gibbons, and S. Seshan, "Cache-and-query for wide area sensor databases," in *SIGMOD*, 2003, pp. 503–514.
[10] Y. Kotidis, "Snapshot queries: Towards data-centric sensor networks," in *ICDE*, 2005.
[11] C. Olston, B. T. Loo, and J. Widom, "Adaptive precision setting for cached approximate values," in *SIGMOD*, 2001.
[12] C. Olston and J. Widom, "Best-effort cache synchronization with source cooperation," in *SIGMOD*, 2002, pp. 73–84.
[13] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense sensor network," *IEEE Signal Processing Magazine*, vol. 1, 2002.
[14] A. Sharaf, J. Beaver, A. Labrinidis, and K. Chrysanthis, "Balancing energy efficiency and quality of aggregate data in sensor networks," *The VLDB Journal*, vol. 13, no. 4, pp. 384–403, 2004.
[15] A. Silberstein, R. Braynard, C. S. Ellis, K. Munagala, and J. Yang, "A sampling-based approach to optimizing top-k queries in sensor networks." in *ICDE*, 2006, p. 68.
[16] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: on energy-efficient continuous monitoring in sensor networks," in *SIGMOD*, June 2006.
[17] M. Li, D. Ganesan, and P. J. Shenoy, "Presto: Feedback-driven data management in sensor networks," in *NSDI*, 2006.
[18] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *IPSN*, 2004, pp. 124–133.
[19] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *ACM Trans. Networking*, vol. 11, no. 1, pp. 2–16, 2003.
[20] W. G. Cochran, *Sampling Techniques.* John Wiley & Sons, New York, NY, 1977.
[21] S. Chaudhuri, G. Das, and U. Srivastava, "Effective use of block-level sampling in statistics estimation," in *SIGMOD*, 2004, pp. 287–298.
[22] B. Arai, G. Das, D. Gunopulos, and V. Kalogeraki, "Approximating aggregations in peer-to-peer databases." in *ICDE*, 2006.
[23] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS*, vol. 8, 2000.
[24] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. on Mobile Computing*, vol. 03, no. 4, pp. 366–379, 2004.
[25] D. P. Bertsekas, "Constrained optimization and lagrange multiplier methods," *Academic Press*, 1982.
[26] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.
[27] S. C. Wang and S. Y. Kuo, "Communication strategies for heartbeat-style failure detectors in wireless ad hoc networks," in *DSN*, 2003, p. 361.