Sensors, Uncertainty Models and Probabilistic Queries

Reynold Cheng and Sunil Prabhakar
Department of Computer Sciences, Purdue University
{ckcheng,sunil}@cs.purdue.edu

INTRODUCTION

Sensors are often used to monitor the status of an environment continuously. The sensor readings are reported to the application for making decisions and answering user queries. For example, a fire-alarm system in a building employs temperature sensors to detect any abrupt change in temperature. An aircraft is equipped with sensors to track the wind speed, and radars are used to report the aircraft's location to a military application. These applications usually include a database or server to which the sensor readings are sent. Limited network bandwidth and battery power imply that it is often not practical for the server to record the exact status of an entity it monitors at every time instant. In particular, if the value of an entity (e.g., temperature, location) being monitored is constantly evolving, the recorded data value may differ from the actual value. Querying the database can then produce incorrect results. Consider a simple example where a user asks the database: "which room has a temperature between 10^oF and 20^oF?". If we represent temperature values of rooms A and B stored in the database by A_0 and B_0 respectively, we can see from Figure 1(a) that the answer to the user query is "Room B". In reality, the temperature values of both rooms may have changed to newer values, A_1 and B_1, as shown in Figure 1(b), where the true query answer should be "Room A". Unfortunately, because of transmission delay, these newest pieces of information are not propagated in time to the system to supply fresh data to the query, and consequently the query is unable to yield a correct answer.

In general, the incorrectness of query results is due to **Sensor Uncertainty**, an inherent property of any sensor database where each recorded data item is only an older, approximate version of the corresponding entity being monitored. Apparently, providing meaningful answers in face of sensor uncertainty appears to be a futile exercise. In many situations, however, the values of sensors cannot change drastically in a short period of time – the degree and/or rate of change of a sensor value is constrained. This helps us to alleviate the problem. In the previous example, if we can guarantee that the actual temperatures of room A and B are no more than some deviations from A_0 and B_0 respectively, as in Figure 1(c), then we can state with confidence that room A does not satisfy the query.

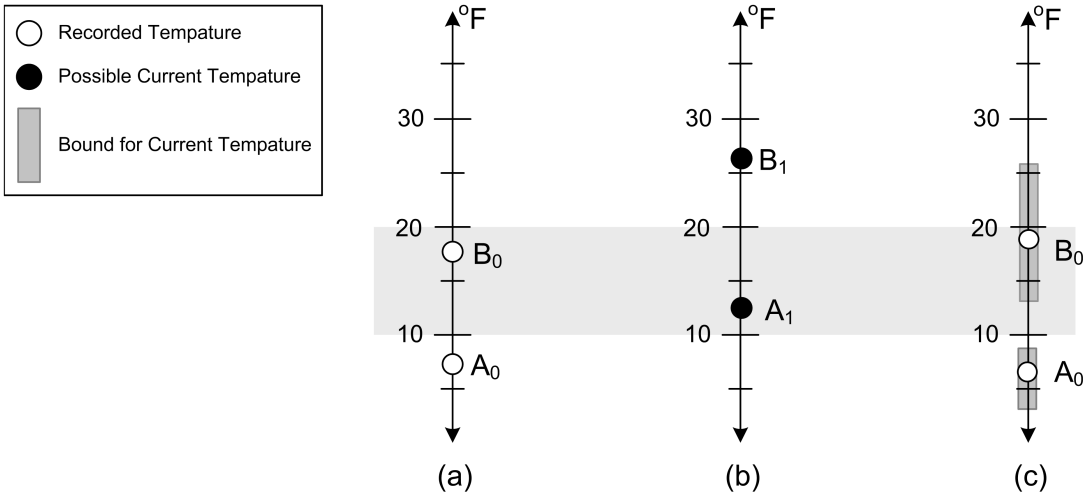

Figure 1: Example of data uncertainty and a range query for temperature values.

Whether room B satisfies the query is less obvious. In Figure 1(c), it is not clear whether B has a temperature between 10°F and 20°F. However, the fact that the uncertainty associated with B's temperature is *bounded* makes it possible to decide the degree of likelihood that B satisfies this query. In general, bounded uncertainty allows us to augment different levels of confidence with each answer (e.g. as a probability), instead of providing a definite answer. Queries that augment answers with probability values, based on uncertain information, are known as **probabilistic queries**. In the previous example, a probabilistic query produces answers such as: "Room A has a probability of 0 being between 10°F and 20°F, while Room B has a probability of 0.7". Contrast with a query that gives incorrect answers because of stale data, a probabilistic query provides more confidence in the answers since uncertainty is considered.

Depending on the nature of the probabilistic query, confidence in a probabilistic query answer can be expressed in different forms. We will study different classes of probabilistic queries, which have different forms of answers and evaluation techniques. We also examine the concept of "quality" of a probabilistic query result which is related to the ambiguity of the result.

The rest of this chapter is organized as follows. We first present the background and related works. Then we examine in detail how sensor data uncertainty can be modeled. Based on the data models, we present a classification of probabilistic queries. For each query class, we examine factors that determine quality of probabilistic query answers. Finally, we outline future research issues.

BACKGROUND

Approximate answers to queries based on a subset of data have been well studied. Vrbsky & Liu (1994) studied approximate answers for set-valued queries (where a query answer contains a set of objects) and single-valued queries (where a query answer contains a single value). An exact answer E can be approximated by two sets: a certain set C which is the subset of E, and a possible set P such that $C \cup P$ is a superset of E. Other techniques like precomputation (Poosala & Ganti, 1999), sampling (Gibbons & Matias, 1998) and synopses (Acharya, Gibbons, Poosala and Ramaswamy, 1999) are used to produce statistical results. While these efforts investigate approximate answers based upon a subset of (exact) values of the data, this chapter addresses imprecise answers that assume all (uncertain) values of the data.

There are a number of works about evaluation of intervals. Olston et al. discuss the problem of balancing the trade-off between precision and performance for querying replicated data (Olston & Widom, 2000; Olston, Loo & Widom, 2001; Olston & Widom, 2002). In their model, the cache in the server cannot keep track of the exact values of sensor sources due to limited network bandwidth. Instead of storing the actual value in the server's cache, an interval for each item is stored, within which the current value must be located. A query is then answered by using these intervals, together with the actual values fetched from the sources. The problem of minimizing the update cost within an error bound specified by aggregate queries is studied.

Probabilistic range querying over uncertainty of moving objects is presented by Wolfson, Sistla, Chamberlain & Yesha (1999). Cheng, Kalashnikov & Prabhakar (2004) present a solution for probabilistic nearest neighbor queries over moving objects. A generalized taxonomy of uncertainty models is presented by Cheng & Prabhakar (2003). Evaluation techniques and quality metrics are studied by Cheng, Kalashnikov & Prabhakar (2003).

SENSOR UNCERTAINTY MODELS

Uncertainty of sensor values can be represented in three forms, from no attention to uncertainty at all, to the highest resolution of uncertainty information (Cheng & Prabhakar, 2003). For notational convenience, let us assume that a real-valued attribute a of a set of database objects T is queried. We name the ith object of T as T_i, and the value of a for T_i as $T_i.a$ (where $i = 1,\ldots, |T|$).

1. Point Uncertainty. This is the simplest model, where we assume there is no uncertainty associated with data at all. Each data item, $T_i.a$, is supposed to be a correct representation of the external entity being monitored. Queries use these exact values to evaluate results. Although manipulating real values is relatively easy for a query, the example in Figure 1 illustrates how such data can lead to incorrect query results.

2. Interval Uncertainty. Instead of representing the exact sensor value in the database, an *uncertain interval*, denoted by $U_i(t)$, is stored. Specifically, $U_i(t)$ is a close interval $[l_i(t),u_i(t)]$, where $l_i(t)$ and $u_i(t)$ are real-valued functions of t,

bounding the value of $T_i.a$ at time t. This model represents imprecision of data in the form of an interval. An example model of $U_i(t)$ is an interval bounding all values within a distance of $(t\text{-}t_{update}) \times r$ of $T_i.a$, where t_{update} is the time that $T_i.a$ is last updated, and r is the current rate of change of $T_i.a$. Thus $U_i(t)$ expands linearly with time until the next update of $T_i.a$ is received. Another realization of this model can be found in Wolfson, Sistla, Chamberlain & Yesha (1999).

3. Probabilistic Uncertainty. This model is proposed by Cheng, Kalashnikov & Prabhakar, 2003. Compared with interval uncertainty, it requires one more piece of information – the probability density function (pdf) of $T_i.a$ within $U_i(t)$. We call this function an *uncertain pdf of* $T_i.a$ at time t, denoted by $f_i(x,t)$. Notice that $\int_{l_i(t)}^{u_i(t)} f_i(x,t)dx = 1$ and $f_i(x,t)$ equals 0 if $x \notin U_i(t)$. Further, the exact form of $f_i(x,t)$ is application-dependent. For example, in modeling sensor measurement uncertainty, where each U_i is an error range containing the mean value, $f_i(x,t)$ can be a normal distribution around the mean value. Another example is the modeling of one-dimensional moving objects, where at any point in time, the actual location is within a certain bound, d, of its last reported location value. If the actual location changes further than d, then the sensor reports its new location value to the database and possibly changes d. Then $U_i(t)$ contains all the values within a distance of d from its last reported value (Wolfson, Sistla, Chamberlain & Yesha, 1999), and $f_i(x,t)$ can be a uniform distribution, i.e.,

$$f_i(x,t) = 1/[\, u_i(t) - l_i(t)] \text{ for } T_i.a \in U_i(t)$$

which models the scenario when $T_i.a$ has an equal chance of locating anywhere in $U_i(t)$. Alternatively, one may perform an estimation of the pdf based on time-series analysis, the discussion of which is beyond the scope of this chapter, and interested readers are referred to Chatfield (1989) for details.

As we can see, the probabilistic uncertainty model is the most complicated model of sensor uncertainty among the ones we presented. The complexity of the model is paid off, however, by the fact that probabilistic queries can be defined, which we discuss in the next section.

CLASSIFICATION OF PROBABILISTIC QUERIES

A **Probabilistic Query** assumes that data inputs are characterized under the probabilistic uncertainty model, and augments the answers with confidence, expressed in probability values. In this section we examine different types of probabilistic queries. In particular, we present a classification scheme of probabilistic queries proposed by Cheng, Kalashnikov & Prabhakar (2003).

Probabilistic queries can be classified in two ways.[1] First, we can classify them according to the forms of answers required. An **entity-based query** is one that returns a set of objects (e.g., list of objects that satisfy a range query), whereas a **value-based query** returns a single numeric value (e.g., value of a particular sensor). Another criterion is based on whether an aggregate operator, such as SUM and MAX, is used to produce results. An **aggregate query** is one where interplay

[1] We only consider "atomic" queries that perform a single operation (e.g., range query). Complex queries that involve two or more operators (e.g., a SUM operation over a range query) are not covered by this classification scheme.

between objects determines the results. For example, to decide whether an object holds the minimum value of $T_i.a$ requires us to examine other objects as well. In contrast, for a **non-aggregate** query, the suitability of an object as the result to an answer is independent of other objects. A range query is a typical example – whether an object satisfies the range query is not affected by the values of other objects.

Based on this classification, we obtain four classes of probabilistic queries. For the names of the queries defined, the first letter is either *E* for entity-based query or *V* for value-based query.

1. **Value-based Non-Aggregate Class.** This query class returns a single value for a given object as the only answer, and does not involve any aggregate operations. An example is the *Probabilistic Single Value Query* (**VSingleQ**), which returns the probabilistic uncertainty information (i.e., $U_i(t)$ and $f_i(x,t)$) of a given object T_i.
2. **Entity-based Non-Aggregate Class.** This query class returns a set of objects, where the satisfiability of each object to the query is independent of others. One such query is the *Probabilistic Range Query* (**ERQ**): given a closed interval $[l,u]$, a list of tuples (T_i,p_i) are returned, where p_i is the non-zero probability that $T_i.a \in [l,u]$.

3. **Entity-based Aggregate Class.** In this query class, an aggregate operator is involved, and a set of objects together with their probability values i.e., a list of n tuples (T_i,p_i) is returned, with $\sum_{i=1}^{n} p_i = 1$. A typical example is the *Probabilistic Minimum Query* (**EMinQ**): a set of tuples (T_i,p_i) are returned, where p_i is the non-zero probability that $T_i.a$ is the minimum among all items in T.

4. **Value-based Aggregate Class.** An aggregate operation is involved in deciding a single value to be returned. As an example, a *Probabilistic Sum Query* (**VSumQ**) yields $l,u \in \Re$ as well as $\{p(x) \mid x \in [l,u]\}$, where X is a random variable for the sum of values of a for all objects in T, and $p(x)$ is a pdf of X such that $\int_{l}^{u} p(x)dx = 1$.

This classification scheme allows us to develop evaluation algorithms for different types of probabilistic queries. Consider the evaluation of an ERQ. Since each object T_i can be evaluated independently of other objects, we first compute the overlapping interval *OI* of the two intervals, $U_i(t)$ and $[l,u]$. The probability that T_i satisfies the ERQ i.e., p_i, is then calculated by integrating $f_i(x,t)$ over OI. On the other hand, evaluating an EMinQ requires a completely different algorithm, details of which can be found in Cheng, Kalashnikov & Prabhakar (2003).

QUALITY OF PROBABILISTIC RESULTS

As mentioned before, a probabilistic query returns answers augmented with probability values, as opposed to definite answers produced by a traditional query. A certain degree of imprecision is incorporated into the query answer, expressed in the form of probability values. The imprecision of a query answer is due to the fact that data supplied to the probabilistic query are inherently uncertain. In particular, if the data interested to a query contain a lot of uncertainty (e.g., their uncertain intervals are large), a probabilistic query can yield ambiguous answers which may not be very useful to the user. Figure 2(a) shows four items with large uncertain intervals that overlap each other. Assume these items have the same uncertain pdf. We can see that it is difficult to decide which item has a larger chance of being the minimum: the probability that satisfies the EMinQ is roughly the same for the four items. The quality of the result to this EMinQ is said to be "poor", since its gives a vague answer, and reflects that there is too much uncertainty for the query to yield a reasonable answer. On the contrary, the uncertainty intervals in Figure 2(b) are smaller and apparently item *D* has a higher probability of having the minimum value, yielding a better quality for an EMinQ.

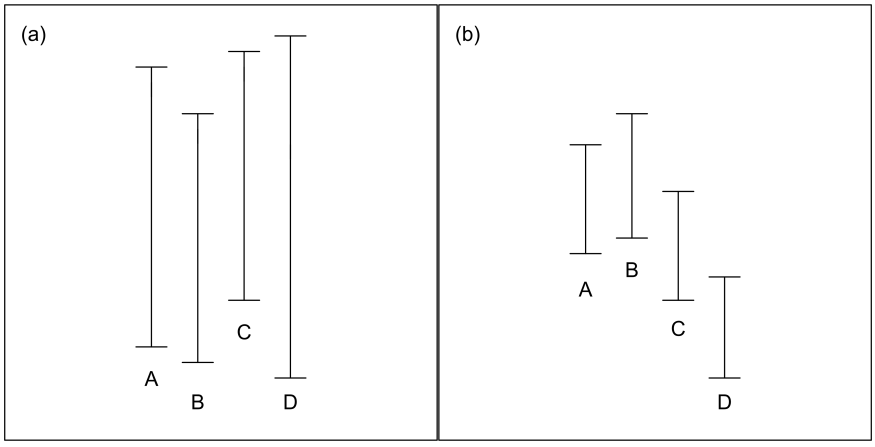

Figure 2: Uncertainty and Quality of Results. In (a), it is not clear which item has a dominant probability value of satisfying EMinQ, and the quality of the result is poor. In (b), item *D* has a higher chance of satisfying EMinQ, yielding a higher quality result.

The quality of a probabilistic result is an important issue. It enables us to determine if the uncertainty of any sensor data needs to be reduced (e.g., request an immediate update from the sensor) in order to improve the quality of result. In this section, we study two key factors that affect quality -- *size of uncertain intervals* and *entropy of probabilistic results*.

1. The size of an uncertain interval indicates the extent of error inherent to a data value. A larger uncertain interval can results in poorer answer quality, as illustrated in Figure 2. Another example is the evaluation of VSumQ over two uncertain data items T_i and T_j. The resulting value (i.e., $T_i.a + T_j.a$) lie in the error range $[l_i(t)+l_j(t), u_i(t)+u_j(t)]$. It is desirable to have smaller $U_i(t)$ and $U_j(t)$ in order to obtain a smaller error range, or higher answer quality.

2. Entropy of probabilistic results. Consider two sets of answers to an EMinQ: $\{(T_1,0.3),(T_2,0.2),(T_3,0.5)\}$ and $\{(T_1,0.8), (T_2,0.2)\}$. How do we know which answer is more informative? Here **entropy** gives us a convenient metric for

measuring uncertainty of probabilistic results. According to Shannon (1949), entropy is defined as follows:

Let $X_1,\ldots,X_n$ be all possible messages, with respective probabilities $p(X_1),\ldots, p(X_n)$ such that $\sum_{i=1}^{n} p(X_i) = 1$. Then the entropy of a message $X \in \{X_1,\ldots,X_n\}$ is defined as $H(X) = \sum_{i=1}^{n} p(X_i) \log_2 \frac{1}{p(X_i)}$.

Intuitively, $H(X)$ measures the amount of uncertainty associated with a random message X. The larger the value of $H(X)$, the more amount of uncertainty is in X.

We can use the idea of entropy to quantify the quality of query answers. Consider the entity-based non-aggregate query class, where each object is evaluated independent of other queries. The quality of each answer tuple (T_i,p_i) can be specified in terms of entropy. For example, in an ERQ, each object T_i has a probability p_i of satisfying the query, and $(1-p_i)$ otherwise. The entropy of (T_i,p_i) is then equal to $-(p_i \log_2 p_i+(1-p_i)\log_2(1- p_i))$. The overall entropy of an ERQ result is then equal to the average of entropy values over all tuples of (T_i,p_i) where $p_i \neq 0$.

For Entity-based Aggregate queries, recall that a set R of tuples (T_i,p_i) are returned, with $\sum_{i=1}^{n} p_i = 1$. We can thus use $H(R)$ to measure the answer uncertainty to these queries. The quality of the answer is lower if $H(R)$ is higher.

A continuous version of the entropy definition can be used for Value-based queries, defined as

$$\hat{H}(X) = \int_{l}^{u} p(x) \log_2 p(x) dx$$

where $\hat{H}(X)$ is called the *differential entropy* of continuous random variable X with probability density function $p(x)$ defined in the interval $[l,u]$.

For more details on answer quality metrics for different query classes, and heuristics that improve query quality, readers are referred to Cheng, Kalashnikov & Prabhakar (2003).

F U T U R E T R E N D S

Our next work is to study the indexing of probabilistic uncertain data in order to provide efficient access mechanisms to uncertain data. We also plan to study the join operation of uncertain data. We will also examine the efficient execution of a **probabilistic threshold query**, which is a probabilistic query with threshold λ – only objects with probability value greater than λ are included in the answer. A preliminary study by Cheng & Prabhakar (2003) reveals that this variant of probabilistic queries is worth further study because it can improve the efficiency of probabilistic query algorithms significantly, by exploiting the fact that only objects with probability values higher than λ are returned.

CONCLUSION

In this chapter, we studied how uncertainty information can be incorporated to data values. Under the probabilistic uncertainty model, we discussed probabilistic queries where confidence in the result is expressed in terms of probability values. Probabilistic queries can be classified into two dimensions, according to the nature of a query answer, as well as whether aggregation operations are involved. Each query class requires a different evaluation algorithm, as well as an answer quality metric. We also discuss how the sizes of uncertain intervals and entropy of the answer affect the quality of a probabilistic result.

REFERENCES

Acharya, S., Gibbons, P., Poosala, V., & Ramaswamy, S. (1999). Join synopses for approximate query answering. *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, 1999.

Chatfield, C. (1989). The analysis of time series an introduction. *Chapman and Hall*.

Cheng, R., Kalashnikov, D., & Prabhakar, S. (2003). Evaluating probabilistic queries over imprecise data. *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, June 2003.

Cheng, R., Kalashnikov, D., & Prabhakar, S. (2004). Querying imprecise data in moving object environments. *IEEE Transactions on Knowledge and Data Engineering*, 2004 (to appear).

Cheng, R., & Prabhakar, S. (2003). Managing uncertainty in sensor databases. *SIGMOD Record issue on Sensor Technology*, December 2003.

Gibbons P., & Matias, Y (1998). New sampling-based summary statistics for improving approximate query answers. *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, 1998.

Olston, C., Loo, B.T., & Widom, J.(2001). Adaptive precision setting for cached approximate values. *Proc. of the ACM SIGMOD 2001 Intl. Conf. on Management of Data*, 2001.

Olston, C., & Widom, J. (2000). Offering a precision-performance tradeoff for aggregation queries over replicated data. *Proc. of the 26th Intl. Conf. on Very Large Data Bases*, 2000.

Olston, C., & Widom, J. (2002). Best-effort cache synchronization with source cooperation. *Proc. of the ACM SIGMOD 2002 Intl. Conf. on Management of Data*, pages 73–84, 2002.

Poosala, V., & Ganti, V. (1999). Fast approximate query answering using precomputed statistics. *Proc. of the 15th Intl. Conf. on Data Engineering*, page 252, 1999.

Shannon, C. (1949). The Mathematical Theory of Communication. *University of Illinois Press*, 1949.

Vrbsky, S. V., & Liu, J. W. S.(1994). Producing approximate answers to set- and single-valued queries. *The Journal of Systems and Software*, 27(3), 1994.

Wolfson, O., Sistla, P., Chamberlain, S., & Yesha, Y.(1999). Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3), 1999.

Terms and Definitions

Sensor Uncertainty: An inherent property of a sensor-based application, where each recorded data item in the database is only an older, approximate version of the entity being monitored in the external environment**.**

Point Uncertainty: A model of sensor uncertainty where each stored data item is assumed to be a correct representation of the entity being represented.

Interval Uncertainty: A model of sensor uncertainty where each stored data value is represented by a closed interval, called *uncertain interval*.

Probabilistic Uncertainty: A model of sensor uncertainty, where each data value is represented by its *uncertain interval*, together with a probabilistic density function (called *uncertain pdf*) that describes the distribution of the values within the interval.

Probabilistic Query: A query which assumes data are characterized by probabilistic uncertainty, and returns query answers augmented with probability values.

Entity-based Query: A probabilistic query that returns a set of objects.

Value-based Query: A probabilistic query that returns a single value.

Probabilistic threshold query: A probabilistic query with probability threshold λ, where only objects with probability values greater than λ are included in the answer.