Cost-Efficient Processing of Min/Max Queries over Distributed Sensors with Uncertainty

Zhenyu Liu
University of California
Los Angeles, CA 90095
vicliu@cs.ucla.edu

Ka Cheung Sia
University of California
Los Angeles, CA 90095
kcsia@cs.ucla.edu

Junghoo Cho
University of California
Los Angeles, CA 90095
cho@cs.ucla.edu

ABSTRACT

The rapid development in micro-sensors and wireless networks has made large-scale sensor networks possible. However, the wide deployment of such systems is still hindered by their limited energy which quickly runs out in case of massive communication. In this paper, we study the cost-efficient processing of aggregate queries that are generally communication-intensive. In particular, we focus on `MIN/MAX` queries that require both *identity* and *value* in the answer. We study how to provide an error bound to such answers, and how to design an "optimal" sensor-contact policy that minimizes communication cost in reducing the error to a user-tolerable level.

Categories and Subject Descriptors

F.0 [**Theory of Computation**]: General

General Terms

Theory

Keywords

MIN/MAX Query Processing, Query Answering with Uncertainty

1. INTRODUCTION

Sensor networks have tremendous potential to extend our capability in sensing and interacting with the surrounding environment. The rapid development in low-power micro-sensors and wireless networks has enabled prototype sensor networks to be deployed in controlled environments [1]. However their wide deployment is still hindered by their relatively short lifespan, because sensors typically operate on a small battery and become inoperable once the battery runs out. It is therefore of paramount importance to minimize their battery use to extend the lifespan of a sensor network.

In a typical setting, sensors communicate through wireless channels due to the ease and low cost of initial deployment. In this scenario, the communication among the sensors becomes the principal source of power usage [1], and a number of ongoing research investigates various mechanisms to minimize sensor communication [2, 3, 4, 5, 1]. In this paper, we study how we can process *aggregate queries* efficiently on a sensor network with minimum communication among the sensors. As our next example illustrates, aggregate queries are one of the most "expensive" queries to support because we potentially have to retrieve the current value from *every* sensor.

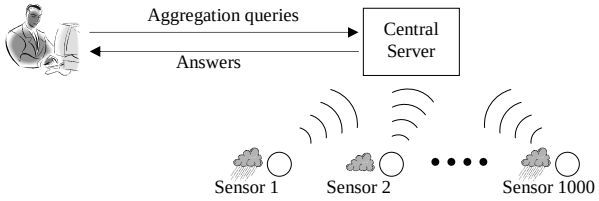


Figure 1: A sensor network consisting of 1000 precipitation sensors distributed at various locations

Example 1 Consider a sensor network consisting of 1,000 rainfall sensors (Figure 1). The user issues queries to a central server that contacts each sensor to retrieve its current rainfall reading. The current values of all the sensors can be viewed as a 1000-tuple table, `Sensors(id, rainfall)`, where `id` is the identity of each sensor and `rainfall` is the current rainfall value. Given this view, the following SQL query computes the maximum rainfall among the sensors:

```
SELECT MAX(rainfall)
FROM Sensors
```

Note that to compute the *exact* answer to this query, we potentially have to contact all the sensors. This process will incur significant communication overhead on the entire sensor network. □

In many applications, however, getting the exact (or 100% accurate) answer may not be required, and users may tolerate a certain level of error in order to save the communication cost.

In this context, we note that `MIN` and `MAX` are the two most difficult queries among the five SQL aggregate functions (`MIN`, `MAX`, `AVG`, `SUM`, `COUNT`) to get an "approximate" answer. For `AVG`, `SUM` and `COUNT` queries, all we need is a value in the answer. To process these "value-oriented" queries, we can simply "sample" the current values of a few sensors and get an approximation. Furthermore, it is also possible to estimate the error bound of the approximation using, say, the central limit theorem [6].

In contrast, for `MIN` or `MAX` queries on a sensor network, we often want to know not only the *value* but also the *identity* of the `MIN/MAX` sensor. For instance, suppose we want to query the above rainfall sensor network to find out possible flood conditions. To do that, we issue a `MAX` query to locate the sensor with the heaviest precipitation. Knowing the identity and knowing the value of the `MAX` sensor are equally crucial in this task: the sensor's identity

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

tells us where the most severe raindrop condition occurs, and the value tells us how severe the condition is.

Requiring both identity and value in the answer poses a fundamental challenge to the cost-efficient answering of MIN/MAX queries. While we might still find an approximate MIN/MAX value via sampling, can we still find an "approximate identity" by sampling a few sensors? For instance, suppose we identify the MAX sensor out of 10 samples drawn from 1000. How do we measure the error between this "approximate identity" and the "true MAX identity" in the 1000? How can we combine such an error with the error in the value component, and further provide an overall error bound?

In this paper, we study how we can find both the identity and the value to the answer of MIN/MAX queries using minimum communication among sensors. We believe this paper makes the following contributions:

- We propose a formal framework to process MIN/MAX queries efficiently when the user can tolerate a certain level of error in the answer (Section 3).
- We describe an algorithm that selectively contacts a few sensors in order to reduce the error in the answer (Section 4). We also explain in what order the server should contact the sensors, so that it can obtain the most "accurate" answer while contacting the minimum number of sensors (Sections 5 and 6).
- We conduct extensive experiments to study the effectiveness of our proposed policies (Section 7). Our results shows that in certain cases, the optimal policy may save about 90% communication cost compared to a naive policy.

2. RELATED WORK

Data gathering with minimum communication cost has been studied in the sensor network community. In earlier works [2, 3, 4, 5] researchers have largely focused on designing efficient routing protocols that gather precise answers. Recently researchers have also investigated using statistical summaries to provide approximate answers to aggregate queries, so that the communication cost is reduced [7, 8]. Our work differs from the existing ones by providing both the identity and the value for MIN/MAX queries, plus we provide an error guarantee to the user so that the user can control the quality of the result.

Answering MIN/MAX queries over imprecise sensor data has been studied by Cheng et al. and Deshpande et al. [9, 10]. In their works, the answer's error is also defined as a probabilistic measure. Sensor probing policies are proposed to reduce this error by observing more sensors and obtaining their latest readings. While these works propose various heuristics in designing sensor probing policies, in this study we focus on how to achieve *optimality* in designing such sensor probing policies.

Our research is also related to answering aggregate queries over imprecise data replications [11, 12, 13]. In contrast to our work, these works are mainly concerned about reducing the error bounds of the returned answers, while our work tries to return both the identity and the value for MIN/MAX queries.

3. FRAMEWORK

In this section, we describe our framework for processing MIN/MAX queries with errors. Note that answering MIN queries is symmetric to answering MAX queries, thus in the remainder of this paper we only deal with MAX.

Roughly, our high-level framework is as follows: We assume that the user issues queries to a central server that is responsible for communicating with the distributed sensors and collecting their values (Figure 1). The server may not know the exact values of the sensors at the query time, due to infrequent communications with the sensors. In our paper, the server models the current value of the i^{th} sensor as a random variable X_i with an associated probability distribution $f_i(x)$. We also assume that when the user issues a query, she also specifies how much "error" in the answer she can tolerate. Given the user-specified error tolerance and the probability distribution of each sensor, the server decides which sensors to contact, so that it can reduce the answer's error to the tolerable level. In the rest of this section we describe this high-level framework more precisely. In Section 3.1 we first explain how the server communicates with the sensors and how it represents the current value of a sensor. Then in Section 3.2 we formalize the notion of "error" in the answer.

3.1 Data Model and Sensor Probing

Periodically, the server communicates with the sensors in the network to obtain their current values. The communication may be initiated either by the sensors or by the server. For example, a sensor may decide to report its current value to the server for the following reasons:

- *Regular update* [1]: The sensor reports its current value to the server regularly at a certain interval, say once per hour.
- *Value-bound violation* [11, 9]: The sensor reports its current value whenever the value exceeds a certain range. For example, a rainfall sensor may be required to report the current precipitation whenever the value deviates +/-50 millimeters from the previous report. In this case, if the previously-reported value was 100mm, the current value must be reported whenever it exceeds 150mm or drops below 50mm.

Note that unless the sensors constantly report their current values to the server, the server does not know the current sensor values for sure. In this paper, the server models the value of sensor i as a random variable X_i with an associated probability density function $f_i(x)$. The value of X_i may lie either in a bounded range or in an unbounded range depending on how sensors communicate with the server.

- *Bounded model* : The server knows that the value of X_i lies within a bounded range $[l_i, u_i]$ (i.e., $l_i \leq X_i \leq u_i$). This model is appropriate for the above value-bound violation scenario. For example, assuming that a rainfall sensor previously reported 100mm, and it must report again whenever the value deviates more than +/-50mm. Without receiving any further report, the server knows that the current value is bounded within [50mm, 150mm].
- *Unbounded model* : X_i may take any value in an unbounded range. This model may be appropriate if the sensor regularly updates the server without providing any value-bound guarantee.

Note that the exact distribution of $f_i(x)$ can usually be obtained either based on the physical model of how X_i's value evolves over time or based on the historical data collected from the sensor. For example, if historical data indicate that the March raindrop at sensor i follows a bell-shaped distribution centered around 100mm, we may assume X_i follows a normal distribution in this range (Figure 2(a)).

Sensor Probing Besides the sensor-initiated communication, the server may also contact sensor i and retrieve its current value, which we refer to as *sensor probing*. Note that after probing X_i, its value changes from a probability distribution to a single value (Figure 2).

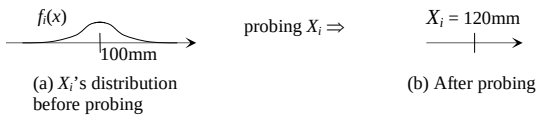


Figure 2: Sensor probing

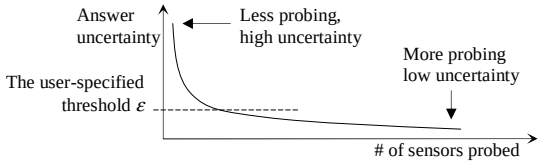


Figure 3: Tradeoff between uncertainty and probing cost

Probing X_i has an associated cost, c_i. Depending on the configuration of the network, the cost of probing two sensors may be the same or different, which leads to the following two cost models.

- *Unit-probing cost model*: Probing every sensor incurs a unit cost, i.e., $c_1 = ... = c_i = ... = 1$.
- *Variable-probing cost model*: Probing different sensors incurs different costs.

3.2 Answer Uncertainty and Uncertainty Threshold

Our goal is to discover $\langle X_{max}, x_{max}\rangle$, the *identity* X_{max} and the *current value* x_{max} of the MAX sensor. Given this problem formulation, we note that an answer $\langle X_{max}, x_{max}\rangle$ can be wrong if there exists another sensor with a higher value than x_{max}. Thus, we define the *answer uncertainty* as follows:

Definition 1 (Answer Uncertainty, $U(X_{max}, x_{max})$) The uncertainty of an answer $\langle X_{max}, x_{max}\rangle$, denoted as $U(X_{max}, x_{max})$, is the probability that another sensor has a value larger than x_{max}. Mathematically,

$$U(X_{max}, x_{max}) = P(X_1 > x_{max} \text{ OR } ... \text{ OR } X_n > x_{max}) \quad (1)$$

where we assume there are n sensors in the network. □

Intuitively, $U(X_{max}, x_{max})$ represents how likely the returned answer is wrong. For example, when $U(X_{max}, x_{max}) = 0.1$, the user may get a wrong answer out of every 10 queries. This answer uncertainty, thus, can be used by the user to specify how much error is tolerable. We refer to the user-specified error-tolerance level as the *uncertainty threshold*, represented by the symbol ε. Depending on the application, the uncertainty threshold can be either zero or positive.

- *Zero-uncertainty tolerance*, $\varepsilon = 0$. Due to the strict requirement of the application, the user tolerates no uncertainty in the answer.
- *Nonzero-uncertainty tolerance*, $\varepsilon > 0$. The user is willing to tolerate errors up to the level of ε in order to save probing cost.

Intuitively, as we probe more sensors, the uncertainty of the answer decreases. Figure 3 shows the intuitive relationship among the uncertainty, the number of probing and a nonzero uncertainty threshold ε. The horizontal axis is the number of probing, and the vertical axis is the uncertainty of our answer. The upper left corner of the curve represents a high uncertainty level when we probe few variables. The lower right corner represents an uncertainty level close to zero after extensive probing. The uncertainty threshold ε represents the user-specified tradeoff point.

Finally, given that the server needs to return both the identity and the current value of the MAX sensor, we note that a sensor can be returned as an answer only when it has already been probed.

Symbol	Meaning
$\{X_1,...,X_n\}$	The set of random variables, each representing a sensor's value
$[l_i, u_i]$	The value range for X_i, where l_i is the lower bound and u_i is the upper bound
$f_i(x)$	The probability density function (*p.d.f.*) of X_i
c_i	The cost of probing X_i
X_{max}	The identity of the maximum variable
x_{max}	The value of the maximum variable
$U(X_{max}, x_{max})$	The uncertainty of answer $\langle X_{max}, x_{max}\rangle$
ε	The user-specifi ed uncertainty threshold
$\mathcal{P}$	A probing policy
$\mathbb{X}$	The set of unprobed variables
$X_i \prec X_j$	A precedence relation between X_i and X_j
s_i	The slack upper bound of X_i
$G_i(x)$	$P(X_i > x)$

Figure 4: Symbols used throughout the paper

Algorithm 4.1 MAXFinder($\{X_1, ..., X_n\}, \varepsilon$)
Input:
$\{X_1, ..., X_n\}$: *the entire set of variables to be probed*
ε: *the uncertainty threshold specified by the user*
Output:
$\langle X_{max}, x_{max}\rangle$: *the identity and value of the max variable*
Procedure
1. $\mathbb{X} \leftarrow \{X_1, ..., X_n\}$, $\mathbb{X}' \leftarrow \emptyset$
2. **DO**
3. $\quad \mathcal{P} \leftarrow$ ChooseProbingPolicy($\mathbb{X}, \varepsilon$)
4. $\quad$ *Probe the first variable in* $\mathcal{P}$
5. $\quad \mathbb{X} \leftarrow \mathbb{X} - \{$*variable just probed*$\}$
6. $\quad \mathbb{X}' \leftarrow \mathbb{X}' \cup \{$*variable just probed*$\}$
7. $\quad \langle X_{max}, x_{max}\rangle \leftarrow$ *the identity and value of the max variable in* $\mathbb{X}'$
8. **WHILE** $U(X_{max}, x_{max}) > \varepsilon$
9. **RETURN** $\langle X_{max}, x_{max}\rangle$

Figure 5: A general probing algorithm

In Figure 4 we summarize the symbols in this paper. As we continue our discussion, we will introduce some symbols that have not been explained yet.

4. SENSOR PROBING ALGORITHM

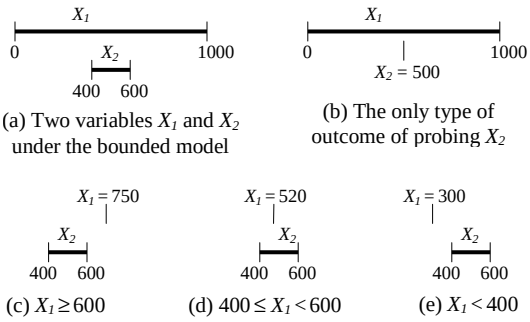
We have illustrated that we can use probing to reduce the answer uncertainty until it meets the uncertainty threshold ε. Our main goal is to optimize the probing cost in this process. In this section, we first describe a high-level probing algorithm and then discuss the core challenge in optimizing the probing cost.

4.1 A General Probing Algorithm

Algorithm *MAXFinder* in Figure 4.1 depicts the general probing procedure. We highlight the main features and concepts of this algorithm as follows.

Core data structure. At any time the system maintains two sets of variables, $\mathbb{X}$ corresponding to all the *unprobed* sensors, and $\mathbb{X}'$ corresponding to all the *probed* sensors. These two sets are initialized accordingly in Step 1 to reflect that no sensor has been probed in the beginning.

Probing policy. In Step 3 of the algorithm, the system considers all potential probing sequences for the unprobed variables and picks the one that is most likely to incur the minimal probing cost. We refer to a sequence of unprobed variables in $\mathbb{X}$ as a *probing policy*, denoted as $\mathcal{P}$: $X_1^{\mathcal{P}} \rightarrow ... \rightarrow X_{|\mathbb{X}|}^{\mathcal{P}}$. For example, a probing policy on four unprobed variables $\{X_1, X_2, X_3, X_4\}$ may be $X_4 \rightarrow X_1 \rightarrow X_2 \rightarrow X_3$. After considering potential probing policies and picking the "best" one $\mathcal{P}$ in Step 3, the system probes



(a) Two variables X_1 and X_2 under the bounded model

(b) The only type of outcome of probing X_2

(c) $X_1 \geq 600$

(d) $400 \leq X_1 < 600$

(e) $X_1 < 400$

Figure 6: Searching for the maximum between X_1 and X_2 with zero uncertainty

the first variable in $\mathcal{P}$ in Step 4. If the maximum answer so far still does not meet the uncertainty threshold, ε (condition in Step 8), the system continues to its next probing.

Policy-selection. The effectiveness of Algorithm *MAXFinder* is primarily determined by Step 3, where the system evaluates all potential probing policies and picks the one that will lead to the minimum probing cost. Unfortunately, at this stage the system does not know the sensors' current values which determine the *exact* cost of each policy. As a result, the system can only "guess" the cost of each policy and choose the "best" on that basis. We use a simple example to illustrate how the system can evaluate the "likely cost" of various probing policies.

Example 2 Consider searching for the maximum between two sensors when the user tolerates zero uncertainty (i.e., $\varepsilon = 0$). As shown in Figure 6(a), the value of these two sensors, X_1 and X_2 are bounded in $[0, 1000]$ and $[400, 600]$, respectively. To simplify the example, we assume both variables follow uniform distributions and incur unit-probing cost. In the *ChooseProbingPolicy* step, we need to compare the costs of two possible policies, $\mathcal{P}_a$: $X_2 \rightarrow X_1$ and $\mathcal{P}_b$: $X_1 \rightarrow X_2$.

- *The cost of* $\mathcal{P}_a$: $X_2 \rightarrow X_1$. Given its value range, the outcome of probing X_2 always falls within X_1's range (Figure 6(b)). Thus, even after probing X_2, X_1 still has a chance to be the maximum. We have to further probe X_1 to obtain the zero-uncertainty answer. Therefore $\mathcal{P}_a$ always leads to a probing cost of 2.
- *The cost of* $\mathcal{P}_b$: $X_1 \rightarrow X_2$. The cost depends on the probing outcome of X_1. There are three important cases to consider:
  - $X_1 \geq 600$ (Figure 6(c)): X_1 is greater than X_2's upper bound and is thus the maximum. In this case, we can stop probing and return X_1 as the answer. The overall probing cost is 1.
  - $400 \leq X_1 < 600$ (Figure 6(d)): Since X_1 is within X_2's range, we cannot decide whether X_1 is the maximum. As a result, we have to further probe X_2 and the overall probing cost is 2.
  - $X_1 < 400$ (Figure 6(e)): In this case X_2 is the maximum. However, because we need to return both the identity and the value of the maximum answer, we still need to probe X_2 to get its value. Thus the overall cost is still 2.

Note that we need to make a decision between $\mathcal{P}_a$ and $\mathcal{P}_b$ before we probe any variable. Thus, we have to consider all possible values of the variables and compute the *expected cost* of $\mathcal{P}_a$ and $\mathcal{P}_b$. The expected cost of $\mathcal{P}_a$ is 2. The expected cost of $\mathcal{P}_b$ is $1 \cdot P(X_1 \geq 600) + 2 \cdot P(400 \leq X_1 < 600) + 2 \cdot P(X_1 < 400)$. Given X_1's uniform distribution between 0 and 1000, the expected cost of $\mathcal{P}_b$ is $1 \cdot 0.4 + 2 \cdot 0.2 + 2 \cdot 0.4 = 1.6$. Since $\mathcal{P}_b$ incurs lower expected cost, we pick $\mathcal{P}_b$ as the "best" probing policy. □

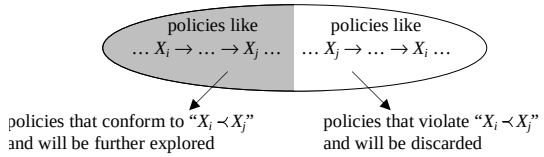


Figure 7: Search space pruning from precedence relation: "$X_i \prec X_j$"

The above example illustrates that the best policy is chosen based on its *expected probing cost*. The expected cost of a policy depends on 1) the variable sequence in the policy, 2) the probability distribution of each variable, and 3) the user-specified threshold ε. Due to space constraint, we provide the exact formula in the extended version of this paper [14].

Adaptiveness in *MAXFinder*. Note that at the beginning of each probing iteration, the system *re-evaluates* all probing policies in Step 3 again. This re-evaluation is necessary to adjust the probing policy adaptively based on the probing results so far. For example, before probing any variable, the optimal policy on four variables may be $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4$. But after probing X_1 and knowing its value, the optimal policy on the remaining three variables may change to $X_4 \rightarrow X_3 \rightarrow X_2$ from the initial $X_2 \rightarrow X_3 \rightarrow X_4$.

4.2 Reducing The Policy-Search Space

In Step 3 of Algorithm *MAXFinder*, note that we potentially have to enumerate *all* probing policies and compute their expected cost. This enumeration is computationally very expensive when we have a reasonably large number of variables. For example, if we have 100 variables, we need to consider $100! \approx 10^{158}$ potential permutations of the variables. In this section, we discuss how we may reduce this large search space. Our main idea for the reduction is as follows: If we know that probing X_1 before X_2 always leads to a lower probing cost, we can discard any policy that probes X_2 before X_1. We formalize this idea by introducing the notion of *precedence relation*.

Definition 2 (Precedence Relation, $X_i \prec X_j$) We define that the variable X_i should *precede* X_j, denoted as $X_i \prec X_j$, if and only if for any policy $\mathcal{P}$: ... $\rightarrow X_i \rightarrow$ ... $\rightarrow X_j \rightarrow$ ..., simply switching the positions of X_i and X_j always leads to another policy $\mathcal{P}'$: ... $\rightarrow X_j \rightarrow$ ... $\rightarrow X_i \rightarrow$ ... with a higher expected cost. □

Basically, $X_i \prec X_j$ means X_i should be probed before X_j under all circumstances. In other words, any policy that probes X_j before X_i can be safely thrown away. Figure 7 illustrates the pruning effect of a precedence relation. As the figure shows, a precedence relation $X_i \prec X_j$ allows us to partition the policy space by half and throw away one of the partitions. Thus, we can significantly reduce the policy-search space. In the next two sections, we explain how we can identify the precedence relations between variables. Our study shows that the solution for the zero-uncertainty-tolerance case differs significantly from that of the nonzero-uncertainty-tolerance case. Hence, we discuss these two cases separately. Section 5 discusses the zero-uncertainty case. Section 6 discusses the nonzero-uncertainty case.

5. OPTIMAL PROBING POLICY UNDER ZERO-UNCERTAINTY TOLERANCE

In this section, we study how to determine the precedence relations among the sensors under zero-uncertainty tolerance. Our study shows that

- When sensor values lie in bounded ranges (the bounded model in Section 3), there exists a very simple mechanism to find the optimal probing policy.
- When the sensor values are unbounded (the unbounded model in Section 3), we essentially have to probe all sensors in order to return a zero-uncertainty answer.

In Section 5.1 we present our findings for the bounded model. Section 5.2 is for the unbounded model. The results apply to both the unit-probing cost and the variable-probing cost models.

5.1 Optimal Policy for The Bounded Model

To motivate our finding, we use another simple example.

Example 3 We search for the maximum between two sensors, X_1 and X_2, under a zero-uncertainty setting. The values of X_1 and X_2 are bounded in $[0, 1000]$ and $[700, 900]$, respectively. Probing X_1 costs 2 and probing X_2 cost 1. Can we determine the precedence relation between these two variables without evaluating the expected cost of *every* possible policy? What criteria can we use? Should we first probe X_2 given that its mean value is likely to be higher than that of X_1, plus probing X_2 is cheaper? □

The following theorem is our answer to the above question:

Theorem 1 *Under zero-uncertainty tolerance and the bounded-variable model, $X_i \prec X_j$ if and only if $u_i > u_j$.* □

Proof Please refer to the extended version of this paper [14]. ■

According to the above theorem, we can determine the precedence relation simply by examining the upper bounds of the variables. That is, a variable X_i should be always probed before X_j as long as X_i has a larger upper bound. We emphasize that this theorem does not assume what distributions the variables should follow within their bounds.

Note that this theorem provides a simple and efficient way to discover the optimal policy. Since a variable must be probed before others if it has a larger upper bound, we can obtain the optimal probing policy by sorting all unprobed variables by their upper bounds. The following corollary formalizes this finding.

Corollary 1 *Let $\mathbb{X} = \{X_1, X_2, ..., X_{|\mathbb{X}|}\}$ be the set of unprobed variables, ranked in a descending order of their upper bounds, i.e., $u_1 \geq u_2 \geq ... \geq u_{|\mathbb{X}|}$. Under zero-uncertainty tolerance, policy $\mathcal{P} : X_1 \rightarrow X_2 \rightarrow ... \rightarrow X_{|\mathbb{X}|}$ is optimal.* □

Proof Please refer to the extended version of this paper [14]. ■

In summary, under zero-uncertainty tolerance, we can find the optimal policy by sorting all unprobed variables by their upper bounds in the descending order.[1]

5.2 Optimal Policy for The Unbounded Model

In the unbounded model, a sensor's value can be arbitrarily large. As long as we still have one sensor left unprobed, there is a slight chance that this unprobed sensor may turn out to be the maximum. Thus, to guarantee zero uncertainty we essentially have to probe all sensors. In the extended version [14] we have provided a theorem that formalizes this finding.

[1]Given that the optimal policy is solely determined by the upper bound of each variable, the optimal policy does not change depending on the probing outcome. Thus, we do not need to re-evaluate the optimal policy in every probing iteration in this case.

6. OPTIMAL PROBING POLICY UNDER NONZERO-UNCERTAINTY TOLERANCE

In the previous section we learned that the upper bounds play a very important role in deriving precedence relations under the zero-uncertainty setting. In this section we show that under some conditions, we can use a similar concept, called a *slack upper bound*, to derive precedence relations for the nonzero-uncertainty setting. We first motivate this concept in Section 6.1 using a simple two-sensor scenario with unit-probing cost. We then generalize the result to a multi-sensor scenario with unit-probing cost in Section 6.2, and further to the scenario with variable-probing cost in Section 6.3. The solution in this section applies to both the bounded- and unbounded-variable models, so we no longer differentiate them.

6.1 Optimal Policy in A Two-Sensor Scenario with Unit-Probing Cost

We use the following example to motivate the concept of slack upper bound.

Example 4 We revisit the two-sensor scenario in Figure 6(a). This time we set the uncertainty threshold $\varepsilon = 0.15$.

Under the previous setting of zero-uncertainty tolerance, we return X_1 as the max without probing X_2, if and only if the value of X_1 is larger than the upper bound of X_2, i.e., $X_1 \geq u_2 = 600$.

Under nonzero-uncertainty tolerance, however, we can return X_1 even when its value is slightly less than 600 because the user can tolerate an uncertainty up to 0.15. We note that as long as X_1 is larger than 570, we can consider X_1 as larger than X_2 with uncertainty less than 0.15. □

The above example shows that under the nonzero-uncertainty setting, the value 570 (which satisfies $P(X_2 > 570) = 0.15$) plays a similar role to X_2 as X_2's upper bound u_2 does under the zero-uncertainty setting. Thus, we define 570 as the *slack upper bound* of X_2:

Definition 3 (Slack Upper Bound, s_i) Given the uncertainty threshold ε, the slack upper bound of variable X_i, denoted as s_i, is a point such that

$$P(X_i > s_i) = \varepsilon$$ □

Figure 8 shows the relationship among $P(X_i > x)$, ε, and s_i. By definition, $P(X_i > x)$ is a monotonically decreasing function towards zero. The slack upper bound of X_i, s_i, is the cross point at which $P(X_i > x)$ goes below the uncertainty threshold ε. To simplify our future discussion, we define the function $G_i(x)$ as

$$G_i(x) = P(X_i > x)$$

Under this definition, $G_i(s_i) = \varepsilon$.

The following lemma shows that in a two-sensor scenario with unit-probing cost, the slack upper bound indeed plays the same role as the upper bound in determining the precedence relation between variables:

Lemma 1 *In a two-sensor scenario with unit-probing cost, $X_1 \prec X_2$ if and only if $s_1 > s_2$.* □

Proof Please refer to the extended version [14]. ■

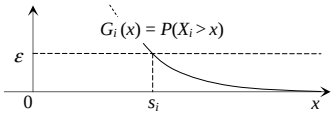


Figure 8: The relationship among $G_i(x)$, ε and the slack upper bound s_i

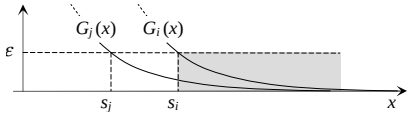


Figure 9: Visualizing the second condition in Theorem 2

6.2 Optimal Policy in A Multi-Sensor Scenario with Unit-Probing Cost

Lemma 1 suggests that in a simple scenario, we can derive the optimal policy by sorting the variables' slack upper bounds. Unfortunately, when there are more than two sensors, our study shows that $s_i > s_j$ does not necessarily imply $X_i \prec X_j$.[2] Thus, in the remainder of this section, we first identify an additional condition that, when combined with $s_i > s_j$, is sufficient to derive a precedence relation between X_i and X_j for three-or-more-sensor scenarios. After we discuss this condition we explain how we handle situations when the condition is not met. The following theorem provides such an additional condition:

Theorem 2 *Under nonzero-uncertainty tolerance and unit-probing cost, $X_i \prec X_j$ if*

- $s_i > s_j$, *and*
- $\forall x > s_i, G_i(x) > G_j(x)$

Proof Please refer to the extended version [14]. ■

In Figure 9 we show the graphical meaning of the second condition in Theorem 2. According to the condition, to the right of s_i (the shaded region), $G_i(x)$ is always above $G_j(x)$. Given that when $x > s_i$, both functions are below the horizontal line of ε, we can rephrase the condition as "$G_i(x)$ and $G_j(x)$ should not intersect below ε."

The following corollary shows that if the second condition is met for every pair of variables, we can derive the optimal probing policy directly by sorting the slack upper bounds.

Corollary 2 *Let $\mathbb{X} = \{X_1, X_2, ..., X_{|\mathbb{X}|}\}$ be the set of unprobed variables with unit-probing cost, ranked in a descending order of their slack upper bounds, i.e., $s_1 \geq s_2 \geq ... \geq s_{|\mathbb{X}|}$. Policy $\mathcal{P}$: $X_1 \rightarrow X_2 \rightarrow ... \rightarrow X_{|\mathbb{X}|}$ is optimal if the second condition in Theorem 2 holds for any X_i and X_j with $i < j$.* □

Proof Please refer to the extended version [14]. ■

Figure 10 depicts the overall relations of $G_i(x)$'s when the condition in Corollary 2 is met. Basically, no two $G_i(x)$ functions intersect beneath the horizontal line of ε. This condition may hold in certain special cases, for example, if all variables follow exponential distributions with different λ values.

In general, however, the condition in Corollary 2 may not hold for all pairs of variables. For example, in Figure 11, $G_2(x)$ and

[2] An example provided in the extended version [14] illustrates this finding.

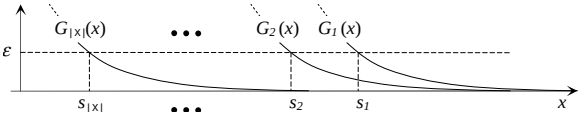


Figure 10: A sample scenario in which Corollary 2 applies

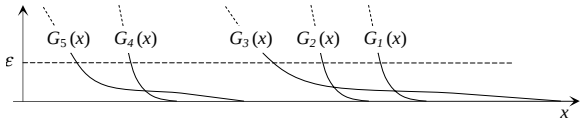


Figure 11: A sample scenario in which Corollary 2 does not apply

$G_3(x)$ intersect with each other underneath ε. As a result, we cannot apply Corollary 2 to directly derive the optimal policy. In this case, we need to first identify all pair-wise precedence relations based on Theorem 2, further reduce the policy search space using the derived precedence relations, and finally find the optimal policy by computing the expected cost of each policy in this reduced search space.

For instance, in case of Figure 11, Theorem 2 gives us seven precedence relations ($X_1 \prec X_2$, $X_1 \prec X_4$, $X_1 \prec X_5$, $X_2 \prec X_4$, $X_2 \prec X_5$, $X_3 \prec X_4$, and $X_3 \prec X_5$), so we can reduce the policy search space to only 6 candidates.[3]

6.3 Optimal Policy in A Multi-Sensor Scenario with Variable-Probing Cost

In this section we show that with variable-probing cost, we need an extra condition to derive the precedence relation between X_i and X_j:

Theorem 3 *Under nonzero-uncertainty tolerance and variable-probing cost, $X_i \prec X_j$ if*

- $s_i > s_j$,
- $\forall x > s_i, G_i(x) > G_j(x)$, *and*
- $c_i < c_j$.

Proof Please refer to the extended version [14]. ■

Thus, in the most general case, we need to derive precedence relations using Theorem 3 to reduce the policy-search space.

6.4 A Greedy Policy

In certain cases, the size of the sensor network may be too large, so we may not be able to examine even the pruned search space exhaustively. In this section, we present a greedy approach that can handle such cases. This greedy approach, as our later experimental results will show, often yields near-optimal performance.

The greedy approach is based on the following idea: The probing procedure (Algorithm 4.1) stops as soon as the answer uncertainty $U(X^k_{max}, x^k_{max})$ decreases below the level of ε. Here we use $\langle X^k_{max}, x^k_{max} \rangle$ to represent the answer after the k^{th} probing iteration. Thus, the greedy policy picks the unprobed variable that, once probed in the $k + 1^{th}$ iteration, will make $U(X^{k+1}_{max}, x^{k+1}_{max})$ decrease the most. In the variable-probing cost scenario, this decrease should also be normalized by the variable's probing cost.

To implement this policy, we first illustrate how to predict $U(X^{k+1}_{max}, x^{k+1}_{max})$ assuming we will probe X_i in the $k + 1$ iteration. Here we give a sketch of the analysis, and the detailed formula can be found in the extended version [14]. Computing $U(X^{k+1}_{max}, x^{k+1}_{max})$ is not straightforward because x^{k+1}_{max} depends on the outcome of probing X_i. Without the actual probing, we must investigate how different values of X_i, x_i, may affect x^{k+1}_{max}:

- $x_i \leq x^k_{max}$. Thus x^k_{max} is still the maximum value in the $k + 1^{th}$ iteration, i.e., $x^{k+1}_{max} = x^k_{max}$.
- $x_i > x^k_{max}$. Thus x_i becomes the maximum value in the next iteration, i.e., $x^{k+1}_{max} = x_i$.

[3] In the extended version of this paper [14], we present a *Divide-N-Conquer* algorithm that reduces the search space even further. When we use *Divide-N-Conquer* on Figure 11, we need to compare only 3 policies out of $5! = 120$ potential policies.

Without knowing x_i, we need to consider both cases above. Since in each case we know the concrete value x_{max}^{k+1}, we can compute $U(X_{max}^{k+1}, x_{max}^{k+1})$ in that case using Eq. (1). Afterwards, we combine the $U(X_{max}^{k+1}, x_{max}^{k+1})$ value with the probability of each case and compute the expected uncertainty in the $k+1^{th}$ iteration, $E[U(X_{max}^{k+1}, x_{max}^{k+1})]$.

After we compute the "expected" decrease of uncertainty after probing X_i, we need to combine the decrease with X_i's probing cost c_i. Thus the greedy policy would pick X_i that maximizes

$$\frac{U(X_{max}^{k}, x_{max}^{k}) - E[U(X_{max}^{k+1}, x_{max}^{k+1})]}{c_i}$$

7. EXPERIMENTAL RESULTS

In this section we experimentally study the behavior of the policies discussed in the previous sections. We perform the study on both a synthetic dataset and a real precipitation-sensor dataset. Due to space constraints, we include the results on the synthetic dataset in this paper and report the results on the real dataset in the extended version [14]. In the current stage we assume unit-probing cost, and we defer the experimental study of the variable-probing cost model to the future work.

7.1 Dataset and Experimental Setup

We simulate a scenario with 1000 sensors. The value of each sensor is bounded and follows a uniform distribution. Thus we have 1000 variables following the bounded model. To generate such a dataset, for each variable X_i we need to 1) generate a value range, which is kept by the central server and 2) come up with the value for that variable, which represents the current reading of the corresponding sensor. The value of each variable is only known when it is probed.

In generating the value ranges for the 1000 variables, we want to control the overlapping among those variables. We expect that when the variables overlap a lot, more probing is needed to clarify the situation and find the maximum. To study how various overlapping scenarios affect the behavior of a probing policy, we introduce a parameter R called the *max radius*. With a given R, we assign a value range for X_i as follows. First we randomly pick a point in [0, 1000] as the middle point for this value range, say, m_i. We then randomly choose a number r_i between 0 and R as the radius for this interval, and assign $[m_i - r_i, m_i + r_i]$ as the value range for X_i. A larger R value results in larger value ranges on average, and causes more variables to overlap. For example, when $R = 50$, a variable might roughly overlap with 50 other variables.

After generating a value range for X_i, we need to come up with the actual value for X_i. Given X_i's uniform distribution, we pick a random value x_i uniformly from $[m_i - r_i, m_i + r_i]$. After this step, we create one test case for these 1000 variables. For a fixed max radius R, we generate 5000 of such test cases, and we repeat this process for $R = 25$, 50, 75 and 100.

7.2 Zero-Uncertainty Tolerance, $\varepsilon = 0$

In this section we study how efficient the optimal policy is under zero-uncertainty tolerance. That is, we want to see how many sensors the optimal policy probes, compared to some naive method that requires excessive probing.

A naive probing method. In the bounded model, if the upper bound of variable X_i is smaller than the lower bound of X_j, X_i is always smaller than X_j and can never be the maximum. Thus, in what we call the "naive probing method," we probe only the variables whose upper bound is larger than all other lower bounds. For example, when the ranges for X_1, X_2 and X_3 are $[700, 870]$, $[850, 950]$ and $[900, 925]$, respectively, X_1 is always smaller than X_3, so the naive method only probes X_2 and X_3. We use the result of this naive method as the base comparison point against our optimal policy.

R	25	50	75	100
avg # of sensors probed, naive	18.75	33.69	48.45	62.65
avg # of sensors probed, optimal	5.76	7.73	8.75	9.74

Figure 12: Efficiency of the optimal policy under the zero uncertainty setting and various overlapping scenarios

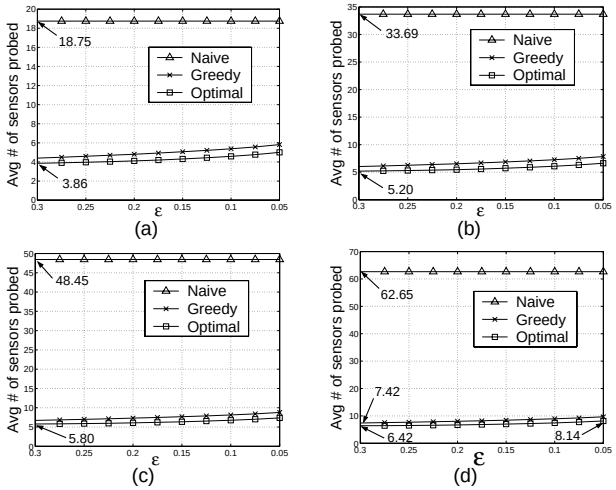


Figure 13: Avg # of sensors probed by the optimal and the greedy polices on the synthetic dataset: (a) $R = 25$ **(b)** $R = 50$ **(c)** $R = 75$ **(d)** $R = 100$

Efficiency of the optimal policy. We compare the optimal policy with this naive method to study the former method's efficiency. For each test case under a fixed R, we execute both the optimal policy and the naive method. We then average the number of sensors probed by both methods over all the 5000 test cases, and summarize the results in Figure 12. For example, on the test cases with $R = 100$, on average the optimal policy probes 9.74 sensors, and the naive method probes 62.65. This represents a 84.4% saving on the side of the optimal policy.

The impact of R on the number of probing. Recall that a larger R, the max radius, causes more variables to overlap and should lead to more probing. As Figure 12 shows, the optimal policy (the third row) does require more probing in larger R settings.

7.3 Nonzero-Uncertainty Tolerance, $\varepsilon > 0$

In this section, our primary goal is to study the efficiency of the optimal and the greedy policy under nonzero-uncertainty tolerance. Further, we empirically study the difference between the optimal and the greedy policies.

Efficiency of the optimal and the greedy policies. We compare both the optimal and the greedy policies against the naive probing method described in Section 7.2, and see how much probing each method saves. We perform the study under the following parameter settings: 1) The max radius R: 25, 50, 75 and 100. 2) The uncertainty threshold ε: from 0.3 to 0.025 at each interval of 0.025.

Figure 13 summarizes the results for various R settings. In each figure, the x-axis shows the different ε values, and the y-axis shows the average number of sensors probed by the three methods: 1) the naive probing method, 2) the greedy policy and 3) the optimal policy. The figures suggest that in general both the optimal and the greedy save a great amount of probing compared to the naive method. For example, under the setting of $R = 100$ and

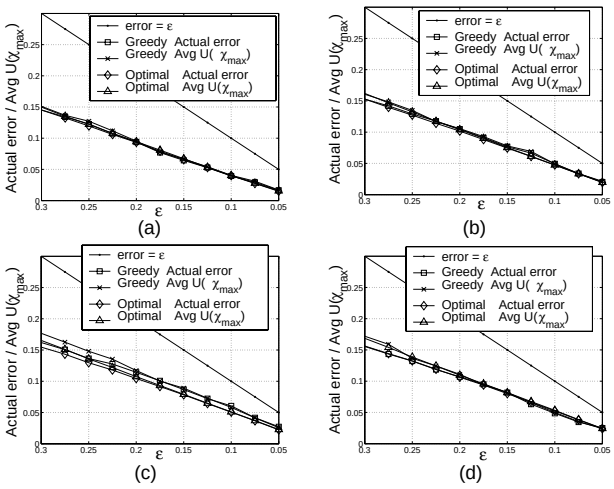


Figure 14: Actual inaccuracy of the optimal and the greedy policy on the synthetic dataset: (a) $R = 25$ **(b)** $R = 50$ **(c)** $R = 75$ **(d)** $R = 100$

$\varepsilon = 0.3$ (Figure 13(d)), the optimal policy probes 6.42 sensors on average, representing 89.8% saving compared to the naive method; the greedy policy probes 7.42 sensors, representing 88.2% saving.

Comparing the optimal policy with the greedy policy. In general the optimal policy saves about 10%-15% on the number of sensors probed. For example, in Figure 13(d), for an uncertainty threshold $\varepsilon = 0.3$, the optimal policy probes 6.42 sensors, whereas the greedy policy probes 7.42 sensors. In this particular case, the optimal policy saves 14.1% compared to greedy. This result suggests that in real applications when searching for the optimal policy may be time-consuming, the greedy policy provides a relatively close approximation.

The impact of ε on the number of sensors probed. We expect that as the user requires less uncertainty, more probing is needed. This general trend can be observed in all sub-figures. In Figure 13(d) for example, the optimal policy probes 6.42 sensors to reach the uncertainty level of 0.3, whereas it probes 8.14 sensors to reach 0.05.

The impact of R on the number of probing. As we have seen in the zero-uncertainty case, a larger R value leads to more probing. In the nonzero-uncertainty case, the results exhibit the same trend. For example, to reach the uncertainty level of 0.3, the optimal policy probes 3.86 sensors in $R = 25$, 5.20 in $R = 50$, 5.80 in $R = 75$ and 6.42 in $R = 100$.

Actual error in the returned answer. The last task performed on the synthetic dataset is to study whether the actual error in the answers matches the uncertainty that the system "guarantees." For example, if in all testing cases the system returns an answer with a "claimed" uncertainty of 0.05, then asymptotically we expect to observe 5 wrong answers out of every 100 testing cases.

- *Computing the "claimed" uncertainty.* In the experiment, for each testing case Algorithm 4.1 may terminate with different uncertainty level $U(X_{max}, x_{max})$ (as long as $U(X_{max}, x_{max}) < \varepsilon$). Thus we average $U(X_{max}, x_{max})$ over all 5000 cases as the uncertainty level that the system "claims."

- *Computing the actual error.* We first compute the correct `MAX` answer for each testing case. By comparing against the correct answers in all cases, we can check the percentage of the cases that the system is actually wrong.

We compare these two measurements on every R and ε setting, and summarize the results in Figure 14. In each sub-figure, the x-axis lists different ε settings, and the y-axis shows 1) the average $U(X_{max}, x_{max})$ and 2) the actual error by both the optimal and the greedy policies. The diagonal line represents "error $= \varepsilon$." From the figures we observe the following:

- In all cases, both the average $U(X_{max}, x_{max})$ and the actual error are always below the ε level. This happens because $U(X_{max}, x_{max}) < \varepsilon$ is the stopping condition for all probing processes (Step 8 in Algorithm 4.1).
- In all cases, the actual error always agrees with the average $U(X_{max}, x_{max})$. Basically this result means we have obtained the desirable error level as we have requested.

8. CONCLUSION AND FUTURE WORK

In this paper we studied the cost-efficient processing of `MIN/MAX` queries over distributed sensors. We first used a probabilistic measure to define the uncertainty embedded in the answers. Further, we analyzed the optimal sensor-probing policy that uses minimum probing to reduce the answer uncertainty to a user-tolerable level. Our analysis was performed under both zero-uncertainty tolerance and nonzero-uncertainty tolerance. Finally, we experimentally evaluated the behavior of both the optimal policy and a proposed greedy policy. The results show that the processing of `MIN/MAX` queries can be very cost-efficient for both policies.

In our current experiments we focused on the unit-probing-cost scenario. In the future we plan to perform experimental study in the variable-probing-cost scenario.

9. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0347993. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The authors gratefully acknowledge Carlos Brito for earlier discussion and feedbacks.

10. REFERENCES

[1] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[2] H.Ö. Tan and Ï. Körpeoğlu. Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Record*, 32(7):66–71, 2003.

[3] Bhaskar Krishanamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *International Workshop of Distributed Event Based Systems (DEBS)*, 2002.

[4] S. Lindsey and C.S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Proceedings of IEEE Aerospace Conference*, 2002.

[5] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of 33rd Annual Hawaii International Conference on System Sciences*, 2000.

[6] D.D. Wackerly, W. Mendenhall III, and R.L. Scheaffer. *Mathematical Statistics with Applications*. Duxbury, 6th edition, 2002.

[7] K. Kalpakis, V. Puttagunta, and P. Namjoshi. Accuracy vs. lifetime: Linear sketches for approximate aggregate range

queries in sensor networks. Technical report, Computer Science and Eletrical Engineering Department, University of Maryland Baltimore County, 2004.

[8] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of ICDE '04*, 2004.

[9] R. Cheng, D.V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of ACM SIGMOD '03*, 2003.

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

[11] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of VLDB '00*, 2000.

[12] T. Feder, R. Motwani, R. Panigrahy, C. Olston, and J. Widom. Computing the median with uncertainty. In *32nd ACM Symposium on Theory of Computing (STOC)*, 2000.

[13] S. Khanna and W.C. Tan. On computing functions with uncertainty. In *Proceedings of ACM PODS '01*, 2001.

[14] Z. Liu, K.C. Sia, and J. Cho. Cost-efficient processing of min/max queries over distributed sensors with uncertainty. Technical report, Computer Science Department, UCLA, 2004.