This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Continuous Multi-dimensional Top-k Query Processing in Sensor Networks

Hongbo Jiang[1] Jie Cheng[1] Dan Wang[2] Chonggang Wang[3] Guang Tan[4]

[1]Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China
[2]Department of Computing, Hong Kong Polytechnic University, HK
[3]NEC Laboratories America, Princeton, NJ 08540
[4]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China
[1]{hongbojiang2004,jiecheng2009}@gmail.com, [2]csdwang@comp.polyu.edu.hk
[3]cgwang@ieee.org, [4]guang.tan@siat.ac.cn

***Abstract*—Top-k query has long been an important topic in many fields of computer science. Efficient implementation of the top-k queries is the key for information searching. With the new frontier such as the 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, different users may request different set of information, with different priorities and at different times. Thus, the top-k search not only should be multi-dimensional, but also across time domain. From the system's perspective, the 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 the two ends. The sensor network maintains an efficient dominant graph data structure for data readings. A simple top-k extraction algorithm is used for the user query processing and two schemes are proposed to further reduce communication cost. Our proposed methods can be used for top-k query with any linear convex query function. To the best of our knowledge, this is the first work for continuous multi-dimensional top-k query processing in sensor networks; and our simulation results show that our schemes can reduce the total communication cost by up to 90%, compared with the centralized scheme or a straightforward extension from previous top-k algorithm on one-dimensional sensor data.

I. Introduction

The recent development of sensor networks has made it possible for people to search information not only in the cyber space, but also in the physical world. To illustrate by a concrete example, it would be not in the remote future that people would be able to enjoy searching the temperature, humidity, light, smoke of various time, 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 the light and temperature. These application requirements ask for a key function from the system design, an efficient processing of the continuous multi-dimensional top-k queries in sensor networks.

Top-k query processing has long been a topic in various research communities [12], [18] 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 multi-dimensional nature has made many algorithms [2], [16], [17] developed for snapshot, one-dimensional top-k queries unsuitable or less efficient if a naive extension is made. With multi-dimensional 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, the large body of centralized schemes developed in database community such as [18] is not applicable. As an example, ordinary nodes have no information about the user preference and if all sensor data is extracted, it could incur a large amount of communication.

To this end, we develop a framework based on *dominant graph* (DG for short) [18]. DG is a layered data structure to build a relationship between different data points in multi-dimensions. To successfully apply DG in distributed sensor networks, we face many challenges. In [18], the server extracts top-k results using the given query function. This is a pure centralized operation. In a distributed sensor network, 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, is of crucial importance for system efficiency, and requires more specific 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 multi-dimensional 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

978-1-4244-9921-2/11/$26.00 ©2011 IEEE

our understanding, we are the first to develop solutions for continuous multi-dimensional top-k query processing in sensor networks. Our contributions are summarized as follows:

- We propose a novel framework efficiently realizing DG in distributed sensor networks. The framework supports top-k queries for arbitrary user-defined linear convex query functions over multi-dimensional 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 which further reduces the communication traffic.
- We evaluated our framework on both real and synthetic data sets. Our simulation results show that our schemes can reduce the total communication cost by up to 90%, compared with the centralized scheme or a straightforward extension from previous top-k algorithm designed for one-dimensional sensor data.

The remaining part of this paper proceeds as follows: Section II outlines the background of top-k (preference) query in multi-dimensional data and the framework we proposed. Section III is devoted to the top-k query processing algorithms. We evaluate our schemes in Section IV. Section V presents related work and finally, Section VI concludes the paper.

II. Architecture Design

A. The Problem

Assume a data set $D = \{d_1, d_2, \cdots, d_n\}$. Each d_i is an m-dimensional data point represented by an $(m+2)$-tuple $d_i = (d_i.x1, d_i.x2, \cdots, d_i.xm, d_i.id, d_i.t)$, 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 [13], [16], [17] 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 [12], [18], we only consider the typical linear convex query functions here. Also known as *aggregate monotone functions* [7], this kind of functions satisfy $F(x1, \cdots xm) \leq F(x1', \cdots xm')$ if $xj \leq xj'$, for all j. Without loss of generality, we present our design with 2 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 could assign a set of weights representing his own preference, the sink typically retrieves more than k data points (we say them top-k results, denoted by RS) from the sensor network to allow arbitrary user-defined linear convex query functions over multidimensional sensor data. In Section III, a formal definition of the top-k results RS will be given.

B. The Dominant Graph

We say that the data point $d \in D$ dominates $d' \in D$ in multi-dimensional space if and only if: 1) $d.xj \geq d'.xj$ for each dimension j; 2) there exists l such that $d.xl > d'.xl$. 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 [6], [18], 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* $\mathtt{L}_1$ (we call it a *layer* for short in the rest of this paper, and note that in some previous works [12] it is also called a *skyband*) as the set of maximal points in D. In addition, the kth layer $\mathtt{L}_k$ is the set of maximal points in $D - \cup_{l=1\cdots k-1}\mathtt{L}_l$. Fig. 1 shows an example including the first layer, consisting of {P1,P2,P3,P4}, and the second layer, consisting of {P5,P6,P7}, on a 2-dimensional space.

A *dominant graph* (DG) proposed in [18] 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. 1(c) shows an example of DG. $\mathtt{L}_k$ represents the kth layer of the dominant graph. To build a DG, many previous algorithms [5], [18] 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. In this paper, we do not elaborate the DG (for the interested reader, more details can be found at [18]).

III. Our Algorithms

Our distributed algorithms aim to retrieve information for top-k queries. While the proposed 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 [10], can be used in conjunction with our approach. Here we simply assume that such a routing tree has been constructed, in which each node knows its parent/children nodes.

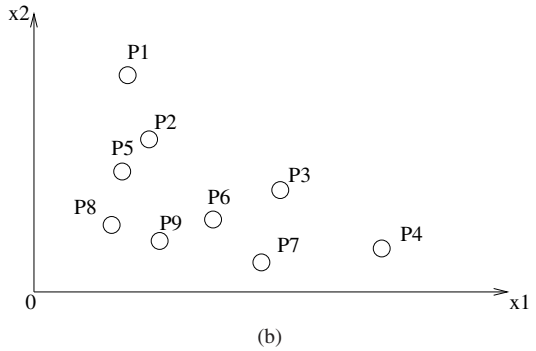
A. Distributed Top-k Extraction Algorithm

We first focus on the top-k extraction algorithm given a DG. As we mentioned in Section I the extracted top-k results should support arbitrary query functions (thus the extraction algorithm proposed in [18] can not be used).

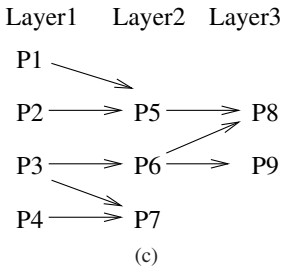
We observe that in Fig. 1 for every posed top-1 query, one of {P1,P2,P3,P4} is returned to the user according to the query

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

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 can not 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. This theorem immediately leads to the following result.

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

This definition implies that, in Fig. 1, P5P7P8P9 are non-top-2 query results for any query function F since any of them is dominated by at least two data points.

In our 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 non-top-k results. Furthermore, all their successors in DG are non-top-k results too. Our 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 will 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 (we will discuss this later).

Algorithm 1 Top-k Extraction Algorithm

Require: Input: a DG of valid dataset 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 finish the efforts on the distributed top-k extraction algorithm (Algorithm 1) given a DG. A straightforward approach to continuous multi-dimensional top-k query processing in sensor networks accordingly works as follows. 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 sensor 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 such that the traffic can be reduced. Next we discuss two schemes working together with Algorithm 1 by which we strive to reduce the communication cost.

B. 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 non-top-k results for continuous monitoring.

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 non-top-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 non-decreasing 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 [18]). Accordingly, we can find a data set $\{d^0, d^1, d^2, \cdots, d^{k-1}\}$ (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 to the DG, we have two cases. The first case is that we can find a chain $\{d^0, d^1, d^2, \cdots, d^{k-1}\}$ 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, the layer where d resides 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 layer number of the data point d 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 layer number of the data point d 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 (non-top-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 such that the nodes are capable of filtering out most non-top-k results locally.

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 could dominate most local ones. In a realistic sensor dataset [1], we observed that most data points of RS_{sink} also dominate the local newly collected ones since the data characteristics exhibit stable over time. Besides the sink, the sensor nodes dynamically maintain their local DGs: insert new readings and remove outdated ones. We do not elaborate on the insertion and deletion of DG; the details can be found in [18].

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. After that, 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.

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).

Note that in our basic scheme, each node generates a snapshot of dominant graph with up to k layers. All sensor nodes, in our design, 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 emphasize that we have proposed a novel data aggregation scheme when forwarding the partial dominant graph in the routing tree in this paper.

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**

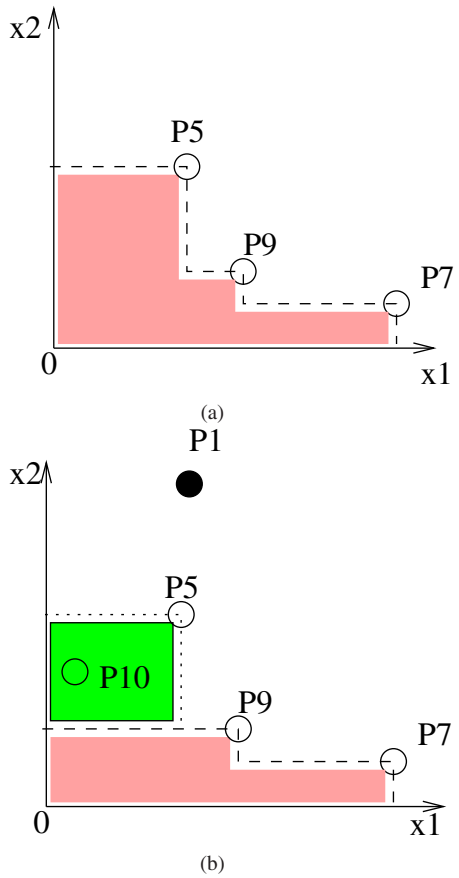


Fig. 2. The filter based on the data set in Fig. 1.

C. 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 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 can be reduced. Thus the problem becomes how to extract these necessary data of RS_{sink}. Fig. 2(a) 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.

1) Filter Construction: We define the filter as a data set $FL = \{d_1, d_2, \cdots, d_F\}$ which are diffused by the sink to all nodes. In the case of two-dimensional space, the d_f in the filter is a tuple $< d_f.x1, d_f.x2, d_f.expired >$.

Definition 3. *We call the data point d is dominated by the filter $FL = \{d_1, d_2, \cdots, d_F\}$ 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. 2(a) will not be sent since they are dominated by the filter (the filter we propose in this paper is different from the skyband in [12]. 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. 2(b), 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. 2(b). 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).

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},*

Algorithm 3 Filter Construction Algorithm

Require: Input: a DG of valid dataset 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**

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. ■

2) Filter Update: It is infeasible for the sink to continuously diffuse its updated filter as a large amount of communication will be incurred. 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 be always valid at the individual nodes even without being informed by the sink. As shown in Fig. 2(b), when $P5$ is expired, the node uses $\{P9, P7\}$ as its filter. The downside is that this filter, in spite of its correctness, will become more and more *conservative* because the new data points that are non-top-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 non-top-k results based on Theorem 3, is by no means a necessary condition. Fig. 3 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 that those data are non-top-k results (that is, it is useless to transmit them), the communication cost to transmit those data is large. To address this problem, the sink should re-diffuse the updated filter when the old one is too conservative as shown in Fig. 3.

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. 3, denoted by $Counter$. 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. $\bar{P}$

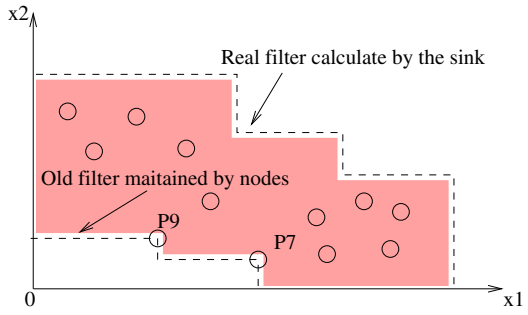


Fig. 3. Filter update.

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 set-up phase of the network. $|FL_{sink}| \cdot N$ presents the incurred traffic to diffuse the new filter. When there is new incoming data or some data expires in the DG at the sink, it re-calculates 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. 3), $Counter$ will be increased by one to indicate how many non-top-k results have been sent by nodes (Lines 5-7). In addition, the sink records those nodes denoted by NS that really send the non-top-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 so as to keep the communication low.

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: $Counter \leftarrow Counter + 1$; $NS \leftarrow NS \cup \{i\}$
7: **end if**
8: **if** $Counter \cdot \bar{P} > |FL_{sink}| \cdot |NS|$ **then**
9: The sink diffuses FL_{sink} to all nodes in NS
10: $Counter \leftarrow 0$; $NS \leftarrow \emptyset$
11: **end if**
12: **end if**
13: **end loop**

3) The Node Processing Module: Each node, as in the second step of the basic scheme, will update its local filter and prepare 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: $FL_i \leftarrow FL_{sink}$; $TS_i \leftarrow \emptyset$
4: **end if**
5: **for** each data point $d_{if} \in FL_i$ **do**
6: **if** d_{if} expires according to $d_{if}.expired$ **then**
7: $FL_i \leftarrow FL_i \backslash \{d_{if}\}$ //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; $TS_i \leftarrow TS_i \cup \{d'\}$
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 can not be in the top-k results
18: **end if**
19: Send d_i to i's parent; $TS_i \leftarrow TS_i \cup \{d_i\}$
20: **end if**
21: **end loop**

The enhanced scheme differs from the basic scheme in several respects. *First*, each node does not hold a DG any more. 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 who 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.

IV. Performance Evaluation

To evaluate the performance of our algorithms, we conduct a series of computer-simulated experiments. We first describe the datasets we used and the 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.

A. The Datasets

1) Intel Berkeley Lab Data: This publicly available sensor data-set was collected by the Intel Berkeley Research Lab during a one-month period [1]. The data consists 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 dataset of temperature and voltage over a one-week period (Feb 29th to Mar 6th, 2004). A node close to the center of the area is assumed to be the sink in each experiment.

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 nodes 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 6 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 $x_0^i \sim U(0, 1)$.

B. Alternative Techniques

1) Centralized Exact: In TinyDB [11], all sensor values are always reported to the sink. This technique offers an error-free propagation of data.

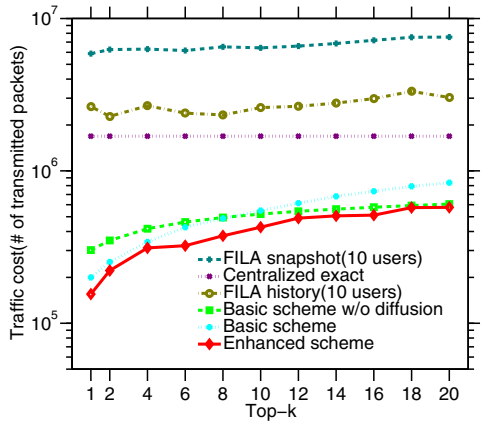
2) FILA: Two variants of FILA [16] are used as a benchmark for our comparative study. Since FILA focuses on snapshot query instead of history query, and on one-dimensional data, the sink generates a query function F for each user request with weights (w_j). The node receives the query function, calculates 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.

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}.

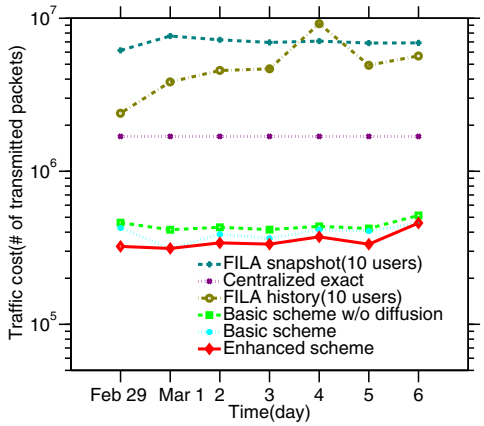
C. Evaluation Results

We evaluate the algorithms' performance in terms of the total number of transmitted packets by all sensor nodes. For a fair comparison, we assume that each packet contains a single double-precision floating-point number (8 bytes). Accordingly, a 2-dimensional 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 one-hour. Obviously the sink is capable of dealing with the query with a s-hour ($s \geq 1$) window size in that the sink maintains top-k results all the time as we mentioned in Section III.

Figure 4(a) 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. 4(a)), the traffic using FILA is even higher than that using a centralized algorithm where all data is sent back to the sink. Intuitively, the performance gets worse with an increased number of users using FILA, since it is designed for one-dimensional dataset without considering the user preference across dimensions. *Second*, all the three algorithms proposed in this paper, 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% 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. *Finally*, 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.



(a)



(b)

Fig. 4. Comparison using the Intel lab data: (a) with varying k values; (b) over a one-week period(k=6).

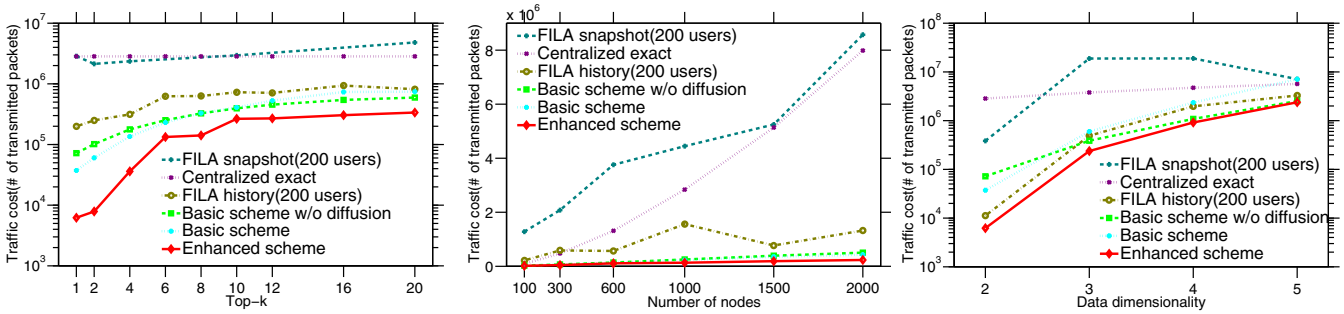
We also fix k and study the stability of our algorithms over time. Fig. 4(b) depicts the communication cost of answering top-6 queries over one-week period on the Intel data with different algorithms. Again, our algorithms outperform previous ones including the centralized algorithm and FILA. Specifically, the enhanced algorithm, compared with centralized algorithm where all nodes continuously send back their data, achieves much better performance with only 10-20% as much communication cost.

Fig. 5 compares the different algorithms with the synthetic data. First we set the number of sensor nodes to be 1,000 for the 2-dimensional dataset. Since we set 10 users in Fig. 4, a total of 200 users are set in Fig. 5 according to the network size for a fair comparison. Fig. 5(a) 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. 4(a). Our algorithms, again, show superior performance with only less that 10% of the communication cost caused by the centralized algorithm and FILA. Fig. 5(b) presents the results of answering top-6 queries for the 2-dimensional dataset with a range of network sizes. Obviously, the communication costs using our algorithms only increases linearly with the number of sensor nodes. 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% for 2,000 nodes) compared with centralized algorithms. Fig. 5(c) 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).

V. Related Work

Many studies [4], [8], [9], [14], [15] have explored various data aggregation techniques for query processing in sensor networks, so as to reduce communication and thus the energy consumption, motivated by the fact that energy conservation is crucial to the prolonged lifetime of a sensor network. Among all those aggregates, top-k query is preliminary for many sensor network applications [16], [17]. Some previous studies strive to propose energy-efficient processing approach for top-k query such that the list of k highest (or lowest) sensor readings are retried. Silberstein *et. al* [13] 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* [17] proposed to use non-uniform thresholds on the queried attribute in order to minimize the number of tuples transferred towards the querying node. Similarly, Wu *et. al* [16] proposed to use a filter at sensor nodes to suppress unnecessary sensor readings.

All these studies unfortunately, despite the intrinsic multi-dimensionality of sensor data [1], focus on one-dimensional dataset. Our distributed, multi-dimensional, and historical query processing framework is motivated by some previous



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

Fig. 5. Comparison with the synthetic dataset.

works [3], [12], [18] in database community. Babcock *et. al* [3] worked on the one-dimensional and error-tolerant top-k monitoring over distributed nodes by setting up the local parameterized constrains which are calculated by the centralized node. Mouratidis *et. al* [12] studied centralized continuous monitoring of top-k queries over a fixed-size sliding window via partially precomputing the future changes. In [18] a dominant graph based algorithm was introduced for centralized, multi-dimensional, and snapshot query processing. In sensor networks, however, energy is of crucial importance which requires more special treatment.

VI. Conclusion

This paper presented the first work on continuous multidimensional top-k query processing in wireless sensor networks. We 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 non-top-k results. The simulation shows that our algorithms reduce the communication cost by up to 90% as compared to the centralized scheme and a straightforward extension from previous top-k algorithm on one-dimensional sensor data.

We are interested in several directions in the future. First, we 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.

Acknowledgement

This work was supported in part through National Natural Science Foundation of China (No.60803115, No.61073147) and National Natural Science Foundation of China - Microsoft Research Asia (No.60933012). Dan Wang's work is supported by grant Hong Kong PolyU/G-YG78, A-PJ19, 1-ZV5W, and RGC/GRF PolyU 5305/08E.

References

[1] http://db.lcs.mit.edu/labdata/labdata.html.

[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. In *Proceedings of IEEE ICDE (demo)*, 2009.

[3] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proceedings of ACM SIGMOD*, 2003.

[4] M. Bhardwaj and A. P. Chandrakasan. Bounding the lifetime of sensor networks via optimal role assignments. In *Proceedings of IEEE INFOCOM*, 2002.

[5] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of IEEE ICDE*, 2001.

[6] T. H. Corment, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithm*. The MIT Press, 2001.

[7] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of ACM PODS*, 2001.

[8] A. Kamra, V. Misra, and D. Rubenstein. Counttorrent: Ubiquitous access to query aggregates in dynamic and mobile sensor networks. In *Proceedings of ACM Sensys*, 2007.

[9] W. Liu, Y. Zhang, W. Lou, and Y. Fang. A robust and energy-efficient data dissemination framework for wireless sensor networks. *Wireless Networks*, 12, 2006.

[10] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad hoc sensor networks. In *Proceedings of USENIX OSDI*, 2002.

[11] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[12] K. Mouratidis, S. Bakiras, and D. Papadias. Continous monitoring of top-k query over sliding windows. In *Proceedings of ACM SIGMOD*, 2006.

[13] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang. A sampling-based approach to optimizing top-k queries in sensor networks. In *Proceeding of IEEE ICDE*, 2006.

[14] X. Tang and J. Xu. Extending network lifetime for precision-constrained data aggregation in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2006.

[15] D. Wang, J. Xu, J. Liu, and F. Wang. Mobile filtering for error bounded data collection in sensor networks. In *Proceedings of IEEE ICDCS*, 2008.

[16] M. Wu, J. Xu, X. Tang, and W.-C. Lee. Top-k monitoring in wireless sensor networks. *IEEE Trans. on Knowledge and Data Engineering*, 19(7), 2007.

[17] 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. In *Proceedings of Workshop Data Management for Sensor Networks (DMSN)*, 2005.

[18] L. Zou and L. Chen. Dominant graph: An efficient indexing sturcture to answer top-k queries. In *Proceedings of IEEE ICDE*, 2008.