

Optimizing Energy-Efficient Query Processing in Wireless Sensor Networks

Ross Rosemark Wang-Chien Lee Bhuvan Urgaonkar
 Department of Computer Science and Engineering
 The Pennsylvania State University
 University Park, PA 16802, USA
 {rosemark, wlee, bhuvan}@cse.psu.edu

Abstract—This paper studies the issues of energy-efficient query optimization for wireless sensor networks. Different from existing query optimization techniques that consider only query plans for extracting data from sensors at individual nodes, our approach takes into account both of the sensing and communication cost in query plans. Central to our study is a cost-based analysis, based on which the energy cost of candidate plans for a given query are estimated to determine a query plan that is likely to consume the least energy for execution. Simulation results show that the query plan chosen in our approach consumes significantly less energy than an approach that optimizes on sensing cost only.

I. INTRODUCTION

This paper focuses on energy-efficient query processing for wireless sensor networks (WSNs). For a given query there exists multiple *query plans*, representing alternatives for retrieving data for the query. Choosing an efficient query plan is known as the problem of *query optimization* in database research. A query optimizer evaluates a set of query plans, choosing one that potentially incurs the minimum number of *disk accesses* [4]. For query processing in WSNs, a query optimizer can be deployed at the access point (AP) to choose a query plan with minimal estimated *energy cost*.

Two primary operations primarily contribute to the energy consumption of WSNs: 1) *sensing cost* - periodical sampling the sensors at the nodes; and 2) *communication cost* - routing queries to involved nodes and collecting results to the AP. Nevertheless, existing query optimizers for WSNs typically generate query plans that only instruct nodes *how to extract data from their sensors* but not taking communication cost into consideration. As shown in existing research [9, 12, 16], there exist multiple routing protocols to route a query to the participating nodes and collect sensor data to the AP. Thus, in this paper, we broaden a query plan to take into account all aspects of query execution, including *routing, sensing and data/metadata collection*.

Traditionally, metadata plays an important role in query optimizers for database management systems. To determine how to extract data from the WSN, global metadata that defines the data profiles (e.g., statistics describing the distribution of the values generated by the sensors) of the nodes participating in the query is also crucial for query optimization. However, to collect metadata from sensor nodes to the query optimizer (i.e. the AP) requires an energy overhead. For queries that consume little energy in execution (referred to as *light queries*), this overhead can be too costly. However, if metadata is not

collected, the query optimizer may not accurately estimate the energy cost of query plans to make a good choice. For queries that consume significant energy in execution (referred to as *heavy queries*), this overhead is a worthy investment.

This paper makes several significant contributions:

- 1) Our query optimizer, to the best of our knowledge, is the first to consider both sensing and communication costs.
- 2) The derivation of a cost formulae for estimation of energy consumption in various operations of query execution provides a concrete basis for estimating the overall energy cost of a query plan.
- 3) To the best of our knowledge, this is the first study that exploits the trade-off between the cost of metadata collection and the improved accuracy of estimating energy costs provided by metadata.
- 4) Extensive simulation validates that our approach utilizes significantly less energy than an approach that optimizes on sensing cost only.

The rest of this paper is organized as follows. In Sections 2, we give an overview of existing work related to our study. We then introduce our query optimization framework in Section 3 and present the derivation of cost models in Section 4. Next, we present in Section 5 simulation results that validates the efficiency of our approach. Finally, in Section 6 we conclude our work and present directions for future research.

II. RELATED WORK

Two well known query processing engines for WSNs are TinyDB and Cougar [13, 14, 17]. In these systems, a query is injected at the AP. Upon receiving the query, the AP collects metadata in an aggregated form from all nodes participating in the query. Based on this metadata, the query optimizer located at the AP selects a query plan that defines the order a node samples its sensors. Next, the AP utilizes a built-in network infrastructure (i.e., a minimum spanning tree) to disseminate the chosen query plan to all nodes participating in the query. Nodes receiving the query execute the query plan and send the resulting data to the AP via the minimum spanning tree.

In sensor networks, it's possible to equip each node with a query optimizer in order to re-optimize long-term query processing at run-time [7, 10]. An aggressive strategy proposed in Eddies [5, 8, 15] combines query optimization and query execution. In this approach, a query plan is defined at runtime on the granularity of a tuple. While this is an interesting direction for future study, our paper considers centralized query optimization at the AP.

III. QUERY OPTIMIZATION

This section discusses our assumptions and provides a high-level overview of our query optimizer.

Assumptions. In this paper, we focus on optimization of a *single query* (Q) which contains p predicates $\mathbb{P}_Q = \{P_1, P_2, \dots, P_p\}$ and involves n nodes, $\mathbb{N}_Q = \{N_1, N_2, \dots, N_n\}$ ¹. Similar to well-known query processing engines (e.g., TinyDB [14]), we assume that nodes are stable (i.e., they do not move or fail) and each node has s sensors, $\mathbb{S} = \{S_1, S_2, \dots, S_s\}$. For each sensor $S_i \in \mathbb{S}$, a node maintains a histogram to capture the distribution of reading values generated by S_i .

Overview. In our study, the query optimizer generates a set of query plans with respect to Q , denoted as $\mathbb{Q}_Q = \{QP_1, QP_2, \dots, QP_m\}$, where m is the total number of query plans generated. A query plan $QP \in \mathbb{Q}_Q$ specifies how to execute the query in the sensor network instead of just specifying how to sample sensors in a node.

There are two stages in our query optimization process:

- 1) **Classification Stage:** Determine if metadata should be collected by evaluating each query plan within \mathbb{Q}_Q . Based on this evaluation, the query optimizer chooses $QP_{min} \in \mathbb{Q}_Q$ that utilizes minimal energy. If QP_{min} requires no metadata collection, it indicates that the extra cost of metadata collection may not be justifiable for the energy saving obtainable from finding a better query plan. Thus, QP_{min} is distributed to each node $N \in \mathbb{N}_Q$, which in accordance with QP_{min} samples its sensors and routes the results back to the AP. On the other hand, if QP_{min} dictates that metadata collection is necessary, our query optimizer enters the second stage as below.
- 2) **Refinement Stage:** Collect metadata to re-evaluate each query plan in \mathbb{Q}_Q choosing QP'_{min} . Finally, QP'_{min} is distributed to each node $N \in \mathbb{N}_Q$, which samples its sensors and routes back the desired tuples to the AP in accordance with QP'_{min} .

In a large network, there exists numerous alternative query plans. Generating and evaluating all query plans is unnecessary [6, 11]. In this paper, our query optimizer only evaluates query plans with the following properties. First, each node transmits its data to only a single neighbor node. Second, to save energy, we stagger communication such that a node far away (in terms of the number of hops) sends its data before a node with fewer hops. This ensures that a node receives data from all its children before transmitting its data. Finally, each node utilizes the *Run Length* [3] compression algorithm to compresses its data before sending it back to the AP.

IV. COST ESTIMATION

In this section, we derive cost formulae for the various operations specified in query plans.

A. Assumptions

A query optimizer estimates the cost of query plans based on the selectivity of predicates $P_i \in \mathbb{P}_Q$ where $i = 1..p$. We assume that for each node N_i where $i = 1..n$, metadata has been maintained as a set of histograms $H_{i,j}$, where $j = 1..s$,

¹We plan on extending our approach to efficiently support multiple queries simultaneously in the future work.

to represent the probability of sensor S_j at node N_i generates a given value. Metadata may be requested by the AP periodically (specified by the system administrator) or on-demand. The frequency of metadata collection is denoted as MCR in our analysis. When metadata is not collected, the query optimizer assumes that each node's histograms are uniformly distributed. The *duration* and *epoch* of a query Q is denoted as D_Q and E_Q , respectively. Additionally, we make the following assumptions: 1) a sensor requires θ Joules to sample, 2) the energy needed to send and receive a bit 1 hop are β and γ joules, respectively. This includes the energy required to startup and shutdown the radio component of a node.

B. Classification Stage

In this section we discuss how to estimate the energy cost associated with a given query plan $QP \in \mathbb{Q}_Q$ at the classification stage.

1) *Metadata Collection:* In this section we estimate the energy associated with metadata collection. The cost of metadata collection is the sum of the energy associated with: a) notifying each node that metadata should be collected, b) collecting the metadata from each node. If the query plan denotes that metadata is not collected, the energy cost is 0.

In our approach, the AP floods a message to each node $N \in \mathbb{N}_Q$ instructing it to return the selectivity of each predicate $P \in \mathbb{P}_Q$. Assuming this message is v bits, then the dissemination cost (denoted as DC) can be estimated as $DC = n * v * (\beta + \gamma)$. Next, each node $N \in \mathbb{N}_Q$ returns the requested metadata to the AP. As discussed, a node receives metadata from all its children before transmitting its metadata. As such, a node will send its own metadata and its children's metadata in one packet to save energy (in terms of powering up and down the radio component only one time). The energy cost incurred by sending metadata at a node is proportional to the size of metadata. Let $MR(N)$ and $MS(N)$ denote the size of metadata received and sent by node N , respectively. A leaf node only sends its own metadata, but an interior node needs to first receive the metadata from its children to send along with its own metadata. Thus, for an interior node N_I that has c children, $N_{C(1)}, N_{C(2)} \dots N_{C(c)}$, the energy cost associated with receiving metadata from its children is: $RC(N_I) = \gamma * \sum_{i=1}^c MS(N_{C(i)})$. The energy associated with sending metadata is: $SC(N_I) = \beta * (MR(N_I))$. In summary, the energy cost to return a node's metadata to the AP is:

$$MC(N) = \begin{cases} MR(N) * \beta & \text{if } N \text{ is a leaf node} \\ RC(N) + SC(N) & \text{if } N \text{ is an interior node} \end{cases}$$

Therefore, the total cost of one metadata collection is: $C_{MC}(QP) = DC + \sum_{i=1}^n MC(N_i)$.

2) *Sensor Sampling:* In this section we estimate the energy cost associated with obtaining sensor readings at nodes. Again, it is important to note that in the classification stage, we try to estimate the benefit of metadata collection to the sensor sampling cost for query plans that specifies metadata collection. This estimation tends to be inaccurate since fresh metadata is not available at this stage. In our approach, the query optimizer utilizes its existing metadata to approximate the metadata to be collected and uses this temporary metadata to figure out the cost of the query plan under consideration.

The approximation proceeds as follows. Let a query plan under consideration, QP , specifies that a chosen node $N \in N_Q$ shall send to the AP its metadata corresponding to predicate $P \in \mathbb{P}_Q$, represented as a list of buckets $\langle B_1, B_2, \dots, B_m \rangle$. Each bucket B in the list consists of: a) $Range(B)$: the range of readings fallen in B , and b) $Prob.(B)$: the probability of a reading fallen within $Range(B)$. The query optimizer approximates these buckets with the metadata it currently has. Utilizing these approximated buckets, the query optimizer estimates the selectivity of a predicate P at node N as: $Selectivity(P, N) = \sum_{i=1}^h Prob.(B_{H(i)})$ where $B_{H(1)}, B_{H(2)}, \dots, B_{H(h)}$ are the h buckets resolving P . Finally, assuming that QP specifies that predicate $P_{R(i)}$ at node N should be resolved before predicate $P_{R(i+1)}$, the energy cost for sensor sampling is:

$$SS(N) = \theta * \sum_{i=1}^p Selectivity(P_{R(i)}, N) * \prod_{j=1}^{i-1} Selectivity(P_{R(j)}, N)$$

Thus the energy associated with each sensor sampling is: $C_{SS}(QP) = \sum_{i=1}^n SS(N_i)$.

3) *Result Reporting*: Here we estimate the energy cost associated with result reporting. The cost of reporting is the sum of the energy a node consumes on: a) receiving the reports generated by its children, b) sending a compressed report by aggregating its own and its children's reports to its parent.

To estimate the cost of result reporting, the AP must first determine the number of *qualified reading tuples* (QRTs) generated by each node. Briefly, a QRT consists of readings that satisfies query predicates. We assume that at each *epoch* a node generates 0 or 1 QRT. The probability that a node N generates a QRT at a given *epoch* can be inferred from the query optimizer's metadata.

Algorithm 1 Determines the probability that a given node N generates a QRT that satisfies the query predicates.

- 1: Determine the set of QRT $\mathbb{T} = \{T_1, T_2, \dots, T_t\}$ that satisfy the predicates.
 - 2: Utilizing the query optimizer's metadata, calculate the probability that node N generates a QRT T_i as $Prob.(T_i)$.
 - 3: return $\sum_{i=1}^t Prob(T_i)/t$ (i.e., average probability that a QRT is generated)
-

Algorithm 1 (referred to as function $PF(N)$) describes how to determine the probability that a given node N generates a QRT. Next, the query optimizer calculates the cost associated with sending/receiving reports, which is proportional to the size of reporting packets sent. Let $RS(N)$ denote the size of reporting packet sent by N . The size of a report sent by a leaf node is: $RS(N) = PF(N)$. Estimating the size of report sent by an interior node is more difficult since it depends on the reports the node receives from its children. Assume that based on QP , a given node N has c children nodes $N_{C(1)}, N_{C(2)}, \dots, N_{C(c)}$. Also assume that a child node N_C sends to node N a data packet of size $RS(N_C)$. The total energy cost for node N to receive reports from its children nodes is: $RRC(N) = \gamma * \sum_{i=1}^c RS(N_{C(i)})$. The total energy cost for node N to send a report to its parent is $RSC(N) = \beta * RS(N)$. In summary, node N consumes the following energy performing result reporting:

$$RC(N) = \begin{cases} PF(N) * \beta & \text{if } N \text{ is a leaf node} \\ RRC(N) + RSC(N) & \text{if } N \text{ is an interior node} \end{cases}$$

Therefore the cost of result reporting in QP is defined as: $C_{RR}(QP) = \sum_{i=1}^n RC(N_i)$.

For a given node N , we now calculate how to determine the size of its report, $RS(N)$. The size of the reporting packet transmitted by N is based on two factors: a) how efficiently the compression algorithm reduces the size of report sent by node N , and 2) the size of the reports received from its children. We assume that the *run length* compression algorithm is adopted. Utilizing this algorithm a node analyzes the aggregate of the reports it generated (if any) and received from its children to replace duplicate tuples with a single instance of the tuple and its frequency. To determine the number of duplicate tuples at node N , the query optimizer calculates the probability that at a given *epoch* the aggregate tuples received/generated by it contains d duplicate tuples. To determine this, the query optimizer first consults its metadata to determine the probability that a node generates a given tuple T . Assume that the probability for a node N to generate a given tuple T is $Prob.(N, T)$. Also assume that based on the query plan, N has c children $N_{C(1)}, N_{C(2)}, \dots, N_{C(c)}$. The query optimizer determines the average probability that N or its children generate tuple T is:

$$Prob.(T) = \frac{Prob.(N, T) + \sum_{i=1}^c Prob.(N_{C(i)}, T)}{c}$$

Next, utilizing this equation the query optimizer determines the probability that N generates and/or receives (from its children) tuple T . To determine this, the query optimizer utilizes the binomial distribution which calculates given $Prob.(T)$ the probability that node N generates or receives the tuple T for k times.

$$Freq(T, k) = \frac{c}{(k!)(c-k)!} * Prob.(T)^k * (1 - Prob.(T))^{c-k}$$

Thus, the number of times tuple t exists in N 's report packet at a given *epoch* E is $Freq(T) = \frac{\sum_{i=0}^c Freq(T, i)}{c}$. Based on this assumption, the query optimizer determines the average number of duplicate tuples in N 's data packet. For each duplicate tuple, it replaces all duplicates with a single tuple and its frequency (as discussed). Assume that we use a 32-bits integer to represent the frequency and two integers (i.e., 64 bits) to represent frequency and tuple. Finally, assume that each node can only potentially generate tuples T_1, T_2, \dots, T_h . The following equation represents the size of report packet (in bits) sent by node N to its parent:

$$RS(N) = \sum_{i=1}^h \begin{cases} 32 \text{ bits} & \text{if } Freq(T_i) = 1 \\ 64 \text{ bits} & \text{otherwise} \end{cases}$$

4) *Building an Infrastructure for Reporting*: Once a query plan QP is determined by the query optimizer, the AP disseminates it to the nodes involved in the query and builds an infrastructure for reporting. For simplicity, we assume that the AP disseminates QP by flooding it throughout the network. In this section we estimate the energy cost associated with building an efficient infrastructure for returning reports. Assume that the size of the QP is denoted as $|QP|$. Based on this assumption each node consumes $(\beta + \gamma) * |QP|$ to disseminate and receive messages for building the infrastructure. Thus, the

total energy cost required for building the infrastructure is:
 $C_{BI}(QP) = (\beta + \gamma) * |QP| * n$.

5) *Total Energy*: The total energy of a $QP \in \mathbb{Q}_Q$ is:

$$C(QP) = (C_{MC}(QP) + C_{SS}(QP) + C_{RR}(QP) + C_{BI}(QP)) * \frac{D_Q}{E_Q}$$

Utilizing this equation the query optimizer calculates the energy associated with each query plan $QP_i \in \mathbb{Q}_Q$ where $i = 1..m$, choosing the best query plan QP_{min} with the least estimated energy cost. If QP_{min} requires an on-demand metadata collection, the query optimizer collects the metadata and enters the refinement stage (see the next section). If QP_{min} specifies no metadata collection, the query plan is disseminated to the nodes for execution.

C. Refinement Stage

In this section we discuss how to estimate the energy associated with a given query plan $QP \in \mathbb{Q}_Q$ in the refinement stage. Since fresh metadata has been collected, new query plans only need to specify how to perform sensor sampling, result reporting, and infrastructure construction. Note that our cost formulae developed for the classification stage are applicable here but our estimates are now based on the newly collected metadata. Hence, the total cost of a query plan QP is:

$$C(QP) = (C_{SS}(QP) + C_{RR}(QP) + C_{BI}(QP)) * \frac{D_Q}{E_Q}$$

Utilizing this equation the query optimizer calculates the energy cost for all query plans to choose the best query plan QP'_{min} with the least estimated energy cost. The query plan QP'_{min} is then disseminated to the nodes for execution.

V. PERFORMANCE EVALUATION

To evaluate our approach we conduct experiments using NS-2 network simulator with the CMU wireless extensions [2].

A. Experimental Settings

All experiments simulate a sensor network comprised of 50 static nodes uniformly distributed in a fixed area (600m x 600m). The readings of sensor nodes are generated by scaling down and fitting the dataset available at The Joint Institute for the Study of the Atmosphere and Ocean (JISAO) [1] to our network nodes. In our experiments, the local metadata of a node, was obtained based on the JISAO dataset. The calculation of energy consumption in our simulation adopts the parameter settings and experimental measurements of [14], which is based on Mica Motes by Crossbow Technology Inc. We also assume that θ , i.e., energy cost for sampling a sensor, is 1500 uJ (default) and metadata is collected once a week.

B. Effectiveness of Classification

The classification component in our query optimizer determines whether on-demand metadata collection is needed. While up-to-date metadata is necessary to accurately estimate the energy cost of query plans, there is an energy overhead for collecting metadata. Thus, only *heavy queries* which consume significant energy performing sensor sampling and result collection, may trade the overhead of metadata collection for an actual energy saving.

There is not an absolute threshold to determine whether a query is heavy or light. For a given query, the following factors

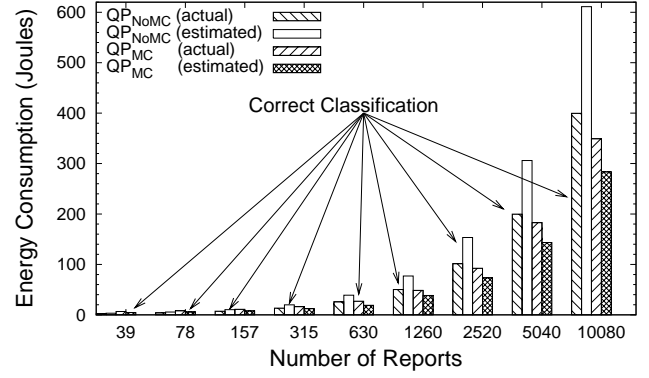


Fig. 1. Impact of Reporting

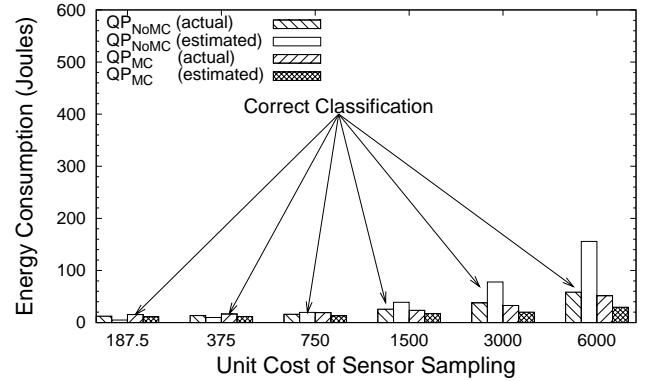


Fig. 2. Impact of Sensor Sampling

significantly impact its total energy consumption: *the number of reportings* and *the unit energy cost of each sensor sampling*. The number of reportings is determined by the *epoch* and *duration* of a query. Since each reporting operation consists of sensor sampling (to generate a report) and delivering a report to the AP, the more reports are generated the higher the total energy cost. On the other hand, the higher unit energy cost of each sensor sampling the higher the total cost. In addition to the above two factors, *the freshness of metadata* a query optimizer currently has also plays a role on classification. If the metadata is fresh, the estimates of query plans are more accurate and thus may not require on-demand metadata collection. Therefore, for each of the three above-mentioned factors, we conducted an experiment to determine its impact on our query optimizer.

In each experiment, we run the classification component to estimate the energy cost of query plans and select the best query plan that dictates: 1) metadata is collected (denoted as QP_{MC}), 2) metadata is not collected (denoted as QP_{NoMC}). As discussed, if the estimated energy cost of QP_{NoMC} is greater than the estimated energy cost of QP_{MC} , our optimizer determined this is a heavy query, which should go ahead to collect metadata and enter the refinement stage of query optimization. To determine the effectiveness of our query optimizer in correctly deciding whether to collect metadata, we compare the actual energy cost of performing QP_{NoMC} and QP_{MC} . If our query optimizer correctly decide a query plan should collect metadata, QP_{MC} should result in less actual energy consumption than QP_{NoMC} .

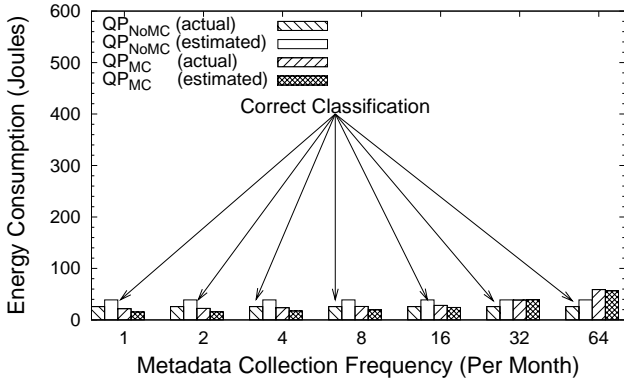


Fig. 3. Impact of Metadata Freshness

We evaluated multiple queries each sampling a various ranges of readings. To save space we only plot figures for the following query.

```

SELECT Temperature
FROM Sensors
WHERE Latitude < 600 and Latitude > 300
and Longitude < 450 and Longitude > 200
and Precipitation < 150 and Temperature < 25 and Pressure < 107.5
EPOCH  $E$  minutes DURATION 1 Month

```

In the first experiment, we compare QP_{MC} and QP_{NoMC} in terms of estimated and actual energy consumption by increasing the number of reports from 39 times per month (i.e., $E = 21$ hours) to 10080 times per month (i.e., $E = 4$ minutes). Basically, the query optimizer classifies the query *correctly* if the decision based on estimated cost is consistent with the actual cost. As illustrated in Figure 1, where the labels of bars are self-explained, our query optimizer correctly decided that the query needs to collect metadata (when the number of reports is greater than or equal to 1260) and correctly decided that the query does not need to collect metadata (when the number of reports is less than or equal to 78). Also shown in the figure, our query optimizer may incorrectly classify the query when the number of reports in the give query is between 157 and 630, due to the relative closeness in energy consumption between the two estimated query plans. Since this error only occurs when the difference (in energy cost) between QP_{NoMC} and QP_{MC} is relatively small, it does not significantly degrade the performance of our query optimizer.

Next, to analyze the impact of sensor sampling cost on effectiveness of the classification, we fixed the epoch at $E = 64$ and vary the unit energy cost for sampling a sensor from 187.5 uJ to 6000 uJ. As illustrated in Figure 2, our query optimizer correctly decided that the query needs to collect metadata (when the sensor sampling cost is greater than or equal to 1500 uJ) and correctly decided the query do not need to collect metadata (when the sensor sampling cost is less than or equal to 375 uJ). The only experiment that our query optimizer did not correctly classified the query is when the sampling cost is 750 uJ. This again does not significantly degrade the performance of our query optimizer.

Finally, we test the impact of metadata freshness on the classification of our query optimizer by assuming metadata

has been periodically collected by the query optimizer. We vary the frequency of metadata collection from 1 time per month to 64 times per month. As illustrated in Figure 3, our query optimizer correctly decided that there is no need to collect metadata when the metadata collection frequency is greater than or equal to 32 and correctly decided there is a need to obtain fresh metadata when the sensor sampling cost is less than or equal to 4. Our query optimizer decided to make an un-necessary on-demand metadata collection when the frequency of periodical metadata collection equals to 8 or 16 per month. However, as shown in the figure, this incorrect decision does not significantly degrade the performance of our query optimizer.

C. Effectiveness of Query Optimizer

In this section, we conducted experiments to evaluate if our query optimizer chooses one of the most energy-efficient query plan for execution. To achieve this, we modified our simulator to execute each query plan in the query space of a given query Q . For each query plan, we obtain both of the estimated and actual energy costs. In this experiment we evaluate over a thousand query plans, however, due to space constraints we only illustrate a subset.

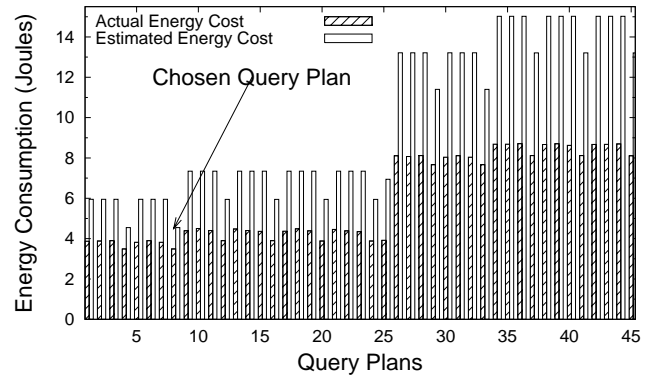


Fig. 4. Light Query

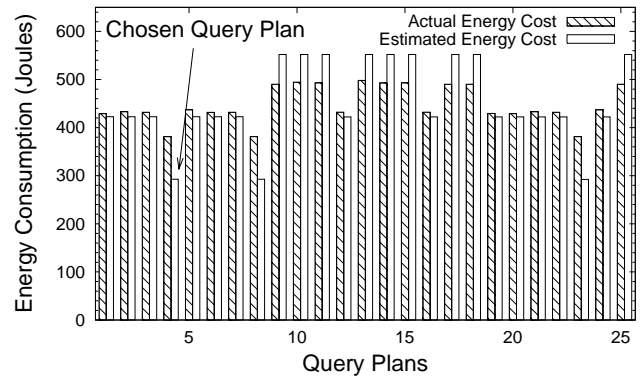


Fig. 5. Heavy Query

In this experiment, we consider two query cases by resetting the above mentioned query with two different *epochs*: 1) $E = 21$ hours and 2) $E = 4$ minutes. The former, a representing a *light query*, is to simulate a situation where metadata collection is too costly, and thus the query optimizer does not enter the refinement stage. The latter, representing a *heavy query*, is to

simulate a situation where metadata collection maybe a worthy investment, and thus the query optimizer performs an on-demand metadata collection to refine the query optimization.

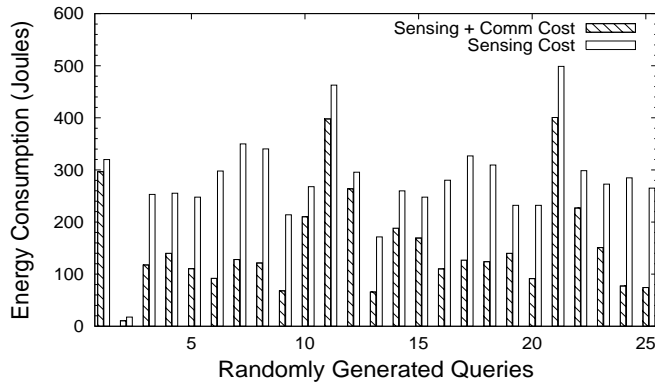


Fig. 6. Targets of Optimization

As illustrated in Figure 4, our estimates for the light query tends to be inaccurate due to the stale metadata. However, this inaccuracy is acceptable since the energy overhead of metadata collection supersedes the gain from a good query plan. For instance, as shown in Figure 4, query plans 1..25 and 26..45 represent two sets of plans which do and do not collect metadata, respectively. As we observed, not collecting metadata is a good strategy for a light query in general because their energy cost are small relative to the overhead associated with metadata collection. In contrast, as illustrated in Figure 5, our cost estimates are accurate for the heavy query, thus a good query plan is chosen. These estimates are accurate because they are based on fresh metadata just collected from the nodes participating in the query. As illustrated, for a heavy query, choosing a good query plan is important because the difference in energy cost between the best and the worst query plans are quite significant.

D. Optimization on Communication Cost

The last experiment is to compare our query optimizer which takes into account both communications and sensing cost in optimization with one that considers only the sensing cost. The goal is to determine the gain from optimizing the communication cost.

In this experiment, we simulated 25 queries, by randomly setting the following parameters:

- Topology: 5 topologies each consisting of 50 nodes.
- Sensor Histogram: uniformly distribution or as defined in accordance with JISAO dataset.
- Predicates: 1 to 5 randomly chosen predicates.
- Duration: 1 day to 3 months.
- Epoch: 4 minutes to 3 months.
- Periodical Metadata Collection: 1 to 64 times a month.

For each query, we compare two optimization approaches: a) sensing cost only; and b) sensing and communication cost. As illustrated in Figure 6, the approach optimizing both sensing and communication cost consumes (on average) 35% less energy than the other approach. This experimental result indicates that optimizing on the communication cost may achieve significant energy saving.

VI. CONCLUSION

In this paper we examine the design of a query optimizer for WSNs. Unlike existing approaches which only define different sensor sampling orders, our approach optimizes on routing, sensing and data/metadata collection. A salient feature in our approach is the estimation and tradeoff of the metadata collection cost for the potential gain in more accurately estimated query plans. Our system collects metadata only for queries that will benefit from more accurate cost estimation based on fresh metadata. Through simulation we showed that our proposal chooses good query plans and utilizes significantly less energy than an approach that only considers sensing cost.

In the future we plan to extend our approach to support multiple queries simultaneously. Additionally, we plan on researching how to dynamically adapt query plans to changes of sensor readings and networks.

VII. ACKNOWLEDGMENT

This work was supported in part by National Science Foundation grants CNS-0626709, IIS-0534343, IIS-0328881.

VIII. REFERENCES

- [1] Joint institute for the study of the atmosphere and ocean. <http://www.ncgc.nrcs.usda.gov/products/datasets/>.
- [2] ns-2: The Network Simulator. <http://www.isi.edu/nsnam/ns/>.
- [3] Run length encoding. http://en.wikipedia.org/wiki/Runlength_encoding.
- [4] M. M. Astrahan and et al. System R: Relational Approach to Database Management. *ACM Transactions on Database Systems*, 1(2):97–137, 1976.
- [5] R. Avnur and J. M. Hellerstein. Eddies: Continuously Adaptive Query Processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 261–272, Dallas, Texas, USA, May 2000.
- [6] P. Van Bommel and Th. P. Van Der Weide. Reducing the Search Space for Conceptual Schema Transformation. *Data Knowledge Engineering*, 8(4):269–292, 1992.
- [7] A. Deshpande, C. Guestrin, W. Hong, and S. R. Madden. Exploiting Correlated Attributes in Acquisitional Query Processing. In *International Conference on Data Engineering (ICDE)*, pages 143–154, Tokyo, Japan, April 2005.
- [8] A. Deshpande and J. Hellerstein. Lifting the Burden of History from Adaptive Query Processing. In *Proceedings of the International Conference on Very Large Data Bases*, pages 948–959, Toronto, Canada, September 2004.
- [9] C. Intanagonwivat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 56–67, Boston, MA, USA, August 2000.
- [10] Y. E. Ioannidis, R. T. Ng, K. Shim, and T. K. Sellis. Parametric query optimization. *Very Large Database Journal*, 6(2):132–151, 1997.
- [11] Yannis E. Ioannidis. Query optimization. *ACM Computing Survey*, 28(1):121–123, 1996.
- [12] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data centric routing in wireless sensor networks. Technical Report CENG 02-14, University of Southern California, 2002.
- [13] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Proceedings of Symposium on Operating Systems Design and Implementation (OSDI)*, pages 131–146, Boston, MA, USA, December 2002.
- [14] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 491–502, San Diego, CA, USA, June 2003.
- [15] S. R. Madden, M. Shah, J. M. Hellerstein, and V. Raman. Continuously Adaptive Continuous Queries over Streams. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 49–60, Madison, WI, USA, June 2002.
- [16] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the Sixteenth IEEE Conference on Computer Communications (INFOCOM)*, page 1405, Washington, DC, USA, 1997.
- [17] Y. Yao and J. Gehrke. Query processing in Sensor Networks. *IEEE Pervasive Computing*, 3(1):46–55, January-March 2004.