Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 681368, 15 pages
http://dx.doi.org/10.1155/2014/681368

Research Article

Histogram Estimation for Optimal Filter Skyline Query Processing in Wireless Sensor Networks

Haixiang Wang,[1] Jiping Zheng,[1,2] Baoli Song,[1] and Yongge Wang[1]

[1] *College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, No. 29 Yudao Street, Qinhuai Distrinct, Nanjing 210016, China*
[2] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*

Correspondence should be addressed to Jiping Zheng; jzh@nuaa.edu.cn

Received 13 January 2014; Revised 4 June 2014; Accepted 22 June 2014; Published 15 July 2014

Academic Editor: Arumugam Nallanathan

The skyline query processing technique plays an increasingly important role for multicriteria decision making applications in wireless sensor networks. The technique of saving energy to prolong the lifetime of sensor nodes is one of the dominating challenges to resource-constrained wireless sensor networks. In this paper, we propose an energy-efficient skyline query processing algorithm, called the histogram filter based algorithm (HFA), to efficiently retrieve skyline results from a sensor network. First, we use historical data at the base station to construct histograms for further estimating the probability density distributions of the sensor data. Second, the dominance probability of each tuple is computed based on the histograms, and the optimal tuple which has the largest possibility of dominance/filtering capability is obtained using in-network aggregation approach. After that, the base station broadcasts the optimized tuple as the global filter to each sensor node. Then, the tuples which do not satisfy the skyline query semantics are discarded to avoid unnecessary data transmissions. An extensive experimental study demonstrates that the proposed HFA algorithm performs more efficiently than existing algorithms on reducing data transmissions during skyline query processing, which saves the energy and prolongs the lifetime of wireless sensor networks.

1. Introduction

Rapid advances of embedded systems, sensing, and wireless communication technologies have fostered the developments of wireless sensor networks. The sensor nodes perceive, gather, and process information of monitoring area, such as light, temperature, and humidity, and transmit the information to remote users via wireless communication. Wireless sensor networks have been widely applied to many fields, such as defense military, environmental monitoring, and traffic management. The sensor nodes are generally battery powered and have limited energy. In real applications, sensor nodes are often deployed in harsh environments, which makes changing batteries unpractical. Therefore, applications over wireless sensor networks need an energy-efficient method to process data gathered by sensor nodes [1].

To support data query processing in wireless sensor networks, TinyDB [2] and Cougar [3] systems have been proposed which developed some basic aggregation operations, such as MAX and MIN. Other types of user queries for a wireless sensor network such as Join queries [4, 5], kNN queries [6], top-k queries [7–9], and skyline queries [10, 11] which have been widely studied in recent years are not supported by these systems. In this paper, we focus on the skyline query as it is one of the important means of multicriteria decision making problems. Börzsönyi et al. [12] first proposed skyline queries and introduced a classic example to describe it in database community. Suppose one person is going on holiday to Nassau and queries the hotel database to find one that is cheap and close to the beach. Generally, the hotels near the beach tend to be more expensive and ones with low prices are far from the beach, which lead to the fact that the database cannot return one best result to the user. However, there may exist some interested hotels not worse than any other hotel considering both distance and price. The set of interesting hotels forms a skyline. Specifically, skyline query is to select tuples which are not dominated by any other tuple from a given set of tuples T of D dimensions space.

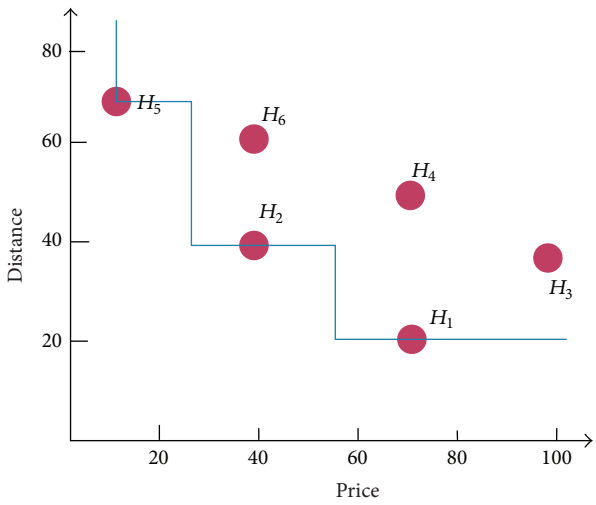


Figure 1: Example of Nassau beach.

A tuple dominates another one if it is as good or better in all dimensions and better in at least one dimension. Take Table 1, for example; there are six hotels listed in the table. Although H_1 is more expensive than H_2, H_1 is closer to the beach than H_2, and no other hotels dominate them; they are both in the skyline. H_3 is dominated by H_1 as it is more expensive and farther from the beach. Similarly, H_4 and H_6 are dominated by H_2. Therefore, the skyline results contain H_1, H_2, and H_6. Figure 1 shows that the hotels lying in the blue line are the results of the skyline query.

The skyline query can be used in many applications in wireless sensor networks. Take monitoring wild animal's behavior, for example. Outdoor biologists can collect information of birds to analyze birds' behavioral habit through wireless sensor networks. If a biologist wants to study the ecological relationship between the groups of birds in a region, he may hope that the observation area has more species and lager quantity of birds. Sensors can be deployed to monitoring these conditions in the forest to find the most suitable area for monitoring birds' behavior. Besides, scientists can also use wireless sensor networks to study wild fish. Sensors can be deployed to monitor river regions with specific water temperature and water flow speed, so that these regions can be used to study the specific kind of fish.

In wireless sensor networks, each sensor node collects data continually, so there is a mass quantity of data in the network. Sensors cannot deal with such a large amount of data due to their limited computing ability. Besides, there is no significance to computing the skyline over the whole time. Therefore, we use a sliding window to constrain the data and retrieve the skyline from the latest data in the sliding window. Most previous studies have resolved the skyline query problem in a centralized way. However, if all sensor data is collected to the base station to conduct a skyline query, it will greatly increase the number of data transmissions. Because the wireless transmission in sensor networks is the main aspect of energy consumption, the large number of transmissions will spend a lot of energy, which leads to the sensor nodes malfunction. If we install a filter in each sensor node to discard tuples which will not be included in final skyline results, it will avoid a large amount of data to transmit so as to reduce the energy consumption and prolong the lifetime of the sensor network. In this paper, we propose a histogram filter based algorithm (HFA) to handle a sliding window skyline query problem. First, we use the historical data at the base station to construct histograms for further estimating the probability density distributions of the sensor data. Second, the optimal tuple is computed based on the histograms and a maximal dominance region mechanism. After that, the base station broadcasts the optimal tuple as a global filter to each sensor node. Then, the tuples which do not satisfy the skyline query semantics are discarded to avoid unnecessary data transmission. The contributions of this paper can be summarized as follows.

Table 1: Information of hotels.

Name	Price	Distance
H_1	70	20
H_2	40	40
H_3	100	40
H_4	70	50
H_5	10	70
H_6	40	60

(1) We take a sliding window skyline query into account, which does not need to collect all sensor data to answer a skyline query and is more practical in real applications.

(2) We take advantages of the multivariate histograms to estimate the distribution of sensor data, thus, capture tuples with the largest capability of the dominance. And a maximal dominance region mechanism is used to compute the optimal tuple. The base station broadcasts the optimal tuple as a global filter to each sensor node so as to prune dominated tuples.

(3) We conduct extensive experiments to evaluate our proposed method. And the experimental results show that the HFA algorithm is energy-efficient.

The rest of this paper is organized as follows. Section 2 introduces the related work of skyline query processing algorithms. In Section 3, a sliding window mechanism is described and a multivariate histogram constructing policy is introduced. In Section 4, we propose an energy-efficient skyline query algorithm based on multivariate histograms. Experimental results are illustrated in Section 5 and Section 6 concludes this paper.

2. Related Work

The skyline query processing techniques have been widely studied in recent years. After Börzsönyi et al. [12] introduced the skyline operator into the database community, a lot of improved algorithms were proposed, which can be divided into two types: with/without indexes. The block-nested-loop algorithm (BNL) and divide-and-conquer algorithm (D&C)

[12] are two typical methods without indexes. Tan et al. [13] proposed a Bitmap algorithm to compute skyline, which progressively return the interesting points. Bitmap exploits a bitmap structure to quickly identify whether a point is an interesting point or not. As the mapping way Bitmap utilizes may cause the length of string to increase rapidly due to increasing values of different dimensions, it will take up a lot of storage space. Chomicki et al. [14] proposed an algorithm named sort-filter-skylines (SFS) as a variant of BNL. And Godfrey et al. [15] introduce the linear elimination sort for skyline (LESS) based on SFS. Börzsönyi et al. [12] also proposed methods with indexes using B-tree and R-tree to handle skyline query problem. Tan et al. [13] introduced Index algorithm. The algorithm exploits a transformation mechanism and a B+-tree index to return skyline points in batches. Each point is transformed into a single dimensional space and stored in a B+-tree structure. Points with some common features are clustered together. Kossmann et al. [16] proposed nearest neighbor algorithm (NN), and it is the first user-friendly skyline computation method. Papadias et al. [17] took advantage of R-tree to establish index for the data and proposed a progressive algorithm called BBS (branch and bound skyline) which is optimal in terms of node accesses.

In addition to a centralized environment, the skyline query has been widely used in distributed, peer-to-peer (P2P), Web, and road network environment. Hose and Vlachou [18] described in detail the skyline query processing algorithm in the distributed environment. Wang et al. [19] proposed Skyframe, which consists of two querying methods for efficient skyline processing in Peer-to-Peer systems. And they introduced a method to balance the query loads among the peers in Skyframe through both load induced data space partitioning and dynamic load migration. The Skyframe algorithm can quickly respnd to a query and has lower communication cost. Tao and Papadias [20] studied skyline computation in stream systems that consider only the tuples that arrived in a sliding window covering the W most recent timestamps. Balke et al. [21] put forward an effective distributed skyline algorithm in Web information systems for the first time according to the characteristics of the independent Web resources distribution. Deng et al. [22] considered multisource relative skyline queries in road networks for the first time. Three algorithms are proposed to process multisource skyline queries in a constrained space.

In the past years, the skyline query problem in wireless sensor networks has been studied in the literature. Huang et al. [23] proposed a filter-based approach (FA) to retrieve skyline results from mobile ad hoc networks, which aims to reduce the cost of the communication among the mobile devices and the cost of query execution on the devices. And this method can be easily extanded to wirelss sensor networks. Kwon et al. [24] proposed a MFT-applied aggregation approach (MFTA) for in-network processing for skyline queries. Chen et al. [25] studied the problem of continuous skyline monitoring in wireless sensor network for the first time. They proposed an advanced approach that employs hierarchical thresholds at nodes and a sophisticated MinMax-threshold approach (MINMAX), which aim to minimize the transmission traffic of the entire network. Xin et al. [26] put forward a sliding window skyline monitoring algorithm (SWSMA) to continuously maintain skyline in the network. There are two filter methods, one is tuple filter approach and another is grid filter approach. Su et al. [10] proposed a skyline sensor algorithm (SkySensor) to efficiently retrieve skyline results from a sensor network. A cluster-based architecture is designed in SkySensor to collect all sensor readings from sensor nodes. Chen et al. [11] studied skyline query optimization and maintenance problems in wireless sensor networks.

The energy consumption of wireless communication has a great effect on the lifetime of sensor networks. It is not practical to collect all sensor readings to the base station to conduct a skyline query. We observe that the algorithms exploited in centralized, distributed, Web, and data stream environment are not applicable to the wireless sensor networks. Existing skyline query algorithms in sensor networks are generally based on filtering some dominated tuples to efficiently answer the skyline queries, so as to avoid unnecessary wireless communications in order to reduce energy consumption of sensor nodes. These methods mostly focus on the local filters; if all the data of one sensor node does not belong to the final skyline set, the local filter approach will not prevent these data from transmitting, which consumes a certain amount of energy. Therefore, a global filter method is necessary to design to prune nonskyline tuples as many as possible for energy-efficient skyline query processing in sensor networks.

3. Preliminaries

Assume that the tuple set of a sensor network is denoted by T, and every tuple has d dimensions. So each tuple t_i with d real-valued attributes can be conceptualized as a d-dimensional point $(t_{i,1}, t_{i,2}, \ldots, t_{i,d}) \in \mathscr{R}^d$, where $t_{i,j}$ is the jth attribute of t_i. In the following sections, we first define the skyline query. Then, we introduce the sliding window skyline query. After that, a policy to construct a multivariate histogram is discussed.

3.1. The Sliding Window Skyline Queries. In this paper, we focus on the skyline query in a wireless sensor network. The skyline operator is first proposed in [12]. It is represented in Definition 1.

Definition 1. The skyline query is to find the skyline set of the given database T, which retrieves tuples that are not dominated by any other tuple. Given two tuples $t_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,d})$ and $t_j = (t_{j,1}, t_{j,2}, \ldots, t_{j,d}) \in \mathscr{R}^d$, t_i dominates t_j if and only if we have $t_{i,k} \leq t_{j,k}$ for $1 \leq k \leq d$ and $t_{i,k} < t_{j,k}$ for some $1 \leq k \leq d$, denoted by $t_i \prec t_j$.

Notice that we consider the smaller value of every attribute as a good one. There is no need that every tuple of the skyline needs to dominate a tuple of T. For instance, in Figure 1, while hotels H_1 and H_2 each dominate two other hotels, hotel H_1 dominates no hotel.

In wireless sensor networks, every sensor node collects data periodically. When a skyline query is conducted to

a sensor network, if all sensor readings are gathered at the base station to process the query, it will consume a large amount of energy, which is not energy efficient. Therefore, the sensor nodes generally do not transmit mass data to base station. Instead, all sensor data is stored locally. The sensor nodes will generate large amounts of data over time. And it is impossible to store all data due to the limited storage ability of nodes. Besides, the data that is collected earlier makes no sense to the skyline query at current moment. So we don't consider these earlier collected data. On the other hand, it is not viable to process skyline query after all data has been collected because the sensor data are continuously generated. Instead, we consider the skyline of data that is falling in a sliding window. The sliding window skyline based on time stamps is represented in Definition 2 [26].

Definition 2. When the sensor node collects data, there is a time stamp $t.c$ to indicate the tuple t's collected time. If the size of the sliding window is S at each sensor node and the current moment is τ, then, if the tuple t in the network satisfies (1), it will be used to compute the sliding window skyline results:

$$\tau - S \leq t.c \leq \tau. \tag{1}$$

3.2. The Construction of Histogram. We propose a histogram based approach to estimate the distribution of data in a wireless sensor network, as histograms can intuitively show the the distribution status of data. If the probability density function of data is known in advance, the tuple's dominance probability can be calculated [26] to obtain the optimal tuple which has the largest capability of the dominance.

3.2.1. The Construction of Univariate Histograms. A univariate histogram is to construct a histogram for data with one dimension. Assume that, in a wireless sensor network, there are n historical tuples stored in base station: $t_1, t_2, \ldots, t_n, t_i \in \mathscr{R}$. Then, the base station trains the tuples to construct a univariate histogram. The histogram consists of many bins. Assume that the kth bin of the histogram is denoted by $B_k = [v_k, v_{k+1})$. Suppose that $v_{k+1} - v_k = h$ for all k; then, the histogram is said to have fixed bin width h. Let $n_k = \mathrm{num}_k(t_i)$ denote the number of tuples in the kth bin, that is, the number of tuples falling in bin B_k; then the probability density function obtained from the histogram is defined as

$$p(B_k) = \frac{n_k}{n} = \frac{\mathrm{num}_k(t_i)}{n} \tag{2}$$

and $t_i \in [v_k, v_{k+1}), i \in [1, n]$.

The tuples of the sensor network are falling in m bins. Assume that $v_{\max}$ and $v_{\min}$ are the largest and smallest values of the training set, respectively; then the bin width h is determined by

$$h = \frac{v_{\max} - v_{\min}}{m}. \tag{3}$$

After that, the interval of every bin of the histogram is calculated using (3). The initial position of the kth bin of the histogram v_k is

$$v_k = v_{\min} + (k-1) * h. \tag{4}$$

Then, the number of tuples falling in each bin is computed. And according to (2), the probability of each bin is calculated, so as to obtain the probability density function of the whole sensor network.

3.2.2. The Construction of Multivariate Histograms. Processing multidimension data query is meaningful to skyline queries in a wireless sensor network; so a multivariate histogram is needed. Assume that tuples in the sensor networks have d dimensions, and the base station trains n historical tuples: $t_1, t_2, \ldots, t_n, t_i \in \mathscr{R}^d$ to construct a multivariate histogram. A multivariate histogram is determined by a partition of the space. Consider a regular partition by hyperrectangles of size $h_1 \times h_2 \times \cdots \times h_d$, choosing hyper cubes as bins would be sufficient if the data is properly scaled, where h_j denotes the width of the jth dimension. Then, the number of bins of the histogram is $m_1 \times m_2 \times \cdots \times m_d$, where m_j denotes the count of the jth dimension.

Constructing a multivariate histogram for a wireless sensor network is as follows. Firstly, the tuples of the network are divided into d groups. Let $t_{1,j}, t_{2,j}, \ldots, t_{n,j}$ denote tuples of the jth group, where $t_{i,j}$ is the value of the jth dimension of tuple t_i. Secondly, the theory used to construct a univariate histogram is utilized to construct a multivariate histogram. Let $v_{j.\max}$ and $v_{j.\min}$ be the largest and smallest values of the jth dimension of the training set, respectively. Then, the width of the jth dimension h_j is determined by

$$h_j = \frac{v_{j.\max} - v_{j.\min}}{m_j}. \tag{5}$$

Assume that $b_{j,k} = [v_{j,k}, v_{j,k+1})$ denotes the interval of the kth block of the jth dimension and $v_{j,k+1} - v_{j,k} = h_j$; so

$$v_{j,k} = v_{j.\min} + (k-1) * h_j. \tag{6}$$

For the tuples of the whole dimensions, let $B(k_1, k_2, \ldots, k_d)$ denote the interval of corresponding bin, where k_j is the k_jth block of the jth dimension and $k_j = 1, 2, \ldots, m_j$:

$$B(k_1, k_2, \ldots, k_d) = \left(b_{1,k_1}, b_{2,k_2}, \ldots, b_{d,k_d}\right). \tag{7}$$

After that, the number of tuples falling in each bin is computed using (7), and the probability of each bin is

$$p(B(k_1, k_2, \ldots, k_d)) = \frac{n_{B(k_1,k_2,\ldots,k_d)}}{n}, \tag{8}$$

where $n_{B(k_1,k_2,\ldots,k_d)}$ denotes the number of tuples falling in bin $B(k_1, k_2, \ldots, k_d)$.

Figure 2 shows the histogram bins of 2-dimensional data. Assume the sensor data has two attributes, data of 1st dimension is divided into 3 groups while data of 2nd dimension is divided into 4 groups, and h_1 and h_2 are bin widths of corresponding dimensions.

4. Histogram Filter Based Algorithm

Many methods focus on pruning nonskyline tuples to efficiently answer the skyline query in a wireless sensor network,

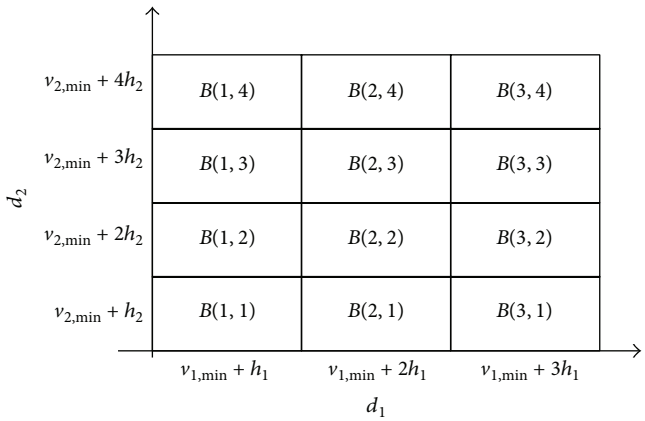


Figure 2: Bins of a multivariate histogram.

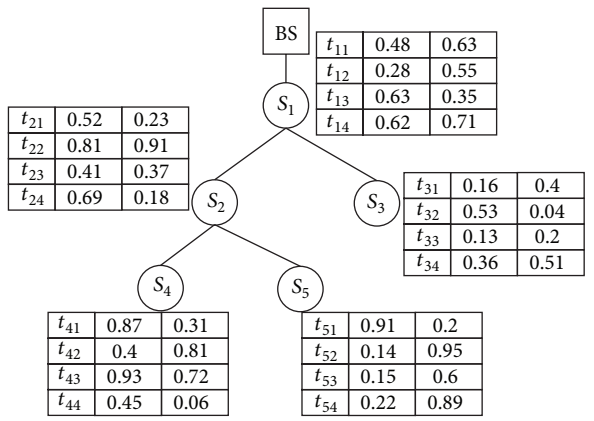


Figure 3: Topology of a sensor network.

so as to reduce the data transmission. In this section, we propose our HFA algorithm. First, a naive approach to carry out a skyline query in a wireless sensor network is introduced. Second, we discuss two methods used to compute the skyline results, known as FA [23] and MFTA [24]. Third, the details of our proposed method are described.

4.1. Basic Approach. The most direct method to compute skyline in a wireless sensor network is to collect all data in the sliding window to the base station and carry out the skyline query utilizing a centralized method. However, only a little part of the tuple set belongs to the final skyline results generally. If all tuples are delivered to the base station, a lot of nonskyline tuples will transmit in the network, which increases the energy consumption of the sensor network.

Because the skyline operator is decomposable [26], the in-network aggregation method can be used to compute skylines, which is called naive approach (NA). A spanning tree is established with the base station as the root. So each intermediate sensor node will not transmit the data to its parent until it has received all its children's skyline results. In this way, the data is combined to a message, so that it reduces the cost of wireless communication. First, each leaf node computes its local skyline and transmits the result to its parents. Second, each intermediate node merges its local skyline and the results delivered by its children and thens send the new results to its parent. Third, after the root node has received all children's results, the final skyline results can be obtained.

4.2. Methods Used to Compute Skyline Results in Wireless Sensor Networks. In the NA approach, each sensor node computes the skyline result locally and prunes some of the nonskyline tuples, so that it reduces the data transmission. However, every sensor node needs to send its local skyline to its parent. It cannot avoid some tuples to transmit because there are some tuples which belong to the local skyline but do not belong to the final skyline. If there is a filter in each sensor node, it will efficiently prevent many of the nonskyline tuples form transmitting and greatly reduce the cost of communication.

Assume that Figure 3 shows the topology structure of a sensor network. There are five sensor nodes $S_1, \ldots, S_5$ in the network. Tuples in each node are as shown in Figure 3. Suppose all tuples are in the sliding window, and we will use the FA algorithm and MFTA algorithm to compute the skyline of given tuples.

4.2.1. FA Algorithm. In FA algorithm, a tuple t_j at the root node can be chosen as a filter because the skyline query is also carried out in the root sensor node. Assume that the data range of the kth dimension of sensor data is $[s_k, b_k]$; then, for a tuple $t_j = (t_{j,1}, t_{j,2}, \ldots, t_{j,d})$, its dominance region is defined as $\mathrm{VDR}_j = \prod(b_k - t_{j,k})$, where k is from 1 to d. The root sensor node sends skyline request as well as the filter tuple to its children. The child nodes prune tuples dominated by the filter tuple and compute the local skyline as well as the VDR of tuples in the local skyline. If there exists one tuple t_k whose VDR_k is larger than VDR_j, then the filter tuple is updated by t_k. In the end, the merge approach is used to compute the final skyline results of the whole network.

Take Figure 3, for example, assume that the base station conducts a skyline query to S_1. The local skyline of S_1 is $\{t_{12}, t_{13}\}$. The upper bound of each dimension is 1. A filter tuple is chosen from the local skyline of S_1, as $\mathrm{VDR}_{12} = (1 - 0.28) * (1 - 0.55) = 0.324$ and $\mathrm{VDR}_{13} = 0.2404$, $\mathrm{VDR}_{12} > \mathrm{VDR}_{13}$, t_{12} is chosen as the filter tuple. S_1 sends the query as well as t_{12} to its child nodes S_2 and S_3. Unfortunately, the filter tuple t_{12} does not dominate any tuple of S_2 and S_3. So S_3 sends its local skyline $\{t_{32}, t_{33}\}$ to S_1. In node S_2, as $\mathrm{VDR}_{23} = 0.3717 > \mathrm{VDR}_{12} = 0.324$, the filter tuple is updated by t_{23}. Filter tuple t_{23} dominates t_{42} at sensor node S_4 but dominates no tuples at S_5. S_4 sends its local skyline $\{t_{42}, t_{44}\}$ to S_2 and S_5 sends its local skyline $\{t_{51}, t_{52}, t_{53}\}$ to S_2. In the end, the base station retrieves the final skyline results $\{t_{32}, t_{33}, t_{44}\}$ using the merge approach.

4.2.2. MFTA Algorithm. MFTA algorithm utilizes a min-score filter tuple (MFT) to prune nonskyline tuples so as to avoid these tuples transmitting between sensor nodes. Tuples closer to the original point have bigger chance to dominate other tuples; so the MFTA algorithm chooses the tuple which

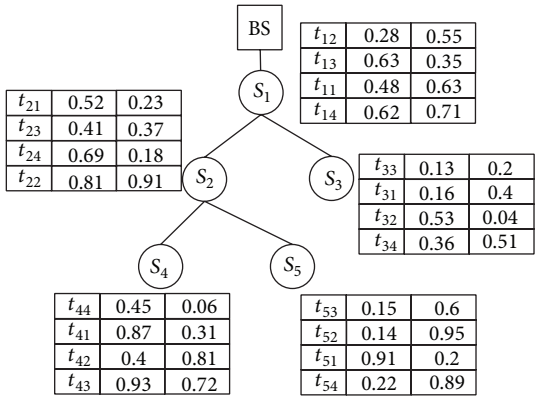


Figure 4: MFTA approach.

is closest to the original point as the MFT. Each sensor node sorts the sensor reading locally according to ascending order of $\sum t_{i,j}$, where j is from 1 to d, as shown in Figure 4. The MFT is chosen based on the following rules. (a) The tuple that makes $\sum t_{i,j}$ smallest is chosen as MFT. (b) If there is more than one tuple satisfying rule (a), then the tuple that makes $\prod(b_k - t_{i,j})$ biggest is chosen as MFT. (c) If more than one tuple satisfies rules (a) and (b) at the same time, then the first sensed tuple is chosen as MFT.

In MFTA algorithm, each nonleaf node computes its local MFT. When the root sensor node receives a skyline query, it will send the query as well as its local MFT to its child nodes and compute its local skyline. When the intermediate node receives the skyline query, it will compare the received MFT and its local MFT to determine whether to update the MFT or not. After that, it will send the MFT and skyline query to its child nodes and compute its local skyline. After all leaf nodes have retrieved the local skyline set, the results are gathered to the base station. And the final skyline result is obtained using an in-network aggregation method.

Take Figure 3, for example; the tuples in Figure 4 have been sorted. When the sensor node S_1 receives the skyline query from base station, S_1 chooses t_{12} as MFT_1 and sends the skyline query as well as MFT_1 to S_2 and S_3. S_1 computes its local skyline = $\{t_{12}, t_{13}\}$. In sensor node S_2, the MFT_1 is updated by t_{21}. S_3 sends the query and $\text{MFT}_2 = t_{21}$ to S_4 and S_5 and computes its local skyline = $\{t_{21}, t_{23}, t_{24}\}$. S_4 and S_5 use MFT_2 to prune dominated tuples and compute their local skyline. S_4, t_{41}, and t_{43} are dominated by MFT_2, so S_4 sends $\{t_{42}, t_{44}\}$ to S_2. Similarly, S_5 sends $\{t_{51}, t_{52}, t_{53}\}$ to S_2. Then, the node S_2 computes the current local skyline $\{t_{23}, t_{44}, t_{52}, t_{53}\}$ and sends the result to S_1. As there is no tuple dominated by MFT_1, S_3 sends its local skyline $\{t_{32}, t_{33}\}$ to S_1. In the end, S_1 retrieves the final skyline $\{t_{32}, t_{33}, t_{44}\}$ by merging its children's results.

It can be seen from the above two examples that although FA algorithm and MFTA algorithm utilize filters at sensor nodes, the pruning effect is not obvious. We observe that the tuple t_{33} in node S_3's local skyline has small value in both dimensions by analyzing the characteristic of the data of the network. If we can choose it as the filter before processing the queries, more nonskyline tuples will be discarded. For example, sensor node S_4 just needs to send t_{44} to its parent, and tuples at node S_5 are all dominated by t_{33}, which prevents these tuples from transmitting in network and greatly reduces the data transmission and energy consumption.

4.3. Histogram Filter Based Algorithm. If one tuple t_i belongs to the local skyline but does not belong to the final skyline, there must exist one tuple t_j that can dominate t_i in the tuple set. Suppose t_i is located at sensor S_i; if t_j is set to be the filter at node S_i, then tuple t_i will be discarded and prevented from transmitting. When a tuple t_{flt} which can dominate the most number of tuples is found and broadcasted to the whole network as a global filter, it will avoid a large number of nonskyline tuples transmitting in the network and reduce the energy consumption. Therefore, we propose a histogram based algorithm to obtain the best global filter. We utilize sensor data on the base station to construct histograms for estimating probability distributions and the dominance probability of each tuple is computed based on the histograms. The optimal filter is the one with the largest possibility of dominance/filtering capability.

4.3.1. The Dominance Probability. Suppose tuple $t_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,d})$ falls in the bin $B(k_1, k_2, \ldots, k_d)$ of the multivariate histogram; then, the dominance probability of t_i can be calculated by computing the bin $B(k_1, k_2, \ldots, k_d)$'s dominance probability, which can be obtained by

$$\begin{aligned} p\_\text{dom}(t_i) &= p\_\text{dom}\left(B\left(k_1, k_2, \ldots, k_d\right)\right) \\ &= \sum_R p\left(B\left(k_1', k_2', \ldots, k_d'\right)\right), \end{aligned} \tag{9}$$

where $R = \{(k_1', k_2', \ldots, k_d') \mid k_1' > k_1, k_2' > k_2, \ldots, k_d' > k_d\}$. If there exists $j \in \{1, 2, \ldots, d\}$, which makes k_j equal to m_j, then $p\_\text{dom}(t_i)$ is set to zero.

4.3.2. The Maximal Dominance Region Mechanism. Using (9), we can obtain the bin which has the largest capability of dominance because (9) is used to compute the dominance probability of the bin which tuple t_i falls in. There may exist more than one tuple fall in the bin; so only utilizing (9) cannot determine which tuple in the bin will be set to be the filter. Therefore, we use method in [23] to compute the dominance region of tuples in this bin to measure the tuples' dominance ability. The maximal dominance region mechanism is also applicable for tuples satisfying $p\_\text{dom}(t_i) = 0$.

Assume that t_i and t_j are two 2-dimensional tuples, their dominance regions are shown in Figure 5. It is obvious that tuples falling in $t_i \cdot \text{DR}$ will be dominated by t_i. Therefore, the larger the value of $t_i \cdot \text{DR}$ is, the more the tuple t_i is able to dominate other tuples. Figure 5 shows that t_j has higher dominance ability than t_i as the value of $t_j \cdot \text{DR}$ is larger than the value of $t_i \cdot \text{DR}$. Let u_j denote the upper bound of the jth dimension of the tuple t_i and let $t_{i,j}$ denote the value of the jth dimension of the tuple t_i, then the dominance region of t_i can be obtained by

$$t_i \cdot \text{DR} = \prod_{j=1}^{d} \left(u_j - t_{ij}\right). \tag{10}$$

```
(1)  n ← sensor numbers
(2)  for i = 1 to n do
(3)      Initialization: LS(i) = ∅ ▷ initializing each node's local skyline;
(4)      sort tuples in node(i) in ascending order of the first dimension
(5)      for each tuple tp in node(i) do
(6)          for each tuple ltp in LS(i) do
(7)              if tp is not dominated by ltp then
(8)                  insert tp to LS(i) ▷ insert the tuple to the skyline
(9)              end if
(10)         end for
(11)     end for
(12)     compute_probability(LS(i)) ▷ compute the dominance probability of each local skyline tuple
(13)     maxp_tuple ← tuples with the biggest dominance probability
(14)     if num(maxp_tuple) > 1 then
(15)         compute_DR(maxp_tuple) ▷ compute the dominance region of every tuple in maxp_tuple
(16)         filter(i) ← tuple with the largest value of DR
(17)     end if
(18) end for
(19) aggregation(filter(i)) ▷ use the in-network approach to find tuple with best dominance ability
(20) broadcast(filter) ▷ broadcast the filter to the network
(21) for each tuple tp in LS(i) do
(22)     if tp is dominated by filter then
(23)         remove tp from LS(i)
(24)     end if
(25) end for
(26) aggregation(LS(i)) ▷ use the in-network approach to compute the final skyline
```

Algorithm 1: HFA algorithm.

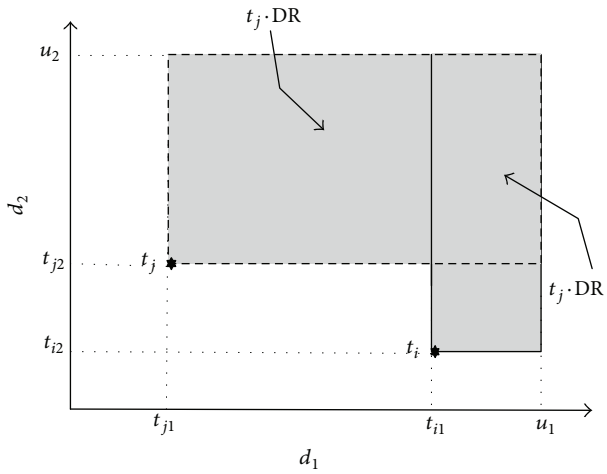


Figure 5: The dominance regions of t_i and t_j.

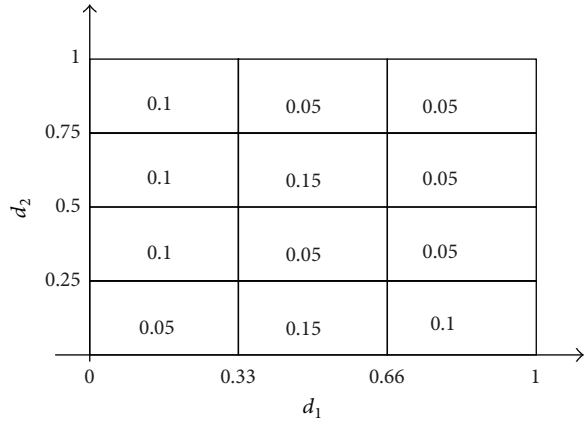


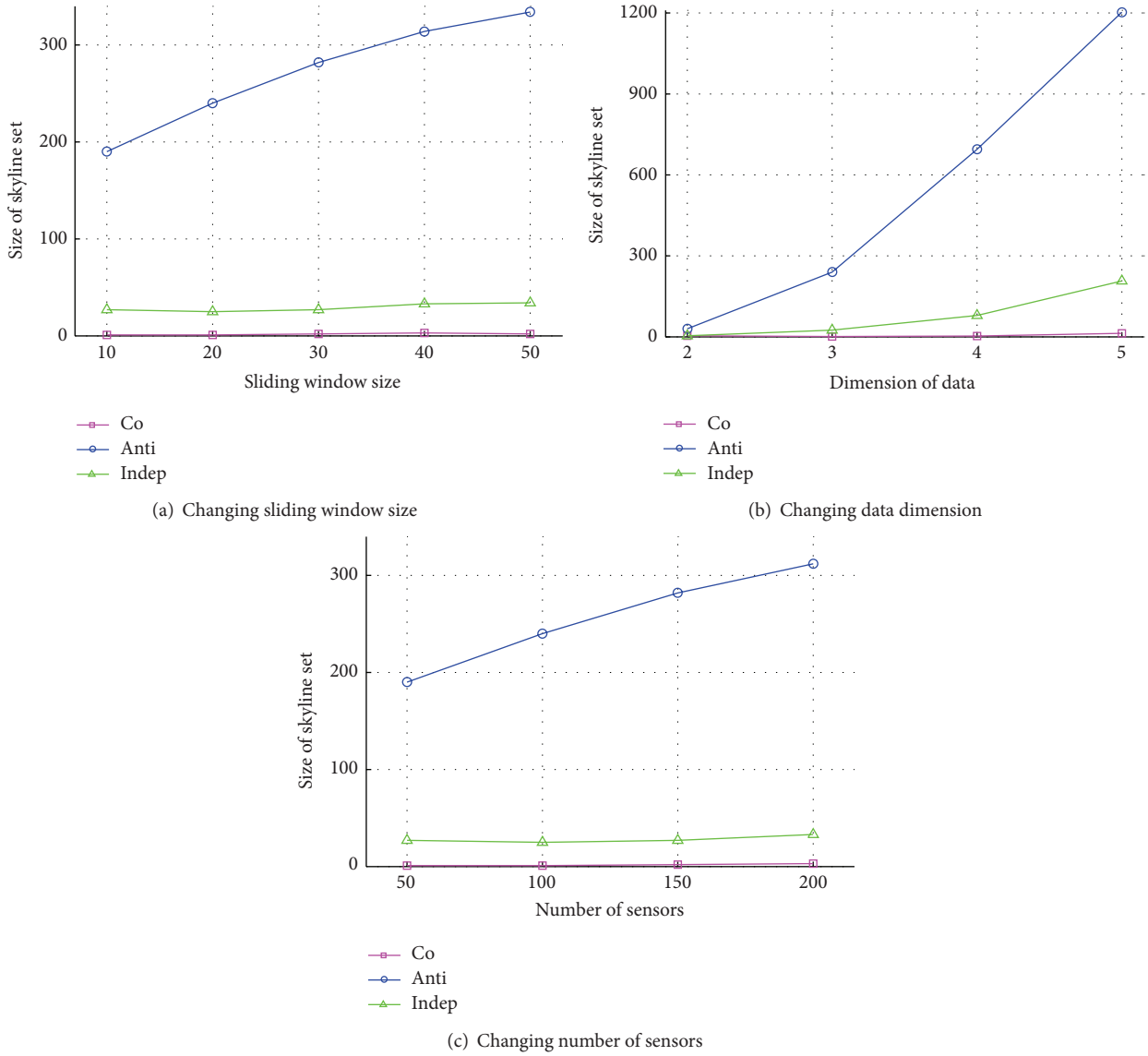
Figure 6: The probability density distribution.

Finally, the tuple which makes (9) and (10) the largest is chosen as final filter tuple t_{flt}.

4.3.3. Description of the HFA Algorithm. Algorithm 1 is the pseudo-code of our proposed HFA algorithm. First, the base station uses the historical data to construct histograms for further estimating the probability density distributions of the sensor data. Second, the dominance probability of every tuple is computed based on the histograms, and the optimal tuple which has the largest dominance ability is obtained using the in-network aggregation approach. After that, the base station broadcasts the optimal tuple as the global filter to each sensor node. Then, the tuples which don't satisfy the skyline query semantics are discarded to avoid unnecessary data transmission. In the end, the final skyline result is calculated using the aggregation approach.

For example, in Figure 3, the tuples are used to construct a multivariate histogram based on Figure 2. Figure 6 shows the probability density distribution of sensor data. Take sensor node S_3, for example; when the skyline query is executed, the sensor node first computes its local skyline $\text{LS}(S_3) = \{t_{32}, t_{33}\}$. Then, the two tuples' dominance probability is computed:

(1) $p\_\text{dom}(t_{32}) = p\_\text{dom}(B(2,1)) = p(B(3,2)) + p(B(3,3)) + p(B(3,4)) = 0.15$;



(a) Changing sliding window size

(b) Changing data dimension

(c) Changing number of sensors

Figure 7: Effect of different synthetic datasets.

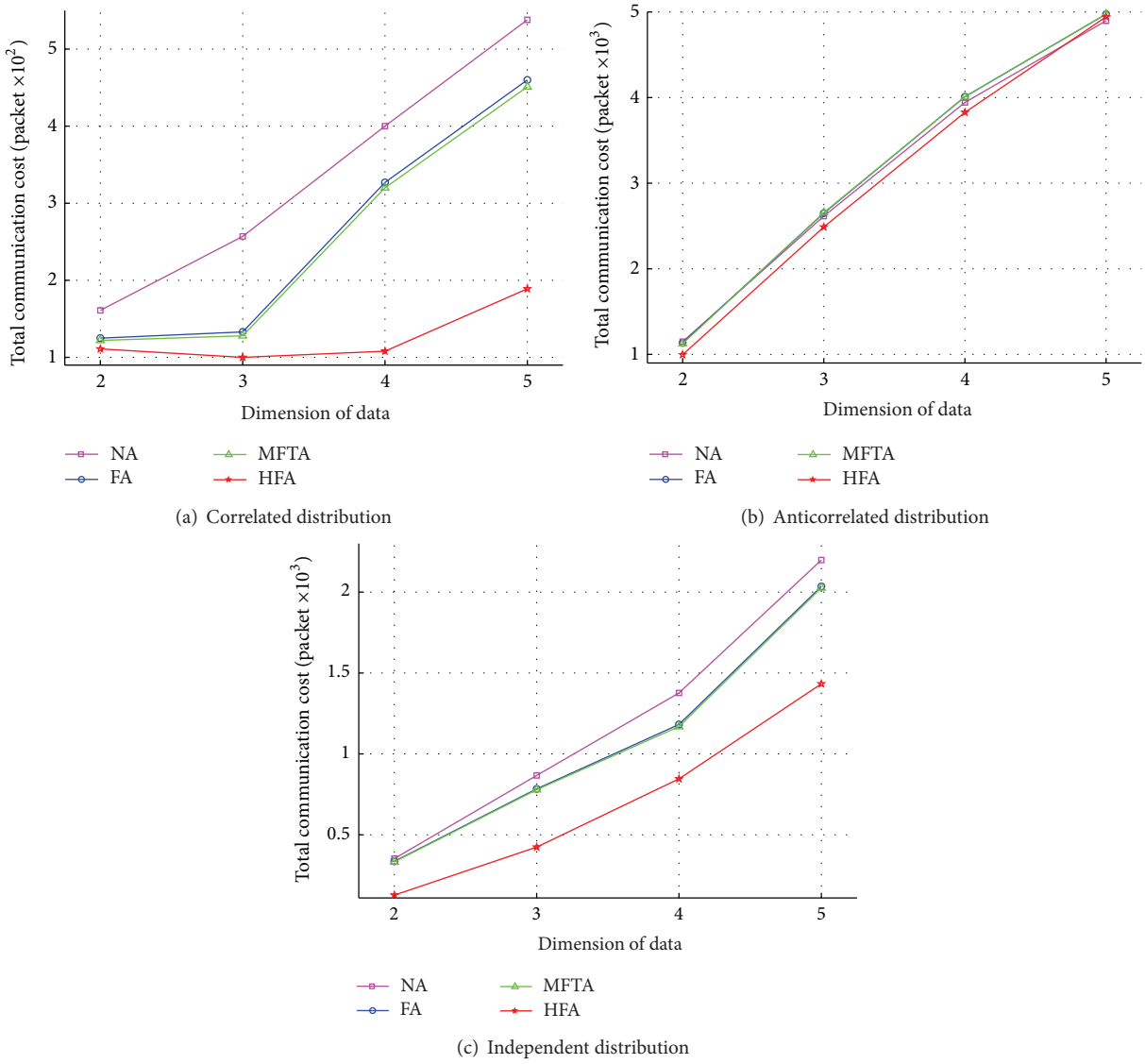
(2) $p\_\mathrm{dom}(t_{33}) = p\_\mathrm{dom}(B(1,1)) = p(B(2,2)) + p(B(2,3)) + p(B(2,4)) + p(B(3,2)) + p(B(3,3)) + p(B(3,4)) = 0.4$.

We can see that $p\_\mathrm{dom}(t_{33})$ is larger than $p\_\mathrm{dom}(t_{32})$, so tuple t_{33} is chosen as the filter tuple. The other sensor nodes obtain their filter tuples in the same way. The tuple t_{33} is chosen as a global filter by using the in-network aggregation method. After that, the base station broadcasts t_{33} as the global filter t_{flt} to each sensor node. In this way, tuples at the leaf node S_5 are all dominated by t_{flt} and there is no need to send any tuple to node S_2. In sensor node S_4, t_{41}, t_{42}, and t_{43} are dominated by t_{flt}, and only t_{44} needs to transmit to S_2. Similarly, the sensor node S_3 only sends t_{32} and t_{33} to S_1. The intermediate node S_2 merges its local skyline and the results transmitted by its children and sends t_{44} to S_1. In the end, the sensor node S_1 retrieves the final skyline set $\{t_{32}, t_{33}, t_{44}\}$. Thus, the HFA algorithm needs to send less data than the methods mentioned in Section 4.2; thus, it is more energy-efficient.

Clearly, when the bin width h_j of every dimension decreases, the number of groups m_j of the corresponding dimension increases. The number of bins grows exponentially as the dimension of sensor data increases, which increases the computational complexity of each sensor node. To facilitate the calculation, we assume that the group number of each dimension is fixed, $m = 3$, so as to reduce the computational cost of sensor nodes.

5. Experimental Evaluation

In this section, we utilize synthesized datasets and real datasets to compare and evaluate the performances of naive approach NA, filter based approach FA, min-score filter tuple approach MFTA, and our proposed HFA algorithm by



(a) Correlated distribution

(b) Anticorrelated distribution

(c) Independent distribution

Figure 8: Effect of dimension of data of synthetic datasets.

MATLAB simulations. All simulations are conducted on a desktop PC running on MS Windows XP Professional. The PC has a Pentium 2.8 GHz CPU and 512 MB memory.

5.1. Experimental Setup. Suppose that a sensor network is monitoring one region whose size is $n \times n$. There are n^2 sensor nodes with communication radius of $2\sqrt{2}$ evenly deployed in the network. The average area one sensor node occupied is set to 1. In the experiment, datasets with correlated, anticorrelated, and independent distributions are generated using the method mentioned in [12]. In addition, we use Intel Lab Data [27] and Washington State Climate Data [28] as the real datasets to conduct our experiments. The Intel Lab Data consists of 54 sensor nodes gathering temperature, humidity, voltage, and light from real world. The Washington State Climate Data is atmospheric data collected from 32 sensors in Washington and Oregon states, which has three dimensions of data: temperature, wind-speed, and humidity. Table 2 shows the experimental parameters and their settings which are used to evaluate performance of the algorithms. The number of sending packets for each sensor node is recorded for performance evaluation.

Table 2: Parameters used in eExperiments.

Parameter	Default	Rang of variation
Dimension of data	3	2, 3, 4, 5
Size of sliding window	20	10, 20, 30, 40, 50
Number of sensors	100	50, 100, 150, 200

5.2. Experimental Results

5.2.1. Experimental Results of Synthetic Datasets. First, we use the synthetic datasets to conduct experiments and compare

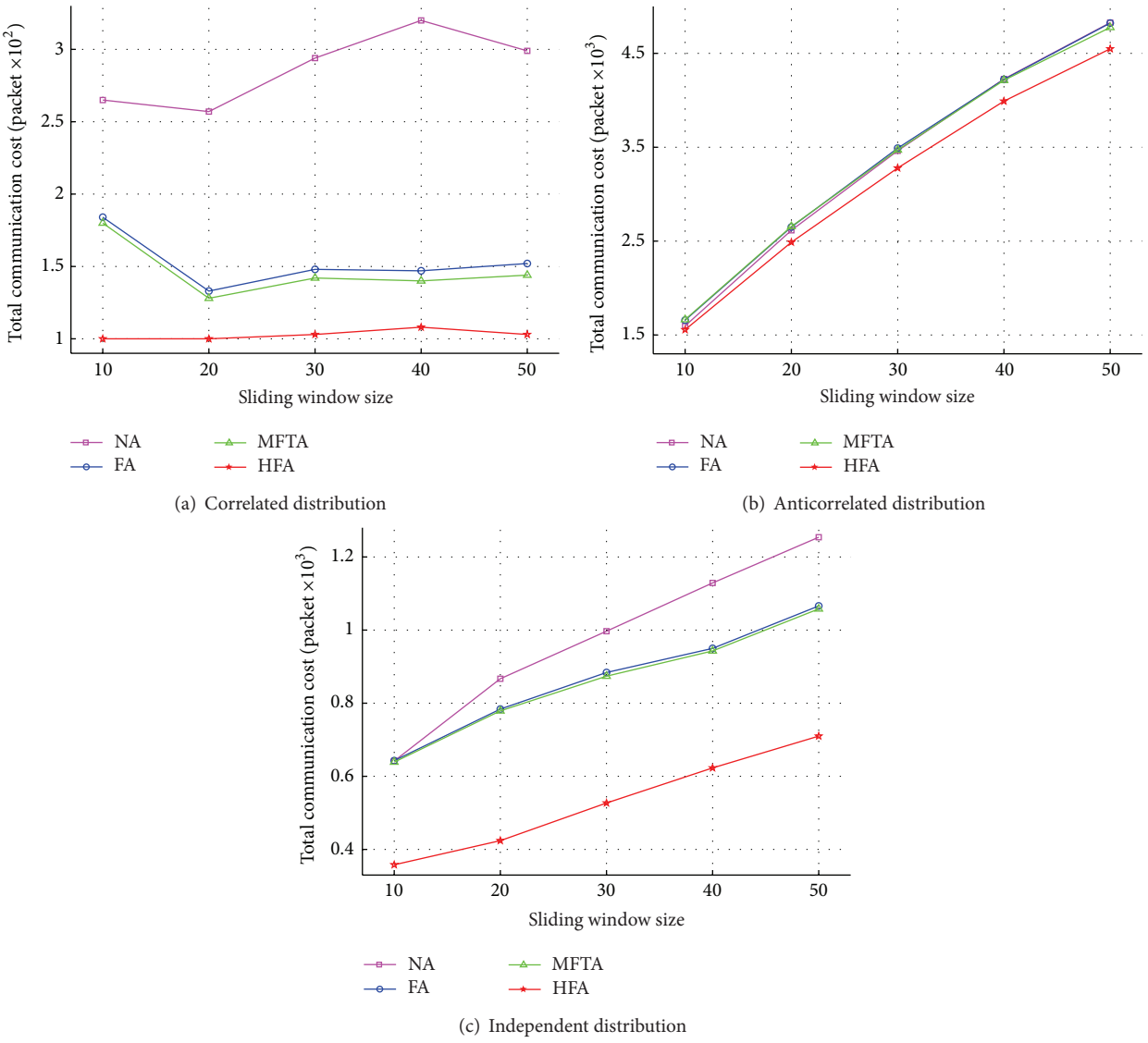


Figure 9: Effect of sliding window size of synthetic datasets.

the effect of our proposed algorithm with other methods. The experiments evaluate the performance of the algorithms by changing the dimension of sensor data, the size of sliding window, and the number of sensor nodes.

Before comparing performance of the algorithms, we study how different datasets affect the size of the skyline set. Figure 7 shows the skyline size under different datasets while changing the size of sliding window, the dimension of sensor data, and the number of sensor nodes. Figure 7 indicates that the skyline size is very large in anticorrelated distribution while the skyline size is very small in correlated distribution. As the dimension of data increases, it mostly influences the skyline size of anticorrelated distribution and has little influence on independent distribution and almost has no influence on correlated distribution. The reason is that the increase of dimension adds the probability of two tuples that are not dominated by each other in anticorrelated distribution data, which makes the size of the skyline set larger. As the sliding window size grows and the number of sensors in the network increases, the size of the skyline set grows. Because the larger the sliding window size and the number of sensors, the more the number of data in the network, resulting in the increase of the skyline set size.

Figure 8 shows performance of the algorithms on different datasets while changing the dimension of data. Figures 8(a), 8(b), and 8(c) represent the effect of correlated distribution, anticorrelated distribution, and independent distribution datasets, respectively. It can be observed that the HFA algorithm performs best in each dimension. As the HFA algorithm takes advantage of the historical data to learn the global knowledge of the data distribution, it chooses a filter with the best quality. Therefore, HFA algorithm performs best in the four algorithms. Meanwhile, the total sending packets of the network increase along with the increase of dimension of sensor data for all algorithms, thus, resulting in a higher query processing cost. The reason is that the increase

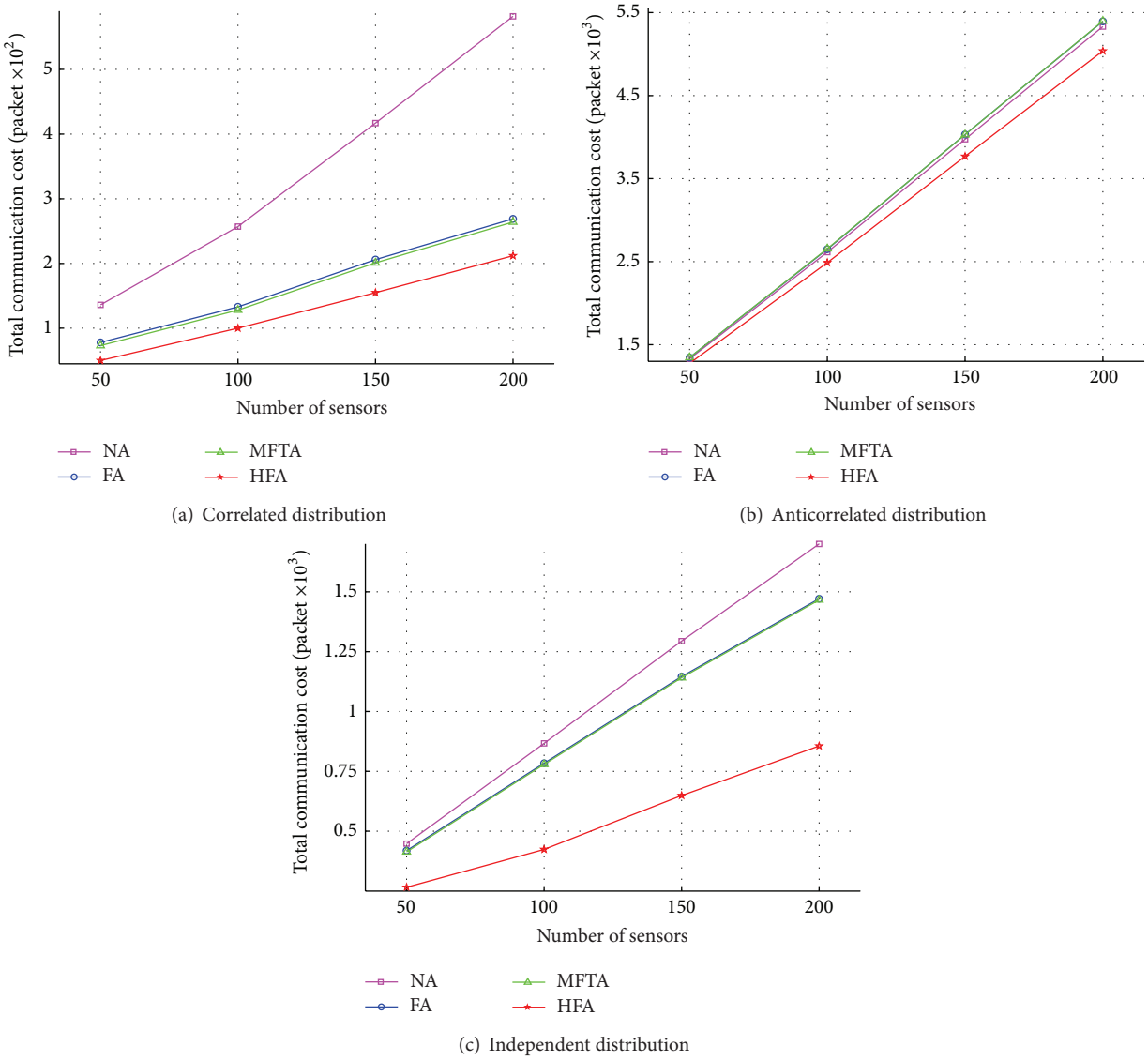


FIGURE 10: Effect of number of sensors of synthetic datasets.

of dimension adds the probability of two tuples that are not dominated by each other, which makes the size of the skyline set larger, as shown in Figure 7(b). Because the strategy of choosing filter tuple in FA algorithm is similar to MFTA algorithm, there is no significant difference of total number of sending packets between FA and MFTA. Both of them can prune some nonskyline tuples in correlated and independent distributions, but the filtering effect is not so good as anticorrelated distribution.

Figure 9 shows the effect of different datasets on the algorithms while changing the sliding window size. Figures 9(a), 9(b), and 9(c) represent the effect of correlated, anticorrelated, and independent distributed datasets, respectively. It shows that the HFA algorithm is more efficient than other algorithms. The reason is that the HFA algorithm considers the global characteristic of sensor data and chooses an optimal tuple from the network as the global filter. Figure 9 indicates that all the algorithms require the lowest communication cost when the data distribution is correlated. When the data distribution is anticorrelated, all the algorithms need a mass of energy to transmit data. The reason is that the size of the skyline set in the correlated distribution is very small as shown in Figure 7, which causes the algorithms to only need to transfer a small amount of data in the network. However, the size of the skyline set in the anticorrelated distribution is typically large, which leads to a large amount of sending packets.

Figure 10 shows the effect of different datasets on the algorithms while changing number of sensor nodes. Figures 10(a), 10(b), and 10(c) represent the effect of correlated distribution, anticorrelated distribution, and independent distribution datasets, respectively. It indicates that under different network sizes, the HFA algorithm sends less packets than the other methods, and the HFA algorithm performs

Table 3: Skyline size versus attribute combinations.

(a) Intel Lab data (dimension = 2)

Combinations of attributes	Skyline size
TV	5
TL	28
TH	68
VL	31
VH	37
LH	49

(b) Intel Lab data (dimension = 3)

Combinations of attributes	Skyline size
TVL	34
TVH	134
TLH	208
VLH	157

(c) Washington State Climate Data (dimension = 2)

Combinations of attributes	Skyline size
TS	3
TH	14
SH	6

best in the four methods. The total number of sending packets increases along with the increment of sensor nodes for all algorithms. The reason is that tuples of the whole network become large when the number of sensor nodes increases, so that the tuples belong to the skyline increase, as shown in Figure 7(c), which leads to the increment of communication cost.

5.2.2. Experimental Results of Real Datasets. We use Intel Lab Data and Washington State Climate Data for real datasets to conduct experiments and evaluate the effect of our proposed HFA algorithm. The experiments evaluate the performance of algorithms through changing the dimension of sensor data and the size of sliding window. We do not compare the influence of number of sensors on the algorithms because the number of sensor nodes in the network is fixed.

Table 3 shows how different combinations of attributes affect the size of the skyline set (the sliding window size is set to 20). For the Intel Lab Data, there are four attributes; suppose T represents temperature, V represents voltage, L represents light, and H represents humidity, respectively. For the Washington State Climate Data, there are three attributes, and T represents temperature, S represents wind-speed, and H represents humidity, respectively. As shown in Table 3, there are C_4^2 combinations when data dimension is set to 2 in the Intel Lab Data. Similarly, there are C_4^3 combinations when data dimension is 3 in the Intel Lab Data and C_3^2 combinations when data dimension is 2 in the Washington State Climate Data. It can be seen from the table that even though the length of dimension is fixed, different combinations of attributes make different influence on the skyline size. Take Table 3(a) for example, when we consider temperature and voltage, the skyline size is very small, while when we consider temperature and humidity, the skyline size becomes large. That is because different attribute combinations cause different final skyline sizes.

Table 4: Skyline size of different datasets.

(a) Changing data dimension

Data dimension	Skyline size (Intel)	Skyline size (Washington)
2	49	3
3	157	38
4	335	/

(b) Changing sliding window size

Sliding window size	Skyline size (Intel)	Skyline size (Washington)
10	18	2
20	34	3
30	61	3
40	79	3
50	89	3

Figure 11 shows the effect of the two real datasets on the algorithms while changing the combinations of attributes of data (sliding window size = 20). Figures 11(a), 11(b), and 11(c) represent 2 dimensions in Intel Lab data, 3 dimensions in Intel Lab data, and 2 dimensions in the Washington State Climate Data, respectively. It demonstrates that our proposed HFA algorithm performs best in the four algorithms. The total communication of cost, that is, the number of sending packets, increases or decreases along with the skyline size. When the skyline size is small, the communication cost is low, while when the skyline size is large, the communication cost is high. The reason is that when the skyline size of the network becomes large, there is a lot of data needed to be transmitted, thus, causing a large amount of data to be delivered in the network, which results in a high energy cost.

Figure 12 shows performance of the algorithms on real datasets while changing the dimension of data (sliding window size = 20). For the Intel Lab data, we consider the light and humidity attributes when data dimension is 2 and consider voltage, light, and humidity when data dimension is 3. For the Washington State Climate Data, we consider temperature and wind-speed attributes when data dimension is 2. Table 4(a) shows the skyline size when changing data dimension. Figures 12(a) and 12(b) represent the effect of the Intel Lab Data and The Washington State Climate Data, respectively. As the HFA algorithm chooses a best tuple as a global filter, it performs best in the four algorithms. Meanwhile, the total sending packets of the network increase along with the increase of dimension of sensor data for all algorithms, thus, resulting in a higher energy cost. The reason is that the increase of dimension adds the probability of two tuples which are not dominated by each other, which makes the size of final skyline set larger, as shown in Table 4(a).

Figure 13 shows the effect of the two real datasets on the algorithms while changing the sliding window size.

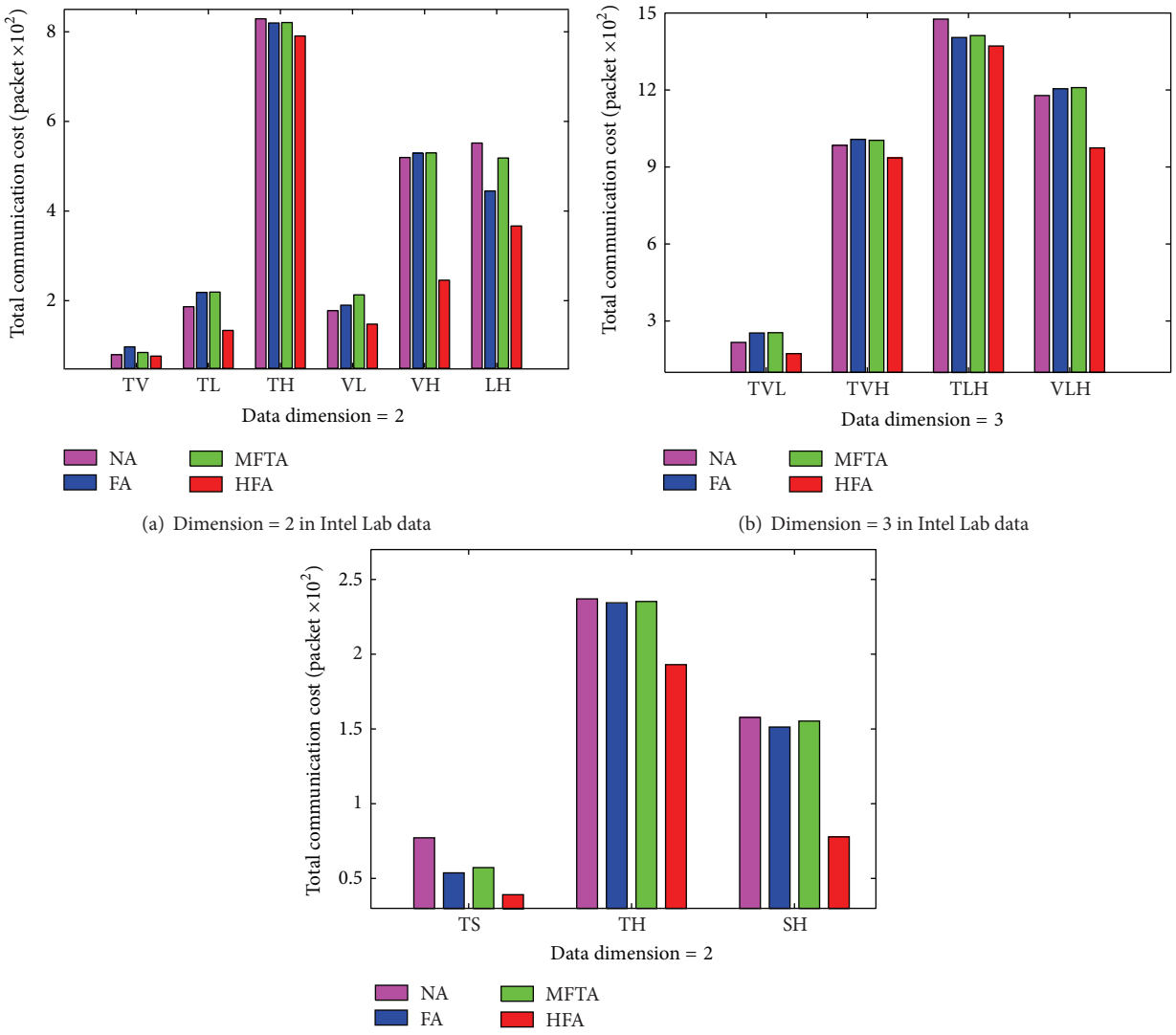
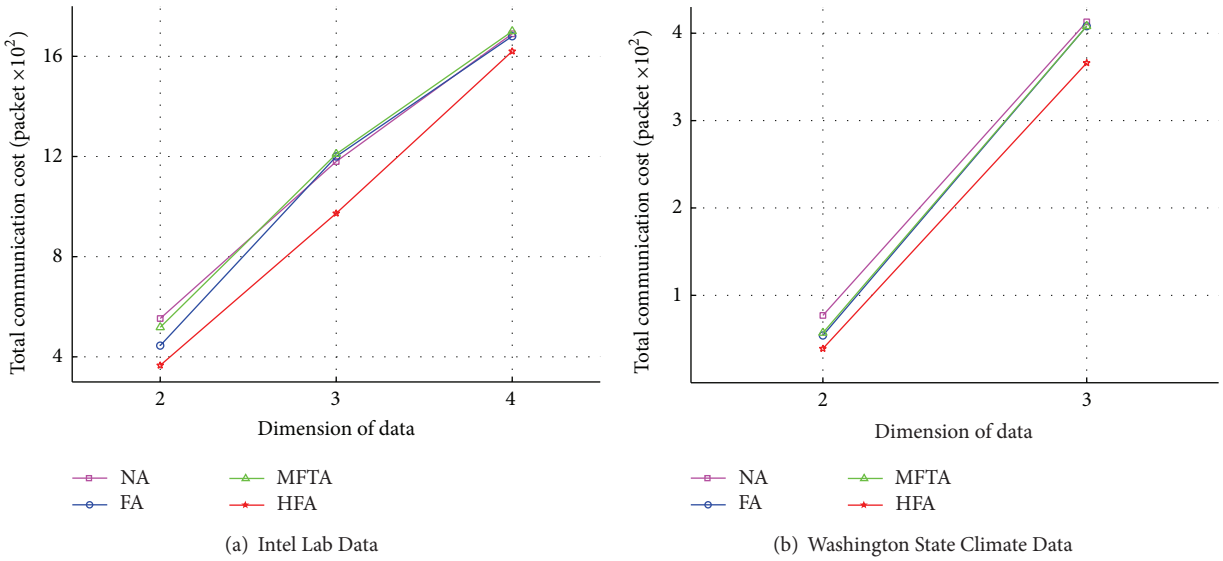


Figure 11: Effect of different combinations of attributes of real datasets.

For the Intel Lab data, we consider 3 dimensions of data: temperature, voltage, and light attributes. For the Washington State Climate Data, we consider 2 dimensions of data: temperature and wind-speed. Table 4(a) shows the skyline size when changing sliding window size. Figures 13(a) and 13(b) represent the effect of sliding window size of the Intel Lab Data and The Washington State Climate Data, respectively. It demonstrates that our proposed HFA algorithm performs well. It can be seen that when the sliding window size increases, the number of sending packets increases in the Intel Lab Data, while it has little influence on the Washington State Climate Data. That is because when the sliding window size increases, the Washington State Climate Data has a smaller skyline size as shown in Table 4(b), and the sliding window size almost has no influence on the skyline size. For the two datasets, when the sliding window size grows, the number of data in the network increases, and more data needs to be transmitted, which results in a little more delivering energy.

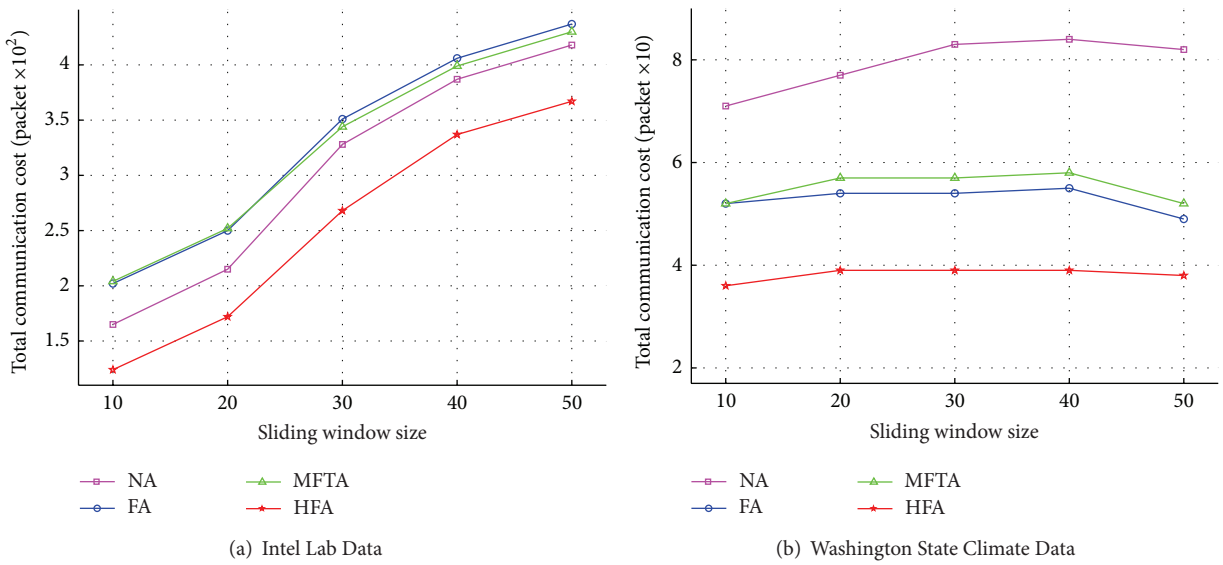
6. Conclusions

In this paper, we propose a histogram filter based algorithm HFA to process the sliding window skyline query problem in wireless sensor networks. HFA utilizes historical data to construct a multivariate histogram to estimate the probability density function of the sensor data. Then, an optimal filter tuple is chosen based on the histograms, which can prune nonskyline tuples to the maximum extent. The experimental results demonstrates that the HFA algorithm outperforms NA, FA, and MFTA algorithms and reduces the data transmissions effectively in wireless sensor networks which saves the energy and prolongs the lifetime of wireless sensor networks. Further interesting work is to extend our HFA



(a) Intel Lab Data (b) Washington State Climate Data

FIGURE 12: Effect of data dimension of real datasets.



(a) Intel Lab Data (b) Washington State Climate Data

FIGURE 13: Effect of sliding window size of real datasets.

algorithm to processing some variations of skyline queries, such as continuous skyline queries and k-dominant skyline queries.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by NUAA Research Funding of China under Grant no. NS2013089. The authors would also like to thank the reviewers for their helpful comments and advices to improve the presentation of the paper.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] 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*, vol. 30, no. 1, pp. 122–173, 2005.

[3] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.

[4] D. J. Abadi, S. Madden, and W. Lindner, "REED: robust, efficient filtering and event detection in sensor networks," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, pp. 769–780, September 2005.

[5] M. Stern, K. Böhm, and E. Buchmann, "Processing continuous join queries in sensor networks: a filtering approach," in *Proceedings of the International Conference on Management of Data (SIGMOD '10)*, pp. 267–278, June 2010.

[6] Y. Yao, X. Tang, and E.-P. Lim, "Continuous monitoring of knn queries in wireless sensor networks," in *Proceedings of the International Conference on Mo bile Ad-hoc and Sensor Networks*, pp. 663–674, 2006.

[7] M. Wu, J. Xu, X. Tang, and W. Lee, "Top-k monitoring in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 962–976, 2007.

[8] B. Malhotra, M. A. Nascimento, and I. Nikolaidis, "Exact top-K queries in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 10, pp. 1513–1525, 2011.

[9] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang, "A sampling-based approach to optimizing top-k queries in sensor networks," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*, pp. 68–78, April 2006.

[10] I.-F. Su, Y.-C. Chung, C. Lee, and Y.-Y. Lin, "Efficient skyline query processing in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 70, no. 6, pp. 680–698, 2010.

[11] B. Chen, W. Liang, and J. X. Yu, "Energy-efficient skyline query optimization in wireless sensor networks," *Wireless Networks*, vol. 18, no. 8, pp. 985–1004, 2012.

[12] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering (ICDE '01)*, pp. 421–430, April 2001.

[13] K.-L. Tan, P.-K. Eng, and C. O. Beng, "Efficient progressive skyline computation," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pp. 301–310, 2001.

[14] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proceedings of the 19th International Conference on Data Ingineering (ICDE '03)*, pp. 717–719, March 2003.

[15] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, pp. 229–240, Trondheim, Norway, September 2005.

[16] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: an online algorithm for skyline queries," in *Proceeding of the 28nd International Conference on Very Large Data Bases (VLDB '02)*, pp. 275–286, 2002.

[17] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 41–82, 2005.

[18] K. Hose and A. Vlachou, "A survey of skyline processing in highly distributed environments," *The VLDB Journal*, vol. 21, no. 3, pp. 359–384, 2012.

[19] S. Wang, Q. H. Vu, B. C. Ooi, A. K. H. Tung, and L. Xu, "Skyframe: a framework for skyline query processing in peer-to-peer systems," *The VLDB Journal*, vol. 18, no. 1, pp. 345–362, 2009.

[20] Y. Tao and D. Papadias, "Maintaining sliding window skylines on data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 377–391, 2006.

[21] W.-T. Balke, U. Gntzer, and J. X. Zheng, "Efficient distributed skylining for web information systems," in *Proceedings of the Advances in Database Technology (EDBT '04)*, pp. 256–273.

[22] K. Deng, X. Zhou, and H. T. Shen, "Multi-source skyline query processing in road networks," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 796–805, April 2007.

[23] Z. Huang, C. S. Jensen, H. Lu, and B. C. Ooi, "Skyline queries against mobile lightweight devices in MANETs," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*, p. 66, April 2006.

[24] Y. Kwon, J.-H. Choi, and C. Yon-Dohn, "In-network processing for skyline queries in sensor networks," in *Proceedings of the Institute of Electronics, Information and Communication Engineers (IEICE '07)*, pp. 3452–3459, 2007.

[25] H. Chen, S. Zhou, and J. Guan, "Towards energy-efficient skyline monitoring in wireless sensor networks," in *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN '07)*, pp. 101–116, 2007.

[26] J. Xin, G. Wang, and L. Chen, "Continuously maintaining sliding window skylines in a sensor network," in *Proceedings of the 12th International Conference on Database Systems for Advanced Applications(DASFAA '07)*, pp. 509–521, 2007.

[27] Intel berkeley research lab, http://www.select.cs.cmu.edu/data/labapp3/.

[28] "Earth climate and weather," University of Washington, http://www-k12.atmos.washington.edu/k12/grayskies/.