Deterministic Data Reduction in Sensor Networks

Hüseyin Akcan
Computer & Information Science Department
Polytechnic University, Brooklyn, NY 11201
hakcan01@cis.poly.edu

Hervé Brönnimann
Computer & Information Science Department
Polytechnic University, Brooklyn, NY 11201
hbr@poly.edu

***Abstract*— The processing capabilities of wireless sensor nodes enable to aggregate redundant data to limit total data flow over the network. The main property of a good aggregation algorithm is to extract the most representative data by using minimum resources. From this point of view, sampling is a promising aggregation method, that acts as surrogate for the whole data, and once extracted can be used to answer multiple kinds of queries (such as AVG, MEDIAN, SUM, COUNT, etc.), at no extra cost. Additionally, sampling also preserves the correlation info within multi-dimensional data, which is quite valuable for further data mining. In this paper, we propose a novel, distributed, weighted sampling algorithm to sample sensor network data and compare to an existing random sampling algorithm, which to the best of our knowledge is the only algorithm to work in this kind of setting.**

I. Introduction

In this paper, we focus on sampling from a set of sensor nodes linked by a network and consider the problem of extracting answers to queries about conditions inside the sensor network. These queries may be based on snapshots, or may be continuous. The data collected by the sensors is usually highly redundant, and thus one need not collect and process all of it, approximate query results are usually sufficient. Hence, substantial savings may be obtained by either aggregating or processing the data in-network, or by obtaining a much smaller but representative sample which can then be processed by a centralized more powerful unit.

One particular kind of data we focus on is the multi-dimensional count data that arises fairly often from market basket data or census-like applications, or in the context of sensor networks, from monitoring several parameters of an environment. We contend that collecting and maintaining a representative sample of the data from the sensor network is a promising solution for this problem because samples, unlike other synopsis structures, are general-purpose and can be used as a surrogate for the (expensive) network[2]. In other domains, sampling has long been used to answer approximately many aggregation queries[7] such as MEDIAN, AVG, COUNT, and MODE.

Although random sampling is an easy and intuitive way to answer approximate queries, random deviations in the sampling process, however, make random sampling somewhat less precise and unpredictable. In the context of transactional data sets, a simple deterministic procedure (EASE [3], refined in Biased-L2 [1]) produces samples whose error is consistently an order of magnitude better than that of a random sample. In the context of sensor networks, we find that this translates into order-of-magnitude communication and energy savings, namely, for an equivalent RMS error, our deterministic sample is much smaller thus requires lower energy costs (including extra communication requirements of the algorithm). Alternatively, for the same resource spending, one gets a much more accurate picture.

We give an overview of the previous work in Section II. In Section III, we present an arbitrary aggregation structure for sampling, discuss the motivation for weighted sampling in this kind of aggregation structures and finally present our deterministic sampling algorithm. In Section IV we study experimentally our algorithm by comparing to other (non-deterministic) sampling algorithms, while focusing on issues such as sample quality, energy and communication costs. Finally, we conclude in Section V with both discussion and future work.

Our Contributions

- We propose a novel deterministic weighted sampling algorithm as a new aggregation method for sensor network data. To the best of our knowledge, this is the first distributed deterministic sampling algorithm for sensor networks.
- Our algorithm weights the samples and adjusts the weights dynamically, which enables to work on networks organized in any arbitrary topology.
- In our deterministic weighted sampling algorithm, data aggregation via sampling is done on all (participating) nodes, which equally distributes sampling work over the network, and prevents any node from being a bottleneck (both CPU and communication based).

II. Related Work

In this section, we give an overview of the previous work and state the relations and differences compared to our work. In [1], Biased-L2 is presented, which improves on EASE[3], [4] to deterministically sample streaming count data. Both algorithms sample by introducing penalty functions using the support of each item. EASE and Biased-L2 are designed to work on centralized database settings, where data is stored in a database or comes from a single stream, so they are not directly applicable to sensor network settings, where data is distributed and network topology is unknown. In our paper, we propose a novel deterministic sampling algorithm that uses similar ideas to Biased-L2, but extended to handle weights and distributed sources, essential for sensor network data extraction.

Recently, data aggregation in sensor networks attracted great attention from the research community. In sensor networks, where the in-network processing of various aggregate queries is paramount, data aggregation inside the network could drastically reduce the communication cost (consequently, prolong the battery life) and ensure the desired bounds on the quality of data. Madden *et al.*[12] propose an effective aggregation tree (TAG-tree), which works on well defined epochs, and reduces the communication by using an optimum tree structure. We also used a tree generation algorithm similar to TAG-tree, where iterative broadcast from sink to the rest of the network is used to create a tree that covers the whole network.

Although the tree structure is optimal for communication, it is not robust enough for sensor networks. The robustness issue led researchers to propose a DAG architecture instead of an aggregation tree. Addressing the duplication problem of the DAG architecture, Considine *et al.*[6] and Nath *et al.*[14] independently proposed using duplicate-insensitive sketches for data aggregation in sensor networks. While Considine *et al.* focus only on efficiently counting SUM aggregates, Nath *et al.* give a general framework of defining and identifying order and duplicate insensitive (ODI) synopsis, including a uniform random sample (DIR sample as we call it in this paper). In our paper, we compare our algorithm to DIR, which computes a uniform random sample of the sensor data on any arbitrary hierarchical network structure, using an optimum number of messages.

Manjhi *et al.*[13] by combining the tree and multi-path aggregation structures, propose a tributary-delta structure, that benefits from both. Both structures exist together, and convert to each other, so the tributary-delta can behave as effective as a tree and as robust as a multi-path, depending on the need and network topology. Shrivastava *et al.* propose an aggregation technique for medians in [15], specifically for sensor networks. Greenwald and Khanna [8] extended their space-efficient order statistics algorithms for sensor networks.

In their work, Heinzelman *et al.*[9] and Chen *et al.*[5] present energy formulas for wireless transmission and receive. According to [5], the energy usage during idle::receive::transmit is respectively 1::1.2::1.7. We also used the same energy formulas in our evaluations.

III. Distributed Deterministic Sampling

In this section we discuss the shortcomings of the existing deterministic sampling algorithms for arbitrary sensor network topologies, and present the necessities for using weights in the sampling algorithm. Later we present our Deterministic Weighted Sampling (DWS) algorithm, which is designed to run on distributed environments such as sensor networks. The simplicity and effectiveness of DWS is most appropriate to run on resource-restraint hardware such as sensor nodes. In this version, the notation required to understand the algorithms are omitted for the sake of brevity, and can be found in [1].

A. Aggregation data structure.

In this setting, the data is distributed over the nodes, hence each node x holds a subset D_x of D such that $\cup_x D_x = D$. (In the extreme case, each node holds a single value, which may be updated over time.) We assume that an aggregation tree structure is already available for our algorithms. We do not require any specific property on the tree, other than its connectedness (it must cover all the nodes). Once the aggregation tree is built, sensor nodes, starting from the leaves of the tree, create a sample of size $s = \alpha n$ from their data, and forward these samples to their parents. Nodes in the middle levels of the tree wait until they gather samples from all their children (or for a timeout), and then do sampling on all the gathered data, including their own, to create a sample of size s. This sampling scheme is similar to the DIR [14], and maintains a sample of size s for every node in the network.

B. Motivation for weighted sampling

One important challenge here is that, since we work on arbitrary aggregation trees, and the tree topology changes with the underlying sensor network topology, each node in the tree has an arbitrary number of children. In particular, the sampling rate on each node varies depending on the number of children. Simply gathering all the children's samples in a big chunk of data and

DWS(D, W, x, α)

1: $w_x \leftarrow 1/(\alpha \cdot W_x)$
2: **for** each child y **do**
3: $\quad w_y \leftarrow W_y/(\alpha \cdot W_x)$
4: **for** each item i **do**
5: $\quad n_i \leftarrow r_i \leftarrow 0$
6: **for** each record j in $D_x \cup (\cup_y D_y)$ **do**
7: $\quad sum_n \leftarrow 0, sum_r \leftarrow 0$
8: $\quad$ **for** each item i in j **do**
9: $\quad\quad sum_n \leftarrow sum_n + n_i$; $sum_r \leftarrow sum_r + r_i$
10: $\quad\quad n_i \leftarrow n_i + w_j$
11: $\quad R \leftarrow sum_r - \alpha \cdot sum_n$
12: $\quad K \leftarrow 2 \cdot \alpha \cdot w_j - (2 \cdot R)/\text{size}(j)$
13: $\quad$ **if** $K > 1$ **then**
14: $\quad\quad$ Insert j into the sample S_x
15: $\quad\quad$ **for** each item i in j **do**
16: $\quad\quad\quad r_i \leftarrow r_i + 1$
17: **return** (S_x, W_x)

Fig. 1. The Deterministic Weighted Sampling algorithm

deterministically or randomly sampling over this chunk would introduce misrepresentations in the sample of a node. Namely, the sample values of nodes closer to the sink in the tree would have more chance to appear in the final sample. This is the main reason existing centralized algorithms (EASE, Biased-L2) don't work on sensor networks. To overcome this difficulty in our algorithm, we introduce weights for each sample, which are simply the representation of how many other nodes this sample stands for. Deterministically sampling the gathered data using the weights, we guarantee that each node's data has the same chance to belong to the final sample, independent from its provenance in the network.

C. Deterministic Weighted Sampling (DWS)

We discussed the motivation for weighted sampling in distributed environments in the previous section. In this section, we present the main weighted sampling algorithm. DWS (pseudo-code given in Figure 1) handles weighted sampling at a node x, where the data comes locally from D_x with a weight of 1, and otherwise from the samples in its children y, each with a weight W_y. Thus $W_x = 1 + \sum_y W_y$, and these weights can be accumulated in the network at a very low cost. These weights are normalized by αW_x as in lines 1–3, and we abuse the notation slightly to use $w_j \leftarrow w_y$ for each record j coming from source y.

For each item i, the algorithm uses $Q_i = (r_i - \alpha n_i)^2$ as the penalty function. The total penalty is $Q = \sum_{i \in I} Q_i$. For each $i \in j$, accepting a record j increases r_i by 1 and n_i by w_j, while rejecting a record only increases n_i by w_j, where w_j is the weight of the record j in dataset. During sampling, each record is added to the sample with a weight of 1. After sampling the record weights are updated to reflect the total weight of that sample. For sake of space, the details of calculating the acceptance rule used in Figure 1 are omitted here.

As the communication costs are the dominating factor in sensor networks, we present the performance of the algorithm in number of messages. Assuming each sample value fits in a single message, the sample size of each node is s, and the total number of nodes is m, our algorithm performs a full sampling with $O(sm)$ messages. The memory requirement of the algorithm is $O(m + \alpha n T_{avg})$, based on storing the counts for each item, weights for each record and the raw sample. Practically speaking, if we assume we are using a dataset with 5 tuples per record, we need to use 6 integers (24 bytes) to store a record with its weight. Typically, keeping a sample of 1000 records requires 24 Kbytes. If we assume each tuple has granularity 100, we need 2 Kbytes to store integer counts for the items. Since a typical sensor node (ex. Berkeley Motes) currently has 128 Kbytes of memory, we assume these are reasonable memory requirements.

IV. Experiments

In the experiments section, we demonstrate our work using the wireless sensor network simulator Shawn [11].

In Section IV-A we give the simulation results of our deterministic algorithm (DWS), and compare the results to other algorithms such as naive random sampling and Duplicate Insensitive Random (DIR) sampling. We show that our algorithm has many advantages such as better sample quality and less communication rate compared to other methods.

All energy usage calculations in the experiments are based on total number of sent and received messages. Using the weights from[5], the received messages are weighted with 1.2 and the sent messages are weighted with 1.7 and energy usage for a node is the total. Each tree building message is calculated as a single message, also when exchanging samples between nodes, each record is assumed to fit in exactly one message, also the weights of the whole sample are assumed to fit in one message.

A. Distributed Data Reduction

In this section, we give the simulation results of our Distributed Weighted Sampling algorithm on sensor networks. The evaluation is based on two category, the sample quality and the energy usage. In order to demonstrate our work, we also include the naive random sampling algorithm and a more advanced, Duplicate Insensitive Random (DIR) sampling algorithm from Nath *et al.*[14].

The synthetic dataset (T10I6D1500K), which is an association rule dataset, is created using the IBM data

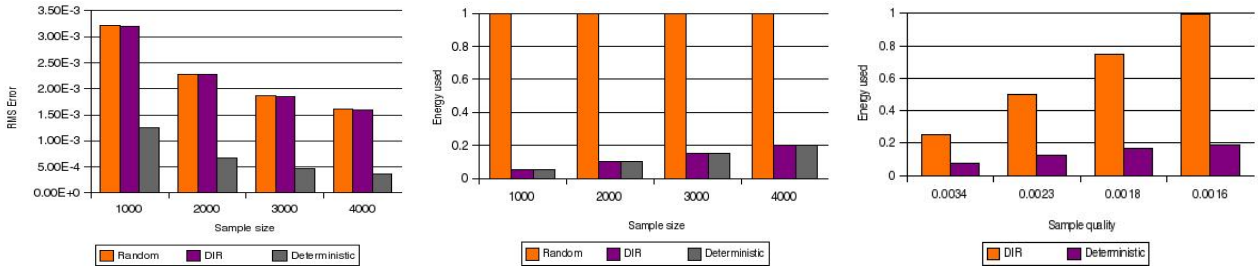

Fig. 2. Results for synthetic dataset. (left) RMS results vs. sample size, (center) energy usage vs. sample size, and (right) energy usage vs. sample quality.

generator [10]. The synthetic dataset includes 1,500,000 records, each having different number of items. Each item represents a different sensor reading, and a record represents a total reading from a sensor, with various items. For simulations, we used 227 nodes, randomly deployed in a 60x25 area, sink on the center.

The simulation results show the averages of running DWS and random algorithms on synthetic dataset 100 times. Figure 2 (left) shows the RMS error values of the final sample generated by the algorithms for four different sample sizes 1000, 2000, 3000 and 4000. We can see that our DWS algorithm generates up to 3 times better quality samples than the other two algorithms.

Figure 2 (center) shows the total energy used in the network while generating the samples. The energy usage is calculated based on the total number of messages sent and received as described in Section IV. Energy usage results are scaled for a clear presentation. As expected, the energy usage of DIR and DWS are the same, and are superior to that of the naive random sampling.

To clearly compare DIR and DWS algorithms, in Figure 2 (right), we generate two samples having the same quality based on RMS error values and compare the energy used by both algorithms. Here, DIR algorithm still uses samples of sizes 1000, 2000, 3000 and 4000. The sample sizes of DWS algorithm are 314, 505, 685, and 766. From the figures it is clear that, we can have the same quality sample by using considerably less energy if we use our DWS algorithm.

V. Concluding Remarks and Future Work

We have presented a novel deterministic weighted sampling algorithm as a new aggregation method for network of wireless sensors. Deterministic Weighted Sampling is simple enough to not consume too many resources locally, and we validate through experiments that the sample it provides is vastly superior to other distributed sampling methods exist for sensor networks. Our algorithm is designed to work on arbitrary network topologies, by introducing weights for samples and dynamically updating these weights throughout the sampling. DWS by design effectively distributes the aggregation work over all the nodes by enabling each node to generate a fixed sized sample and prevents any node from being a bottleneck.

One criticism of our approach is that loss of connection in the aggregation tree structure induced by link or node failure can have drastic effects on the representativity of our sample, since an entire subtree may no longer be contributing to the sample. This can be handled by allowing a multi-path aggregation structure [14], [13]. In the context of aggregation, however, one must then address the problem posed by the duplication of the data along these multi-paths. The duplicate-insensitive solutions provided by Nath *et al.* [14] and Considine *et al.* [6] do not extend to our deterministic algorithm, and we leave it as a matter for future research to handle robust connectivity with our deterministic sampling algorithm.

References

[1] H. Akcan, A. Astashyn, H. Brönnimann, and L. Bukhman. Sampling Multi-dimensional data. *Technical Report TR-CIS-2006-01*, CIS Department, Polytechnic University, February 2006.

[2] D. Barbará, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. E. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik. The New Jersey Data Reduction Report. *IEEE Data Engineering Bulletin* 20(4):3–45, 1997.

[3] H. Brönnimann, B. Chen, M. Dash, P.J. Haas and P. Scheuermann. Efficient data reduction with EASE. *Proc. ACM KDD'03*, pp. 59–68, 2003.

[4] H. Brönnimann, B. Chen, M. Dash, P. J. Haas, Y. Qiao and P. Scheuermann. Efficient data-reduction methods for on-line association rule discovery. Chapter 4 of *Selected papers from the NSF Workshop on Next-Generation Data Mining (NGDM'02)*, pp. 190–208, MIT Press, 2004.

[5] B. J. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy Efficient Coordination Algorithm for Topology Maintenance in Ad-Hoc Wireless Networks. *Wireless Networks* 8(5):481-494, 2002.

[6] J. Considine, F. Li, G. Kollios and J. Byers. Approximate Aggregation Techniques for Sensor Databases *Proc. IEEE ICDE'04*, pp. 449, 2004.

[7] P. B. Gibbons, S. Acharya, Y. Bartal, Y. Matias, S. Muthukrishnan, V. Poosala, S. Ramaswamy, and T. Suel. Aqua: System and techniques for approximate query answering. Technical report, Bell Labs, 1998.

[8] M. B. Greenwald, S. Khanna. Power-Conserving Computation of Order-Statistics over Sensor Networks. *Proc. PODS'04*, pp. 275–285, 2004.

[9] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy Efficient Communication Protocol for Wireless Microsensor Networks. *Proc. of 33rd Hawaii Int. Conf. System Sci.*, p. 8020, 2000.

[10] Intelligent Information Systems. *Synthetic Data Generation Code for Associations and Sequential Patterns*. Research group at the IBM Almaden Research Center.

[11] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. *Proc. Design, Analysis, and Simulation of Distributed Systems (DASD05)*, pp. 117-124, 2005.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong. TAG: A tiny aggregation service for ad hoc sensor networks. *Proc. USENIX (OSDI)*, pp. 131–146, 2002.

[13] A. Manjhi, S. Nath, P. B. Gibbons. Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams. *Proc. SIGMOD'05*, pp. 287–298, Baltimore, Maryland, USA, 2005

[14] S. Nath, P. B. Gibbons, S. Seshan, Z. R. Anderson. Synopsis Diffusion for Robust Aggregation in Sensor Networks. *Proc. SenSys'04*, pp. 250–262 , Baltimore, Maryland, 2004.

[15] N. Shrivastava, C. Buragohain, D. Agrawal. Medians and Beyond: New Aggregation Techniques for Sensor Networks. *Proc. SenSys'04* pp. 239–249, 2004.