A General Framework for Efficient Continuous Multidimensional Top-k Query Processing in Sensor Networks

Hongbo Jiang, *Member*, *IEEE*, Jie Cheng, Dan Wang, *Member*, *IEEE*, Chonggang Wang, *Senior Member*, *IEEE*, and Guang Tan, *Member*, *IEEE*

Abstract—Top-k query has long been a crucial problem in multiple fields of computer science, such as data processing and information retrieval. In emerging cyber-physical systems, where there can be a large number of users searching information directly into the physical world, many new challenges arise for top-k query processing. From the client's perspective, users may request different sets of information, with different priorities and at different times. Thus, top-k search should not only be multidimensional, but also be across time domain. From the system's perspective, data collection is usually carried out by small sensing devices. Unlike the data centers used for searching in the cyber-space, these devices are often extremely resource constrained and system efficiency is of paramount importance. In this paper, we develop a framework that can effectively satisfy demands from the two aspects. The sensor network maintains an efficient dominant graph data structure for data readings. A simple top-k extraction algorithm is used for user query processing and two schemes are proposed to further reduce communication cost. Our methods can be used for top-k query with any linear convex query function. The framework is adaptive enough to incorporate some advanced features; for example, we show how approximate queries and data aging can be applied. To the best of our knowledge, this is the first work for continuous multidimensional top-k query processing in sensor networks. Simulation results show that our schemes can reduce the total communication cost by up to 90 percent, compared with a centralized scheme or a straightforward extension from previous top-k algorithm on 1D sensor data.

Index Terms—Sensor networks, algorithm/protocol design, top-k extraction.

1 Introduction

The recent development of sensor networks [8], [10], [19], [20], [21], [22] has made it possible for users to search information not only in the cyber-space, but also in the physical world. It can be imagined that in the near future people would be able to enjoy searching temperature, humidity, light, smoke of various times, in a forest, according to their own preferences. The fire service department may focus more on the temperature and smoke of the region, while the zoologists may be more interested in light and temperature. These application requirements call for a key function from the system design, an efficient processing of the continuous multidimensional top-k queries in sensor networks.

Top-k query processing has long been an important task in various research domains [29], [39] where the k highest (or lowest) data points are retrieved from a large data set. The unique challenges we face in the aforementioned applications come from two aspects. First, from the client's perspective, there can be a large number of different users. Each of them has his own preference; they not only put different weights on different dimensions of data but also on different time periods of data. This multidimensional nature has made many algorithms [2], [34], [38] developed for 1D top-k queries unsuitable or inefficient. With multidimensional sensor data, they have to pose as many queries as the number of user requests since each user could assign a set of weights representing his own preference, which results in huge communication overhead. Second, from the system's perspective, the sensing devices are distributed and usually with great resource constraints. Especially, for the energy limited sensor nodes, the communication should be tightly optimized. Therefore, most centralized schemes developed in the database community such as [39] cannot be applied here. As an example, ordinary nodes have no information about the user preference and if all sensor data are extracted, it could incur a large amount of communications.

To this end, we develop a framework based on *dominant graph* (DG) [39]. DG is a layered data structure to build a relationship between different data points in multidimensions. To successfully apply DG in distributed sensor networks, we face many challenges. In [39], the server extracts top-k results using the given query function. This is a pure centralized operation. In a distributed sensor network,

- *H. Jiang and J. Cheng are with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China. E-mail: {hongbojiang2004, Jiecheng2009}@gmail.com.*
- *D. Wang is with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. E-mail: csdwang@comp.polyu.edu.hk.*
- *C. Wang is with the InterDigital Communications, 781 Third Avenue, King of Prussia, PA 19406. E-mail: cgwang@ieee.org.*
- *G. Tan is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, 1068 Xueyuan Avenue, Shenzhen University Town, Shenzhen 518055, China. E-mail: guangtan@gmail.com.*

Manuscript received 27 Jan. 2012; accepted 31 Jan. 2012; published online 8 Feb. 2012.
Recommended for acceptance by S. Papavassiliou, N. Kato, Y. Liu, C.-Z. Xu, and X. Wang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-2012-01-0063.
Digital Object Identifier no. 10.1109/TPDS.2012.69.

unfortunately, it is impossible for the sensor nodes to know the query functions since the query is performed at the sink. To successfully carry out the top-k query in sensor networks, there must be interactions between the sink and the sensors in the network. The incurred communication traffic dominates the energy consumption, and is of crucial importance for system efficiency, thus it requires special treatment.

We believe the best way to handle such difficulties is to develop a framework for the sensor network. In the framework, the complexity of multidimensional top-k query processing and system efficiency can be clearly assigned to sinks and the sensor nodes, respectively. In this paper, we discuss our framework and the associated algorithms. To the best of our understanding, we are the first to develop solutions for continuous multidimensional top-k query processing in sensor networks. Our contributions are summarized as follows:

- We propose a novel framework that efficiently realizes DG in a distributed sensor network. The framework supports top-k queries for arbitrary user-defined linear convex query functions over multidimensional sensor data.
- For each sensor, where the query function is unknown, we develop a top-k extraction algorithm to efficiently retrieve necessary data points that will be sent to the sink.
- We propose schemes working with the top-k extraction algorithm for reducing communication traffic. Our key observation is that the continuously collected sensor data does not change abruptly. With previously generated global top-k results, we develop novel scheme for each node to update and maintain its local dominant graph for later data suppression. In addition, we propose local filters to further reduce the communication traffic.
- We develop a comprehensive set of modules and interactions. Our framework is general enough so that many modules can be added piece by piece. We show two examples on how to handle data aging and accommodate approximate queries so as to adapt to different user requirements and system performance.
- We evaluate our framework on both real and synthetic data sets. Simulation results show that our schemes can reduce the total communication cost by up to 90 percent, compared with the centralized scheme or a straightforward extension from previous top-k algorithm designed for 1D sensor data.

The remainder of this paper proceeds as follows: Section 2 describes the background of top-k (preference) query in multidimensional data and our framework; Section 3 is devoted to the top-k query processing algorithms; in Section 4 issues related to overhead reduction and complexity analysis are discussed. We evaluate our schemes in Section 5; Section 6 presents related work and finally, Section 7 summarizes the paper and gives future plan.

2 Architecture Design

2.1 The Problem

Assume a data set $D = \{d_1, d_2, \ldots, d_n\}$. Each d_i is an m-dimensional data point represented by an $(m+2)$-tuple $d_i = (d_i.x1, d_i.x2, \ldots, d_i.xm, d_i.id, d_i.t)$, where $d_i.x1$ through $d_i.xm$ represent its values in the m dimensions, and $d_i.id$ and $d_i.t$ are its unique ID [30], [34], [38] and arrival timestamp (used in slide window queries), respectively. Let a user-defined query function be $F(d_i) = \Sigma_{j=1}^{m} w_j \cdot d_i.xj$ where w_j represents the weight on the jth dimension. A *top-k preference query* (or top-k query for short) is to retrieve k data points from D whose values of function F are the highest. Consistent with [29], [39], we only consider the typical linear convex query functions here. Also known as *aggregate monotone functions* [13], this kind of functions satisfy $F(x1, \ldots\ xm) \leq F(x1', \ldots\ xm')$ if $xj \leq xj'$, for all j. Without loss of generality, we present our design with two dimensions. That is, each data point d_i is a 4-tuple $< d_i.x1, d.x2, d_i.id, d_i.t >$, including its values of two attributes, its unique ID, and its arrival timestamp.

A user starts a query with: 1) a set of user-defined weights; 2) k, the number of data points to be retrieved; and 3) a time period $s \geq 1$ during which the top-k query results should be retrieved. With a sliding window model, the top-k results in the previous time window of size s is returned.

Since each user may assign a set of weights representing his own preference, the sink typically retrieves more than k data points (i.e., the top-k results, denoted by RS) from the network to allow arbitrary user-defined linear convex query functions over multidimensional sensor data. In Section 3, a formal definition of the top-k results RS will be given.

2.2 The Dominant Graph

We say that a data point $d \in D$ dominates $d' \in D$ in multidimensional space if and only if: 1) $d.xj \geq d'.xj$ for each dimension j; 2) there exists l such that $d.xl > d'.xl$. The second condition holds to avoid a special case from the first condition: $d.xj = d'.xj$ for each dimension j. This dominating relationship always holds in top-k queries:

Lemma 1. *If d dominates d', we have, for any aggregate monotone function F, $F(d) \geq F(d')$.*

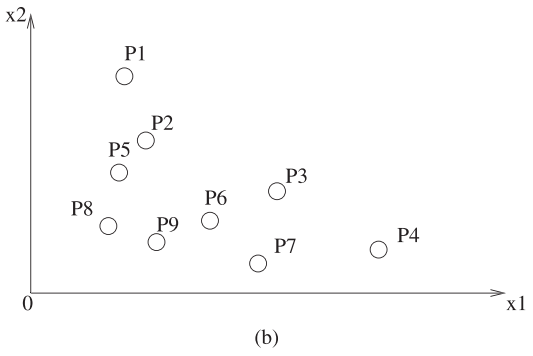
Proof. Since d dominates d', $d.xj \geq d'.xj$ for each dimension j. According to the definition of aggregate monotone function, we have $F(d) \geq F(d')$. □

Following [11], [39], we call the data point d a *maximal point* if it is not dominated by any other data point in D. We refer to the first *maximal layer* L_1 (hereinafter called a *layer*; in some previous works [29] it is also called a *skyband*) as the set of maximal points in D. In addition, the kth layer L_k is the set of maximal points in $D - \cup_{l=1\ldots k-1} L_l''$. Fig. 1 shows an example including the first layer, consisting of {P1, P2, P3, P4}, and the second layer, consisting of {P5, P6, P7}, in a 2D space.

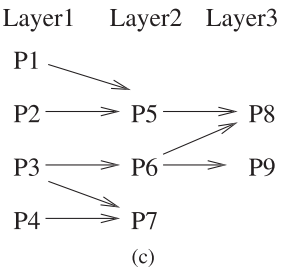
A *dominant graph* (DG) [39] is defined by a set of bipartite graphs g_k where there exist direct edges each of which represents a data point d in the kth layer that dominates a data point d' in the $(k+1)$th layer. Fig. 1c shows an example of DG. L_k'' represents the kth layer of the dominant graph. Many previous algorithms [5], [39] can be used to find each layer of DG. As a result, we build the *parent-children relationship* between the kth layer and the $(k+1)$th layer.

ID	x1	x2
1	1.8	4.0
2	2.1	2.8
3	4.7	1.8
4	6.5	0.8
5	1.7	2.2
6	3.3	1.3
7	4.3	0.5
8	1.5	1.2
9	2.3	0.9

(a)



(b)



(c)

Fig. 1. An example of multiple layers.

2.3 The Multidimensional Top-k Framework

We propose a framework as shown in Fig. 2. This framework resides on top of the data routing layer. We note that the framework does not rely on any routing infrastructure such as [12], [14], [28]. In this framework, there are interrelated base station (or sink) actions and sensor network actions. The main flows of the framework are the data flow and the control flow (shown in the white and gray arrows in Fig. 2). In addition, the framework is open to additional modules.

The data flow (shown in the white arrows) is mainly associated with sensor network actions. Based on its sensor "Data," the individual node is able to set up a local "Dominate Graph." Based on "Query Process" requirements, when a node receives an RS (local top-k results) from its downstream node, it will go through its own DG (to be explained in Section 3), and then the RS ("Local top-k results") will be aggregated and sent along upstream nodes ("Data Aggregation") until reaching the sink.

The control flow involves both sensor network action and base station action. Notice that the base station has the most complete view of the top-k query, the user weight functions, as well as the DG. Therefore, the base station will periodically send the global top-k information ("Feedback Control") to the sensor network (Section 3.2), or adaptively diffuse the global filter information ("Filter") over the whole network (Section 3.3). Ultimately, such information is stored in the query processing unit of the sensor nodes, which will affect the calculation of the top-k results.

With the query processing units in both the base station and the network, the framework is able to incorporate optional units (the Optional Modules in Fig. 2). For example, we can accommodate data aging (detailed in Section 4.2) and approximate queries (Section 4.3). In the remaining part of the paper, we will detail all our designs.

3 Our Algorithms

Our distributed algorithms aim to retrieve information for top-k queries. While the algorithms use a routing tree, the construction of such a tree is out of the scope of this work. Numerous recent studies have focused on this topic, and many of them, for instance [25], can be used in conjunction

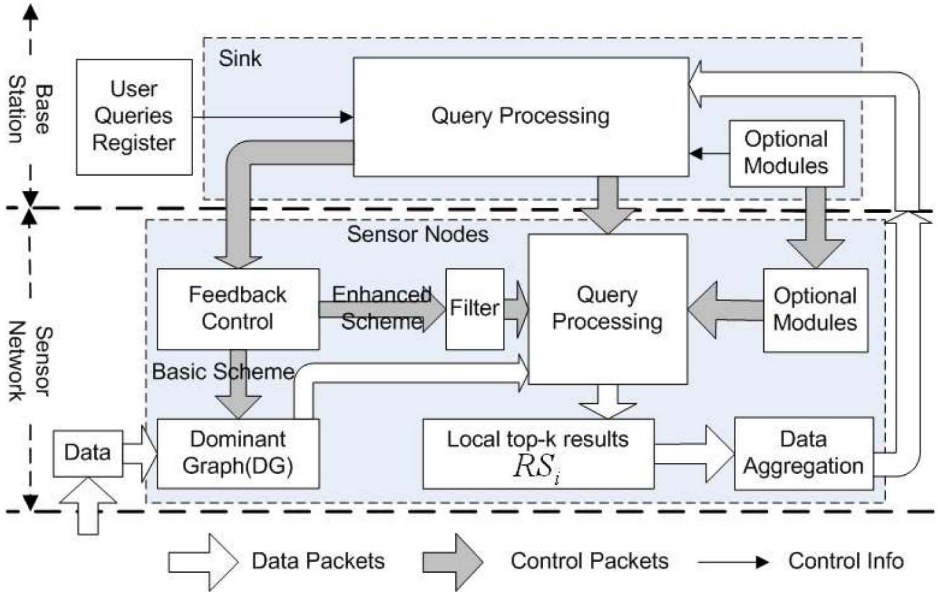


Fig. 2. The framework for multidimensional top-k query processing.

with our approach. Here we simply assume that such a routing tree has been constructed, in which each node knows its parent/children nodes.

3.1 Distributed Top-k Extraction Algorithm

We first consider the top-k extraction algorithm assuming the existence of a DG. As mentioned in Section 1, the extracted top-k results should support arbitrary query functions (thus the extraction algorithm proposed in [39] cannot be used).

Observe that in Fig. 1, for every posed top-1 query, one of {P1, P2, P3, P4} is returned to the user according to the query function F. Recall that our goal is to process multiple queries, each having its own query function F. Accordingly, the sink should retrieve all {P1, P2, P3, P4} for top-1 queries and retrieve {P1, P2, P3, P4, P6} for top-2 queries. It is worth noting that the set of points returned for top-2 queries is a subset of the top two layers: not every point in the second layer needs to be returned. In general, we present a criterion of the top-k query results RS:

Definition 1. *We refer to the top-k query results RS as the data set such that any $d \in RS$ is dominated by less than k data points.*

Note that the top-k results are a subset of the top k layers. Then we have the following theorem.

Theorem 1. *The top-k results built using the rule of Definition 1 enable the sink to answer top-k queries for arbitrary user-defined linear convex query functions over multidimensional sensor data.*

Proof. Proof by contradiction. Assume that there exists data point $d_0 \in RS$ that is dominated by at least k data points. According to Lemma 1, for any aggregate monotone function F, the $F(d_0)$ value cannot be a top-k result. This contradicts the fact that RS contains the top-k results for any query. That is, for any data point $d \in RS$, the number of data points dominating d is less than k. □

This theorem implies that the system often returns more than k data points for top-k query processing. It immediately leads to the following result.

Definition 2. *For any query function F, the data point d is a nontop-k result if it is dominated by at least k data points.*

This definition implies that, in Fig. 1, {P5, P7, P8, P9} are nontop-2 query results for any query function F since any of them is dominated by at least two data points.

In the algorithm, each node first builds its DG locally. We denote by $L_i(d)$ the layer number of a data point d in the local DG (a smaller L means a higher layer) at node i. According to Definition 2, those data points with more than k predecessors are nontop-k results. Furthermore, all their successors in DG are nontop-k results too. The local top-k extraction algorithm at individual nodes is illustrated in Algorithm 1. One observation is that if the data point is in the top-k results, so is its parent. Specifically, when the data point is identified to be a top-k result (Lines 5-6), its children are put into $Q'_{candidate}$ for further identifying in the next loop (Line 7). While Algorithm 1 is performed at ordinary nodes, it can also be used to extract possible top-k results at the sink which can be exploited by our basic and enhanced schemes (to be discussed later).

Algorithm 1. Top-k Extraction Algorithm
Require: Input: a DG of valid data set at node i
Output: the local top-k results, RS_i.
1: All data points at 1st layer of DG are put into RS_i and $Q_{candidate}$.
2: $Q'_{candidate} \leftarrow \emptyset$
3: **while** $Q_{candidate}$ is not empty **do**
4: **for** every data point d in $Q_{candidate}$ **do**
5: **if** d has less than k predecessors in DG **then**
6: $RS_i \leftarrow RS_i \cup \{d\}$ //move d into the result set
7: move all of d's children into $Q'_{candidate}$
8: **else**
9: $Q_{candidate} \leftarrow Q_{candidate} \setminus \{d\}$ //remove d from the candidate set
10: **end if**
11: **end for**
12: $Q_{candidate} \leftarrow Q'_{candidate}$
13: $Q'_{candidate} \leftarrow \emptyset$
14: **end while**

So far we have described the distributed top-k extraction algorithm (Algorithm 1) given a DG. A straightforward approach to continuous multidimensional top-k query processing can then be based on this algorithm. Using Algorithm 1, individual nodes collect top-k result set suitable to support top-k queries for arbitrary user-defined linear convex query functions over multidimensional data. Intermediate nodes aggregate the top-k results RS from the children and the results from their own, and send the aggregated results to parents. The sink also builds a DG to maintain the top-k results to allow user requests of the top-k query with arbitrary time period and weights. However, such an approach may incur heavy traffic. Our algorithm thus allows interaction between ordinary nodes and the sink so that the traffic can be reduced. Next we discuss two schemes to address this issue.

3.2 Basic Scheme

In this section, we propose a basic scheme for query processing. The basic idea behind it is that if ordinary nodes have global top-k information, it is helpful to filter out the nontop-k results for continuous monitoring.

3.2.1 Diffuse Top-k Results RS_{sink} from the Sink

The sink periodically initiates a flooding to disseminate all its top-k results extracted by Algorithm 1 over the whole network. It is noted that the sink continuously collects the top-k results from the sensor network and has a complete view. One fact used in the design is that an ordinary node i in most cases does not need to report all data points in RS_i. Most of data points can be identified to be nontop-k results from the sink's point of view, as long as the node has the knowledge of the previous top-k query results RS_{sink} from the sink.

For the data point d collected by the node i, if d in RS_i is sent to i's parent j, then its layer number has the following property.

Lemma 2. *The layer number of the data point d at node i, $L_i(d)$, is nondecreasing as new data points are inserted into i's DG.*

Proof. Recall that by definition, for the data point d^0 on the kth layer, there is another data point d^1 on the $(k-1)$th layer (see Lemma 2.1 in [39]). Accordingly, we can find a data set $\{d^0, d^1, d^2, \ldots, d^{k-1}\}$ (we call it a *chain*) where d^m dominates d^{m-1} $(1 \leq m \leq k-1)$.

For any new data point d' inserted into the DG, we have two cases. The first case is that we can find a chain $\{d^0, d^1, d^2, \ldots, d^{k-1}\}$ such that: 1) d' is dominated by some d^m while itself dominates d^{m-1}; 2) d is in the chain. In this case, d's layer is increased by 1 if d' dominates d. That is, $L_i(d)$ will be increased after a new data point is inserted into the DG. In the second case $L_i(d)$ remains unchanged. □

Lemma 3. *The data point d's layer number at node i is no more than its layer number at i's parent j. That is, $L_i(d) \leq L_j(d)$.*

Proof. Note that when we compare $L_i(d)$ and $L_j(d)$, due to the aggregation, it is equivalent to the case that there are many new data points to be inserted into the DG at the node j as node j receives many data from its children. According to Lemma 2, we have $L_i(r) \leq L_j(r)$. □

Theorem 2. *The data point d's layer number at the node i is no more than its layer number in the DG at the sink. That is, $L_i(d) \leq L_{sink}(d)$.*

Proof. According to Lemma 3, The layer number of the data point d at node i is no more than its layer number at i's parent. Since the sink is obviously i's predecessor in the routing tree, we have the result. □

Theorem 2 implies that the node often sends "useless" data (nontop-k results) to the sink over the routing tree when those data cannot be included in the final top-k query results. To address this problem, we consider sending all the data points in RS_{sink} back to individual nodes so that the nodes are capable of filtering out most nontop-k results locally.

3.2.2 The Node Processing Module

Each node, after receiving RS_{sink} from the sink, will update its local DG by inserting data of RS_{sink} into its local DG. According to Theorem 2, many data points obtained from the sink may dominate most local ones. In a realistic sensor data set [1], we observed that most data points of RS_{sink} also dominate the local newly collected ones since the data are stable over time. Besides the sink, the sensor nodes dynamically maintain their local DGs: insert new readings and remove outdated ones. For details of the insert and delete operations of DG see [39].

Each node i, after looking through its local DG, sends RS_i calculated by Algorithm 1 in its DG to its parent at the initial phase. Afterward, the node updates its local DG when collecting data itself or receiving data from its children and identify whether the incoming data should be sent. A schematic diagram of the node processing loop is presented in Algorithm 2. First, the node updates its local DG after receiving the set of RS_{sink} (Lines 2-4). Second, since we aim at the sliding window query, when the data point expires, its children have a chance to become the top-k results (Lines 6-10). Similar to Lemma 2, we have the following result.

Algorithm 2. Node i's Processing Module

//Initial Phase

1: Build node i's local DG based on the data points collected by itself and those from its children
2: Calculate RS_i using **Algorithm 1**
3: Send RS_i to node i's parent

//Node Processing Loop

1: **loop**
2: **if** receive the new RS_{sink} diffused by the sink **then**
3: Update its local DG //perform insertion
4: **end if**
5: Remove all outdated data (denoted by ED_i) in DG
6: **if** any data point $d' \in e$'s successor where $e \in ED_i$ **then**
7: **if** d' has less than k predecessors in DG **and** $!d'.sent$ **then**
8: Send d' to node i's parent; $d'.sent \leftarrow$ **TRUE**
9: **end if**
10: **end if**
11: **if** node i has new sensor data d_i **then** {//d_i can be generated by node i or received from i's child}
12: Insert d_i into its DG; $d_i.sent \leftarrow$ **FALSE**
13: **if** d_i has less than k predecessors in its DG **then**
14: Send d_i to node i's parent
15: $d_i.sent \leftarrow$ **TRUE**
16: **end if**
17: **end if**
18: **end loop**

Lemma 4. *For the data point d at node i, $L_i(d)$ does not increase when some expiring data points are removed from the DG.*

Finally, according to Definition 2, the new data point d will be sent if it is dominated by less than k data points in DG (Lines 13-16).

In the basic scheme, each node generates a snapshot of dominant graph with up to k layers. All the sensor nodes will form a hierarchical routing tree. That is, each node will send the snapshot back to the sink in a hop-by-hop fashion with data aggregation at intermediate nodes. While not claiming the credit for the construction of the DG, we note that our contribution here is a novel data aggregation scheme that forwards partial dominant graphs in the routing tree.

3.3 Enhanced Scheme

A drawback of the basic scheme is that the sink has to periodically diffuse all the potential top-k results for later data suppression. When $|RS_{sink}|$ is large, diffusing them will incur a large amount of communication. One fact is that the size of RS_{sink} is often much larger than k. For example, in our experiments, a top-6 query leads to around 40-50 results. In this section, we propose an enhanced scheme to refine the basic scheme and set up a so-called *filter* structure to reduce the diffusion overhead. The basic idea is to avoid diffusing the high layer data in a DG.

In the first step of the basic scheme, the sink diffusion causes around $|RS_{sink}| \cdot N$ traffic. However, if the sink only diffuses the *necessary* data in RS_{sink}, the communication cost

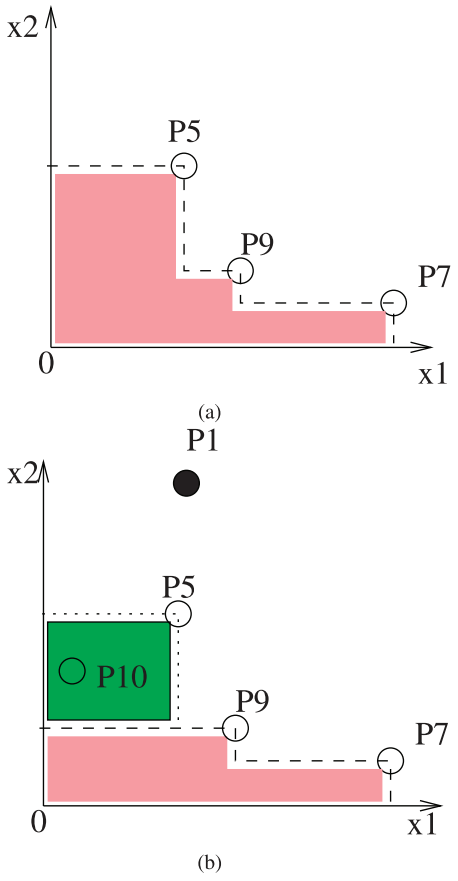


Fig. 3. The filter based on the data set in Fig. 1.

can be reduced. Thus, the problem becomes how to extract these necessary data of RS_{sink}. Fig. 3a shows an intuitive example of our proposed filter ({P5, P9, P7}) to answer a top-3 query according to the data set in Fig. 1.

3.3.1 Filter Construction

We define the filter as a data set $FL = \{d_1, d_2, \ldots, d_F\}$ which are diffused by the sink to all nodes. In the case of 2D space, the d_f in the filter is a tuple $<d_f.x1, d_f.x2, d_f.expired>$.

Definition 3. *We say that the data point d is dominated by the filter $FL = \{d_1, d_2, \ldots, d_F\}$ if there exists at least one data point $d_f \in FL$ that can dominate d.*

After receiving FL_{sink}, the node will not send those data points that are dominated by this filter. For instance, the data points in the pink area in Fig. 3a will not be sent since they are dominated by the filter. (Note that the filter used here is different from the skyband in [29]. For example, the 3-skyband in Fig. 1 is {P8, P9} but the filter for top-3 query is {P5, P7, P9}) Another problem is that given the time window associated with the query, some data may expire. For instance, in Fig. 3b, when P1 expires, P5 cannot be in the filter since it is dominated by only P2 instead of P1P2. In this case, the filter should be composed of only P9 and P7, shown in Fig. 3b. If the new data falls into the green area, say P10, it should be included in the top-3 query results.

Algorithm 3 presents the details of the filter construction. Line 6 shows the calculation of the expiring time of the filter data. The sink then iteratively collects those data which can be dominated by at least $(k-1)$ other data points (Line 7). The final filter will not contain those nodes whose parents have been identified to be included (Lines 10-14).

Algorithm 3. Filter Construction Algorithm
Require: Input: a DG of valid data set at the sink
Output: the filter, FL_{sink}.
1: All data at 1st layer of DG are put into $Q_{candidate}$.
2: $Q'_{candidate} \leftarrow \emptyset$; $FL_{sink} \leftarrow \emptyset$
3: **while** $Q_{candidate}$ is not empty **do**
4: **for** every data point $d \in Q_{candidate}$ **do**
5: **if** d has more than $(k-1)$ predecessors, denoted by $\mathcal{M}$, in DG **then**
6: $d.expired \leftarrow$ minimal expiring time of d and the $(k-1)$ latest-expiring points in $\mathcal{M}$
7: $FL_{sink} \leftarrow FL_{sink} \cup \{d\}$ //move d into the filter
8: **continue**
9: **end if**
10: **for** every child d' of d **do**
11: **if** No parent of d' is in FL_{sink} **then**
12: $Q'_{candidate} \leftarrow Q'_{candidate} \cup \{d'\}$
13: **end if**
14: **end for**
15: **end for**
16: $Q_{candidate} \leftarrow Q'_{candidate}$; $Q'_{candidate} \leftarrow \emptyset$
17: **end while**

The fact that the data point d is dominated by $d_f \in FL_{sink}$ means that there exist at least k data points in the DG at the sink that can dominate d. According to Definition 2, it is easy to prove the correctness of Algorithm 3:

Theorem 3. *If the new data point d is dominated by FL_{sink}, then d is not a top-k result.*

Proof. If the data point d is dominated by FL_{sink}, there exist at least k data points in the DG at the sink that can dominate d. That is, d is not a top-k result. □

3.3.2 Filter Update

It is infeasible for the sink to continuously diffuse its updated filter because of the high communication cost. In addition, by setting an expiring time for the filter data, each node is aware of when the data point should be removed from the filter so that the filter can remain valid at the individual nodes even without being informed by the sink. As shown in Fig. 3b, when $P5$ has expired, the node uses $\{P9, P7\}$ as its filter. The downside is that this filter will become more and more *conservative* because the new data points that are nontop-k results are possibly not filtered out by the filter maintained at the individual nodes. The filter dominating, while guaranteeing the sufficient condition for identifying nontop-k results based on Theorem 3, is unfortunately not a necessary condition. Fig. 4 shows an example where many data points in the pink area will be sent to the sink. While the sink is finally capable of finding the unwanted nontop-k data, a significant of communication resource is wasted. To address this problem, the sink should rediffuse the updated filter when the old one is too conservative as shown in Fig. 4.

To that end, we count the number of data points falling in the region that is dominated by the real filter but not by the old one, that is, the pink region in Fig. 4, denoted by

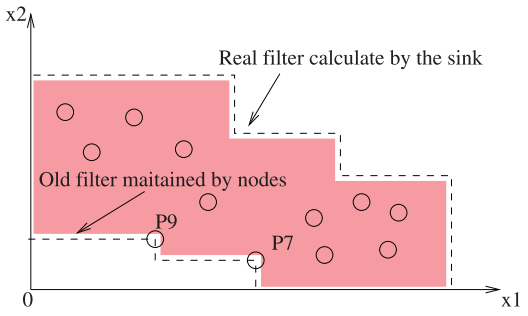


Fig. 4. Filter update.

$Counter$. When the filter is too conservative, that is, $Counter$ is too large, the sink should diffuse its updated filter FL_{sink}. Algorithm 4 presents the detail of filter update process. $\bar{P}$ represents the average path length to transmit data to the sink. The estimation of $\bar{P}$ in the real world can be done by each node sending several small packets to the sink during the setup phase of the network. $|FL_{sink}| \cdot N$ presents the amount of incurred traffic to diffuse the new filter. When there is new incoming data or some data expires in the DG at the sink, it recalculates a new filter (real filter) accordingly (Lines 2-4). If the new data points are not dominated by the old filter maintained by ordinary nodes, but dominated by the new filter (fall in the pink region in Fig. 4), $Counter$ will be increased by one to indicate how many nontop-k results have been sent by nodes (Lines 5-7). In addition, the sink records those nodes denoted by NS that really send the nontop-k results (Line 6). When the old filter is too conservative (Lines 8-11), the sink will diffuse a new filter. It is worth noting that the sink only diffuses the new filter to the nodes in NS (Line 9) instead of all nodes in order to keep a low communication cost.

Algorithm 4. Filter Update Algorithm

```
1:  loop
2:    if new data d is coming at the sink or some data
      points expire in DG then
3:      Update the DG at the sink //perform insertion
        and deletion
4:      Calculate FL_sink based on its DG by Algorithm 3
5:      if new data d (coming from node i) is dominated
        by FL_sink then
6:        Counter ← Counter + 1; NS ← NS ∪ {i}
7:      end if
8:      if Counter · P̄ > |FL_sink| · |NS| then
9:        The sink diffuses FL_sink to all nodes in NS
10:       Counter ← 0; NS ← ∅
11:     end if
12:   end if
13: end loop
```

3.3.3 The Node Processing Module

Like in the second step of the basic scheme, each node updates its local filter and prepares the data set for transmission. The node processing module is illustrated in Algorithm 5. It updates the filter FL_i locally when it receives a new filter data set FL_{sink} diffused from the sink (Lines 2-4). Then the node removes the outdated data in the filter (Lines 5-9). The node looks up those data points which are previously not in the top-k results but should be sent later (Lines 10-14). Finally, if the new data d_i is dominated by FL_i, it will not be sent to the node's parent (Lines 15-20). Here TS_i is the transmitted data set and will be updated at each loop to remove the expiring data.

Algorithm 5. Node i's Processing Module

```
//Initial Phase
... //similar to Algorithm 2
//Node Processing Loop
 1: loop
 2:   if receive the new filter FL_sink diffused by the sink
      then
 3:     FL_i ← FL_sink; TS_i ← ∅
 4:   end if
 5:   for each data point d_if ∈ FL_i do
 6:     if d_if expires according to d_if.expired then
 7:       FL_i ← FL_i\{d_if} //remove outdated data
          points in the filter
 8:     end if
 9:   end for
10:   for each valid data point d' not in TS_i do
11:     if d' is not dominated by FL_i and d' is not
        dominated by at least k data points in TS_i then
12:       Send d_i to i's parent; TS_i ← TS_i ∪ {d'}
13:     end if
14:   end for
15:   if node i has a new data point d_i then {//d_i could be
      generated by node i or received from node i's child}
16:     if d_i is dominated by FL_i or d_i is dominated by at
        least k data points in TS_i then
17:       exit //the data point cannot be in the top-k
          results
18:     end if
19:     Send d_i to i's parent; TS_i ← TS_i ∪ {d_i}
20:   end if
21: end loop
```

The enhanced scheme differs from the basic one in several respects. *First*, each node no longer holds a DG. Instead, a filter is maintained at nodes which roughly represents the last layer in DG for answering top-k queries. *Second*, the updated filter is diffused to those nodes which may not have top-k query results rather than being diffused to all nodes in the network (Line 9 in Algorithm 4). *Finally*, the sink diffusion is performed adaptively instead of periodically, providing a mechanism to improve energy efficiency.

4 Discussion

4.1 Time and Message Complexity

Time and message complexity are important factors for efficient top-k query processing. Let N be the total number of nodes in the network, $\bar{P}$ be the average path length to the sink. $\log N$ could be a very optimistic estimate of $\bar{P}$, but if the sensors are deployed in the real world, say roughly in 2D, typically we have the transmission path length of $\Omega\sqrt{N}$ (or $\Omega N^{1/3}$ in 3D) where Ω is a constant. Obviously the message complexity is $O(N\sqrt{N})$ if all nodes continuously send their readings to the sink. Let M be the cardinality of the data set within the sliding window (i.e., the number of

the active data points over all nodes). Besides, with basic scheme, the sink diffuses its RS_{sink} every T time ticks. For simplicity, we consider the case where the data are uniformly distributed [29]. The following theorems show the scalability of our algorithm.

Theorem 4. *The message complexity of our algorithms is $O(N)$ where N is the network size.*

Proof. First, we consider the basic scheme without diffusion from the sink. In this case, the probability that the new data will be sent is $\overline{|RS_i|}/M$ where $\overline{|RS_i|}$ is the average cardinality of the local top-k results at ordinary nodes (obtained from Algorithm 1). Due to the data aggregation at the intermediate nodes, the number of transmissions for the data is N instead of $\Omega N\sqrt{N}$. Thus, the message complexity of the basic scheme without diffusion from the sink is $C_1 = N \cdot \overline{|RS_i|}/M \in O(N)$.

Second, for the basic scheme, since RS_{sink} diffused by the sink represents the global information of the top-k results, the probability that the new data point will be sent becomes $\frac{|RS_{sink}|}{M \cdot N}$. Therefore, to transmit all these $\frac{|RS_{sink}|}{M \cdot N} \cdot N$ data points, the message complexity is $\frac{|RS_{sink}|}{M \cdot N} \cdot N\Omega\sqrt{N}$. In addition, the sink diffuses RS_{sink} every T time ticks and thus each diffusion causes $|RS_{sink}| \cdot N/(M \cdot T)$ traffic. Overall, the message complexity of the basic scheme is $C_2 = \Omega\sqrt{N} \cdot |RS_{sink}|/M + N \cdot |RS_{sink}| \ /(M \cdot T) \in O(N)$.

Finally, we turn to the enhanced scheme where the sink adaptively diffuses the filter consisting the data points close to the k-layer (as such, the cardinality is roughly $1/k$ of $|RS_{sink}|$). The first part of traffic is similar to that of the basic scheme: $\Omega\sqrt{N} \cdot |RS_{sink}|/M$. But for the second part, the diffusion only causes $|RS_{sink}| \cdot N/(M \cdot T \cdot k)$ traffic. In addition, the sink only diffuses FL_{sink} when the diffusion is necessary (see Lines 8-11 in Algorithm 4). When the diffusion is unnecessary, the traffic cost becomes $N \cdot \overline{|RS_i|}/M$. Therefore, the message complexity of the enhanced scheme is $C_3 = \min\{\Omega\sqrt{N} \cdot |RS_{sink}|/M + |RS_{sink}| \cdot N/(M \cdot T \cdot k), C_1\} \in O(N)$. □

The proof of Theorem 4 implies our algorithms have the following properties:

Claim 1. *The enhanced scheme always outperforms the nondiffusion basic scheme and the basic scheme.*

Claim 2. *The basic scheme can cause a higher communication cost than the basic scheme without diffusion.*

This is because when k is large, $|RS_{sink}|$ could be much larger than $\overline{|RS_i|}$. In this case, $C_2 > C_1$.

As for time complexity, the time complexity at ordinary nodes to update DGs is $O(M^2)$ [39]. Besides, as mentioned above, the average path length is $O(\sqrt{N})$. As a result, the time complexity for our proposed algorithm is $O(M^2 \cdot \sqrt{N})$, that is, when $M << \sqrt{N}$ we have:

Theorem 5. *The time complexity of our proposed algorithms is $O(\sqrt{N})$.*

4.2 Data Aging

Users may be more interested in recent data than in past ones and thus a data aging (called information aging in [33]) function can be applied to each data point. The usage of data aging represents that data vectors are contracted toward the origin (shortened) as they age. In particular, each data point can be multiplied by a factor $fac^{(cur-d.t)}$, where cur is the current time and $d.t$ is the generation time of the data point. fac represents the degree of the data aging. For example, if the data point is considered to account for half of original value from 6 am to 12 pm and the sensor reading is generated every 30 seconds like [1], the aging factor fac is around $0.5^{1/720} \approx 0.999$.

Our algorithms can be easily adapted to realize the aging function with minor modifications. First, for the basic scheme, since this only affects DG (at nodes or the sink), all data points at DGs will be updated every time. It is noted that even the data point has been updated, it can be easily recovered by multiplying $fac^{-(cur-d.t)}$. By doing so, updating the DG inherently represents the preference on recent data point. Second, for the enhanced scheme, the node should update the filter FL_i by multiplying the aging factor (between Lines 5 and 6 in Algorithm 5). We show only representative results in Section 5 concerning the aging factor as they were found to be close to those without aging.

4.3 Approximate Queries

When approximate results are acceptable, a sampling-based approach can be used that reduces communication cost by avoiding data transmission from nodes whose data points are unlikely in the query results. To this end, we propose a sampling-based query planning using linear programming, which can be easily integrated to our basic or enhanced scheme. The goal is to minimize the communication cost, while satisfying the user predefined top-k accuracy. In short, our algorithms can be easily extended to allow a tradeoff between the accuracy of the results and the communication cost.

Let c_i be the number of values coming from the node i in RS_{sink} ($\sum_{i=1}^{N} c_i = |RS_{sink}|$). We refer to h_i as the hop distance between node i and the sink, thus $h_i \cdot c_i$ represents the communication cost incurred by node i reporting its top-k results to the sink. In addition, let $u_i \in [0, 1]$ be the probability that the node i will report its results. If the user defines an accuracy $\varphi \in [0\%, 100\%]$ that the sink could capture the top-k results, we have the following linear program:

$$\begin{aligned} \texttt{Minimize} \quad & \sum_{i=1}^{N} u_i h_i c_i \\ s.t. \quad & \sum_{i=1}^{N} u_i c_i \geq \varphi |RS_{sink}| \\ & 0 \leq u_i \leq 1 \quad \forall i. \end{aligned} \tag{1}$$

The first line of (1) represents the optimization goal of minimizing the communication cost. The second line of (1) specifies the accuracy lower bound of the solution. It is practical to integrate this linear program into our schemes. The sink periodically solves the linear program and then diffuses the probability $u_i (1 \leq i \leq N)$. In practice, the ordinary nodes still perform Algorithms 2 or 5 but send data (Lines 8 and 14 in Algorithm 2; Lines 12 and 19 in Algorithm 5) with the probability u_i. (It is noted that for the data received from children, the node does not perform probabilistic operation but deterministic one. Otherwise the

top-k results are unlikely sent back to the sink) Besides, we cannot assume that the sensor data pattern is unchanged all the time, and therefore the sink informs nodes to transmit with full accuracy periodically (every 30 minutes in our implementation) to refresh RS_{sink}.

By integrating the linear programming into our schemes, it may happen that the actual accuracy slightly drops below the user predefined one φ as we try to minimize communication cost. That is, the accuracy lower bound in (1) is in terms of the probability of sensors returning precise results. When an exact accuracy is required, the sink would flood additional queries into the network if the returned results are not sufficient. In that case, we found this is not an issue as long as we slightly increase the value of k when flooding the queries, say retrieving top-(k+1) results. By doing so, sufficient results are often obtained and no additional queries are required.

4.4 Deployment Issues

An important issue related to the deployment is the flexibility for multiple users whose interests on k may vary, say from top-2 query to top-20 queries. According to our design, a large amount of various k values could affect the system performance greatly. Recall that each node will extract top-k results and send to its parent based on Algorithm 1. The size of diffusion messages from the sink, no matter for the basic scheme or for the enhanced scheme, are highly dependent on the value of k (the upper bound of the message complexity highly relies on RS_{sink} or $\overline{|RS_i|}$ according to the proof of Theorem 4). One approach to handling this problem is that the sink estimates from the historical queries an appropriate k value. For example, if there are more that 90 percent users request only up to top-10 queries, the sink will use 10 as the maximum k value, denoted by K, and each node extracts top-10 query results to send to its parent.

In general, the sink is capable of dealing with the case when the user poses top-k ($k < K$) query. For those queries with $k' > K$, however, it is nontrivial as each nodes and the sink only extracts up to top-K results at their local DG. To address this problem for $k' > K$, we can calculate the additional results beyond top-K results in a pipelined fashion like [30] where each node maintains a heap list containing its own data and the last set of data requested from its children. When a node receives a request from its parents for additional data, the node will check its own heap list and send the request to its children that do not have data in the heap list. Besides, the node will send all data at highest layer in all untransmitted data. This process will be ended when the sink obtains enough data to construct the top-k' results.

5 Performance Evaluation

To evaluate the performance of our algorithms, we conduct a series of simulation experiments. We first describe the data sets used and alternative techniques for comparison. Then we present the results and analysis. Due to the space limit, we only present some representative results in this paper.

5.1 The Data Sets

5.1.1 Intel Berkeley Lab Data

This publicly available sensor data set was collected by the Intel Berkeley Research Lab during a 1-month period [1]. The data consist of environmental data regularly collected from 54 nodes spread around their lab. We observed some missing data values for various nodes at different time epochs, and interpolated them with the average of the values during the previous and subsequent epochs at the same node. In the simulations, we select the complete data set of temperature and voltage over a one-week period (February 29th to March 6th, 2004). A node close to the center of the area is assumed to be the sink in each experiment.

5.1.2 Synthetic Data

To evaluate the algorithms in larger networks and with larger data sets, we also generated synthetic networks and synthetic data. The size of the synthetic networks ranges from 100 to 2,000 nodes and the size of data dimensionality ranges from 2 to 5. We place the nodes in a uniformly random distribution. On average each node has six neighboring nodes within its radio range. Data values on each dimension at every node i is modeled as $x_t^i = \alpha_i x_{t-1}^i + e_t$ where $e_t \sim N(0, 0.1)$ (here $N(\cdot,\cdot)$ represents a Gaussian distribution) and $\alpha_i \sim N(1.0, 0.5)$. For each node, 240 data values were generated (120 values for training and the rest for testing). Every node is initialized with $x_0^i \sim U(0, 1)$.

5.2 Alternative Techniques

5.2.1 Centralized Exact

In TinyDB [26], all sensor values are always reported to the sink. This technique offers an error-free propagation of data.

5.2.2 FILA

Two variants of FILA [34] are used as a benchmark for our comparative study. Since FILA focuses on snapshot query instead of history query, and on 1D data, the sink generates a query function F for each user request with weights (w_j). The node receives the query function, calculates the value for each function it receives, and then sorts the data points to find top-k results sent to the sink. This method is hereafter referred to as "FILA snapshot." An alternative method, called "FILA history," uses top-k results over the period of query time only instead of each time tick. Compared with "FILA snapshot," this method can reduce communication cost.

5.2.3 Basic Scheme without Diffusion

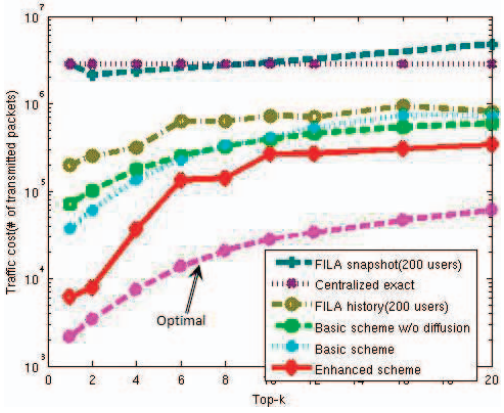
To evaluate the effectiveness of diffusion, we also present the results when the sink does not diffuse RS_{sink} or FL_{sink}.

5.2.4 Optimal Results

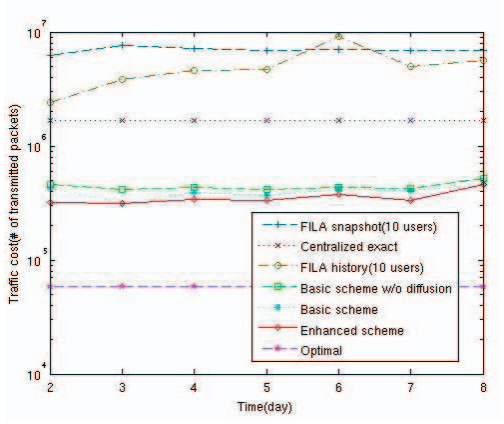
For comparison, we calculate the optimal results of top-k extraction (labeled as "Optimal" in Fig. 5) if the global information is given for sensors. In this case, individual sensors have the global knowledge of all sensor data of the network (note that it is impossible in practice). That is, they are able to calculate exact top-k results and only send those sensor readings in the top-k results back to the sink.

5.3 Evaluation Results

We evaluate the algorithms' performance in terms of total number of transmitted packets by all sensor nodes. For a fair comparison, we assume that each packet contains a



(a)



(b)

Fig. 5. Comparison using the Intel lab data. (a) With varying k values. (b) Over a one-week period ($k = 6$).

single double-precision floating-point number (8 bytes). Accordingly, a 2D data point takes three packets to transmit: two for $d_i.x1$, $d_i.x2$ and one for the timestamp $d_i.t$ and ID $d_i.id$. Besides, we set the minimal sliding windows size to be 1 hour. Obviously, the sink is able to deal with the query with an s-hour ($s \geq 1$) window size since the sink maintains top-k results all the time.

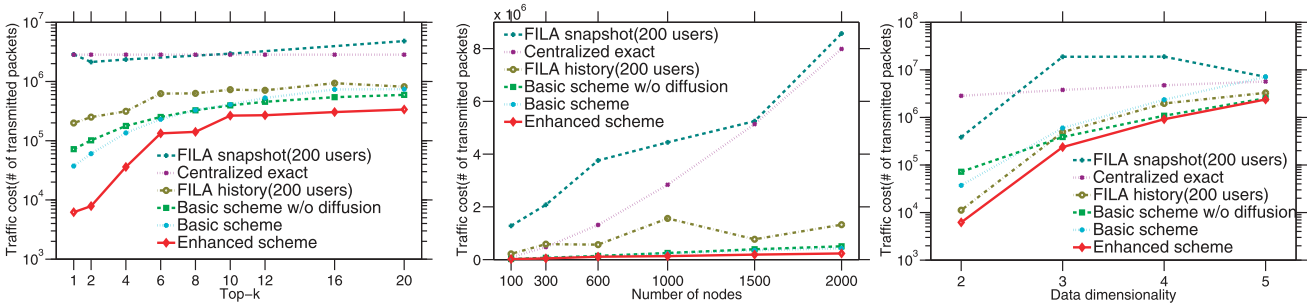
5.3.1 Comparison Using the Intel Lab Data

Fig. 5a shows the total number of transmitted packets, using the Intel Berkeley Lab Data on the first day, as a function of k. *First*, when the number of users is large (say 10 in Fig. 5a), the traffic using FILA is even higher than that using a centralized algorithm where all data are sent back to the sink. Intuitively, the performance gets worse with an increased number of users using FILA, since it is designed for 1D data set without considering user preference across dimensions. *Second*, all the three algorithms, namely the basic scheme without diffusion, the basic scheme with diffusion, and the enhanced scheme, achieve improved performance, thanks to the use of dominant graph which is suitable for high dimensional data. *Third*, compared with the basic scheme, the results show an improvement of about 20-40 percent by the enhanced scheme for a small k (e.g., less than 8). *Fourth*, when k is large, for example more than 10, the basic scheme without diffusion outperforms the basic scheme with diffusion. The reason is that, as indicated by Claim 2 in Section 4.1, with the basic scheme, the sink periodically diffuses RS_{sink} whose cardinality can be large when k is large (a top-6 query often results in 40-50 results). *Fifth*, the enhanced scheme's cost is always lower than that of the basic scheme with diffusion and is close to the results of the basic scheme without diffusion when k is large, since with the enhanced scheme the sink adaptively (not periodically) diffuses its filter FL_{sink}. As a result, it avoids frequently updating the filter at the nodes. *Finally*, compared with optimal results, our algorithms lead to more traffic cost. For example, the cost of "Optimal" is around 30 percent of the enhanced scheme. The reason is that in the case of "Optimal," individual sensors have the global knowledge of all sensor data of the network as we mentioned in Section 5.2.4. In contrast, in practice, the individual nodes have to make decision locally with reliance on the data of their own readings and received from the descendants.

We also fix k and study the stability of our algorithms over time. Fig. 5b depicts the communication cost of answering top-6 queries over 1-week period on the Intel data with different algorithms. Again, our algorithms outperform previous ones including the centralized algorithm and FILA. Specifically, the enhanced algorithm, compared with centralized algorithm where all nodes continuously send back their data, achieves much better performance with only 10-20 percent as much communication cost. Again, the cost of "Optimal" is around 20-30 percent compared to the enhanced scheme.

5.3.2 Comparison Using the Synthetic Data Set

Fig. 6 compares the different algorithms with the synthetic data. First we set the number of sensor nodes to be 1,000 for the 2D data set. Since we set 10 users in Fig. 5, a total of 200



(a)$N = 1000$, 2-dimensional dataset (b)$k = 6$, 2-dimensional dataset (c)$N = 1000$, $k = 1$

Fig. 6. Comparison using the synthetic data set.

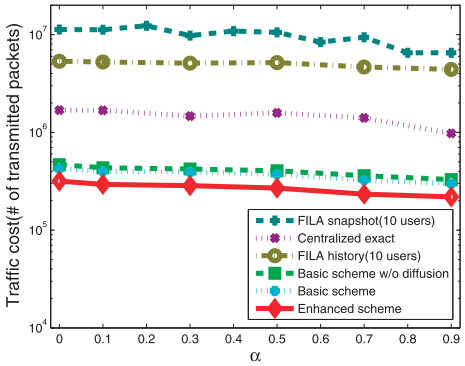


Fig. 7. Comparison under the quasi-UDG radio model.

users are set in Fig. 6 according to the network size for a fair comparison. Fig. 6a depicts the communication cost of answering top-k query with different k values. We find that the results are similar to those shown in Fig. 5a. Our algorithms, again, show superior performance with less that 10 percent of the communication cost caused by the centralized algorithm and FILA. Fig. 6b presents the results of answering top-6 queries for the 2D data set with a range of network sizes. Obviously, the communication costs using our algorithms only increases linearly with the number of sensor nodes, as we discussed in Section 4.1. In contrast, the cost increases at noticeably higher rates with other algorithms. As the network grows in size, our algorithms show very marginal increase in communication cost (for example, less than 10 percent for 2,000 nodes) compared with centralized algorithms. Fig. 6c presents the scalability results of answering top-1 query for 1,000 nodes with data dimensionality varying from 2 to 5. Again, our algorithms produce less traffic compared with others. With higher data dimensionality, all the algorithms have seen a higher communication cost due to the increased packet size (containing high-dimensional data).

5.3.3 Impact of Radio Models

Fig. 7 shows the results for answering top-6 queries under a different radio model: quasi-UDG radio model with parameter $0 \leq \alpha < 1$. In this model, a link exists between two nodes i and j with probability 1 when $dist(i,j) \leq R \cdot (1-\alpha)$ where R is the radio range and $dist(\cdot,\cdot)$ is the distance between two nodes. The link exists with probability $0 < p < 1$ when $R \cdot (1-\alpha) < dist(i,j) < R \cdot (1+\alpha)$ and the link does not exist when $dist(i,j) > R \cdot (1+\alpha)$. We vary α and adjust p so that the average node degree in our tested networks remains nearly the same. Our framework works well with the increase of α, still outperforming others. Besides, we find a slight decrease of traffic cost for all algorithms with an increased α. The reason is that with the node degree fixed, increasing α leads to a decrease of average path length from nodes to the sink (from around 3.7 hops to 2.2 hops in this case). As such, the traffic cost is slightly reduced as shown in Fig. 7.

5.3.4 Efficacy of Optional Modules

We finally evaluate the efficacy of our optional modules. Fig. 8a depicts the communication cost with data aging as a function of k under the assumption that the aging half-life is 6 hours as we mentioned in Section 4.2. We observe that the

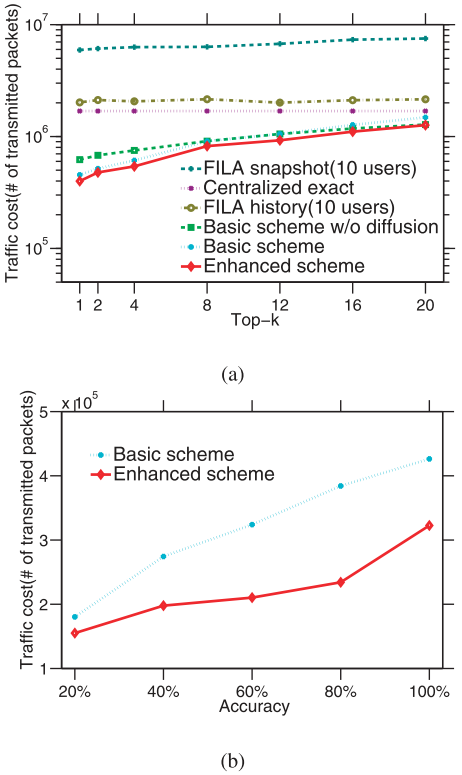


Fig. 8. Results with optional modules using Intel lab data. (a) Data aging. (b) Approximate queries ($k = 6$).

results are similar to Fig. 5a: our proposed algorithms outperform others and the communication cost is increased with larger k values. Besides, Fig. 8a shows data aging leads to overall degraded performance except for centralized scheme. The possible reasons include: 1) some diffused data from the sink, due to data aging, can be useless for data suppression. 2) the recent data are more likely to account for the top-k results and prone to being transmitted. Fig. 8b presents the communication cost, with approximate queries, as a function of accuracy. We observe that with lower accuracy in approximate query, both the basic scheme and the enhanced scheme achieve significant communication reductions. For instance, the communication cost for top-6 query with 20 percent accuracy is only 30-40 percent of that with full accuracy. Another observation is that the communication cost decrease is not linearly proportional to the reduction of predefined accuracy. This is because the sink will periodically ask nodes to transmit data with full accuracy.

6 Related Work

Many studies [4], [9], [16], [17], [18], [23], [31], [32] have explored various data aggregation techniques for query processing in sensor networks in order to reduce communication cost and energy consumption. Some studies strive to design energy-efficient processing methods for top-k queries. Silberstein et al. [30] proposed to use the samples of past sensor readings for query optimization and developed a series of top-k query planning algorithms with linear programming. Zeinalipour et al. [38] proposed to use nonuniform thresholds on the queried attribute in order to

minimize the number of tuples transferred toward the querying node. Similarly, Wu et al. [34] proposed to use a filter at sensor nodes to suppress unnecessary sensor readings. Aiming at continuous top-k monitoring, Zeinalipour-Yazti et al. proposed MINT [36], a framework that maintains the complete results of a query and calculate the upper bounds, so as to reduce the cost of future queries by pruning phase. Instead of using a shortest-path tree for routing, XP [24] employs the cluster tree as the routing infrastructure, in order to aggregate many data points locally before further transmissions. Chen et al. [7] strived to return k highest data points generated within a specified time interval.

All these studies unfortunately have focused on 1D data set. Our distributed, multidimensional, and historical query processing framework is motivated by some previous works [3], [29], [39] in the database community. Babcock and Olston [3] worked on the 1D and error-tolerant top-k monitoring over distributed nodes by setting up the local parameterized constrains which are calculated by the centralized node. Cao and Wang [6] proposed a uniform threshold algorithm (TPUT) to reduce the remote accesses. Besides exact algorithms, Yu et al. proposed TPAT [35], exploiting statistics to further suppress the readings of TPUT. Michel et al. [27] considered approximate top-k queries in distributed environments where at each node the approximation consists of an histogram on the local scores along with a bloom filter. Thus, it is applicable for approximate and on-demand queries. Further, UBK/UBLB-K algorithms [37] addressed exact answers. Mouratidis et al. [29] studied centralized continuous monitoring of top-k queries over a fixed-size sliding window via partially precomputing the future changes. In [39], a dominant graph-based algorithm was introduced for centralized, multidimensional, and snapshot query processing. In sensor networks, however, energy is of crucial importance which requires more special treatment.

7 Conclusion

This paper presents the first work on continuous multi-dimensional top-k query processing in wireless sensor networks. We have developed a framework which effectively monitors user queries and in-network process. More importantly, it incorporates a special data structure, the dominant graph, to maintain top-k query results. The dominant information can facilitate identifying the potential top-k query results for any given preference function and filtering out nontop-k results. The framework is general enough such that optional modules such as approximate query processing, data aging, etc., can be handled. We prove that our algorithms are scalable as the message complexity is proportional to the total number of sensor nodes. The simulation shows that our algorithms reduce the communication cost by up to 90 percent as compared to the centralized scheme and a straightforward extension from previous top-k algorithm on 1D sensor data.

We are interested in several directions in future work. First, we will seek more efficient algorithms to reduce the traffic overhead of our proposed framework. Second, evaluation of our framework on larger scale networks will be carried out. Finally, besides simulations in this paper, a testbed evaluation will be performed.

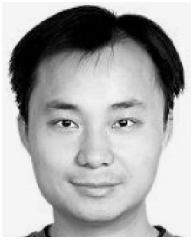
Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 60803115, Grant 60873127, Grant 61073147, and Grant 61173120; by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the Fundamental Research Funds for the Central Universities under Grant 2011QN014; by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the CHUTIAN Scholar Project of Hubei Province; by the Youth Chenguang Project of Wuhan City under Grant 201050231080; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). Dan Wang's work is supported by grant Hong Kong PolyU/G-YG78, A-PJ19, 1-ZV5W, and RGC/GRF PolyU 5305/08E. An earlier version of this work appeared as [15].

References

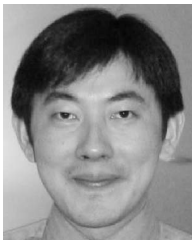
[1] http://db.lcs.mit.edu/labdata/labdata.html, 2012.
[2] P. Andreou, D. Zeinalipour-Yazti, M. Andreou, P.K. Chrysanthis, and G. Samaras, "Kspot: Effectively Monitoring the k Most Important Events in a Wireless Sensor Network," *Proc. IEEE 25th Int'l Conf. Data Eng. (ICDE)*, 2009.
[3] B. Babcock and C. Olston, "Distributed Top-k Monitoring," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03)*, 2003.
[4] M. Bhardwaj and A.P. Chandrakasan, "Bounding the Lifetime of Sensor Networks via Optimal Role Assignments," *Proc. IEEE INFOCOM*, 2002.
[5] S. Borzsonyi, D. Kossmann, and K. Stocker, "The Skyline Operator," *Proc. 17th Int'l Conf. Data Eng.*, 2001.
[6] P. Cao and Z. Wang, "Efficient Top-k Query Calculation in Distributed Networks," *Proc. 23rd Ann. ACM Symp. Principles of Distributed Computing (PODC '04)*, 2004.
[7] B. Chen, W. Liang, and J.X. Yu, "Online Time Interval Top-k Queries in Wireless Sensor Networks," *Proc. Int'l Conf. Mobile Data Management (MDM)*, 2010.
[8] W. Chen, J. Hou, and L. Sha, "Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 3, no. 3, pp. 258-271, July-Aug. 2004.
[9] J. Cheng, H. Jiang, J. Liu, W. Liu, and C. Wang, "On Efficient Processing of Continuous Historical Top-k Queries in Wireless Sensor Networks," *IEEE Trans. Vehicular Technology*, vol. 60, no. 5, pp. 2363-2367, June 2011.
[10] R. Cohen and D. Peleg, "Convergence of Autonomous Mobile Robots with Inaccurate Sensors and Movement," *SIAM J. Computing*, vol. 38, pp. 276-302, 2008.
[11] T.H. Corment, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithm*. The MIT Press, 2001.
[12] S. De, C. Qiao, and H. Wu, "Meshed Multipath Routing: An Efficient Strategy in Wireless Sensor Networks," *Computer Networks*, vol. 43, pp. 481-497, 2003.
[13] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," *Proc. 20th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS '01)*, 2001.
[14] W. Ge, J. Zhang, and G. Xue, "Joint Clustering and Optimal Cooperative Routing in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Comm.*, 2008.
[15] H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan, "Continuous Multi-Dimensional Top-k Query Processing in Sensor Networks," *Proc. IEEE INFOCOM*, 2011.
[16] H. Jiang, S. Jin, and C. Wang, "Parameter-Based Data Aggregation for Statistical Information Extraction in Wireless Sensor Networks," *IEEE Trans. Vehicular Technology*, vol. 59, no. 8, pp. 3992-4001, Oct. 2010.

[17] H. Jiang, S. Jin, and C. Wang, "Prediction or Not? an Energy-Efficient Framework for Clustering-Based Data Collection in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 22, no. 6, pp. 1064-1071, June 2011.
[18] A. Kamra, V. Misra, and D. Rubenstein, "Counttorrent: Ubiquitous Access to Query Aggregates in Dynamic and Mobile Sensor Networks," *Proc. Fifth Int'l Conf. Embedded Networked Sensor Systems (Sensys '07),* 2007.
[19] M. Li and Y. Liu, "Underground Structure Monitoring with Wireless Sensor Networks," *Proc. Sixth Int'l Conf. Information Processing in Sensor Networks (IPSN '07),* 2007.
[20] C. Liu and G. Cao, "Minimizing the Cost of Mine Selection via Sensor Networks," *Proc. IEEE INFOCOM,* 2009.
[21] H. Liu, X. Jia, P. Wan, C. Yi, S. Makki, and N. Pissinou, "Maximizing Lifetime of Sensor Surveillance Systems," *IEEE/ACM Trans. Networking,* vol. 15, no. 2, pp. 334-345, Apr. 2007.
[22] L. Liu, X. Zhang, and H. Ma, "Dynamic Node Collaboration for Mobile Target Tracking in Wireless Camera Sensor Networks," *Proc. IEEE INFOCOM,* 2009.
[23] W. Liu, Y. Zhang, W. Lou, and Y. Fang, "A Robust and Energy-Efficient Data Dissemination Framework for Wireless Sensor Networks," *Wireless Networks,* vol. 12, pp. 465-479, 2006.
[24] X. Liu, J. Xu, and W.-C. Lee, "A Cross Pruning Framework for Top-k Data Collection in Wireless Sensor Networks," *Proc. Int'l Conf. Mobile Data Management,* 2010.
[25] S.R. Madden, M.J. Franklin, J.M. 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.
[26] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "Tinydb: An Acquisitional Query Processing System for Sensor Networks," *ACM Trans. Database Systems,* vol. 30, no. 1, pp. 122-173, 2005.
[27] S. Michel, P. Triantafillou, and G. Weikum, "Klee: A Framework for Distributed Top-k Query Algorithms," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05),* 2005.
[28] S. Misra, G. Xue, and D. Yang, "Polynomial Time Approximations for Multi-Path Routing with Bandwidth and Delay Constraints," *Proc. IEEE INFOCOM,* 2009.
[29] K. Mouratidis, S. Bakiras, and D. Papadias, "Continous Monitoring of Top-k Query over Sliding Windows," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06),* 2006.
[30] A. 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),* 2006.
[31] X. Tang and J. Xu, "Extending Network Lifetime for Precision-constrained Data Aggregation in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2006.
[32] 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),* 2008.
[33] J. Widmer and J.-Y.L. Boudec, "Network Coding for Efficient Communication in Extreme Networks," *Proc. ACM SIGCOMM Workshops Delay-Tolerant Networking (WDTN '05),* 2005.
[34] 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.
[35] H. Yu, H. Li, P. Wu, D. Agrawal, and A.E. Abbadi, "Efficient Processing of Distributed Top-k Queries," *Proc. DEXA,* 2005.
[36] D. Zeinalipour-Yazti, P. Andreou, P.K. Chrysanthis, and G. Samaras, "Mint Views: Materialized in-Network Top-k Views in Sensor Networks," *Proc. Int'l Conf. Mobile Data Management,* 2007.
[37] D. Zeinalipour-Yazti, S. Lin, and D. Gunopulos, "Distributed Spatio-Temporal Similarity Search," *Proc. ACM 15th Conf. Information and Knowledge Management,* 2006.
[38] 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. Workshop Data Management for Sensor Networks (DMSN),* 2005.
[39] L. Zou and L. Chen, "Dominant Graph: An Efficient Indexing Structure to Answer Top-k Queries," *Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08),* 2008.

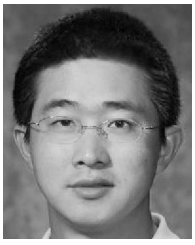


Hongbo Jiang received the BS and MS degrees from Huazhong University of Science and Technology, China. He received the PhD degree from Case Western Reserve University in 2008. After that he joined the faculty of Huazhong University of Science and Technology as an associate professor. His research interests include computer networking, especially algorithms and architectures for high-performance networks and wireless networks. He is a member of the IEEE.

Jie Cheng received the BS and MS degrees from National University of Defense Technology, China. He is currently working toward the PhD degree in Huazhong University of Science and Technology since 2006. His research interests include wireless networking, especially algorithms and architectures for sensor networks.



Dan Wang (S'05-M'07) received the BSc degree from Peking University, Beijing, China, in 2000, the MSc degree from Case Western Reserve University, Cleveland, Ohio, in 2004, and the PhD degree from Simon Fraser University, Burnaby, B.C., Canada, in 2007; all in computer science. He is currently an assistant professor at the Department of Computing, The Hong Kong Polytechnic University. His research interests include wireless sensor networks, Internet routing, and peer-to-peer networks. He is a member of the IEEE.



Chonggang Wang received the PhD degree in computer science from Beijing University of Posts and Telecommunications. He has conducted research with NEC Laboratories America, AT&T Labs Research, and University of Arkansas, and Hong Kong University of Science and Technology. His research interests include future Internet, machine-to-machine (M2M) communications, and cognitive and wireless networks. He has published more than 80 journal/conference articles and book chapters. He is on the editorial board for IEEE Communications Magazine, IEEE Networks Magazine, IEEE Technology News, *ACM/Springer Wireless Networks Journal*, and *Wiley Wireless Communications and Mobile Computing Journal*, *Wiley Security and Communication Networks Journal*. He is a senior member of the IEEE.

Guang Tan received the BS degree from the Chongqing University of Posts and Telecommunications, China, in 1999, the MS degree from the Huazhong University of Science and Technology, China, in 2002, and the PhD degree in computer science from the University of Warwick, United Kingdom, in 2007. He is currently an associate researcher at Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, China, where he works in the area of distributed systems and networks. From 2007 to 2010, he was a postdoctoral researcher at INRIA-Rennes, France. He is a member of the IEEE.