Distributed Processing of Probabilistic Top-k Queries in Wireless Sensor Networks

Mao Ye, Wang-Chien Lee, *Member*, *IEEE*, Dik Lun Lee, *Member*, *IEEE*, and Xingjie Liu, *Student Member*, *IEEE*

Abstract—In this paper, we introduce the notion of *sufficient set* and *necessary set* for distributed processing of probabilistic top-k queries in cluster-based wireless sensor networks. These two concepts have very nice properties that can facilitate localized data pruning in clusters. Accordingly, we develop a suite of algorithms, namely, *sufficient set-based (SSB)*, *necessary set-based (NSB)*, and *boundary-based (BB)*, for intercluster query processing with bounded rounds of communications. Moreover, in responding to dynamic changes of data distribution in the network, we develop an adaptive algorithm that dynamically switches among the three proposed algorithms to minimize the transmission cost. We show the applicability of sufficient set and necessary set to wireless sensor networks with both two-tier hierarchical and tree-structured network topologies. Experimental results show that the proposed algorithms reduce data transmissions significantly and incur only small constant rounds of data communications. The experimental results also demonstrate the superiority of the adaptive algorithm, which achieves a near-optimal performance under various conditions.

Index Terms—Top-k queries, distributed data management, probabilistic databases, wireless sensor networks

1 Introduction

Wireless sensor networks are revolutionizing the ways to collect and use information from the physical world. This new technology has resulted in significant impacts on a wide array of applications in various fields, including military, science, industry, commerce, transportation, and health-care. However, the quality of sensors varies significantly in terms of their sensing precision, accuracy, tolerance to hardware/external noise, and so on. For example, studies show that the distribution of noise varies widely in different photovoltic sensors [1], precision and accuracy of readings usually vary significantly in humidity sensors [2], and the errors in GPS devices can be up to several meters [3]. Thus, sensor readings are inherently uncertain [4], [5], [6].

To facilitate management of uncertain data, research on probabilistic databases have received renewed attentions in the past few years. Most of the recent works on probabilistic data modeling propose to associate a *confidence* (in form of probability) with a data record/tuple to capture the data uncertainty and thus carry a *possible worlds* semantic [7], [8], [9].[1] Accordingly, system issues such as indexing techniques [10], [11] and query processing [8], [6], [12], [13], [14] have been examined. Nevertheless, they have mostly been studied under a *centralized* system setting. In this paper, we explore the problem of processing probabilistic top-k queries in *distributed* wireless sensor networks.

Here, we first use an environmental monitoring application of wireless sensor network to introduce some basics of probabilistic databases.

Running example. Consider a wireless sensor network that consists of a large number of sensor nodes deployed in a geographical region. Feature readings (e.g., moisture levels or speed of wind gust) are collected from these distributed sensor nodes. Due to sensing imprecision and environmental interferences, the sensor readings are usually noisy. Thus, multiple sensors are deployed at certain zones in order to improve monitoring quality. In this network, sensor nodes are grouped into *clusters*, within each of which one of sensors is selected as the *cluster head* for performing localized data processing. By using statistic methods (e.g., [5]), a cluster head may generate a set of data tuples for each zone within its monitored region.[2] In this example, we assume that each tuple is comprised of tuple-id, zone, a derived possible attribute value, along with a confidence that serves as a measurement of data uncertainty. Thus, the data tuples corresponding to the same zone collectively represent the probabilistic distribution of derived possible values for the zone. Since the existence of possible values in these tuples are exclusive to each other, they naturally form a logical tuple, called *x-tuple*.[3]

Fig. 1 shows a wireless sensor network (with a two-tier hierarchical topology) that monitors the speed of wind gust in different zones (denoted as zones A, B, ..., F).[4] In this

1. We follow the conventional relational data model to refer data records as tuples for the rest of this paper.

- *M. Ye, W.-C. Lee, and X. Liu are with the Department of Computer Science and Engineering, Pennsylvania State University, IST Building, University Park, PA 16802. E-mail: {mxy177, wlee, xzl106}@cse.psu.edu.*
- *D.L. Lee is with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: dlee@cs.ust.hk.*

Manuscript received 12 Oct. 2010; revised 11 May 2011; accepted 17 June 2011; published online 23 June 2011.
Recommended for acceptance by J. Li.
For information on obtaining reprints of this article, please send e-mail to: take@computer.org, and reference IEEECS Log Number TKDE-2010-10-0548.
Digital Object Identifier no. 10.1109/TKDE.2011.145.

2. Note that zones and clusters are two independent organizations corresponding to different functions at the application and network levels. For simplicity of presentation, we assume a zone is located within a cluster in this example.

3. It is possible that all these possible values do not exist at the same time.

4. This network topology is widely adopted in data-centric sensor networks, such as LEACH [15] and StoneDB [16].

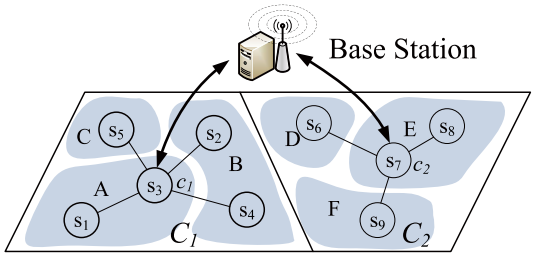


Fig. 1. A wireless sensor network. There are six zones, denoted as A, B, ..., F, which are organized into two clusters C_1 and C_2, with the corresponding cluster heads c_1 (i.e., s_3) and c_2 (i.e., s_7).

Tid	Loc.	Speed	Conf.	
t_1	A	89 km/h	0.9	T_1 (w.r.t. Cluster $\mathcal{C}_1$)
t_2	B	83 km/h	0.1	
t_3	A	81 km/h	0.1	
t_4	B	80 km/h	0.45	
t_5	C	75 km/h	0.5	
t_6	D	99 km/h	0.1	T_2 (w.r.t. Cluster $\mathcal{C}_2$)
t_7	E	94 km/h	0.5	
t_8	E	90 km/h	0.1	
t_9	F	85 km/h	0.2	

$\tau_A = \{t_1, t_3\}, \tau_B = \{t_2, t_4\}, \tau_C = \{t_5\}, \tau_D = \{t_6\}$
$\tau_E = \{t_7, t_8\}$ and $\tau_F = \{t_9\}$

Fig. 2. A global view of wind gust records.

network, sensor nodes are grouped into clusters, where cluster heads are responsible for local processing and to report aggregated results to the base station. As shown, c_1 and c_2 denote the cluster heads for clusters $\mathcal{C}_1$ and $\mathcal{C}_2$, correspondingly.

Fig. 2 shows two tables, T_1 and T_2, representing a global snapshot of gust wind records collected from the sensor network. As shown, each tuple records a *possible* wind speed corresponding to a location. The confidence value associated with a tuple indicates the existence probability of that particular wind speed. For example, there are two data tuples generated for Location A. The wind speeds in these two tuples are both valid (i.e., with measured confidences) but their presence is exclusive. In probabilistic databases, the exclusive presence of tuple instances in an x-tuple τ is dictated by the *x-relation* rule, denoted in the form of $\tau = \{tuple_1, tuple_2, \ldots\}$. In our running example, each location is associated with an x-relation rule. For instance, corresponding to location A, $\tau_A = \{t_1, t_3\}$, i.e., τ_A is an x-tuple for A and t_1 and t_3 are *alternative tuple instances* (or simply called *alternatives*) of τ_A. In our running example, x-tuples for locations A, B, E have two alternatives while those for other locations have only one alternative. Note that the overall confidence of an x-tuple is the sum of its alternatives' confidence values, which is less than or equal to 1 (e.g., the confidence of $\tau_A = 0.9 + 0.1 = 1$ and that of $\tau_B = 0.1 + 0.45 = 0.55$).

Top-k queries on uncertain data have received increasing interests recently [17], [13], [18], [19], [14]. Given a scoring function and a data set, a top-k query finds the highest ranked k tuples over all possible worlds. The answers to this query may vary in different possible worlds, making the problem interesting and challenging. Moreover, various semantics of top-k queries can be employed in different applications. There are several top-k query semantics and solutions proposed recently, including *U-Topk* and *U-kRanks* in [17], *PT-Topk* in [13], *PK-Topk* in [18], *expected rank* in [19], and so on. A common way to process probabilistic top-k queries is to first sort all tuples based on the scoring attribute, and then process tuples in the sorted order (until some termination condition is met) to compute the final answer set. Answer sets for the above-mentioned queries typically consist of highly ranked tuples in the sorted list because tuples positioned lowly usually do not have the required ranks and confidence to be included in the answer sets. While these works produce insightful results, they are conducted on a *centralized* database rather than in a distributed setting as we target on in this paper.

Only recently, a study on processing a probabilistic query with "expected rank" semantic in distributed systems has appeared [14]. Nevertheless, while focusing on optimizing the transmission bandwidth, the proposed techniques require numerous iterations of computation and communication, introducing tremendous communication overhead and resulting in long latency. As argued in [20], this is not desirable for many distributed applications, e.g., network monitoring, that require the queries to be answered in a good response time, with a minimized energy consumption. In this paper, we aim at developing energy efficient algorithms optimized for fixed rounds of communications. As shown later, our distributed algorithms for probabilistic top-k query processing are guaranteed to be completed with no more than two rounds of communications. Moreover, our approach is not designed only for a specific probabilistic top-k query semantics. Our proposal is a *general* approach for efficient processing of the probabilistic top-k queries in distributed wireless sensor networks. These objectives uniquely differentiate our work from [14].

Motivated by previous studies that show promises in processing queries in wireless sensor networks [21], [22], we employ an in-network processing approach that efficiently prunes and aggregates data in order to avoid redundant transmissions. The basic idea is to perform *data pruning* and *aggregation* at cluster heads such that only the data tuples required for final processing are transferred to the base station. Consider the running example in Fig. 2, where a tuple with larger speed value is ranked higher in our application scenario. Suppose a PT-Top-k [13] is issued to retrieve tuples with their top-k probabilities over all possible worlds greater than a predefined threshold p, e.g., $k = 2$ and $p = 0.5$. As shown, the data $T_1 = \{t_1, \ldots, t_5\}$ and $T_2 = \{t_6, \ldots, t_9\}$ are maintained at c_1 and c_2, respectively. Instead of transmitting the whole T_1 and T_2 to the base station, c_1 and c_2 submit only subsets of T_1 and T_2 to save energy. From T_1, we observe that the x-tuples $\tau_A = \{t_1, t_3\}$ has an aggregate probability of 1 and $\tau_B = \{t_2, t_4\}$ has an aggregate probability of 0.55. Obviously, t_5 could never be included in the final answer, as the probability for t_5 to be in the top-2 is not greater than $1 - P(\tau_B) = 0.45 < p$. In fact, any tuples ranked lower than t_4 will never be included in the final answer set. Therefore, from the point of view of cluster head c_1, it is "sufficient" for c_1 to send only the subset $T_1 - \{t_5\}$ to the base station. This observation leads to the notion of "sufficient set" to be introduced later.

Now let's consider T_2. From the local viewpoint of c_2, any tuple in T_2 may possibly be ranked in top-2. However, we can also observe that the probabilities for t_6, t_8, and t_9 to be in the top-2 list would never exceed 0.1, 0.1, and 0.2, respectively, which are lower than the probability threshold $p = 0.5$. Thus, they are for sure not to be part of the final answer. Therefore, for cluster head c_2, a "necessary" set of uncertain tuples $\{t_6, t_7\}$ can be derived for delivery to the base station. This leads to our proposed notion of "necessary set." Note that while t_7 is the only qualified *candidate tuple*, cluster head c_2 has to deliver t_6 to the base station because it anticipates that the confidence of t_6 (ranked higher than t_7) may be required for computing the top-2 probability of t_7. Thus, we name t_6 as a *computation tuple*, which is useful for computing the top-k probability of the final result at the base station. Since the computation tuples are only selected based on the local view of their cluster heads, e.g., t_6 is selected by c_2 because of t_7, it's possible that more tuples may be needed at the base station to assist the probability computation for candidate tuples from other clusters. For example, t_1 is a candidate tuple and t_8 is ranked higher than t_1, thus the base station may request t_8 as supplementary data via an extra round of communication with the cluster head c_2.

While our running example above illustrates the sufficient set and necessary set with data tuples in separate clusters, i.e., T_1 and T_2, respectively, they are applicable to all the local clusters in the network. These two notions, with very nice properties to facilitate efficient localized data pruning in clusters, actually provide a theoretical foundation for our design of energy-efficient distributed query processing algorithms. Notice that, while the necessary set is usually a subset of the sufficient set, it may require extra communications for the base station to request for missing computation tuples. This represents a tradeoff between the two ideas.

Armed with sufficient set and necessary set, we develop a suite of algorithms for processing probabilistic top-k queries in two-tier hierarchical wireless sensor networks with PT-Top$k$ as a case study, including 1) sufficient set-based (SSB) algorithm, 2) necessary set-based (NSB) algorithm, and 3) boundary-based (BB) algorithm. Moreover, we developed an adaptive algorithm that dynamically switches among the three proposed algorithms to minimize the communication and energy overhead, in responding to changing data distribution in the network. Furthermore, we discuss how to apply sufficient set and necessary set to devise a series of algorithms, namely SSB-T, NSB-T, and optimized NSB-T (NSB-T-Opt), for processing probabilistic top-k in a sensor network with tree topology. Finally, we evaluate the proposed algorithms both in two-tier hierarchy (i.e., SSB, NSB, BB) and tree topology (i.e., SSB-T, NSB-T, NSB-T-Opt) in comparison with two baseline approaches. We also perform sensitivity tests to observe the behavior of examined algorithms under various parameter settings. The result validates our ideas and shows the superiority of our proposal.

This paper is the full version of our preliminary work published as a short paper in ICDE'10, where we introduce the targeted problem and propose the idea of employing sufficient set to develop the SSB algorithm for distributed processing of probabilistic top-k queries [23]. In this paper, we extend the idea of sufficient set with the notion of necessary set and sufficient/necessary boundaries; develop new distributed processing algorithms; and demonstrate the extendability our ideas to a sensor network with tree topology. Additionally, we examine a new issue of changing data distribution. In summary, our research makes the following important contributions:

- This work supports in-network top-k query process over uncertain data in a distributed wireless sensor network. Although the case study is on PT-Top$k$ query, the general concepts are applicable to other top-k query variants.
- We develop the notion of sufficient set and necessary set for efficient in-network pruning of uncertain data in a distributed setting. Additionally, we systematically derive boundaries for both sufficient and necessary sets. This notion, along with its nice properties, provides a theoretical basis for the distributed query processing algorithms.
- Based on the notion of sufficient and necessary sets, we propose a suite of algorithms for in-network processing of PT-Top$k$ queries in a two-tier hierarchical sensor network. These algorithms exploit individual and combined strengths of sufficient and necessary sets in query processing.
- We develop a cost model on communication cost of the three proposed algorithms. Accordingly, we propose a cost-based adaptive algorithm that dynamically switches among the three algorithms based on their estimated costs.
- We apply sufficient set and necessary set to sensor networks with tree topology, to further improve query processing performance by facilitating sophisticated in-network filtering at the intermediate nodes along the routing path to the root.
- An extensive performance evaluation is conducted to compare the proposed algorithms along with two baseline approaches. Experimental result validates our ideas and shows that the proposed algorithms reduce data transmissions significantly without incurring excessive rounds of communications.

The rest of this paper is organized as follows. Section 2 provides a background for this study. Section 3 introduces the notion of sufficient set and necessary set. Section 4 presents the proposed query processing algorithms in a two-tier hierarchical network topology and Section 5 introduces an adaptive algorithm. Section 6 extends the ideas of sufficient set and necessary set to sensor networks with tree topology. Section 7 validates our proposals and evaluates their performance. Finally, Section 8 concludes the paper.

2 Preliminaries

In this section, we first define the PT-Top$k$ query, introduce the centralized algorithms for its query processing, and review some related works.

2.1 System Model and Problem Definition

In this paper, we assume that a wireless sensor network consisting of a base station and N sensor nodes is deployed in a monitoring field. With a continuous power supply, the

Possible World	Prob.	Possible World	Prob.
$W_1 = \{t_1\}$	0.2025	$W_7 = \{t_3\}$	0.0225
$W_2 = \{t_1, t_2\}$	0.045	$W_8 = \{t_3, t_2\}$	0.005
$W_3 = \{t_1, t_4\}$	0.2025	$W_9 = \{t_3, t_4\}$	0.0225
$W_4 = \{t_1, t_2, t_5\}$	0.045	$W_{10} = \{t_3, t_2, t_5\}$	0.005
$W_5 = \{t_1, t_4, t_5\}$	0.2025	$W_{11} = \{t_3, t_4, t_5\}$	0.0225
$W_6 = \{t_1, t_5\}$	0.2025	$W_{12} = \{t_3, t_5\}$	0.0225

Fig. 3. Possible worlds of T_1 in Fig. 2.

base station serves as the data collection and processing center to the external users and applications. We assume M clusters are formed and a cluster head is selected for each cluster. The whole sensor network can be logically treated as a *distributed uncertain database*. Sensor readings are collected by cluster heads and transformed into uncertain data, which collectively form as an uncertain table T. In other words, the content of T, i.e., a set of uncertain tuples $\{t_1, t_2, \ldots, t_N\}$, is distributed in cluster heads within the sensor network. To capture the data uncertainty, each tuple $t \in T$ is associated with a confidence $P(t) > 0$, i.e., the uncertain table T carries a possible world semantics [7], [8], [9]. A set of *x-relation rules* are derived for tuples reflecting the same monitored phenomenon (e.g., wind speed in location A). Thus, only one tuple in the same x-relation rule may appear in a possible world. Note that, we assume sensor nodes within the same location are clustered together. Thus x-relation rules, which are derived by cluster heads, are only applicable to tuples in the same cluster. In other words, tuples in the same x-relation rule are correlated and corresponding to the same location. We assume tuples with different x-relation rules are independent and corresponding to different locations. Moreover, we assume that each tuple in T involves in one and only one x-relation rule.

An x-relation rule is specified in the form of $\tau = \{t_1, t_2, \ldots, t_m\}$, where $m = |\tau|$ and t_i $(1 \leq i \leq m)$ are *alternative tuples*, e.g., $\tau_A = \{t_1, t_3\}$. The collection of alternative tuples in the same rule can be seen as one logical tuple and thus is called an *x-tuple*. Let $\Upsilon(T) = \{\tau_1, \tau_2, \ldots, \tau_g\}$ (where g is the total number of x-tuples in T) be the set of x-relation rules specified on T. The *aggregate confidence* of an x-tuple τ is the sum of the confidence values of all its alternative tuples, i.e., $P(\tau) = \sum_{t \in \tau} P(t)$.

Let W denote a possible world which consists of a subset of tuples in T and $\mathcal{W}$ denote the set of all possible worlds. The probability that $W \in \mathcal{W}$ exists is

$$P(W) = \prod_{(\tau \in \Upsilon(T)) \wedge (\tau \cap W = \{t\})} P(t) \prod_{(\tau \in \Upsilon(T)) \wedge (\tau \cap W = \emptyset)} (1 - P(\tau)).$$

Since at most one tuple of each x-tuple exists in any $W \in \mathcal{W}$ and tuples belonging to different x-tuples are independent of each other, the first and second terms above capture the probabilities of the x-tuples that do and do not appear in $W \in \mathcal{W}$, respectively. For illustration, Fig. 3 shows all the possible worlds of the uncertain table T_1 in the example of Fig. 2.

As mentioned, we use PT-Top$k$ [13] as a test case for our proposed ideas, we first define *the top-k probability* of a tuple and then define the PT-Top$k$ query.

Definition 1 (Top-k Probability). *Let A_W denote the top-k answer set in a possible world W. Given a tuple $t \in T$, the top-k probability of t is the aggregate probability of t being in the top-k answers over all $W \in \mathcal{W}$, i.e.,*

$$P_{topk}(t) = \sum_{W \in \mathcal{W}, t \in A_W} P(W). \quad (1)$$

For example, consider a PT-Top$k$ query with $k = 2$ and $p = 0.5$ on data set T_1 in Fig. 2. We can calculate $P_{topk}(t_1) = P(W_1) + P(W_2) + P(W_3) + P(W_4) + P(W_5) = 0.9$ and $P_{topk}(t_5) = P(W_6) + P(W_{12}) = 0.225$.

Definition 2 (Probabilistic Threshold Top-k Query (PT-Topk) [13]). *Given a probability threshold p $(0 < p \leq 1)$, PT-Topk finds the set of tuples whose top-k probabilities are at least p.*

2.2 Centralized PT-Topk Query Processing

In this section, we present a general approach for processing PT-Top$k$ queries in a centralized uncertain database, which provides a good background for the targeted distributed processing problem.

Given an uncertain table T, we first sort T in accordance with the ranking function f such that $t_1 \prec_f t_2 \prec_f \cdots \prec_f t_n$. The query answer can be obtained by examining the tuples in descending ranking order from the sorted table (which is still denoted as T for simplicity). We can easily determine that the highest ranked k tuples are definitely in the answer set as long as their confidences are greater than p since their qualifications as PT-Top$k$ answers are not dependent on the existence of any other tuples. Nevertheless, the qualification of tuple t_{k+1} as a PT-Top$k$ answer is dependent on 1) whether there exist some possible worlds where the tuples in front of t_{k+1} belong to less than k x-tuples; and 2) whether the aggregated confidence of t_{k+1} over these possible worlds are greater than p. If the answer is positive, then tuple t_{k+1} is included in the answer set and tuple t_{k+2} is examined. This process continues until there are no more qualified tuples left to be examined.

To shed more light on probabilistic PT-Top$k$ query processing, let's consider the top-k probability for tuple t_i, where $i > k$. Based on (1), $P_{topk}(t_i)$ is the sum of the probabilities that t_i ranks higher than k in a possible world. In other words, the top-k probability of t_i is the sum of possible world probabilities with less than k tuples ranking higher than t_i. Let $r_{i-1,l}[T]$ denote a probability of possible worlds in which l of the highest $i-1$ tuples in T (i.e., those placed in front of t_i) do exist. We use $r_{i-1,l}$ for simplicity when it causes no confusion. The top-k probability of t_i is as follows:

$$P_{topk}(t_i) = P(t_i) \cdot \sum_{l=0}^{k-1} r_{i-1,l}[T]. \quad (2)$$

The above computation involves only the first $i-1$ tuples in T except for those in the same x-relation with t_i. Thus, we use D_{t_i} to denote $\{t_x | t_x \in T, 1 \leq x \leq i-1\} - \{t_x | t_i \in \tau \wedge t_x \in \tau\}$. Thus,

$$P_{topk}(t_i) = P(t_i) \cdot \sum_{l=0}^{k-1} r_{i-1,l}[D_{t_i}]. \quad (3)$$

In the following, we use an example to show how the top-k probability is calculated. Given a PT-Top2 query issued against the uncertain table T_1 (sorted by decreasing speed) as shown in Fig. 2, the top-2 probabilities of t_1 and t_2, are the same as their confidences, i.e., $P_{top2}(t_1) = P(t_1) = 0.9$ and $P_{top2}(t_2) = P(t_2) = 0.1$. Note that both t_1 and t_2 must be within the top-2 list in any possible world, as long as they exist there. For t_3, we have t_1 and t_3 belong to the same x-relation, i.e., $t_1, t_3 \in \tau_A$. Therefore, $D_{t_3} = \{t_1, t_2, t_3\} - \tau_A = \{t_2\}$, and $P_{top2}(t_3) = P(t_3)(r_{2,1}[D_{t_3}] + r_{2,0}[D_{t_3}]) = P(t_3) \cdot (P(t_2) + P(\overline{t_2})) = 0.1 \cdot (0.1 + (1 - 0.1)) = 0.1$, where $P(\bar{t})$ denotes the probability that tuple t does not exist.

2.3 Related Work

Here, we review representative work in the areas of 1) top-k query processing in wireless sensor networks, and 2) top-k query processing on uncertain data.

Top-k query processing in sensor networks. An extensive number of research work in this area has appeared in the literature [21], [24], [25], [26]). Due to the limited energy budget available at sensor nodes, the primary issue is how to develop energy-efficient techniques to reduce communication and energy costs in the networks. TAG [21] is one of the first studies in this area. By exploring the semantics of aggregate operators (e.g., sum, avg, and top-k), in-network processing approach is adopted to suppress redundant data transmissions in wireless sensor networks. Approximate-based data aggregation techniques have also been proposed [27], [25]. The idea is to tradeoff some data quality for improved energy efficiency. Silberstein et al. develop a sampling-based approach to evaluate approximate top-k queries in wireless sensor networks [26]. Based on statistical modeling techniques, a model-driven approach was proposed in [5] to balance the confidence of the query answer against the communication cost in the network. Moreover, continuous top-k queries for sensor networks have been studied in [28] and [29]. In addition, a distributed threshold join algorithm has been developed for top-k queries [24]. These studies, considering no uncertain data, have a different focus from our study.

Top-k query processing on uncertain data. While research works on conventional top-k queries are mostly based on some deterministic scoring functions, the new factor of tuple membership probability in uncertain databases makes evaluation of probabilistic top-k queries very complicated since the top-k answer set depends not only on the ranking scores of candidate tuples but also their probabilities [8]. For uncertain databases, two interesting top-k definitions (i.e., U-Top$k$ and U-kRanks) and A^*-like algorithms are proposed [17]. U-Top$k$ returns a list of k tuples that has the highest probability to be in the top-k list over all possible worlds. U-kRanks returns a list of k tuples such that the ith record has the highest probability to be the ith best record in all possible worlds. In [13], PT-Top$k$ query, which returns the set of tuples with a probability of at least p to be in the top-k lists in the possible worlds, is studied. Inspired by the concept of dominate set in the top-k query, an algorithm which avoids unfolding all possible worlds is given. Besides, a sampling method is developed to quickly compute an approximation with quality guarantee to the answer set by drawing a small sample of the uncertain data. In [19], the expected rank of each tuple across all possible worlds serves as the ranking function for finding the final answer. In [30], U-Top$k$ and U-kRank queries are improved by exploiting their stop conditions. In [31], all existing top-k semantics have been unified by using generating functions.

Recently, a study on processing top-k queries over a distributed uncertain database is reported in [14] and [23]. Li et al. [14] only support top-k queries with the expected ranking semantic. On the contrary, our proposal is a general approach which is applicable to probabilistic top-k queries with *any* semantic. Furthermore, instead of repeatedly requesting data which may last for several rounds, our protocols are guaranteed to be completed within no more than two rounds. These differences uniquely differentiate our effort from [14]. Our previous work [23] as the initial attempt only includes the concept of sufficient set. In this paper, besides of sufficient set, we propose another important concept of necessary set. With the aid of these two concepts, we further develop a suite of algorithms, which show much better performance than the one in [23]. Probabilistic ranked queries based on uncertainty at the attribute level are studied in [32], [33], and [19]. A unique study that ranks tuples by their probabilities satisfying the query is presented in [12]. Finally, uncertain top-k query is studied under the setting of streaming databases where a compact data set is exploited to support efficient slide window top-k queries [18].

3 Intracluster Data Pruning

In a cluster-based wireless sensor network, the cluster heads are responsible for generating uncertain data tuples from the collected raw sensor readings within their clusters. To answer a query, it's natural for the cluster heads to prune redundant uncertain data tuples before delivery to the base station in order to reduce communication and energy cost. The key issue here is how to derive a *compact* set of tuples essential for the base station to answer the probabilistic top-k queries. This is a very challenging issue for the following reasons: 1) the interplay of probability and ranking due to the semantic of probabilistic top-k queries; and 2) the lack of global knowledge to determine the probability and ranking of candidate tuples *locally* at cluster heads. In this section, we propose the notion of *sufficient set* and *necessary set*, and describe how to identify them from local data sets at cluster heads. Next, we use the PT-Top$k$ query as a test case to derive sufficient set and necessary set and show that the top-k probability of a tuple t obtained locally is an upper bound of its true top-k probability. Thus, *data tuples excluded from the sufficient sets and necessary sets in local clusters will never appear in the final answer set*.

3.1 Definition of Sufficient and Necessary Sets

It would be beneficial if cluster heads are able to find the minimum sets of their local data tuples that are sufficient for the base station to answer a given query. Ideally, sufficient set is a subset of the local data set. Data excluded from the sufficient set, no matter which clusters they reside, will never be included in the final answer set nor involved

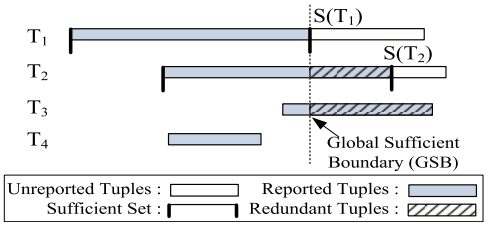


Fig. 4. Data distribution over the wireless sensor network.

in the computation of the final answer set. Here, we define the sufficient set more formally as follows:

Definition 3 (Sufficient Set). *Given an uncertain data set T_i in cluster $\mathcal{C}_i$, if there exists a tuple $t_{sb} \in T_i$ (called sufficient boundary) such that the tuples ranked lower than t_{sb} are useless for query processing at the base station, then the sufficient set of T_i, denoted as $S(T_i)$, is a subset of T_i as specified below:*

$$S(T_i) = \{t | t =_f t_{sb} \text{or} t \prec_f t_{sb}\},$$

where f is a given scoring function for ranking.

Note that a sufficient boundary may not exist for a given data set; then we consider the whole data set as a sufficient set and will discuss it in more detail later.

Theorem 1. *Given M clusters in a sensor network, collecting $S(T_i)$ or T_i (when $S(T_i)$ does not exist for $\mathcal{C}_i$) from clusters $\mathcal{C}_i$ where $i = 1..M$ is sufficient for the base station to answer a query.*

Proof. The proof is sketched below. Meanwhile, we use Fig. 4 for illustration. Suppose there are $M(= M_1 + M_2)$ clusters within the wireless sensor network, where M_1 clusters have sufficient sets, while M_2 clusters do not have one (e.g., $M_1 = 2, M_2 = 2$ in Fig. 4). We call the highest ranked sufficient boundary among all clusters the global sufficient boundary (GSB). Based on Definition 3, any tuple ranked lower than GSB is not in the final answer. Thus, only tuples ranked higher than GSB are of interest to the base station. We consider three cases to prove that all the tuples ranked higher than GSB are in the base station: 1) the clusters without sufficient sets (e.g., T_3 and T_4 in Fig. 4)—all tuples are transmitted; 2) the cluster holding the highest sufficient boundary GSB (i.e., $S(T_1)$)—we can simply agree that all the necessary data tuples are sent to the base station; and 3) any cluster holding a sufficient boundary lower than GSB (i.e., T_2)—all the tuples ranked higher than GSB are safely contained within their sufficient sets and delivered to the base station. Hence, it is sufficient for cluster heads $c_i (i = 1..M)$ to deliver $S(T_i)$ or T_i (if $S(T_i)$ does not exist) for top-k query processing. □

Next, we discuss the notion of necessary set. It consists of a set of data tuples *locally* known to be necessary for the base station to process the query (ideally in one round of communications). It is a subset of the local data set as well, including all the locally qualified candidate tuples and all higher ranked computation tuples absolutely needed for computing top-k probability of the final answer. Thus, the necessary set is defined as follows.

Definition 4 (Necessary Set). *Given a local data set T_i in cluster $\mathcal{C}_i$, assume that A_i is the set of locally known candidate tuples for the final answer and t_{nb} (called necessary boundary) is the lowest ranked tuple in A_i. The necessary set of T_i, denoted as $N(T_i)$, is*

$$N(T_i) = \{t | t \in T_i, t \prec_f t_{nb}\}.$$

Note that, at the base station, for a given cluster, the probability computation of some candidate tuples from other clusters may still require data tuples outside its necessary set. In other words, while the data tuples in the necessary sets include the final answer set, they may not be sufficient to determine the final answer set. In the following theorem, we depict a relationship between the sufficient set and necessary set.

Theorem 2. *Given a data set T_i in cluster $\mathcal{C}_i$ where the sufficient set exists, the necessary set $N(T_i)$ is a subset of the sufficient set $S(T_i)$, i.e., $N(T_i) \subseteq S(T_i)$.*

Proof. We prove the theorem by contradiction. Suppose we have a necessary set denoted by $N(T_i)$ which is a proper superset of the sufficient set $S(T_i)$ (i.e., $N(T_i) \supset S(T_i)$). Let the sufficient boundary be $SB(T_i)$ and the necessary boundary be $NB(T_i)$, then $SB(T_i) \prec_f NB(T_i)$. According to Definition 3, we know that any tuples ranked lower than $SB(T_i)$ can be safely pruned from transmission. On the other hand, if $NB(T_i)$ can be pruned, it cannot be part of the necessary set $N(T_i)$. This contracts our assumption. Therefore, $NB(T_i)$ ranks higher than $SB(T_i)$. According to the definitions of sufficient set and necessary set, we conclude that the necessary set must be a subset of the sufficient set, if the sufficient set exists. □

3.2 Case Study—PT-Top$k$ Query

While the idea we develop is generally applicable to various probabilistic top-k queries, here we use the PT-Top$k$ query to examine the notion of sufficient set and necessary set. To facilitate our discussion, we term the top-k probability computed based on a local data set as *local top-k probability* (denoted as $P_{L,topk}(.)$) and the top-k probability based on the entire data sets in the network as *global top-k probability* (i.e., $P_{topk}(.)$). We show that the local top-k probability of a tuple t is an upper bound of the its global top-k probability.

Lemma 1. *Given a tuple $t \in T_i$ in a cluster $\mathcal{C}_i$, we have $P_{topk}(t) \leq P_{L,topk}(t)$.*

Proof. Let T' represent the aggregation of data sets received from clusters other than $\mathcal{C}_i$. Thus, T_i are to be merged with T' at the base station in order to compute the final answer. For a given tuple $t \in T_i$, we prove that $P_{L,topk}(t)$ is greater than or equivalent to $P_{topk}(t)$ (i.e., the global top-k probability based on $T' \cup T_i$) by induction. As the tuples in T' are grouped in x-tuples, we assume that T' consists of m x-tuples, i.e., $T' = \{\tau_1, \tau_2, \ldots, \tau_m\}$. Let T^0 be T_i. We use $T^p = T^{p-1} \cup \{\tau_p\}$ $(p = 1..m)$ to denote a series of intermediate steps in the merging process and use $P^p_{topk}(t)$ to denote the top-k probability computed based on T^p. Finally, we use $\tau_{p,t}$ to denote the subset of τ_p which includes only those tuples ranked higher than t. The proof is sketched by proving $P^p_{topk}(t) \leq P_{L,topk}(t)$ in the following inductive steps $p = 1..m$.

For the base step $p = 1$, i.e., $T^1 = T_i \cup \{\tau_1\}$. There are two possible cases to consider.

Case 1: $\tau_{1,t}$ is empty, i.e., there is no tuple in τ_1 ranked higher than t. Thus, $P^1_{topk}(t)$ computed based on T^1 remains the same as $P^0_{topk}(t)$ which is $P_{L,topk}(t)$.

Case 2: $\tau_{1,t}$ is not empty. Let $P(\tau_{x,t})$ denote the existence probability of $\tau_{x,t}$. We have

$$P^1_{topk}(t) = P(\tau_{x,t}) \cdot P^0_{top(k-1)}(t) + [1 - P(\tau_{x,t})] \cdot P^0_{topk}(t).$$

Since $P^0_{top(k-1)}(t) \leq P^0_{topk}(t)$, we have $P^1_{topk}(t) \leq P^0_{topk}(t)$.

Combining the above two cases, we conclude that $P^1_{topk}(t) \leq P^0_{topk}(t)$. Since $P^0_{topk}(t) = P_{L,topk}(t)$, we have $P^1_{topk}(t) \leq P_{L,topk}(t)$.

For an inductive step $p = q$, where $1 < q \leq m$, assume that $P^p_{topk}(t) \leq P_{L,topk}(t)$ has been shown true for all the previous steps $p = 2..q-1$. We consider $T^q = T^{q-1} \cup \{\tau_q\}$. Similar to the base step, we can show that $P^q_{topk}(t) \leq P^{q-1}_{topk}(t)$. Hence, $P^q_{topk}(t) \leq P_{L,topk}(t)$ also holds in this step.

Based on the above steps, we conclude that given a tuple $t \in T_i$, we have $P_{topk}(t) \leq P_{L,topk}(t)$. □

Here, we first define the sufficient boundary for PT-Top$k$ queries as follows.

Definition 5 (PT-Topk Sufficient Boundary). *Given an uncertain data set T_i in cluster $\mathcal{C}_i$. The sufficient boundary of T_i, $SB(T_i)$ is the highest ranked tuple t_{sb} such that the following conditions are satisfied*

Condition 1:

$$\forall t_x \in T_i - D, (1 - P(\tau_{x,t_{sb}})) \sum_{j=1}^{k} r_{sb,j-1}[D_{t_x}] < p, \qquad (4)$$

where $\tau_{x,t_{sb}}$ is the subset of x-tuple τ_x which includes only tuples ranked higher than t_{sb}; $P(\tau_{x,t_{sb}}) = \sum_{t_j \in \tau_{x,t_{sb}}} P(t_j)$; $D = \{t_j | t_j \in T_i, 1 \leq j \leq sb\}$; and $D_{t_x} = D - \tau_x$.

Condition 2:

$$\sum_{j=1}^{k} r_{sb,j-1}[D] < p. \qquad (5)$$

In Definition 5, Condition 1 ensures that any local tuple $t \in T_i$ ranked lower than t_{sb} has its top-k probability smaller than p. Let t belong to the x-tuple τ_x. Since $P(\tau_{x,t_{sb}})$ represents the probability of tuples in τ_x that are ranked higher than t_{sb}, we have $P(t) \leq (1 - P(\tau_{x,t_{sb}}))$. Thus, the top-$k$ probability of t is smaller than p. Note that the first condition only considers the local tuples that belong to T_i. To make the locally determined sufficient boundary holds for other clusters, Condition 2 ensures that any nonlocal tuple $t' \notin T_i$ ranked lower than t_{sb} has a its top-k probability smaller than p. Because t' does not belong to T_i, it does not belong to any x-tuple appearing before t_{sb} (i.e., those in D).

Hence, the top-k probability of t' must be smaller than p, even if $P(t') = 1$. Notice that Condition 2 may become redundant when there exists a tuple $t_x \in T_i$ such that $P(\tau_{x,t_{sb}}) = 0$. However, this scenario does not always happen and thus Condition 2 is needed.

Based on the notion of sufficient boundary, we develop Theorem 3 which provides the basis for our development of sufficient-set-based algorithm later.

Theorem 3. *Given an uncertain data set T_i in cluster $\mathcal{C}_i$, any tuple ranked below the PT-Topk sufficient boundary of T_i (i.e., t_{sb}) does not qualify for the final answer.*

Proof. Suppose a tuple t_x is ranked lower than t_{sb}. We consider the local top-k probability of t_x in the following two cases.

Case 1, $t_x \in T_i - D$, where $D = \{t_j | t_j \in T_i, 1 \leq j \leq sb\}$. Due to the mutual exclusiveness of x-tuple members, t_x may appear in a possible world only if no other members of τ_x coexists in this possible world. Therefore, we have

$$\begin{aligned} P_{L,topk}(t_x) &= P(t_x) \sum_{j=0}^{k-1} \left(r_{sb,j}[D_{t_x}] \cdot \sum_{s=0}^{k-j-1} \xi_s \right) \\ &\leq P(t_x) \sum_{j=1}^{k} r_{sb,j-1}[D_{t_x}] \qquad (6) \\ &\leq (1 - P(\tau_{x,t_{sb}})) \cdot \sum_{j=1}^{k} r_{sb,j-1}[D_{t_x}] < p, \end{aligned}$$

where ξ_s is the probability that s tuples between t_{sb} and t_x appears, and therefore, $\sum_{s=0}^{k-j-1} \xi_s \leq 1$.

Case 2, $t_x \notin T_i$. t_x is not a member of any x-tuple in D. Therefore, the local top-k probability of t_x computed based on T_i is

$$\begin{aligned} P_{L,top_k}(t_x) &= P(t_x) \sum_{j=0}^{k-1} \left(r_{sb,j}[D] \cdot \sum_{s=0}^{k-j-1} \xi_s \right) \\ &\leq \sum_{j=1}^{k} r_{sb,j-1}[D] < p. \end{aligned}$$

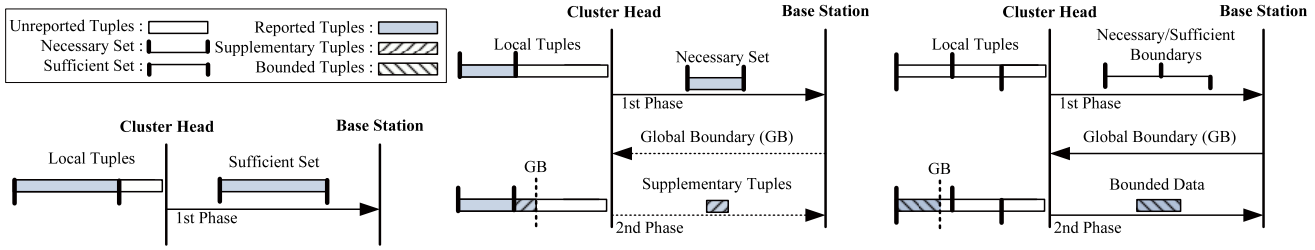
Based on Lemma 1, $P_{topk}(t_x) \leq P_{L,topk}(t_x) < p$. Thus, t_x is not in the final answer. □

In the following, we informally describe the algorithm to obtain the sufficient boundary and sufficient set. To find the sufficient boundary, the cluster head first sorts all the data tuples in ranking order. It then iterates from the highest ranked tuple downward to include tuples into the sufficient set until the first tuple that satisfies the conditions in (4) and (5) is obtained. If the sufficient boundary cannot be found, the sufficient set does not exist. For example, consider a PT-Top$k$ query with $k = 2, p = 0.5$ on the data set T_1 in Fig. 2. Let $r_{m,n}$ denote $r_{m,n}[T_1]$ here. Starting from t_1, we have $D = \{t_1\}$ initially. For $t_2 \in (T_1 - D)$, we check Condition 1 of PT-Top$k$ sufficient boundary (i.e., (4)) and find that

$$\begin{aligned} &(1 - P(\tau_{B,t_1})) \cdot (r_{1,0}[D_{t_2}] + r_{1,1}[D_{t_2}]) \\ &= (1 - 0) \cdot (0.1 + 0.9) = 1 > p. \end{aligned}$$

Since the condition is not met, we move on to t_2, and have $D = \{t_1, t_2\}$. Again, check Condition 1. For $t_3 \in (T_1 - D)$, we have $(1 - P(\tau_{A,t_2})) \cdot (r_{1,0}[D_{t_3}] + r_{1,1}[D_{t_3}]) = (1 - 0.9) \cdot (0.9 + 0.1) = 0.1 < p$; while for $t_4 \in (T_1 - D)$, we have

$$\begin{aligned} &(1 - P(\tau_{B,t_2})) \cdot (r_{1,0}[D_{t_4}] + r_{1,1}[D_{t_4}]) \\ &= (1 - 0.1) \cdot (0.1 + 0.9) = 0.9 > p. \end{aligned}$$



(a) Sufficient Set-based (SSB) Algorithm (b) Necessary Set-based (NSB) Algorithm (c) Boundary-based (BB) Algorithm

Fig. 5. Algorithms for intercluster query processing.

Thus, the condition is not met for t_2 as well. Similarly, we move onward to tuple t_3, and find that t_3 does not satisfy Condition 1 because $(1 - P(\tau_{B,t_3})) \cdot (r_{1,0}[D_{t_4}] + r_{1,1}[D_{t_4}]) = (1 - 0.1) \cdot (0 + 1) = 0.9 > p$. Finally, we check t_4 and thus have $D = \{t_1, t_2, t_3, t_4\}$. For $t_5 (\in T_1 - D)$, we have

$$(1 - P(\tau_{C,t_4})) \cdot (r_{1,0}[D_{t_5}] + r_{1,1}[D_{t_5}]) = (1 - 0) \cdot (0 + 0.45) = 0.45 < p.$$

Meanwhile, $r_{4,0} + r_{4,1} = 0 + 0.45 = 0.45 < p$, i.e., Condition 2 is also satisfied. Thus, the algorithm terminates and return $\{t_1, t_2, t_3, t_4\}$ as the sufficient set in which t_4 is the sufficient boundary. Note that, sufficient boundary does not always exist for a given data set. Take T_2 in Fig. 2 as an example. If there is an independent tuple t_{10} with speed value 84 (km/h) in 1.0 probability in T_2, then t_{10} could possibly be a qualified answer based on the local view of t_6-t_{10}. In this case, any tuple among t_6-t_9 is not a sufficient boundary of T_2.

Next, we first define the necessary boundary for PT-Top$k$ query and then discuss how to obtain the necessary boundary and necessary set via Theorem 4.

Definition 6 (PT-Topk Necessary Boundary). *Given an uncertain data set T_i in cluster C_i, the necessary boundary $NB(T_i)$ is the lowest ranked tuple t_{nb} such that*

$$P_{L,topk}(t_{nb}) \geq p.$$

Theorem 4. *Given an uncertain data set T_i in cluster C_i, the tuples ranked below the PT-Topk necessary boundary of T_i (i.e., t_{nb}) do not qualify for the final answer.*

Proof. Since t_{nb} is the lowest tuple having a "local" top-k probability greater than (or equal to) the threshold p, we know that any tuple ranked lower than t_{nb} has a local top-k probability lower than p. Thus, those tuples have their global top-k probabilities less than p according to Lemma 1, and as a result should not be included in the final answer. Moreover, those tuples ranked below t_{nb} are not needed for computing the top-k probability for any tuple ranked higher than t_{nb}. □

Again, we informally describe the algorithm for finding necessary boundary and necessary set. Similarly, a cluster head first sorts all the tuples in ranking order. The processing starts from the lowest ranked tuple upwards till the first tuple whose top-k probability is no less than the threshold (i.e., the lowest ranked candidate tuple).[5] The tuple satisfying the stop condition is the necessary boundary. Recall the example in Fig. 2. First, we access tuple t_9, and find $P_{top2}(t_9) \leq P(t_9) = 0.2 < p$. Then we move upward to check t_8, and similarly, t_8 is skipped as well. For t_7, we have

$$P_{top2}(t_7) = P(t_7) \times (r_{6,0} + r_{6,1}) = 0.5 \times (0.1 + (1 - 0.1)) = 0.5.$$

Finally, we get the necessary set $\{t_6, t_7\}$, with t_7 as the necessary boundary. Note that, the probability computation of some candidate tuples from other clusters may still require data tuples outside their necessary sets, e.g., in our example, t_8 is needed when computing the top-k probability of t_1. Thus $N(T_2)$ is only necessary but not sufficient.

4 Intercluster Query Processing

In this section, using the notion of sufficient and necessary sets as a basis, we propose three distributed algorithms for processing probabilistic top-k queries in wireless sensor networks, namely 1) Sufficient Set-based method; 2) Necessary Set-based method; and 3) Boundary-based method. Here we focus on addressing the communication overhead which is critical for wireless sensor networks and their applications. For simplicity, we logically assume single-hop transmission in both intracluster and intercluster communications. Nevertheless, our algorithms are not restricted to this assumption and can be extended for the multihop communications. As long as the base station receives all the candidate data tuples and *supplementary tuples*, we are able to compute the final answer with a generic centralized algorithm, e.g., those in [17], [13], [18], and [19]. Note that the supplementary tuples refer to the unqualified data tuples needed for computing the confidence probability of final answer.

4.1 Sufficient Set-Based Algorithm

An intuitive way for in-network data processing is to transmit the sufficient set to base station. As indicated in Theorem 1, the tuples not included in the sufficient set neither have top-k probability higher than p nor affect the top-k probability of qualified tuples in the final answer. Thus, they are subject to pruning. Consequently, the SSB algorithm, as illustrated in Fig. 5a, consists of only one communication phase from cluster heads to the base station. After collecting data tuples from its cluster, a cluster head computes the sufficient set from the local collected tuples and sends it to the base station. Note that if a sufficient set cannot be obtained, all the local data tuples are transmitted. After receiving the transmitted data tuples from all the cluster heads, the base station computes the query answer by a

5. If the sufficient boundary information is available, the sweeping may start from the sufficient boundary instead.

centralized algorithm. Detailed SSB algorithm is presented in Algorithm 1.

Algorithm 1. SSB_ALGORITHM

/* Cluster Head c_i Side */
Compute the sufficient boundary $SB(T_i)$ of T_i
if $SB(T_i)$ exists **then**
  $S(T_i) \leftarrow \{t | t \preceq_f SB(T_i) \wedge t \in T_i\}$
  $T_i' \leftarrow S(T_i)$
else
  $T_i' \leftarrow T_i$
end if
Deliver T_i' to the base station

/* Base Station Side */
Collect T_i' from all c_i $(1 \leq i \leq M)$
$T' \leftarrow \cup_{1 \leq i \leq M} T_i'$
Execute centralized algorithm over T'

The SSB method has a short query latency because only one round of data transmission is needed. However, the sufficient set is not very effective for intracluster pruning.

4.2 Necessary Set-Based Algorithm

The necessary set contains only 1) locally qualified tuples that have local top-k probability higher than p and 2) supplementary tuples ranked higher than those in (1) (Because they are needed to compute top-k probabilities of these qualified tuples). However, even though *all* the tuples that potentially have top-k probabilities higher than p are included in the necessary set, calculating their global top-k probabilities may still need to access some additional supplementary tuples. Therefore, NSB may have two phases when these additional supplementary tuples are needed.

Fig. 5b illustrates the NSB algorithm. As shown, each cluster head first computes its own necessary set and sends the set to the base station. After receiving all the necessary sets, the base station merges all the received tuples into a table $T' = \bigcup_{i=1}^{M} N(T_i)$, and finds the necessary boundary of T' (i.e., $NB(T')$—called the *global boundary* (GB)). Consider T' as a location data set. Based on Lemma 1, any tuple ranked lower than GB is not part of the final result. If GB is ranked higher than the highest ranked necessary boundary (i.e., $NB_{highest}$), we can conclude that all the necessary data have been delivered to the base station; thus, the base station computes the final answer and stop here. Otherwise, entering the second phase, the base station sends the GB back to the cluster heads, which return the supplementary data tuples ranked between its local necessary boundary and GB. Eventually, the base station computes the final answer. Algorithms running in cluster head c_i and the base station are shown in Algorithm 2.

Algorithm 2. NSB_ALGORITHM

/* Cluster Head c_i Side */
Compute the necessary boundary $NB(T_i)$ of T_i
$N(T_i) \leftarrow \{t | t \preceq_f NB(T_i) \wedge t \in T_i\}$
Deliver $N(T_i)$ to the base station
if Receive GB from the base station **than**
  $N'(T_i) \leftarrow \{t | t \preceq_f \text{GB} \wedge t \in [T_i - N(T_i)]\}$
  Send $N'(T_i)$ to the base station
end if

/* Base Station Side */
Collect $N(T_i)$ from all $c_i (1 \leq i \leq M)$
$T' \leftarrow \cup_{1 \leq i \leq M} N(T_i)$
Compute the global necessary boundary (GB) over T'
if GB$\preceq_f NB(T_i)$, where $\forall i$ of $1 \leq i \leq M$ **than**
  Execute centralized algorithm over T'
else
  Broadcast GB to cluster heads
  Once collect all $N'(T_i)$ from all c_i
  $T' \leftarrow T' \cup_{1 \leq i \leq M} N'(T_i)$
  Execute centralized algorithm over T'
end if

4.3 Boundary-Based Algorithm

Instead of directly delivering data tuples to the base station, the boundary-based method first delivers the local knowledge in clusters, in the form of sufficient boundary and necessary boundary, to the base station in order to facilitate a refined global data pruning among clusters later. Fig. 5c illustrates the BB algorithm. As shown, each cluster head first computes its sufficient and necessary sets and sends the boundaries to the base station. After receiving all the sufficient and necessary boundaries, the base station computes the global boundary as follows: Let $SB_{highest}$ denote the highest ranked sufficient boundary and NB_{lowest} denote the lowest ranked necessary boundary. Based on the property of sufficient boundary, we know that any tuple ranked lower than $SB_{highest}$ is not required for query processing. Meanwhile, based on the property of necessary boundary, we know that all tuples ranked higher than NB_{lowest} may be needed for query processing. Therefore, we return the higher boundary between $SB_{highest}$ and NB_{lowest} as GB to the cluster heads. In the second phase, all the data tuples ranked higher than GB are transmitted to the base station, which runs a centralized PT-top$k$ algorithm to compute the final answer. Detailed algorithm is presented in Algorithm 3. Its correctness is shown by Theorem 5.

Algorithm 3. BB_ALGORITHM

/* Cluster Head c_i Side */
Compute $NB(T_i)$ and $SB(T_i)$ of T_i
Send $NB(T_i)$ and $SB(T_i)$ to the base station
Receive GB from the base station
$T_i' \leftarrow \{t | (t \preceq_f GB) \wedge (t \in T_i)\}$
Deliver T_i' to the base station

/* Base Station Side */
Collect $NB(T_i)$ and $SB(T_i)$ from all c_i $(1 \leq i \leq M)$
Let $SB_{highest}$, NB_{lowest} be the highest ranked $SB(T_i)$ and the lowest ranked $NB(T_i)$ respectively, where $(1 \leq i \leq M)$
if $SB_{highest} \prec_f NB_{lowest}$ **than**
  GB $\leftarrow SB_{highest}$
else
  GB $\leftarrow NB_{lowest}$
end if
Broadcast GB to each cluster head
Collect T_i' from all c_i $(1 \leq i \leq M)$.
$T' \leftarrow \cup_{1 \leq i \leq M} T_i'$
Execute centralized algorithm over T'

Theorem 5. *Any tuple ranked lower than GB is not in the final answer.*

Proof. The proof is sketched as below. 1) If GB is the highest ranked $SB(T_i)$ $(1 \leq i \leq M)$, i.e., global sufficient boundary, we make the above conclusion based on Theorem 1. 2) Otherwise, GB is the lowest ranked $NB(T_i)$ $(1 \leq i \leq M)$. According to the definition of necessary boundary, all the candidate results are ranked higher than GB, thus all the tuples ranked lower than GB are useless for the base station. □

5 Adaptive Query Processing

In this section, we first perform a cost analysis on data transmission of the three proposed methods. Since their performance is affected by factors such as the skewness of data distribution among clusters which may change continuously over time, we propose a cost-based adaptive algorithm that keeps track of the estimated cost for all methods in order to switch as appropriate.

5.1 Cost Analysis

Let M denote the number of clusters in the network; and S_q, S_b, and S_d be the sizes of query message, boundary message, and data message, respectively.[6] Also let $|S(T_i)|$ and $|N(T_i)|$ denote the cardinalities of the sufficient set and necessary set of the data set T_i in a cluster $\mathcal{C}_i (1 \leq i \leq M)$, respectively. Notations and their descriptions are listed in Table 1.

The total transmission cost for SSB is

$$C_{SSB} = M \cdot S_q + \sum_{i=1}^{M} |S(T_i)| \cdot S_d. \tag{7}$$

For the necessary set-based algorithm, let $|ST(c_i)|$ be the number of supplementary data tuples collected in the second phase from cluster head c_i. Then, the total transmission cost for NSB can be estimated as follows:

$$C_{NSB} = \begin{cases} M \cdot S_q + \sum_{i=1}^{M} |N(T_i)| \cdot S_d & \text{(1 phase)} \\ 2M \cdot S_q + M \cdot S_b + \sum_{i=1}^{M} |N(T_i)| \cdot S_d & \\ \quad + \sum_{i=1}^{M} |ST(c_i)| \cdot S_d & \text{(2 phases).} \end{cases} \tag{8}$$

Similarly, for the boundary-based approach, let $|BB(c_i)|$ be the cardinality of data tuples delivered in phase 2. Since sufficient boundary and necessary boundary are transmitted in the first phase, and global boundary is transmitted in phase 2, the total transmission cost for BB can be estimated as follows:

$$C_{BB} = 2M \cdot S_q + 3M \cdot S_b + \sum_{i=1}^{M} |BB(c_i)| \cdot S_d. \tag{9}$$

6. While the meanings of query and data messages are self-explained, boundary messages refer to the messages that deliver boundary information.

TABLE 1
Notations and Descriptions

Notation	Description
c_i	cluster head of cluster C_i
M	# of clusters
S_q	size of a query message
S_b	size of a boundary message
S_d	size of a data message
$\|S(T_i)\|$	cardinality of the sufficient set of T_i
$\|N(T_i)\|$	cardinality of the necessary set of T_i
$\|ST(c_i)\|$	cardinality of supplementary data collected from c_i in phase 2 of NSB
$\|BB(c_i)\|$	cardinality of data delivered by c_i in Phase 2 of BB

5.2 Cost-Based Adaptive Algorithm

Here we propose a cost-based adaptive algorithm that switches dynamically among SSB, NSB, and BB as the data distribution within the network changes. Our approach is to keep track of the estimated cost for all three methods under different scenarios in order to trigger the switch as appropriate. The key issue is how to estimate the cost of the running and alternative methods under different scenarios. In the following, we consider three scenarios running SSB, NSB, and BB, respectively, and show how the cost of alternative methods are estimated.

Running SSB. It is easy for the base station to estimate C_{NSB} and C_{BB} under this scenario. According to Theorem 2, we know that the necessary set is always an subset of the sufficient set (if it exists), implying that the base station has required information to mimic NSB to get the C_{NSB}. Furthermore, the total cost C_{BB} for boundary-based method can be estimated at the base station as well.

Running NSB. As shown in (7) and (9), extra information such as $|S(T_i)|$ and $|BB(c_i)|$ are needed to compute C_{SSB} and C_{BB}. Thus, in the first phase of NSB, cluster heads send the base station some extra information, including $|S(T_i)|$, $SB(T_i)$ (the sufficient boundary), and $\rho(c_i)$ (density of data tuples in the scoring range between $NB(T_i)$ and $SB(T_i)$). With $|S(T_i)|$, the base station obviously can obtain C_{SSB} easily.

As for computing C_{BB}, the base station first computes the global boundary for the BB algorithm based on the sufficient and necessary boundaries from the clusters. Then, it compares the necessary sets of clusters against GB. For cluster $\mathcal{C}_i$, if $NB(T_i) \prec_f GB$, $|BB(c_i)|$ is the number of tuples ranked higher than GB; otherwise, $|BB(c_i)| = |N(T_i)| + \rho(c_i) \cdot (score(GB) - score(NB(T_i)))$, where $score(GB)$ and $score(NB(T_i))$ are the score values of GB and $NB(T_i)$, respectively. Notice that, as $SB(T_i)$ is available, the global boundary which is originally derived from necessary sets $N(T_i) (1 \leq i \leq M)$ in NSB can be further refined.

Running BB. The extra information needed from each cluster head c_i to figure out the C_{SSB} and C_{NSB} includes $|S(T_i)|$, $|N(T_i)|$ and $|ST(c_i)|$. Additionally, for NSB, we need to know whether the examined query would be executed in one or two phases. To obtain such information, $|S(T_i)|$ and $|N(T_i)|$ are sent along with the boundary messages in the first phase. With $|S(T_i)|$, the base station can compute the C_{SSB} easily. To figure out C_{NSB}, we consider two cases: 1) $SB_{highest}$ is ranked lower than or equivalent to NB_{lowest}; and 2) $SB_{highest}$ ranked higher than NB_{lowest}.

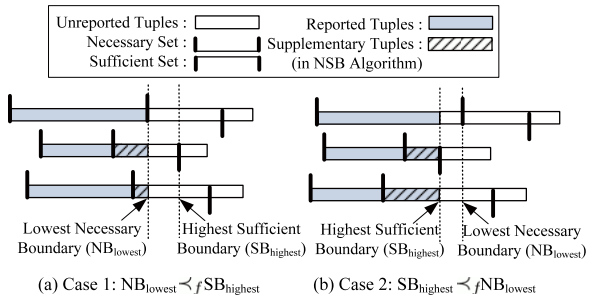


Fig. 6. $SB_{highest}$ and NB_{lowest} in BB Algorithm.

For Case 1 (see Fig. 6a), we know that $BB(c_i)$ covers all necessary sets $N(T_i), i = 1 \ldots M$. Let $T' = \bigcup_{1 \le i \le M} N(T_i)$ and $NB(T')$ denote the boundary of necessary set in T' (i.e., the global boundary in the NSB algorithm). If $GB \preceq_f NB_{highest}$, NSB is completed in one phase, and C_{NSB} can be estimated by taking into account $|N(T_i)|(1 \le i \le M)$; otherwise, supplemental tuple set is constructed as $ST(c_i) = \{t | t \in BB(c_i) \wedge NB(T_i) \prec_f t \preceq_f GB\}$.

For Case 2 (see Fig. 6b), some tuples in the necessary sets have been cut off by intercluster pruning of the BB method. However, with $|N(T_i)|$, we can still construct *partial* necessary sets $PN(T_i)$ from $BB(T_i)$ $(i = 1..M)$ by taking the first $|N(T_i)|$ tuples from $BB(T_i)$. Let $T'' = \bigcup_{1 \le i \le M} PN(T_i)$, and $NB(T'')$ be the necessary boundary of data set T''. In addition, the lowest ranked tuple of $PN(T_i)$ is denoted by $PNB(T_i)$, and $PNB_{highest}$ presents the highest ranked $PNB(T_i)$ $(1 \le i \le M)$. If $NB(T'') \preceq_f PNB_{highest}$, we believe NSB could be completed within the first phase, of which $|N(T_i)|$ information is available; otherwise, NSB needs to enter the second phase to answer the query. Here we let GB be the higher ranked tuple between $NB(T'')$ and $SB_{highest}$, $ST(c_i)$ can be recovered as $ST(c_i) = \{t | t \in BB(c_i) \wedge NB(T_i) \prec_f t \preceq_f GB\}$.

In order to estimate the costs of all three methods under various scenarios, we piggy-back some extra information in data transmissions, which introduces some communication overhead. A detailed derivation that takes into account the overhead is omitted due to space limitation, but the methodology to perform cost analysis is basically similar to what we show in Section 5.1.

The *adaptive* algorithm is shown in Algorithm 4. Basically, the base station collects cost statistics in each processing round but it only decides whether to switch algorithms for every Δ rounds, where Δ is an adjustable system parameter, based on the lowest average transmission cost estimated. The cost statistics is cleared after a switch. Since the switch is triggered by the base station, which is also responsible for tracking the cost statistics, the incurred overhead is limited to the cost of piggy-backing some extra information in the NSB and BB methods.

Algorithm 4. ADAPTIVE_ALGORITHM
// Initialize: $S_{SSB}, S_{NSB}, S_{BB} \leftarrow 0$ and counter $\leftarrow 0$
// Δ is a pre-defined window size.
counter $\leftarrow$ counter $+1$
Estimate the cost of C_{SSB}, C_{NSB} and C_{BB}
$S_{SSB} \leftarrow S_{SSB} + C_{SSB}$
$S_{NSB} \leftarrow S_{NSB} + C_{NSB}$
$S_{BB} \leftarrow S_{BB} + C_{BB}$
if counter $\ge \Delta$ **than**
  if $S_{SSB} = \min\{S_{SSB}, S_{NSB}, S_{BB}\}$ **than**
    Switch to SSB_ALGORITHM.
  end if
  if $S_{NSB} = \min\{S_{SSB}, S_{NSB}, S_{BB}\}$ **than**
    Switch to NSB_ALGORITHM.
  end if
  if $S_{BB} = \min\{S_{SSB}, S_{NSB}, S_{BB}\}$ **than**
    Switch to BB_ALGORITHM.
  end if
  $S_{SSB}, S_{NSB}, S_{BB} \leftarrow 0$ and counter $\leftarrow 0$
end if

6 Extension for Tree-Structured Network Topology

To perform in-network query processing, a routing tree is often formed among sensor nodes and the base station. A query is issued at the root of the routing tree (i.e., the base station) and propagated along the tree to all sensor nodes. Although the concepts of sufficient set and necessary set introduced earlier are based on two-tier hierarchical sensor networks, they are applicable to tree-structured sensor network. Here, we devise a family of algorithms, namely SSB-T, NSB-T, and NSB-T-Opt, for efficient probabilistic top-k query processing in a tree-structured sensor network. Let N and D denote the set of sensor nodes and the entire uncertain data set, respectively. Each sensor node $n_i \in N$, holds a subset of D, denoted by D_i. Note that all D_i $(\forall n_i \in N)$ are disjoint, and $D = \bigcup_{n_i \in N} D_i$. Similarly, we assume tuples belonging to the same x-tuple are obtained in a single sensor node.

SSB-T. Similar to SSB, SSB-T is a one-round query processing algorithm. The basic idea is to deliver the sufficient sets from leaf nodes, along the routing path, back to the root node. More specifically, after the query message is propagated to all sensor nodes, sensor nodes respond as follows: for a leaf sensor node n_i, it computes the sufficient set based on D_i, denoted as S_i, for delivery to its parent. On the other hand, for a nonleaf node n_j, it collects sufficient sets sent by its children (i.e., $\cup_{n_x \in child(n_j)} S_x$), which along with D_j are used to determine the sufficient set S_i for delivery to its parent. When the base station receives all sufficient sets sent by its children. It directly deduces the probabilistic top-k result by running a centralized algorithm. Similar to SSB, the correctness of SSB-T can be proved.

NSB-T. NSB-T is a two-round query processing algorithm, where in the first round necessary sets are collected along the routing tree, from the leaf nodes to the root (i.e., base station). In the second round, the necessary boundary derived at the root is returned to all sensor nodes to obtain additional supplementary tuples for computing the final query result. In the first round, after the query message is propagated to all sensor nodes, NSB-T algorithm runs as follows: for a leaf node n_i, it computes the necessary set based on D_i, denoted as N_i, for delivery to its parent. On the other hand, for a nonleaf node n_j, it collects the necessary sets sent by its children (i.e., $\cup_{n_x \in child(n_j)} N_x$), which along with D_j are used to determine the necessary set N_j for delivery to its parent. When the base station receives all necessary sets sent by its children, it calculates a global necessary set, and sends

back the global necessary boundary (GNB) to all sensor nodes for acquiring supplementary tuples as needed in the second round. After all supplementary data is collected, the base station deduces the probabilistic top-k result by running a centralized algorithm. Similar to NSB, the correctness of NSB-T can be proved.

NSB-T-Opt. Aiming to exploring the in-network caching ability of tree-structured sensor networks, we further extend the idea of NSB-T to optimize its performance, and thus develop a new algorithm called NSB-T-Opt. Note that, in the first round of NSB-T, some data filtered at intermediate nodes may become the supplementary data in the second round. In NSB-T, supplementary data are pulled from their source nodes, but this is not always necessary as some of them might have been delivered to some intermediate nodes. Additionally, with some knowledge about data stored in its subtree, an intermediate node may be able to determine whether there is a need to go further down the tree to retrieve supplementary data.

Based on the ideas discussed earlier, NSB-T-Opt aims to cache at each intermediate node n_j the suppressed data along with a *supplementary boundary* B_i for each of its child node n_i. Note that this boundary information is used to suppress unnecessary supplementary data transmission in the second round. When n_i is a leaf node, B_i is its necessary boundary NB_i, i.e., the lowest ranked tuple of the necessary set from n_i. On the other hand, if n_i is a nonleaf node, B_i will be the supplementary boundary computed and sent along with the necessary set by n_i. It derives B_i from $\cup_{n_x \in child(n_i)} B_x$. Let B_x^* be the boundary holding the highest score among all boundary information of $\cup_{n_x \in child(n_i)} B_x$. If B_x^* ranks higher than NB_i, then $B_i = B_x^*$; otherwise, $B_i = NB_i$.

The above describes the derivation and maintenance of supplementary boundary at intermediate nodes in the first round of NSB-T-OPT. In the second round, the base station disseminates the global necessary boundary along the routing tree to sensor nodes in order to retrieve supplementary data. At an intermediate node n_j, the GNB is not propagated to a child node n_i if B_i is greater than GNB. As such, significant saving in the second round communication can be achieved.

7 Performance Evaluation

In this section, we first conduct a simulation-based performance evaluation on the distributed algorithms for processing PT-topk queries in two-tier hierarchical cluster-based wireless sensor monitoring system. As discussed, limited energy budget is a critical issue for wireless sensor network and radio transmission is the most dominate source of energy consumption. Thus, we measure *the total amount of data transmission* as the performance metrics. Notice that, response time is another important metrics to evaluate query processing algorithms in wireless sensor networks. All of those three algorithms, i.e., SSB, NSB, and BB, perform at most two rounds of message exchange, thus clearly outperform an iterative approach (developed based on the processing strategy in [14]), which usually needs hundreds of iterations. Note that, there is not much difference among SSB, NSB, and BB in terms of query response time, thus we focus on the data transmission cost in the evaluation. Finally, we also conduct experiments to evaluate algorithms, SSB-T, NSB-T, and NSB-T-Opt under the tree-structured network topology.

TABLE 2
Parameter Settings

Parameters	Default	Settings
k: # of top ranked tuples	6	$2 \sim 10$
p: probability threshold	0.5	$0.2 \sim 0.7$
η: data skewness factor	0.5	$0 \sim 1$
λ: mean size of X-tuple	5	$1 \sim 10$
Δ: adaptive window size	15	$1 \sim 50$
M: # of clusters	36	$4 \sim 100$
I: # of interest zones/x-tuples	500	
S_d: data tuple size (bytes)	32	$8 \sim 128$
S_q: query message size (bytes)	8	
S_b: boundary message size (bytes)	8	

7.1 Simulation Model

Here we describe our simulation model. We assume a wireless sensor field consisting of I zones. Each zone is deployed with an average of λ sensor nodes. Here, I can also be seen as the number of x-tuples in the global database and λ is the average size of an x-tuple. The confidence values for sensor readings in an area are assigned randomly. The clusters in the network is realized by a simple grid partition. There are M clusters, which is varied in our experiments. The simulator models sensor mote behavior at a coarse level, similar to the TAG simulator [21] in which time is divided into units of rounds. At the beginning of each round, users may issue PT-Top$k$ queries at the base station and the query messages are passed to cluster heads and sensor nodes without delay since transmission latency is not our main concern in this evaluation.

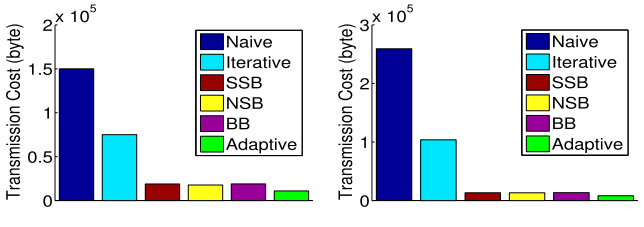
We use both synthetic data and real traces of sensor readings for performance evaluation. Note that, we use tunable parameters to control the distribution of synthetic data as in [34]. For synthetic data, the scoring attribute value of a sensor node n_i is obtained by $f(t_i) = \eta \cdot s(i) + (1 - \eta) \cdot r(i)$, where $0 \leq \eta \leq 1$ controls the data skewness. Here $s(i)$ is a value generated based on the node n_is location in order to inject spatial locality and $r(i)$ is a random variable ranging from 0 to 1. When $\eta = 1$, the data distribution is highly spatial-correlated. When $\eta = 0$, the data distribution is totally random. The real sensor traces are from the Intel Lab Data [35]. We treat the readings for each sensor node in the traces as for an area of interest by assigning λ readings to λ sensor nodes in an area. All the parameter settings are summarized in Table 2.

7.2 Experimental Results

A series of experiments is conducted to evaluate the proposed algorithms for a two-tier network in the following aspects: 1) overall performance, 2) sensitivity tests, and 3) adaptiveness. Additionally, overall performance under the tree topology is evaluated.

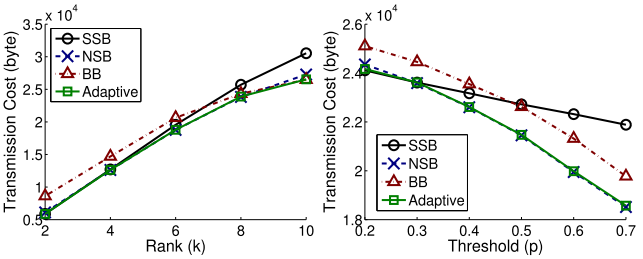
7.2.1 Overall Performance

We first validate the effectiveness of our proposed methods in reducing the transmission cost against two baseline approaches, including 1) a *naive* approach, which simply transmits the entire data set to the base station for query processing; 2) an *iterative* approach devised based on the



(a) Synthetic Data (b) Real Data

Fig. 7. Overall performance.



(a) Varied k (b) Varied p

Fig. 8. Impact of query parameters.

processing strategy explored in [14]. The iterative approach runs as follows: in each round, each cluster head delivers a single data tuple with the highest score in its local data set and the information of current local highest score (after removing the delivered data) to the base station. In response, the base station derives necessary set upon the data collected so far. If the score of necessary boundary is higher than the current local highest score of all cluster heads, the base station can determine the final result based on the available data and thus terminate the algorithm; otherwise, the algorithm goes for the next round. The experiments use both the synthetic data and real traces. Here we randomize all parameter settings within their ranges to show the general performance. Both results on synthetic and real data (see Fig. 7) indicate that all the algorithms proposed in this paper significantly reduce data transmissions against the two baseline approaches by achieving about 70 percent of saving against the Iterative approach in synthetic data sets (see Fig. 7a) and 80 percent of saving against the Iterative approach in real traces (see Fig. 7b). One important reason that our approaches outperform the Iterative approach is due to the small and constant query processing rounds in our approaches. In our experiment, our algorithms complete within two rounds; while the iterative approach incurs about 60-200 rounds. Note that the experiments on adaptive algorithm are conducted on a setting that exhibits dynamic changes with certain temporal locality. Since the algorithm dynamically adapts to the changes by switching to appropriate methods, it provides an additional saving over the other algorithms.

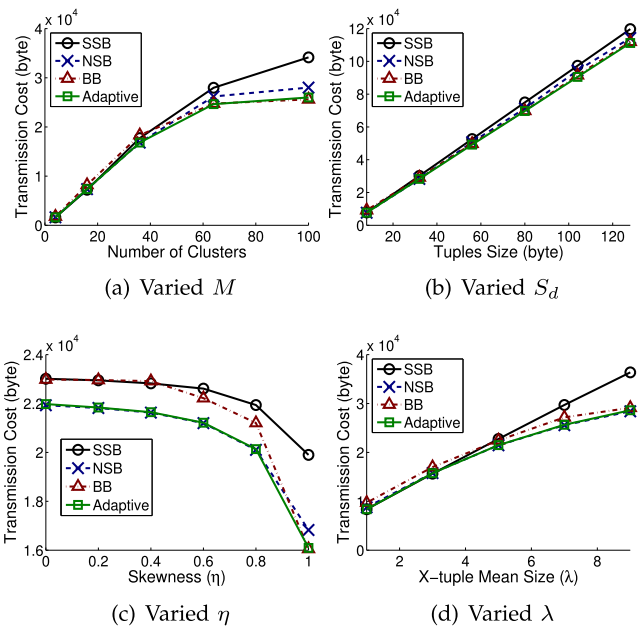
7.2.2 Sensitivity Tests

Next, we examine the impact of a variety of query and system parameters on the performance of the proposed algorithms. In the plots, we do not show the result of baseline approaches for clarity of presentation. We also omit the plots of experiments on real traces due to space limitation.[7]

Here we first show the impact of query parameters, i.e., k and p, on performance. Fig. 8a shows the trend of transmission cost by varying k from 2 to 10. As shown, the transmission cost increases for all algorithms because the number of tuples needed for query processing is increased. Among the SSB, NSB, and BB algorithms, BB does not perform as well as others when k is small but it becomes a good choice when k becomes larger.

Fig. 8b shows the trend of transmission cost by varying query threshold p from 0.2 to 0.7. Performance of all algorithms improve as the threshold p increases because a high threshold in general reduces the size of the result set and hence the amount of data transmitted to the base station. Among the algorithms, SSN performs the best when p is small but becomes significantly worse than the others when p becomes large. This is because NSB and BB are able to use the tighter threshold to filter out more unqualified tuples and reach a tighter necessary boundary. On the other hand, the sufficient boundary does not benefit as much. Finally, we observe that the adaptive algorithm matches very well with NSB.

Next, we present results of our sensitivity tests on a number of system parameters and data distribution. We examine how the algorithms scale up by increasing the number of clusters in the system from 4, 16, 36, 64, to 100. The result, as shown in Fig. 9a, shows that BB scales up better than NSB and SSB. As the number of clusters increases, the total amount of data transmission rises accordingly. Realizing intracluster and intercluster pruning before data transmission is an important factor for BB to outperform others in this examination. The same observation can also be made in the sensitivity test on the size of data tuples (see Fig. 9b). As the primary cost in the network is tuple transmission, SSB exhibits a linear increase of transmission cost at a higher rate than NSB and BB. Again, the result shows that BB is the best choice when the tuple

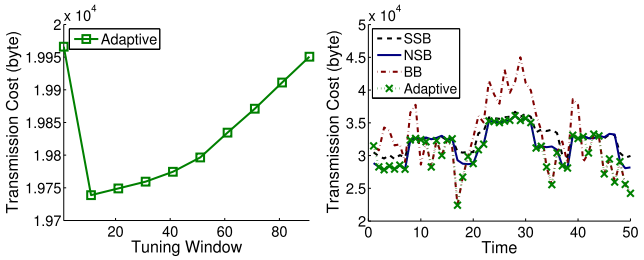


(a) Varied M (b) Varied S_d

(c) Varied η (d) Varied λ

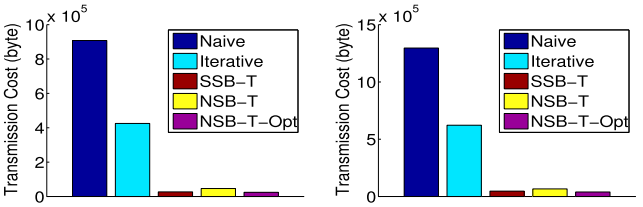
Fig. 9. Sensitivity test on system parameters.

7. The experimental result aligns with conclusion from experiments on synthetic data.



(a) Varied Δ (b) Switching Log

Fig. 10. Tests on Adaptiveness.



(a) Synthetic Data (b) Real Data

Fig. 11. Overall performance in tree topology.

size is large. Overall, the adaptive algorithm matches up the best algorithm under all circumstances.

Fig. 9c shows the impact of data skewness on transmission cost. In the experiments, when the skewness is high the difference among readings in different zones is increased. As shown, the data skewness helps intracluster data pruning as the performance of every algorithm improves when the data distribution becomes more skewed. Additionally, NSB and BB take advantage of the skewed necessary sets and necessary boundaries among local clusters to obtain their global boundaries, respectively, which are very effective for intercluster pruning. Particularly, when $\eta = 1$, BB shows the best performance in terms of transmission cost. Thus, adaptive algorithm approaches BB under this scenario.

Fig. 9d shows the impact of the size of x-tuples λ on the performance. When λ is small, SSB shows a good performance because the probability of highly ranked tuples in an x-tuple is higher. Thus adaptive algorithm chooses SSB as the best candidate. As λ grows, there are more tuples in an x-tuple. Thus, the size of sufficient set is linearly increased as more tuples need to be considered to set the sufficient boundary. On the other hand, although the size of necessary set also increases, it does not increase linearly since many low-probability tuples are filtered out by the probability threshold. By utilizing both sufficient and necessary boundaries, the performance of BB also gets worsen but in a slower pace due to the smaller necessary set. However, due to the overhead for exchanging boundary information, BB may not perform as good as NSB, particularly with light data skewness.

Based on the above results, we find that NSB and BB generally exhibit better performance than SSB does. However, there is no clear winners for NSB and BB. As we discussed above, with larger network scale (M), larger tuple size (S_d), or extreme skewed data distribution (η), BB usually outperforms NSB. However, in other cases, NSB shows better performance over BB.

7.2.3 Adaptiveness

As shown in the above figures, the adaptive algorithm reaches our expectation by achieving the least transmission cost under all circumstances. In this section, we further test its adaptiveness to dynamic sensor network environments.

One important factor that has an impact on the adaptive algorithm is the size of tuning window. To figure out the optimal setting of tuning window size, we conduct an experiment by varying the window size from $1, 10, 20, \ldots,$ to 90. To mock the dynamics of readings, we change η setting among $\{0, 0.5, 1.0\}$ every 50 round. As shown in Fig. 10a, there exists an optimal window size for adapting to the dynamic readings. If the window size is too small, the adaptation will be too sensitive and thus fail to get the optimal performance. On the other hand, if the window size is too large, the adaptation may lag behind the change of readings.

Additionally, we conduct an experiment to show the behaviors of algorithms under a dynamic environment we simulate. Fig. 10b shows that the adaptive algorithm switches timely to match the best algorithms. As shown in the figure, adaptive algorithm switches from the NSB algorithm to the BB algorithm at about time 10, then returns back to NSB at about time 20, and finally switches to the BB algorithm again at about time 30. While the different algorithms may outperform each other at different time, the adaptive algorithm adapts to the dynamic changes to achieve the least transmission cost.

7.2.4 Tree Topology

Finally, we evaluate SSB-T, NSB-T, and NSB-T-Opt in a tree-structured network. Experimental settings are the same as the ones used in two-tier hierarchical network. As shown in Fig. 11, SSB-T, NSB-T, and NSB-T-Opt show much better performance against the two baseline approaches. Notice that, SSB-T outperforms NSB-T, as NSB-T incurs extra overhead for the delivery of supplementary data in the second round. On the other hand, NSB-T-Opt address this issue through in-network suppressing of unnecessary supplementary data transmission. From the experiment, we observe that the query for the supplementary data never reaches the leaf nodes, i.e., requests for supplementary data are fulfilled by some intermediate nodes along the routing paths. In this way, NSB-T-Opt indeed optimizes the performance of NSB-T and shows excellent performance.

8 Conclusion

In this paper, we propose the notion of sufficient set and necessary set for efficient in-network pruning of distributed uncertain data in probabilistic top-k query processing. Accordingly, we systematically derive sufficient and necessary boundaries and propose a suite of algorithms, namely SSB, NSB, and BB algorithms, for in-network processing of PT-Top$k$ queries. Additionally, we derive a cost model on communication cost of the three proposed algorithms and propose a cost-based adaptive algorithm that adapts to the application dynamics. Although our work in this paper is based mainly under the setting of two-tier hierarchical

network, the concepts of sufficient set and necessary set are universal and can be easily extend to a network with tree topology. The performance evaluation validates our ideas and shows that the proposed algorithms reduce data transmissions significantly. While focusing on PT-Top$k$ query in this paper, the developed concepts can be applied to other top-k query variants. We plan to develop algorithms to support other probabilistic top-k queries in the future.

REFERENCES

[1] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A Collaborative Approach to in-Place Sensor Calibration," *Proc. Second Int'l Conf. Information Processing in Sensor Networks (IPSN),* pp. 301-316, 2003.

[2] http://www.veriteq.com/, 2012.

[3] http://www.dashdyno.net/tech/GPS/gps_location.html, 2012.

[4] E. Elnahrawy and B. Nath, "Poster Abstract: Online Data Cleaning in Wireless Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems (SenSys '03),* pp. 294-295, 2003.

[5] A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," *Proc. 13th Int'l Conf. Very Large Data Bases (VLDB '04),* pp. 588-599, 2004.

[6] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03),* pp. 551-562, 2003.

[7] S. Abiteboul, P. Kanellakis, and G. Grahne, "On the Representation and Querying of Sets of Possible Worlds," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '87),* pp. 34-48, 1987.

[8] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04),* pp. 864-875, 2004.

[9] A.D. Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working Models for Uncertain Data," *Proc. 22nd Int'l Conf. Data Eng. (ICDE '06),* p. 7, 2006.

[10] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J.S. Vitter, "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04),* pp. 876-887, 2004.

[11] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, and S. Prabhakar, "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05),* pp. 922-933, 2005.

[12] C. Re, N. Dalvi, and D. Suciu, "Efficient Top-k Query Evaluation on Probabilistic Data," *Proc. Int'l Conf. Data Eng. (ICDE '07),* pp. 896-905, 2007.

[13] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08),* 2008.

[14] F. Li, K. Yi, and J. Jestes, "Ranking Distributed Probabilistic Data," *Proc. 35th SIGMOD Int'l Conf. Management of Data (SIGMOD '09),* 2009.

[15] H.W. Rabiner, C. Anantha, and B. Hari, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. 33rd Hawaii Int'l Conf. System Sciences (HICSS '00),* 2000.

[16] Y. Diao, D. Ganesan, G. Mathur, and P.J. Shenoy, "Rethinking Data Management for Storage-Centric Sensor Networks," *Proc. Conf. Innovative Data Systems Research (CIDR '07),* pp. 22-31, 2007.

[17] M.A. Soliman, I.F. Ilyas, and K.C. Chang, "Top-k Query Processing in Uncertain Databases," *Proc. Int'l Conf. Data Eng. (ICDE '07),* 2007.

[18] C. Jin, K. Yi, L. Chen, J.X. Yu, and X. Lin, "Sliding-Window Top-k Queries on Uncertain Streams," *Proc. Int'l Conf. Very Large Data Bases (VLDB '08),* 2008.

[19] G. Cormode, F. Li, and K. Yi, "Semantics of Ranking Queries for Probabilistic Data and Expected Ranks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '09),* 2009.

[20] P. Cao and Z. Wang, "Efficient Top-k Query Calculation in Distributed Networks," *Proc. 23rd Ann. ACM Symp. Principles of Distributed Computing (PODC),* pp. 206-215, 2004.

[21] S. Madden, M.J. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Proc. Fifth Symp. Operating Systems Design and Implementation (OSDI '02),* 2002.

[22] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, "Processing Window Queries in Wireless Sensor Networks," *Proc. IEEE 22nd Int'l Conf. Data Eng. (ICDE '06),* 2006.

[23] M. Ye, X. Liu, W.-C. Lee, and D.L. Lee, "Probabilistic Top-k Query Processing in Distributed Sensor Networks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '10),* 2010.

[24] D. Zeinalipour-Yazti, Z. Vagena, D. Gunopulos, V. Kalogeraki, V. Tsotras, M. Vlachos, N. Koudas, and D. Srivastava, "The Threshold Join Algorithm for Top-k Queries in Distributed Sensor Networks," *Proc. Second Int'l Workshop Data Management for Sensor Networks (DMSN '05),* pp. 61-66, 2005.

[25] A. Sharaf, J. Beaver, A. Labrinidis, and K. Chrysanthis, "Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks," *Int'l J. Very Large Data Bases,* vol. 13, no. 4, pp. 384-403, 2004.

[26] A.S. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang, "A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks," *Proc. 22nd Int'l Conf. Data Eng. (ICDE '06),* p. 68, 2006.

[27] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy Efficient Data Collection in Distributed Sensor Environments," *Proc. 24th Int'l Conf. Distributed Computing Systems (ICDCS '04),* pp. 590-597, 2004.

[28] M. Wu, J. Xu, X. Tang, and W.-C. Lee, "Top-k Monitoring in Wireless Sensor Networks," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 7, pp. 962-976, July 2007.

[29] D. Wang, J. Xu, J. Liu, and F. Wang, "Mobile Filtering for Error-Bounded Data Collection in Sensor Networks," *Proc. 28th Int'l Conf. Distributed Computing Systems (ICDCS '08),* pp. 530-537, 2008.

[30] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient Processing of Top-k Queries in Uncertain Databases with X-Relations," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 12, pp. 1669-1682, Dec. 2008.

[31] J. Li, B. Saha, and A. Deshpande, "A Unified Approach to Ranking in Probabilistic Databases," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* vol. 2, no. 1, pp. 502-513, 2009.

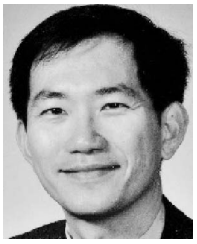
[32] X. Lian and L. Chen, "Probabilistic Ranked Queries in Uncertain Databases," *Proc. 11th Int'l Conf. Extending Database Technology (EDBT '08),* pp. 511-522, 2008.

[33] M.A. Soliman and I.F. Ilyas, "Ranking with Uncertain Scores," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '09),* 2009.

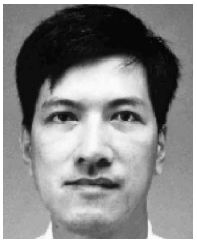
[34] X. Liu, J. Xu, and W.-C. Lee, "A Cross Pruning Framework for Top-k Data Collection in Wireless Sensor Networks," *Proc. 11th Int'l Conf. Mobile Data Management,* pp. 157-166, 2010.

[35] Http://berkeley.intel-research.net/labdata/, 2012.

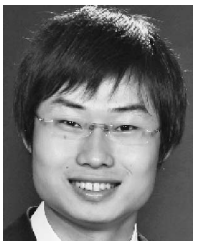
Mao Ye received the bachelor's degree in computer science from the Nanjing University, China, in 2004. He served as a research assistant at City University of Hong Kong from September 2005 to August 2006. Currently, he is working toward the PhD degree in the Department of Computer Science and Engineering, The Pennsylvania State University, University Park. His research interests include data and knowledge engineering and distributed systems.



Wang-Chien Lee received the BS degree from the National Chiao Tung University, Hsinchu, Taiwan, the MS degree from Indiana University, Bloomington, and the PhD degree from the Ohio State University, Columbus. He is working as an associate professor of computer science and engineering at the Pennsylvania State University, University Park, where he leads the Pervasive Data Access (PDA) Research Group to perform cross-area research in database systems, pervasive/mobile computing, and networking. He is particularly interested in developing data management techniques (including accessing, indexing, caching, aggregation,dissemination, and query processing) for supporting complex queries in a wide spectrum of networking and mobile environments such as peer-to-peer networks, mobile adhoc networks, wireless sensor networks, and wireless broadcast systems. Meanwhile, he has worked on XML, security, information integration/retrieval, and object-oriented databases. He is a member of the IEEE.



Dik Lun Lee received the BSc degree in electronics from the Chinese University of Hong Kong, and the MS and PhD degrees in computer science from the University of Toronto, Canada. Currently, he is working as a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He was an associate professor in the Department of Computer Science and Engineering, Ohio State University, Columbus. He was the founding conference chair for the International Conference on Mobile Data Management and served as the chairman of the ACM Hong Kong Chapter in 1997. His research interests include information retrieval, search engines, mobile computing, and pervasive computing. He is a member of the IEEE.



Xingjie Liu (S'03) received the BE degree in electronic engineering from the Tsinghua University, Beijing, China, in June 2003. After that, he began pursuing the PhD degree in computer science and is now working toward the PhD degree at the Pennsylvania State University. His research interests spanned a number of disciplines, including database query engine, query evaluation and optimization, distributed system, and statistical database analysis. He is a student member of the IEEE.