Outlier Detection in Sensor Networks

Bo Sheng, Qun Li, Weizhen Mao
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795, USA
{shengbo, liqun, wm}@cs.wm.edu

Wen Jin
Department of Computer Science
Simon Fraser University
Burnaby, Canada
wjin@cs.sfu.ca

ABSTRACT

Outlier detection has many important applications in sensor networks, e.g., abnormal event detection, animal behavior change, etc. It is a difficult problem since global information about data distributions must be known to identify outliers. In this paper, we use a histogram-based method for outlier detection to reduce communication cost. Rather than collecting all the data in one location for centralized processing, we propose collecting hints (in the form of a histogram) about the data distribution, and using the hints to filter out unnecessary data and identify potential outliers. We show that this method can be used for detecting outliers in terms of two different definitions. Our simulation results show that the histogram method can dramatically reduce the communication cost.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; C.2.4 [**Computer-Communications Networks**]: Distributed Systems

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Data Mining, Histogram, Outlier Detection, Wireless Sensor Networks

1. INTRODUCTION

Sensor networks will be deployed in buildings, cars, and the environment, for monitoring health, traffic, machine status, weather, pollution, surveillance, and so on. It is likely that they will be in use for a long time, generating a large amount of data. Mining this large data repository for useful information will be crucial.

In a simple solution, data collected by sensors can be transmitted to the sink for data mining analysis. This method, however, consumes too much energy because the data volume can be extremely large. The batteries of the sensors will be depleted very quickly leading to network partition and dysfunction. A good method should require little data transmission, but still achieve information extraction from the large amount of data distributed over the network.

In this paper, we consider a very important data mining problem: outlier detection, which defines a process to identify data points that are very different from the rest of the data based on a certain measure. Outlier detection in a central database has been a hot topic in the data mining community, but little work has been done in the context of a sensor network in which data are distributed over thousands or tens of thousands of sensors. Outlier detection in sensor network is needed in many applications that monitor abnormal behaviors, measurements, and events. For example, a sensor network, embedded in a highway bridge around beams or columns for detailed building structural monitoring, can give early warning of any structural weakness or deterioration, reducing the chance of unexpected failures. Outlier detection helps pinpoint the accurate locations of the weakening parts, especially in the early stage of the problem development. Chemical sensors deployed in the environment to monitor toxic spills and nuclear incidents gather the chemical data periodically. Outlier detection can trigger the alarm and locate the source when abnormal data are generated. Habitat monitoring for endangered species is another application in which animals will be attached with small non-intrusive sensors. Outlier detection indicating abnormal behaviors suggests closer observation of an individual animal and maybe more human interactions.

The outlier detection problem addressed in this paper is different from the local outlier detection used in many event detection applications, e.g., vehicle tracking, surveillance monitoring, and so on. Local outlier detection concerns about the abnormal sensor readings in local proximity, which are easy to locate by aggregating data collected by nearby sensors. In this paper, we mainly consider that data are governed by parameters other than locality. More precisely, we aim to find global outliers over the data collected by all sensors, which demands global information about the whole network. For example, in order to determine if a data point on a sensor is an outlier, all sensors that may be distributed remotely but have similar readings must be examined. This requirement is extremely harsh for sensor networks since information aggregation over the entire network is costly due to the network-wide transmission. The critical information in this process is about the data distribution in the network.

MobiHoc'07, September 9–14, 2007, Montréal, Québec, Canada.

In this paper, instead of accumulating all data to the sink for analysis, we extract hints from the network with less information transmitted. We use a histogram to abstract the data distribution. Relevant methods appear in deviant detection for time series database in [18] and [12]. In order to obtain more precise information about the data distribution, we refine the histogram by using a smaller bucket width. We mathematically evaluate the benefit of reducing the bucket width or dredging up the raw data from the network. We consider two definitions of an outlier, which are both based on the distance between a data point and its k-th nearest neighbor. One is defined by a fixed threshold distance [13], and the other is based on the rank among all data points [20].

We make the following contributions in this paper. (1) We study outlier detection in the context of sensor networks. To the best of our knowledge, we propose the first histogram-based detection approach to identify distance-based outliers over sensor networks. (2) We use the histogram-based method to approximate the sensor data distribution and reduce the communication cost under two different detection schemes. We give theoretical analysis for the communication cost incurred in the network. (3) We use a histogram refinement technique for some critical portion of data distribution to gain more information about outliers, and a probabilistic method to estimate the benefit of reducing the bucket width, thus further reducing the communication cost. (4) Our simulation results based on real data collected by Intel demonstrate that our approaches for outlier detection reduce the communication cost dramatically compared with the centralized scheme.

Our schemes can also be extended to support online outlier detection, because histogram information can be accumulated along time. After obtaining an initial histogram, the sink can update the histogram periodically, while each sensor only reports the necessary histogram changes due to the newly generated data. In addition, we believe that the techniques used in this paper will benefit many other data mining problems in sensor networks, such as data clustering and object classification.

2. RELATED WORK

Outlier detection has been well studied in the database research community ([20, 13, 5, 15, 4, 7, 14, 3]) and there are several different definitions of outliers in the literature. This paper considers two popular distance-based definitions proposed in [13] and [20], where outliers are identified by examining data points' nearest neighbors. One major research goal of this problem in the database community is to efficiently detect outliers in a large-scale database ([13, 20, 14] and [5]). In sensor networks, however, data are generated by scattered nodes and transferred via wireless channels. The proposed approaches in the previous work can not be directly applied unless all data are gathered at one node, which is very costly in transmission. Another hot spot for database researchers is high dimensional outlier detection ([3, 4, 15] and [2]). This issue is not covered in this paper, because sensor networks usually only generate low dimensional data, and different attributes, such as temperature and sound, may not be correlated to define outliers, thus can be considered separately as one-dimensional data.

Sensor networks are often treated as databases and SQL queries are common ways to collect data ([17] and [9]). A lot of research work ([16, 11, 8, 23, 22, 21, 10]) has been proposed to handle different types of queries efficiently. However, the distance-based outlier detection has been seldom discussed in this area. As close work, T. Palpanas et. al. [19] study a model-based outlier detection in sensor networks. Normal behaviors are first characterized by predictive models and outliers are detected as the deviations. In [6], J. Branch et. al. propose an in-network scheme for detecting outliers based on data exchanges among neighbors. However, their goal is to reveal outliers to every sensor and the cost is very expensive for common parameter settings in the database literature. In a recent paper [24], S. Subramaniam et. al. present an online outlier detection scheme for sensor networks. Every sensor keeps a sliding window of the historic data and estimates the data distribution to detect the outliers. This method, however, consumes a lot of memory space and may not find all outliers.

In the first part of this paper, queries of detecting outliers have a similar form as general range queries. Previous work in [16] and [11] has provided typical models, in which users can easily specify a range query and obtain the result in an efficient and reliable way. On the other hand, top-k query ([22] and [25]) and order statistical query ([21] and [10]) are similar to finding the rank-based outliers, which is discussed in the latter part of this paper. However, none of these approaches is applicable to our problem, because the range parameter and the order of data in our problem do not depend on the data value, but the distance to neighboring data points. In addition, our problem requires outliers to be exactly returned without approximation.

3. PROBLEM FORMULATION

An *outlier* represents a data point that is very different from the others. It has been defined in different ways in the database literature. In this paper, we consider two popular definitions based on *distance*, which is defined as the the absolute difference between two data points. For each data point p, we can sort all the rest of the data points according to their distances to p in an ascending order. Suppose the sorted list is $p_1, p_2, \cdots, p_k, \cdots$. We have $|p_1 - p| \leq |p_2 - p| \leq \cdots \leq |p_k - p| \leq \cdots$. Let $D^k(p) = |p_k - p|$ represent the distance between data point p and its k-th nearest neighbor (KNN) (p_k). We can define two types of outliers as follows:

DEFINITION 1. *A data point p is called an $O(d, k)$* outlier *if $D^k(p) \geq d$.*

DEFINITION 2. *A data point p is called an $O(n, k)$* outlier *if there are no more than $n-1$ other data points q, such that $D^k(q) > D^k(p)$.*

We are particularly interested in outlier detection in a sensor network composed of N sensors. The sensor network monitors the environment or any object and periodically generates data. Among all the data generated by the sensors, we would like to find all the outliers. We assume that a routing tree rooted at the sink has been constructed by using some simple routing algorithm, in which each sensor is linked to an up-stream and a down-stream node. An outlier detection algorithm will be built on this underlying communication tree. Depending on the choices of outliers, we aim to design algorithms to respond to a query for outliers with parameters d and k for Definition 1 or n and k for Definition 2.

It is possible that the tree topology might be broken or updated due to the wireless link failures. This paper considers a common practice that, when building the tree, only stable links are selected such that errors in transmission due to poor link quality can be reduced and the tree topology can be robust for a long time. We also assume that the communication cost of sending a packet between two nodes with a direct link is proportional to the packet size. For easy exposition, we make an assumption that each data point is represented as an integer with precision to the last digit. It is easy to transform any real data to this format, e.g., 12.34 will be converted to 1234 for precision 0.01. Our algorithms focus on the data points and return all the outliers, although many applications may require the algorithm to return the sensor ID or location of the outliers. An easy solution is, after running our algorithm and obtaining all the outliers, to let the sink diffuse the outlier data points to the sensor network so that the sensors holding the outlier data points will reply with their IDs or locations.

As mentioned earlier, in a naive solution, we may collect data from all sensors to the sink for analysis. We call this method the *centralized scheme*, which is not feasible in practice because transferring a large amount of data is very costly in sensor networks. In Sections 5 and 6, we will introduce energy-efficient algorithms to identify outliers in sensor networks. In Section 7, we will compare our solutions with the centralized scheme and show the difference of the performance.

4. HISTOGRAM QUERY

In this section, we introduce the motivation of using the histogram information in finding outliers. We observe that both definitions are based on the value of $D^k(p)$, the distance between p and its KNN. In the following, we show that the histogram provides useful information to estimate $D^k(p)$ for each data point p, which helps identify outliers.

In this paper, we use the equi-width histogram, because it is easy to aggregate equi-width histogram in sensor networks. We assume that the value range for all data points is uniformly divided into buckets with width w, and each bucket is assigned an ID, consecutively from 1 to the number of buckets. We define bucket i by a value range $[min_i, max_i)$, thus $w = max_i - min_i$ and $min_i = max_{i-1}$. After collecting the histogram, the sink will know the total number of data points in each bucket i, indicated by f_i. For any data point p in bucket i, we can estimate the bounds on $D^k(p)$ based on the histogram information. The following theorems aim to find a pair of values l_i and u_i for any bucket i, such that $\forall p$ in bucket $i, D^k(p) \in (l_i, u_i]$.

THEOREM 1. *If $f_i > k$, then $l_i = 0$ and $u_i = w - 1$ are lower and upper bounds for $D^k(p)$, where p is any data point in bucket i.*

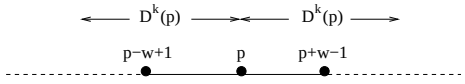

Figure 1: Bounds on $D^k(p)$ in Theorem 1

PROOF. We prove it by contradiction. Referring to Fig. 1, assume there exists a data point p in bucket i, such that $D^k(p) > w - 1$. Let $Q = \{x | x \in (p - D^k(p), p + D^k(p))\}$. On one hand, according to the definition of $D^k(p)$, $|Q| \leq k$. On the other hand,

$$\begin{aligned} & D^k(p) > w - 1 \\ \Rightarrow \quad & (p - D^k(p), p + D^k(p)) \supseteq [p - w + 1, p + w - 1], \end{aligned}$$

which means Q must include all data points in bucket i. Thus, $|Q| \geq f_i > k$. There is a contradiction of $|Q|$. □

THEOREM 2. *We define a function*

$$F(t, i) = \sum_{j=i-t}^{i+t} f_j.$$

If $f_i \leq k$, we can find an integer $s \geq 0$, such that $F(s, i) \leq k$ and $F(s+1, i) > k$. Then, $l_i = s \cdot w$ and $u_i = (s+2) \cdot w - 1$, are the lower and upper bounds for $D^k(p)$, where p is any data point in bucket i.

The proof is similar to Theorem 1 and omitted due to the page limit. As shown above, the histogram information helps us derive lower and upper bounds on $D^k(p)$ for any data point p. We will utilize these theorems in our outlier detection schemes.

5. OUTLIER DETECTION FOR $O(d, k)$

In this section, we propose a histogram-based protocol for detecting outliers defined by Definition 1. Our approach includes two stages. In the first stage, we divide the data value range into uniform buckets and collect equi-width histogram information. The histogram provides us with useful information to identify some data points as outliers or non-outliers. However, the histogram information may not be sufficient to identify every data point. We call those data points *potential outliers* if they cannot be identified by the histogram. In the second stage, the sink gathers the potential outliers from the sensor network and checks the distance to the KNN for each of them. Eventually, the sink will identify all outliers.

In the following, we introduce a basic scheme which uses a single-round histogram information collection and an enhanced scheme which refines the histogram through multiple-round histogram collections.

5.1 Basic Scheme

In this section, we present a basic scheme for $O(d, k)$ outlier detection with a single-round of histogram collection.

5.1.1 Obtain v_{min} and v_{max}

In the first step, the sink queries for the minimum and maximum data values in the sensor network in order to calculate the value range. Let v_{min} and v_{max} be the minimum and maximum values received by the sink. In this step, every sensor sends at most $\log(v_{min} \cdot v_{max})$ bits of information.

5.1.2 Collect Histogram

In the second step, the sink collects the histogram from the sensor network. To obtain the histogram, every sensor and the sink have to agree on the same bucket partition, which can be specified by the bucket width w and the value range $[v_{min}, v_{max} + 1)$. For an easy exposition, we fix the bucket width w to d. We will explain why we set this width rather than other values in the next sub-section. In this step, the sink diffuses a query including d, v_{min} and v_{max} as well

as the other parameter k to the sensor network. Every non-leaf node sends $\log(k \cdot d \cdot v_{max} \cdot v_{min})$ bits of information to forward the query down to its children. Let l be the width of $[v_{min}, v_{max}+1)$ defined as $l = v_{max} - v_{min} + 1$. Sensors divide the value range $[v_{min}, v_{max}+1)$ into $\lceil \frac{l}{d} \rceil$ uniform buckets with width d after they receive the histogram query, i.e., the ith bucket is defined by $[v_{min}+(i-1)\cdot d, v_{min}+i\cdot d)$. Starting from the leaf nodes, each sensor aggregates the number of data points in each bucket from all its descendants. Let g_i^j be the number of data points generated by sensor j in bucket i, and f_i^j be the histogram summary of bucket i sent by sensor j. For a leaf node j, $f_i^j = g_i^j$. For a non-leaf node j, assume it receives summaries from its children $c_1, c_2, \cdots$. All these summaries are aggregated as well as its own summary,

$$f_i^j \Leftarrow g_i^j + f_i^{c_1} + f_i^{c_2} + \cdots .$$

Finally, the aggregated number of data points in each bucket is forwarded to j's parent. In this step, we do not have to maintain the exact histogram as long as we can apply Theorems 1 and 2 later. Therefore, if $f_i^j > k+1$, we will reset it as $k+1$ in order to reduce the communication cost. In this way, every node transfers at most $\lceil \frac{l}{d} \rceil \cdot \log(k+1)$ bits of data back and the sink finally obtains the value of f_i.

5.1.3 Collect Outliers and Potential Outliers

In the third step, the sink applies Theorem 1 and Theorem 2 on every bucket i based on f_i to assign l_i and u_i. If u_i is set by Theorem 1, we will get $u_i = w-1 < d$. On the other hand, if l_i is set by Theorem 2 and greater than 0, we must have $l_i = sw \geq d$. Based on the definition of $O(d,k)$, the sink analyzes the received histogram information as follows:

- Case 1: If $u_i < d$, all data points in bucket i are non-outliers and bucket i is called a *non-outlier bucket*;
- Case 2: If $l_i \geq d$, all data points in bucket i are outliers and bucket i is called an *outlier bucket*;
- Case 3: Otherwise, bucket i is called a *potential outlier bucket* and the data points in it are called *potential outlier*s.

As we mentioned earlier, w can be set to other values. If $w > d$, however, Theorem 1 will not help us identify non-outlier buckets, because there is no derivation from $u_i = w-1$ to $u_i < d$. If $w < d$, on the other hand, we will obtain more detailed information and identify more outlier buckets. However, as we will analyze later, smaller bucket incurs more communication cost and we will probably miss some non-outlier buckets. Therefore, without any prior knowledge of the data, d is a conservative value for w to achieve good performance.

The non-outliers identified in case 1 can be ignored, because they should not be returned in the result. For case 2, the sink can send another query to indicate the outlier buckets and collect all the identified outliers from every sensor. To process the potential outliers in case 3, we use a simple method to first obtain all potential outliers and then, in the next step (explained in Section 5.1.4), find the distance to their individual KNNs to determine whether each potential outlier is actually an outlier or not.

Considering all three cases above, the sink diffuses a query, which includes a vector of length $\lceil \frac{l}{d} \rceil$, $\{q_1 q_2 \ldots q_{\lceil \frac{l}{d} \rceil}\}$, where q_i is a one-bit flag satisfying

$$q_i = \begin{cases} 0 & \text{if bucket } i \text{ is a non-outlier bucket;} \\ 1 & \text{otherwise.} \end{cases}$$

In another word, $q_i = 1$ indicates that all data points in bucket i need to be returned, because they are either outliers or potential outliers. After receiving this query, every sensor will look up its own data set and return all data points in the marked buckets along the routing tree. The query diffusion cost for each non-leaf sensor is $\lceil \frac{l}{d} \rceil$ bits. However, the cost of collecting potential data depends on the histogram obtained in the previous steps. Suppose N_o is the number of the identified outliers and N_{po} is the number of the potential outliers. Thus, the number of data points we need collect in this step is $N_o + N_{po}$. We assume the communication cost is proportional to the data size and the distance between the sender and receiver. Therefore, the cost of collecting data in this step is estimated as

$$(N_o + N_{po}) \cdot \log v_{max} \cdot avgDist,$$

where $avgDist$ is the average hop distance between the sink and sensors.

5.1.4 Diffuse Potential Outliers and Count the Number of Neighbors within d

In the last step, the sink first combs through the collected data points in the potential outlier buckets. Some data points may be identified immediately as outliers or non-outliers because the collected data points may give such information. For example, data points in a potential outlier bucket will be identified if the data points in the two neighboring buckets are also collected. Unfortunately, there are still some data points that could not be identified as either outliers or non-outliers. The sink sends those remaining potential outliers to the sensor network in order to find the actual outliers among them. The query is formed as $\{p_1, p_2, \ldots, \}$, which is the list of all the remaining potential outliers. Every sensor will forward the query to its children until the query reaches the leaf nodes. To answer such a range query, starting from the leaf nodes, every sensor sends a vector of summaries, one for each data point, to its parent, $\{f_1', f_2', \ldots, \}$, where f_i' is the number of the data points within $[p_i - d, p_i + d]$, i.e., the number of p_i's neighbors within distance d. A non-leaf sensor will sum up the summaries from its children as well as the summary of its own data set and forward the aggregated summaries to its parent. Similar to the second step, if $f_i' > k+1$, we will reset it to $k+1$.

This step can be optimized by filtering out some unnecessary diffusion at each node based on the histogram obtained previously. For example, consider a potential outlier p, which belongs to bucket $i = \lceil \frac{p - v_{min} + 1}{d} \rceil$. If a sensor j finds $f_{i-1}^j + f_{i+1}^j = 0$, according to the previous histogram, there is no need to diffuse p_i to its children, because all possible neighbors of p_i in the subtree rooted at this sensor are in bucket j, so we can immediately set the summary for p to f_i^j.

Eventually, the sink receives a value for each potential outlier, which represents the number of data points within distance d of p_i. We may simply scan the summary list and determine whether the ith potential outlier is actually an outlier by examining if $f_i' \leq k$.

In this step, we first diffuse all the collected potential outliers to every sensor. The diffusion cost is at most $N_{nl} \cdot N_{po} \cdot \log v_{max}$, where N_{nl} is the number of non-leaf nodes. And the cost of transferring the return summaries is bounded by $N \cdot N_{po} \cdot \log(k+1)$. To summarize, the total cost of the basic scheme, denoted by C_{basic}, is estimated as

$$\begin{aligned} C_{basic} &= N \cdot \log(v_{min} \cdot v_{max}) \\ &+ N_{nl} \cdot \log(k \cdot d \cdot v_{min} \cdot v_{max}) + N \cdot \lceil \frac{l}{d} \rceil \cdot \log(k+1) \\ &+ N_{nl} \cdot \lceil \frac{l}{d} \rceil + (N_o + N_{po}) \cdot \log v_{max} \cdot avgDist \\ &+ N_{nl} \cdot N_{po} \cdot \log v_{max} + N \cdot N_{po} \cdot \log(k+1). \end{aligned} \quad (1)$$

5.2 Enhanced Scheme

A drawback of the basic scheme is that if there are many potential outliers, i.e., N_{po} in Eq.(1) is very large, collecting and diffusing them will incur a large communication cost. In this section, we propose an enhanced scheme that refines some of the histogram before we query for the potential outliers. We hope that more rounds of histogram queries can help us prune out more data points, i.e., the number of potential outliers can be further reduced. In the following, our enhanced scheme only considers at most one extra round of histogram collection. The algorithm and analysis, however, can be used for more rounds of histogram collection in the same manner.

The first two steps of the enhanced scheme are quite similar to the basic scheme. After receiving the histogram in step 2, however, the sink has two options. First, the sink can follow step 3 in the basic scheme, collecting the potential outliers. In the other option, the sink can send a query for another histogram with a new bucket width $w = d' < d$ and then continue step 3 in the basic scheme. One more histogram with a smaller bucket width leads to more accurate information that helps to reduce ambiguous potential outliers. However, collecting more detailed histogram incurs extra communication cost. Thus, we need to determine if refining histogram is worthwhile and if so what is the appropriate bucket width for the new query. According to Eq.(1), if we do not query for more histogram, the estimated cost of the remaining steps, denoted by $Cost'$, is

$$\begin{aligned} Cost' &= N_{nl} \cdot \lceil \frac{l}{d} \rceil + (N_o + N_{po}) \cdot \log v_{max} \cdot avgDist \\ &+ N_{nl} \cdot N_{po} \cdot \log v_{max} + N \cdot N_{po} \cdot \log(k+1). \end{aligned} \quad (2)$$

In the following, we analyze the cost of collecting more histogram and propose an algorithm to determine the optimal value of the new width d'. The minimum cost will be compared with $Cost'$ to determine which option is better.

In this enhanced scheme, we keep the first two steps of the basic scheme with the following changes:

- In the first step, besides v_{min} and v_{max}, the sink also queries for the total hop distance to the sink and the number of non-leaf nodes. These two values can be aggregated at each node and we use $tolDist$ and N_{nl} to represent the results received by the sink respectively.
- In the second step of collecting the histogram, the upper limit of data points count f_i^j is set to $k \cdot d$, instead of $k+1$. We will explain it in the analysis later.

After step 2, we set $avgDist = \frac{tolDist}{N}$ and estimate $Cost'$ as defined in Eq.(2).

Next, we estimate the cost after step 2 if we send one more histogram query. Essentially, the extra round of histogram query only targets at potential outlier buckets as well as their related neighboring buckets. Assume the new bucket width is set to $d' = \frac{d}{B}$ for the next round of histogram query. To reuse the previously collected histogram information, we choose B to be an integer. The query sent by the sink includes B and a vector of bits which mark the potential outlier buckets identified by Theorems 1 and 2, $\{B, q_1 q_2 \ldots q_{\lceil \frac{l}{d} \rceil}\}$, where $q_i = 1$ if bucket i is a potential outlier bucket. Thus, the cost of query diffusion is $N_{nl} \cdot (\log B + \lceil \frac{l}{d} \rceil)$. After receiving the query message, each sensor will know the new bucket width d' and the potential outlier buckets. The sensors divide each of potential outlier buckets and their neighboring buckets into B uniform sub-buckets. Similarly, every sensor generates a histogram for the new sub-buckets. This information is aggregated bottom-up along the routing tree and finally reaches the sink. One optimization that each sensor can apply is to transfer the first $B-1$ summaries for each target bucket i instead of B summaries, because its parent already knows the total number of data points in bucket i, the last summary can be derived from the available information. The number of buckets involved in the reply can be easily counted as follows:

for $i = 1$ to $\lceil \frac{l}{d} \rceil$ **do**
  if $q_{i-1} + q_i + q_{i+1} > 0$ **then** $Count \leftarrow Count + 1$.

In the return stage, each sensor transfers at most $B \cdot Count \cdot \log(k+1)$ bits of information. Therefore, the total cost of querying and collecting the refined histogram is

$$N_{nl} \cdot (\log B + \lceil \frac{l}{d} \rceil) + N \cdot B \cdot Count \cdot \log(k+1). \quad (3)$$

Assume after obtaining more histogram information, we identify EN_o outliers and EN_{po} potential outliers. Following step 3 in the basic scheme, we need collect outlier data and further check potential outliers. Similar to Eq.(2), the estimated cost of the remaining steps is,

$$\begin{aligned} & N_{nl} \cdot \lceil \frac{l}{d} \rceil + (EN_o + EN_{po}) \cdot \log v_{max} \cdot avgDist \\ + \quad & N_{nl} \cdot EN_{po} \cdot \log v_{max} + N \cdot EN_{po} \cdot \log(k+1). \end{aligned} \quad (4)$$

Therefore, the total cost incurred by refining histogram and its consequence after step 2 is estimated as

$$Cost(d') = \text{Eq.(3)} + \text{Eq.(4)}.$$

In this problem, we aim to find the optimal d' such that $Cost(d')$ is minimized and compare it with $Cost'$ to decide if refining histogram is worthwhile.

In order to calculate $Cost(d')$, we first estimate EN_o and EN_{no} in Eq.(4) based on the histogram information collected in step 2. We assume that data are randomly distributed in each bucket. Let us take a close look at a potential outlier bucket i. After the refined histogram query, we will get B summaries for bucket i as well as bucket $i-1$ and $i+1$. Each new summary is responsible for a sub-bucket of the original buckets. Let us label the jth sub-bucket of the original bucket i as bucket $b_{(i-1)B+j}$. Let $f'_{j'}$ be the number of data points in sub-bucket $b_{j'}$. In the following, we estimate the probabilities that a sub-bucket j' is a non-outlier bucket or outlier bucket, indicated by $P_{no}(j')$ and $P_o(j')$ respectively. The probability of being a potential outlier

bucket is $1 - P_{no}(j') - P_o(j')$. Thus, EN_o and EN_{po} can be derived as

$$EN_o = \sum_{b_{j'}} f'_{j'} P_o(j'),$$

$$EN_{po} = \sum_{b_{j'}} f'_{j'} (1 - P_{no}(j') - P_o(j')).$$

To ensure $b_{(i-1)B+j} (1 \le j \le B)$ is a non-outlier bucket, we must have

$$\sum_{q=(i-1)B+j-B+1}^{(i-1)B+j+B-1} f'_q > k.$$

As illustrated in Fig. 2, the left side can be derived as

$$\sum_{q=(i-2)B+j+1}^{iB+j-1} f'_q = f_i + \sum_{q=(i-2)B+j+1}^{(i-1)B} f'_q + \sum_{q=iB+1}^{iB+j-1} f'_q.$$

Thus, a sub-bucket $b_{(i-1)B+j}$ is a non-outlier bucket if

$$\sum_{q=(i-2)B+j+1}^{(i-1)B} f'_q + \sum_{q=iB+1}^{iB+j-1} f'_q > k - f_i.$$

Let a be the first term (the number of the data points in the rightmost $B-j$ sub-buckets of bucket $i-1$) and b be the second term (the number of the data points in the leftmost $j-1$ sub-buckets of bucket $i+1$). Thus, $P_{no}((i-1)B+j)$ is the probability that $a + b > k - f_i$. We define a function $P(x, y, z)$ to be the probability that the number of data points in the leftmost or rightmost y sub-buckets of bucket x is z. Based on the assumption of random data distribution in every bucket,

$$P(x, y, z) = \begin{pmatrix} f_x \\ z \end{pmatrix} (\frac{y}{B})^z (\frac{B-y}{B})^{f_x - z}.$$

Thus, $P_{no}((i-1)B+j)$ can be calculated as

$$\sum_{a=0}^{f_{i-1}} (P(i-1, B-j, a) \cdot \sum_{b=k-f_i-a+1}^{f_{i+1}} P(i+1, j-1, b)).$$

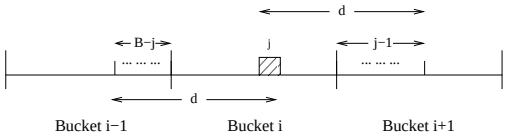

Figure 2: Data involved in identifying a non-outlier sub-bucket

On the other hand, we can claim that $b_{(i-1)B+j}$ is an outlier bucket if the following condition is satisfied:

$$\sum_{q=(i-1)B+j-B}^{(i-1)B+j+B} f'_q \le k.$$

By similar analysis as above, it becomes

$$\sum_{q=(i-2)B+j}^{(i-1)B} f'_q + \sum_{q=iB+1}^{iB+j} f'_q \le k - f_i,$$

as shown in Fig. 3. Let a be the first term and b be the second term. Then, $P_o((i-1)B+j)$ can be calculated as

$$\sum_{a=0}^{f_{i-1}} (P(i-1, B-j+1, a) \cdot \sum_{b=0}^{k-f_i-a} P(i+1, j, b)).$$

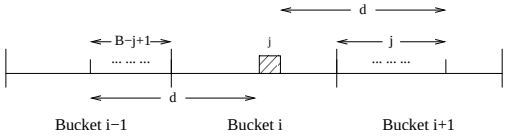

Figure 3: Data involved in identifying an outlier sub-bucket

Besides the above analysis, we use a short-cut estimation if a potential bucket i's neighboring buckets reach the histogram limit $k \cdot d$. Assume $f_{i+1} = k \cdot d$, expectedly, the frequency of each value in bucket $i+1$ is k. Thus, every data point in bucket i has a high probability to be a non-outlier. For such a bucket i, we skip the probabilistic analysis and directly increase EN_o by f_i.

Algorithm 1 determines the optimal bucket width d' for the second round of histogram query. Initially, we scan every bucket and use an array M to mark the potential outlier buckets,

$$M[i] = \begin{cases} 1 & \text{if bucket } i \text{ is a potential outlier bucket;} \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm is constructed by two embedded loops. In the outer loop (lines 7–22), we try different bucket width $d' = \frac{d}{B}$ by testing all possible B. For each B, we estimate the cost incurred by this round of refined query and the subsequent steps for raw data collection. The optimal width yields the minimum value of $Cost$, which is tracked by the variables $optB$ and min in Algorithm 1. If the final value of min is no less than $Cost'$, there is no need to conduct an extra round of histogram query. Otherwise, we set $d' = \frac{d}{optB}$ and do the refined histogram query.

The inner loop of this algorithm (lines 9–17) checks every bucket to estimate the cost. There will be B subbuckets for each requested bucket and reporting a histogram of them needs $B \log(k+1)$ bits of data. A bucket i will be involved only if it is a potential outlier bucket or one of its neighboring buckets is a potential outlier bucket. Additionally, EN_o and EN_{po} are accumulated in the inner loop when checking potential outlier buckets.

The implementations of EstNO and EstO used in Algorithm 1 have similar structures. Due to the page limit, we only show EstNO(B, i) in Algorithm 2 as an instance for explanation. Basically, for every sub-bucket, we calculate its probability of being a non-outlier bucket. The loop variable j set from 2 to $B-1$ is the index of sub-buckets. The first and last sub-buckets are special cases, which will be handled later (lines 16–21). For each sub-bucket j, q_1 is the probability that a data point in bucket $i-1$ resides in the rightmost $B-j$ sub-buckets of bucket $i-1$ and q_2 is the probability that a data point in bucket $i+1$ is within the leftmost $j-1$ sub-buckets of bucket $i+1$. In this algorithm, a represents the number of data points in the rightmost $B-j$ sub-buckets of bucket $i-1$ and b is the number of data points in the rightmost $j-1$ sub-buckets of bucket $i+1$. We enumerate all possible combinations of a and b, which satisfy $a+b > k-f_i$, and calculate the probabilities for values a and

Algorithm 1 Find the Optimal Bucket Width

1: **for** $i = 1$ to $\lceil \frac{l}{d} \rceil$ **do**
2: **if** $f_i \leq k$ and $f_{i-1} + f_i + f_{i+1} > k$ **then**
3: $M[i] = 1$, $N_{po} = N_{po} + f_i$
4: **end if**
5: **end for**
6: $min = Cost' =$ Eq.(2)
7: **for** $B = 2$ to d **do**
8: $Cost = N \cdot (\log B + \lceil \frac{l}{d} \rceil)$
9: **for** $i = 1$ to $\lceil \frac{l}{d} \rceil$ **do**
10: **if** $M[i-1] + M[i] + M[i+1] > 0$ **then**
11: $Cost = Cost + N \cdot (B-1) \log(k+1)$
12: **end if**
13: **if** $M[i] = 1$ **then**
14: $e = \text{EstO}(B, i)$, $EN_o = EN_o + e$
15: $EN_{po} = EN_{po} + f_i - \text{EstNO}(B, i) - e$
16: **end if**
17: **end for**
18: $Cost = Cost + EN_o \cdot avgDist \cdot \log v_{max} + EN_{po} \cdot (avgDist \cdot \log v_{max} + N_{nl} \cdot \log v_{max} + N \cdot \log(k+1))$
19: **if** $Cost < min$ **then**
20: $optB = B$, $min = Cost$
21: **end if**
22: **end for**
23: **if** $min = Cost'$ **then**
24: there is no need for more histogram query
25: **else**
26: $d' = \frac{d}{optB}$
27: **end if**

Algorithm 2 EstNO(B, i)

1: $t = k - f_i$
2: **for** $j = 2$ to $B - 1$ **do**
3: $q_1 = \frac{B-j}{B}$, $q_2 = \frac{j-1}{B}$, $p = 0$
4: **for** $a = 0$ to f_{i-1} **do**
5: $p_1 = \binom{f_{i-1}}{a} q_1^a (1-q_1)^{f_{i-1}-a}$
6: **if** $a > t$ **then** $p_2 = 1$
7: **else if** $t + 1 - a > f_{i+1}$ **then** $p_2 = 0$
8: **else**
9: **for** $b = t + 1 - a$ to f_{i+1} **do**
10: $p_2 = p_2 + \binom{f_{i+1}}{b} q_2^b (1-q_2)^{f_{i+1}-b}$
11: **end for**
12: $p = p + p_1 p_2$
13: **end for**
14: $r = r + p\frac{f_i}{B}$
15: **end for**
16: **if** f_{i-1} or $f_{i+1} > t$ **then**
17: $p = 0$
18: **for** $a = t + 1$ to f_{i-1} or f_{i+1}
19: $p = p + \binom{f_{i-1} \text{ or } f_{i+1}}{a} (\frac{B-1}{B})^a$
20: **end for**
21: $r = r + p\frac{f_i}{B}$
22: return r

b, indicated by p_1 and p_2 respectively. The sum of $p_1 p_2$ for all possible cases becomes the probability that sub-bucket j is a non-outlier bucket. This value is recorded in variable p. On average, there are $\frac{f_i}{B}$ data points in sub-bucket j, so $p\frac{f_i}{B}$ is the expected number of non-outliers in sub-bucket j. After checking every bucket, we store the total number of non-outliers in r and return it as the result.

6. OUTLIER DETECTION FOR $O(n, k)$

In this section, we present a solution to detect the outliers defined by Definition 2. Given k and n, we sort all data points according to the distances to their KNNs. Let the sorted points be $p_1, p_2, \ldots$, where $D^k(p_i) \geq D^k(p_j)$ for $i < j$. The first n data points, $p_1, \ldots, p_n$, are all the $O(n, k)$ outliers we are looking for.

Our approach is still based on equi-width histogram. The sink sends histogram queries for multiple iterations and tries to find a suitable cut-off value c that separates $D^k(p_n)$ and $D^k(p_{n+1})$. The histogram collected in each iteration gives us an estimation for the range of c and helps filter out the buckets that are out of our interests. Then we use the next query to obtain more detailed histogram of the buckets that possibly hold outliers. This query process is repeated till we find all outliers. Note this approach does not fetch and check potential outliers as in the last step of finding the $O(d, k)$ outliers. Checking a potential outlier in this problem is very costly when k is large, because every sensor has to send k data values (k nearest neighbors of the potential outlier).

In the following, we first show that we can estimate bounds for the cut-off value c based on the histogram information. We try to find a pair of values L_c and U_c, such that $c \in (L_c, U_c]$. Suppose the sink sends a histogram query with bucket width w. After receiving the histogram, we first apply Theorem 1 and Theorem 2 on every bucket i to calculate l_i and u_i. Then we calculate L_c and U_c according to the following theorems.

THEOREM 3. *Consider the histogram collected with bucket width w. We have*

$$L_c = \max \{x | \sum_{l_i \geq x} f_i > n, x \text{ is multiple of } w\} < c.$$

PROOF. In the above equation, the condition, $\sum_{l_i \geq x} f_i > n$, means that there are more than n data points (p) satisfying $D^k(p) \geq x$. Based on the definition of the cut-off value c, $x < c$. Thus, any x satisfying the condition can be an exclusive lower bound of c. □

THEOREM 4. *Consider the histogram collected with bucket width w. We have*

$$U_c = \min \{x | \sum_{u_i \geq x} f_i \leq n, \ x + 1 \text{ is multiple of } w\} \geq c.$$

PROOF. The condition, $\sum_{u_i \geq x} f_i \leq n$, means that the number of all possible data points (p) satisfying $D^k(p) \geq x$ is less than or equal to n. According to the definition of c, $x \geq c$. Thus, any x satisfying the condition can be an inclusive upper bound of c. □

Our solution is shown in Algorithm 3. We start a histogram query with an initial bucket width w_{init}. Based on the received histogram, we obtain l_i and u_i for each bucket i and calculate L_c and U_c (lines 4–6). Then, we categorize buckets as follows:

- Case 1: If $u_i \leq L_d$, bucket i is a *non-outlier bucket*;

Algorithm 3 Find $O(n,k)$ Outliers

1: $w = w_{init}$, $hasPO = true$, $q_1 = q_2 = \cdots = 1$
2: **while** $w > 1$ and $hasPO$ **do**
3: Send a query with $< w, q_1 q_2 \cdots >$ and collect the histogram with bucket width w
4: Calculate l_i and u_i for each bucket i
5: $L_d =$ max $\{x | \sum_{l_i \geq x} f_i > n\}$
6: $U_d =$ min $\{x | \sum_{u_i \geq x} f_i \leq n\}$
7: **for** $i = 1$ to $\lceil \frac{l}{w} \rceil$ **do**
8: $q_i = (l_i \geq U_d)$
9: **end for**
10: Send a query with $\{q_i\}$ and collect the outliers
11: $hasPO = false$
12: **for** $i = 1$ to $\lceil \frac{l}{w} \rceil$ **do**
13: **if** $l_i < U_d$ and $u_i > L_d$ **then**
14: $q_i = 1$, $hasPO = true$
15: **else** $q_i = 0$
16: **end for**
17: $w = \frac{w}{2}$
18: **end while**

- Case 2: If $l_i \geq U_d$, bucket i is an *outlier bucket*;
- Case 3: Otherwise, bucket i is a *potential outlier bucket*.

Similar to the $O(d,k)$ outlier detection, we ignore the non-outliers in case 1 and send another query to collect the data values in the outlier buckets (lines 7–10). For case 3, we query for more histogram information of potential outlier buckets, which are marked by the variable q_i (lines 12–16). The bucket width of the new query is set to the half of the current bucket width. Upon receiving the query, sensor nodes calculate the histogram of the marked buckets (by q_i) with new bucket width, and send it back to the sink in a bottom-up direction. We repeat this process until all outliers are found. In the worst case, we need $\log w_{init}$ iterations.

7. PERFORMANCE EVALUATION

7.1 Network Settings and Datasets

In this simulation, we use real datasets collected from Intel Lab [1]. The data were collected from 54 sensors during a one-month period. The details of the dataset can be found at Intel Lab's web site [1]. We consider a 100×100 network field, where the sink is placed in the center. We deploy 54 sensor nodes randomly in the field and assume sensors communicate in a multi-hop fashion. The communication range is set to 18 for good connectivity in a random topology. Two important parameters used in our algorithms, the number of non-leaf nodes and the average hop distance, are shown in Table 1. The values are average measurements of 1000 connected random topologies.

In our simulation, we select the entire temperature records on two dates (03/01 and 03/20) as two datasets. The dataset for 03/01 represents a regular temperature distribution with mean value around 24 degrees. The dataset for 03/20, however, displays a large deviation from the average value. In the latter dataset, for some reason, 50 degrees is reported for many times, and a lot of data are sparsely scattered between 35 degrees and 50 degrees. In this simulation, we use precision 0.01 to round temperature values and scale them by 100 times in order to obtain integer values. The relevant parameters are also listed in Table 1 and Table 2.

Table 1: Network Setup

Number of Sensors(N)	54
Number of Non-leaf Nodes(N_{nl})	25.7
Radio Range	18
Avg. Hop Distance($avgDist$)	4.26

Table 2: Data Characteristics

	03/01 Dataset	03/20 Dataset
Number of Data Points	91468	76871
Maximum Value	3424	5008
Minimum Value	1499	363
Value Range	1926	4646

7.2 Performance Results

In this subsection, we show the performance of our algorithms in terms of the total communication cost for finding all the outliers, which is the sum of all sensors' communication costs. We assume that the cost of transferring a message is proportional to the payload size, which includes the actual data size and necessary control information, e.g., the message type. Thus, in the following, the total communication cost is measured by the total size of the messages transferred in the whole network. We first measure the communication costs for the *centralized scheme* through 1000 independent simulation runs and use the average value as the baseline. In the centralized scheme, the whole network transfers 575K bytes data for the 03/01 dataset on average and 514K bytes for the 03/20 dataset. The deviations for two datasets are 119K bytes and 113K bytes respectively. In addition, to evaluate our algorithms, we conduct 100 independent simulations for each parameter setting. We normalize the average communication costs in our algorithms against the baselines of the centralized scheme and show them as percentage values in the rest of this section. In our simulation, the ratio of standard deviation over average value is always less than 0.14. Since the deviation is small as a percentage value as shown later, the average communication cost is representative for the performance.

7.2.1 $O(d,k)$ Outlier Detection

To compare the basic scheme and enhanced scheme with different parameters, we vary d and k separately. First, we fix $k = 100$ and vary d from 20 to 70 for the 03/01 dataset and from 50 to 450 for the 03/20 dataset. Fig. 4 shows the numbers of outliers with various d. We find the two datasets differ dramatically. For the 03/01 dataset, when we set $d = 70$ (i.e., 0.7 degree in original data), no outlier exists in the entire set. For the 03/20 dataset, however, when we use a large distance with $d = 100$ (1 degree in the original data), 126 outliers appear. We keep increasing d to 400 (4 degree), we still find one outlier. This figure indicates that the 03/20 dataset contains more scattered data points and yields more outliers for a certain (d,k) setting.

The performance of basic and enhanced schemes is illustrated in Fig. 5. First, as shown, the enhanced scheme is always superior to the basic scheme. Secondly, both schemes

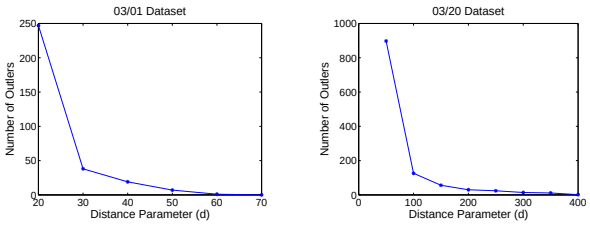

Figure 4: Number of outliers for varying d ($k = 100$)

greatly reduce communication costs. In the worst case in Fig. 5, the basic scheme consumes less than 5.5% of the cost of the centralized scheme.

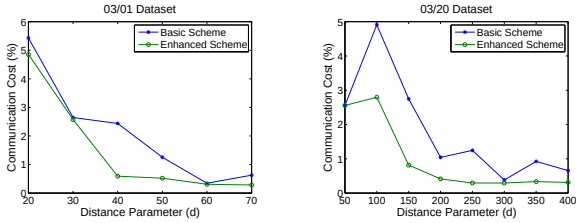

Figure 5: Communication costs for varying d ($k = 100$)

In the following, we will analyze and compare the performance of the basic and enhanced schemes. The communication cost in both schemes is comprised of histogram collection and raw data transfer (collection and diffusion). For a fixed value range, larger bucket width yields smaller number of buckets and less cost in histogram collection. On the other hand, larger bucket width provides less detailed histogram information, which may increase the number of potential outliers and the cost in transferring raw data.

In the 03/01 dataset, when we query for outliers, most non-outliers are identified after the first round of histogram collection, and the number of potential outliers is limited. Thus, the cost of histogram collection is the dominant factor compared with the raw data transfer. As shown in Fig. 5, the performance keeps decreasing along the increasing d. In the 03/20 dataset, a lot of data are sparsely distributed over an abnormal range and the number of outliers is dramatically larger than that in the 03/01 dataset. Since we set larger bucket width for the 03/20 dataset, the cost of histogram collection is less than that in the 03/01 dataset. On the other hand, as we mentioned above, larger bucket width may increase the cost of raw data transfer due to insufficient histogram information. Therefore, the cost of histogram collection is no longer dominant as the difference with the cost of raw data transfer is alleviated. Sometimes, raw data transfer is even more significant than histogram collection. These two types of cost interact with each other and show unstable curves for the 03/20 dataset in Fig. 5. The enhanced scheme outperforms the basic scheme in both datasets by filtering out more potential outliers.

In our simulation, we also fix d and study the performance on variable k. Fig. 6 shows the change of the number of outliers and Fig. 7 is the performance comparisons. Similarly, we find that the enhanced scheme is better than the basic scheme and both schemes are very efficient. In this case, the cost of histogram collection is fixed for basic scheme and

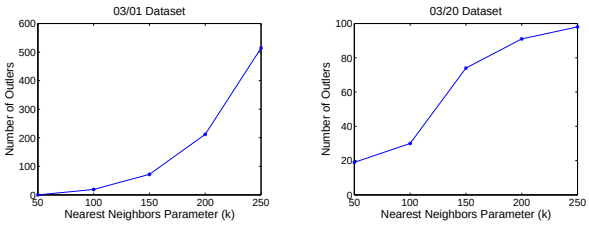

Figure 6: Number of outliers for varying k ($d = 40$ for the 03/01 dataset and $d = 200$ for the 03/20 dataset)

the communication cost only depends on the number of potential outliers. For a given k, the data points, which have roughly k neighbors within distance d, have a high probability to be potential outliers, because it is hard to distinguish these data points by coarse histogram information. Thus, the trend of the curves in Fig. 6, which indicate the number of nearby potential outliers, has an impact on the communication cost. As we can see, for the 03/01 dataset, there is a sharp increase of outliers when $k \in [150, 250]$, which means many potential outliers will be transferred as raw data when we search for outliers. Correspondingly, we see an increase of communication cost around that range in Fig. 7. Additionally, in the 03/20 dataset, the number of outliers has a jump from $k = 100$ to $k = 150$. It also yields an increased cost of basic scheme in Fig. 7. For both datasets, the enhanced scheme smooths the impact of the increased potential outliers and significantly improves the performance.

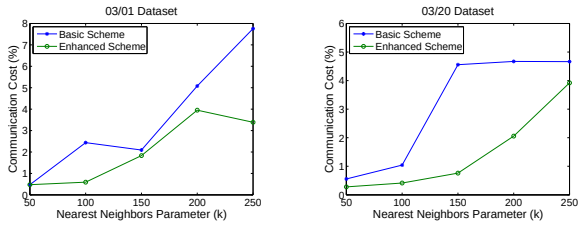

Figure 7: Communication costs for varying k ($d = 40$ for the 03/01 dataset and $d = 200$ for the 03/20 dataset)

As a summary, the proposed histogram approach is very efficient for the $O(d, k)$ outlier detection. The enhanced scheme consumes less than 4% of the cost of the centralized scheme in most cases.

7.2.2 $O(n, k)$ Outlier Detection

In this simulation, we set $k = 100$, and vary n from 10 to 80 for both datasets. The initial bucket width is set to a large value of 1500, i.e., $w_{init} = 1500$. Fig. 8 shows the values for $D^k(p_n)$ and the communication cost is presented in Fig. 9.

The simulation results show that our approach is cost-efficient for the $O(n, k)$ outlier detection. Compared with the centralized solution, our approach significantly reduces the communication cost. For the abnormal 03/20 dataset, it takes less than 1% of the cost to find all top-80 outliers. For the normal 03/01 dataset, our scheme consumes less than 1.5% of the cost in all the cases.

Due to the page limit, we omit the detailed analysis of the performance. Basically, larger value of $D^k(p_n)$ requires

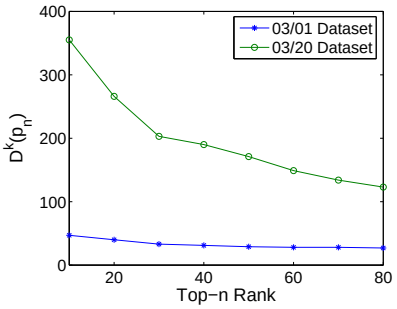

Figure 8: Values of $D^k(p_n)$ for varying n ($k = 100$)

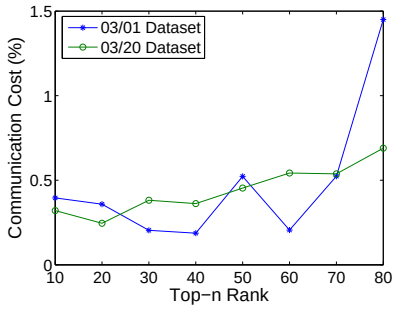

Figure 9: Communication costs for varying n ($k = 100$)

fewer number of iterations to find the cut-off value, and sharper change of $D^k(p_n)$ indicates that it is easier to distinguish the data points around the cut-off value. Both help reduce the communication cost.

8. CONCLUSION

In this paper, we consider the problem of finding two types of distance-based outliers in sensor networks. We propose to use a histogram method to extract hints about the data distribution from the sensor network. The histogram information can help us filter out the non-outliers and identify the potential outliers within a certain range. Those potential data ranges can be further refined with more histogram information or identified by individual check. The simulation shows that the histogram method reduces the communication cost dramatically relative to the centralized scheme.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation award CCF-0514985.

9. REFERENCES

[1] Intel Lab Data, http://berkeley.intel-research.net/labdata/, Apr. 2004.

[2] C. Aggarwal and S. Yu. An effective and efficient algorithm for high-dimensional outlier detection. *The VLDB Journal*, 14(2), 2005.

[3] C. C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *SIGMOD Rec.*, 30(1), 2001.

[4] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *SIGMOD '01*, New York, NY, USA, 2001.

[5] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD '03*, New York, NY, USA, 2003.

[6] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *ICDCS '06*, Lisboa, Portugal, July 2006.

[7] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *SIGMOD '00*, New York, NY, USA, 2000.

[8] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *ICDE '06*, 2006.

[9] W. F. Fung, D. Sun, and J. Gehrke. COUGAR: the network is the database. In *SIGMOD '02*, New York, NY, USA, 2002.

[10] M. B. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *PODS '04*, New York, NY, USA, 2004.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00*, New York, NY, USA, 2000.

[12] H. V. Jagadish, N. Koudas, and S. Muthukrishnan. Mining deviants in a time series database. In *VLDB '99*, 1999.

[13] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB '98*, San Francisco, CA, USA, 1998.

[14] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB '99*, San Francisco, CA, USA, 1999.

[15] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *KDD '05*, New York, NY, USA, 2005.

[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI), 2002.

[17] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1), 2005.

[18] S. Muthukrishnan, R. Shah, and J. S. Vitter. Mining deviants in time series data streams. In *SSDBM*, Santorini Island, Greece, June 2004.

[19] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Rec.*, 32(4), 2003.

[20] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD '00*, New York, NY, USA, 2000.

[21] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys '04*, New York, NY, USA, 2004.

[22] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang. A sampling-based approach to optimizing top-k queries in sensor networks. In *ICDE '06*, 2006.

[23] A. Silberstein, K. Munagala, and J. Yang. Energy-efficient monitoring of extreme values in sensor networks. In *SIGMOD '06*, Chicago, Illinois, USA, June 2006.

[24] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB'2006*, 2006.

[25] 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 *DMSN '05*, New York, NY, USA, 2005.