2012 IEEE 28th International Conference on Data Engineering

Detecting Outliers in Sensor Networks using the Geometric Approach

Sabbas Burdakis [#1], Antonios Deligiannakis [#2]

[#]*Department of Electronic and Computer Engineering, Technical University of Crete, Greece*
[1]sbourdakis@softnet.tuc.gr
[2]adeli@softnet.tuc.gr

***Abstract*—The topic of outlier detection in sensor networks has received significant attention in recent years. Detecting when the measurements of a node become "abnormal" is interesting, because this event may help detect either a malfunctioning node, or a node that starts observing a local interesting phenomenon (i.e., a fire). In this paper we present a new algorithm for detecting outliers in sensor networks, based on the geometric approach. Unlike prior work. our algorithms perform a distributed monitoring of outlier readings, exhibit 100% accuracy in their monitoring (assuming no message losses), and require the transmission of messages only at a fraction of the epochs, thus allowing nodes to safely refrain from transmitting in many epochs. Our approach is based on transforming common similarity metrics in a way that admits the application of the recently proposed geometric approach. We then propose a general framework and suggest multiple modes of operation, which allow each sensor node to accurately monitor its similarity to other nodes. Our experiments demonstrate that our algorithms can accurately detect outliers at a fraction of the communication cost that a centralized approach would require (even in the case where the central node lies just one hop away from all sensor nodes). Moreover, we demonstrate that these bandwidth savings become even larger as we incorporate further optimizations in our proposed modes of operation.**

I. Introduction

Recent advances in microelectronics have enabled the development of large scale sensor networks for a variety of monitoring applications, ranging from wildlife monitoring, healthcare, traffic monitoring, agriculture, production monitoring, battlefield surveillance etc. In such applications, detecting events of interest may require monitoring whether sensors collect measurements that are deemed as "similar" to the measurements of nearby sensors [6]. Detecting when the measurements of two nodes become dissimilar is interesting, because this event may help detect either (i) a malfunctioning node, or (ii) a node that starts observing a local interesting phenomenon (i.e., a fire).

The aforementioned detection process is often referred to as *outlier detection*. We need to note that, while several ways of classifying and detecting nodes as outliers exist, our work targets the important case where testing the similarity of measurements between different nodes is required by the outlier detection process. It, thus, is not applicable to other scenarios, for example when the measurements of a node may be classified as an outlier solely based on the node's past measurements.

For monitoring applications, it is often crucial to be able to detect interesting events with absolute accuracy. For example, in applications where sensors are deployed in order to detect natural phenomena, such as tsunamis, landslides or avalanches, a failure to quickly detect and report the offset of such phenomena may result in a failure to safely evacuate people from areas that are in danger. In other applications, extinguishing fires in forests is much easier when the fire is detected early, but extremely harder after the fire has escalated. In equipment monitoring, failure to accurately detect a malfunctioning part may result in the destruction of a much larger and expensive system. In all the above applications, it is crucial to have an outlier detection system that reports outlier nodes with 100% accuracy. Of course, to ensure the longevity of the sensor network, the requirement for accurate detection needs to coexist with data processing techniques that efficiently reduce the number of messages transmitted by sensor nodes.

A significant amount of effort [2], [4], [6], [8], [27], [29], [33] has been recently placed on detecting outliers. However, none of the existing techniques has tackled the general problem of being able to detect with 100% accuracy the similarity amongst any desired pair of sensor nodes, while at the same allowing, in some cases, sensor nodes to refrain from transmitting any information regarding their measurements. In a nutshell, existing outlier detection techniques suffer from one, or more, of the following drawbacks: (i) they cannot identify the similarity of two nodes with 100% accuracy (assuming no message losses); or (ii) they require the transmission of information that is comparable in size to a centralized query evaluation; or (iii) they require nodes to perform transmissions at each epoch; or (iv) they are tailored to the evaluation of specific similarity functions and/or cannot handle similarity functions over the measurements collected at different nodes (such as the correlation coefficient, or the L_∞ norm).

In this paper we present an outlier detection framework that is based on the recently proposed geometric approach [21], [22], [23], [24]. The geometric approach allows us to accurately (and efficiently) monitor whether a complex, potentially non-linear function, computed over the *average* of vectors maintained at all the sensor nodes, is above or below a specified threshold. In a nutshell, each sensor is automatically assigned a *monitoring zone*, which is a subset of the

A. Deligiannakis was supported by the European Commission under ICT-FP7-LIFT-255951 (Local Inference in Massively Distributed Systems).

1084-4627/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDE.2012.85

domain space, and examines whether the monitored function at any point within its monitoring zone may have crossed the threshold. Each sensor that detects a potential threshold violation makes a transmission (e.g., to a coordinator). While monitoring zones of different shapes may be chosen [24], without loss of generality, in this work we adopt the simplest case where the shape of each monitoring zone is a sphere [22]. What is guaranteed by the geometric approach is that, at any time, the true average vector (which is unknown to the sensor nodes) will lie within the *union* of the local monitoring zones examined by the sensor nodes. Thus, if no sensor signals a potential threshold violation, then the monitored function will not have crossed the threshold either, since the value of the monitored function for the true average vector has certainly been examined by at least one sensor node.

As we demonstrate in this paper, several common similarity functions (depicted in Figure 2) can be transformed in a way that they allow the application of the geometric approach. Then, the similarity between any pair of nodes is simply expressed as a function whose value is desirable to lie above/below a specified threshold. For example, we may consider two nodes to be similar if their vector of measurements have an L_1 distance that is below a threshold, or have a cosine similarity above a given threshold. Of course, the number of functions that we need to monitor in our application scenario may increase quadratically (in the worst case $O(\binom{n}{2})$) to the number n of sensor nodes. Our transformations allow us to utilize the geometric approach and develop a generic framework that: (i) supports a variety of similarity functions; (ii) performs the monitoring with 100% accuracy; and (iii) allows nodes to often refrain from any communication thus, as shown in our experimental evaluation, consuming only a fraction of the bandwidth that centralized techniques would require. This is in contrast to recently proposed outlier detection techniques [6], [8] that require each node to perform a transmission at each epoch. Extensions to allowing the specification of a minimum support (i.e., how many nodes need to be similar to me, so that I am not considered an outlier?) are also trivially incorporated in our framework.

Our contributions can be summarized as follows:

- We demonstrate that several common similarity functions used in outlier detection can be transformed in a way that allows the application of the geometric approach.
- We propose a generic framework for outlier detection in sensor networks using the geometric approach and propose various modes of node operation that achieve the desired monitoring task. Our framework computes with absolute accuracy (assuming no message losses) the similarity of two nodes, a property that stems directly from the geometric approach transformation.
- We examine cases of monitoring the similarity amongst sensor nodes when the communication between these sensors is direct, or not. We also demonstrate how our framework allows the evaluation of minimum support queries.
- We perform an extensive experimental evaluation using real world data. Our analysis demonstrates that our algorithms can accurately detect outliers at a fraction of the communication cost that a centralized approach would require (even in the case where the central node lies just one hop away from all sensor nodes). Moreover, we demonstrate that these bandwidth savings become even larger as we incorporate further optimizations in our proposed modes of operation.

The paper proceeds as follows. In Section II we present related work. Section III provides the necessary background on the geometric approach. Section IV details the setup that we consider in this paper. In Section V we demonstrate how several common similarity functions can be transformed in order to allow the application of the geometric approach. Section VI contains our framework and algorithms for monitoring the similarity of a node with any other sensor that our application deems necessary. Our experimental evaluation is presented in Section VII, while Section VIII contains concluding remarks and future directions.

II. Related Work

In recent years, significant effort has been placed on determining and designing the necessary primitives for data acquisition based on sensor networks [17], [30]. Multiple ways of organizing the network have been proposed, including hierarchical (i.e., tree-like) organizations such as the aggregation tree [17], [26], [32], clustered formations [3], [9], [20], [31], or even completely ad-hoc formations [1], [13], [16].

Due to their inexpensive hardware, sensor nodes are prone to producing outlier readings. Thus, many techniques that seek to determine nodes with "abnormal" behavior have been have been proposed [34]. In [10], [11], a declarative data cleaning mechanism over data streams produced by the sensors is proposed. Similarly, the work of [7] introduces a data cleaning module designed to capture noise in sensor streaming data based on the prior data distribution and a given error model $N(0, \delta^2)$. In [18] kalman filters are adopted during data cleaning or outlier detection procedures. Nonetheless, without prior knowledge of the data distribution the parameters and covariance values used in these filters are difficult to set. The data cleaning technique presented in [36] makes use of a weighted moving average which takes into account both recent local samples and corresponding values by neighboring motes to estimate actual measurements. A wavelet-based value correction process is discussed in [35] while outliers are determined utilizing the Dynamic Time Warping (*DTW*) distance of neighboring motes' values. The work in [28] proposes a fuzzy approach to infer the correlation among readings from different sensors, assigns a confidence value to each of them, and then performs a fused weighted average scheme. A histogram-based method to detect outliers with reduced communication cost is presented in [25].

The work in [4], [29] addresses the problem of identifying faulty sensors using a localized voting protocol. However, localized voting schemes are prone to errors when motes that observe interesting events generating outlier readings are not in direct communication [6]. Furthermore, the framework of [29] requires a *correlation network* to be maintained.

TABLE I
NOTATION

Symbol	Definition
S_i	The i-th sensor node
CN_i	The Comparison Neighborhood of S_i: With which nodes does S_i compute its similarity with?
W	Dimensionality of the measurements vector
d	Dimensionality of the local statistics vector
T	The similarity threshold
$\vec{e}$	The estimate vector. Its dimensionality is d
$\vec{v}_i$	The local statistics vector of S_i. Its dimensionality is d
$\vec{v}$	The true (not known by the sites) global statistics vector
$\Delta\vec{v}_i$	The delta vector of S_i. Calculated as the difference of the current local statistic vector from the last local statistic vector that S_i has transmitted.
$\vec{u}_i$	The drift vector of S_i. Equal to $\vec{e} + \Delta\vec{v}_i$
$B(\vec{e}, \vec{u}_i)$	The sphere having $\vec{e}$ and $\vec{u}_i$ as its diameter
$Conv(\vec{e}, \vec{u}_1, \ldots, \vec{u}_n)$	The convex hull determined by vectors $\vec{e}$, $\vec{u}_1, \ldots, \vec{u}_n$
$minSupp$	The specified minimum support

In [15], the authors discuss a framework for cleaning input data errors using integrity constraints, while in [2], [33] unsupervised outlier detection techniques are used to report the top-k values that exhibit the highest deviation in a network's global sample. However, these techniques provide no means of directly controlling the bandwidth consumption, thus often requiring comparable bandwidth to centralized approaches [10] for outlier detection [2].

In [12], a probabilistic technique for cleaning RFID data streams is presented. In [27] the authors introduce a novel definition of an outlier, as an observation that is sufficiently far from most other observations in the data set. A similar definition is adopted in [19] where a distributed outlier detection approach for dynamic data sets is presented. The framework of [6] is used to identify and remove outliers during the computation of aggregate and group-by queries posed to an aggregation tree [5], [17]. The TACO [8] framework operates on top of a clustered network organizations and attempt to identify outliers based on compressed (LSH) representations of the collected data.

The algorithms in [6], [8], [27] have significant drawbacks compared to our proposed technique. Perhaps most important, they provide no strong guarantees of detecting outlier nodes, but rather follow a best-effort approach ([6], [27]), or at best offer (TACO) some probabilistic guarantees, which however are poor when the similarity of two sensors is close the similarity threshold. Furthermore, the work in [27] is tailored to different similarity functions and cannot directly handle the similarity functions that we target in this paper. The bandwidth savings of [6], compared to centralized alternative algorithms, are modest, while TACO cannot operate at each epoch (without being prohibitively expensive in bandwidth), but rather operates in tumbles. Both [6] and TACO require each node to perform a transmission at each epoch. On the contrary, our proposed algorithms (i) guarantee that the similarity (or not) of two nodes can always be performed with absolute certainty (assuming reliable communication), (ii) can operate in a continuous fashion, thus computing the similarity of nodes at each epoch, and (iii) enable sensor nodes to safely refrain from transmitting in many epochs, thus achieving significant bandwidth savings.

The geometric approach was first presented in [22]. The monitoring zones used in [22] were spheres, an approach that we also adopt in our work due to its lower computational requirements. The work in [14], [24] demonstrated that using monitoring zones of different shapes (i.e., triangles or ellipses) may be more beneficial in terms of the number of transmitted messages. In Section VI we also exploit (for some functions such as the L_∞, L_1, L_2 metrics) the notion of safe zones introduced in [14] in order to further reduce the number of transmitted messages. [21] presents a framework for the efficient evaluation of threshold queries of general functions over distributed data (as opposed to distributed data streams settings used in [14], [22], [24]). The work in [23] is the only work that we are aware of that applies the geometric approach over sensor networks (in order to answer aggregate threshold queries). In [23] the aggregate function is computed over the entire network. Besides the difference in the type of the monitored function with our problem, in this paper we are interested in evaluating multiple pair-wise similarity functions, a setting in which [23] is not efficient.

III. BASICS - THE GEOMETRIC APPROACH

We now describe in more detail the geometric approach for function monitoring over a distributed system of n sites. Table I summarizes the most important notation used in this paper. The corresponding definitions appear at appropriate areas of the text. Figure 1 demonstrates the basic ideas of the geometric approach that we discuss in this section.

Each site S_i maintains a local d-dimensional vector, termed as the *local statistics vector*, with the j-th ($j = 1 \ldots d$) element of the local statistics vector of S_i denoted as $\vec{v}_{j,i}$. All sites contain a vector of the same dimensionality (i.e., number of elements). The *global statistics vector* $\vec{v}$ is computed as the average[1] amongst all local statistics vectors. Thus, the j-th component of the global statistics vector, denoted as $\vec{v}_j$ is computed as: $\vec{v}_j = \frac{1}{n}\sum_{i=1}^{n} \vec{v}_{j,i}$.

For the framework to be applicable, any supported monitoring function $f : \mathcal{R}^d \rightarrow \mathcal{R}$ must be expressed over the global statistics vector $\vec{v}$ (thus, over the average of all local statistics vectors). An important feature is the wide applicability of the geometric approach, as the threshold function can in general be non-linear. Given a threshold T, the framework in [21], [22], [23], [24] can safely determine whether $f(\vec{v}) > T$.

The geometric approach decomposes the monitoring task into a set of constraints (one per site) that each site can monitor *locally*. To achieve this, during the operation of the algorithm,

[1]The same framework also applies when the global statistics vector is calculated as a weighted average of the local statistics vectors.

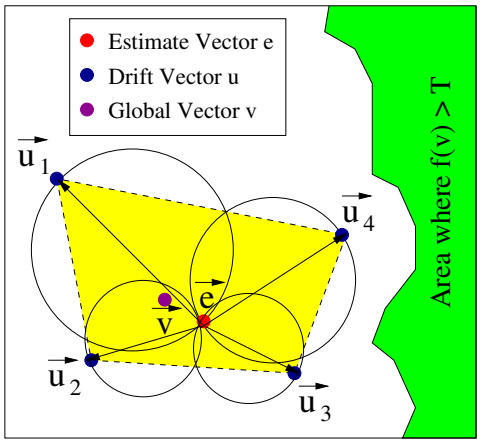

Fig. 1. Local constraints using the Geometric Approach. Each node constructs a sphere with diameter the drift vector $\vec{u}$ of the node and the estimate vector $\vec{e}$. The global statistics vector $\vec{v}$ is guaranteed to lie in the convex hull of $\vec{e}$, $\vec{u}_1$, $\vec{u}_2$, $\vec{u}_3$, $\vec{u}_4$. The union of the local spheres covers the convex hull.

each site S_i maintains (i) the estimate vector $\vec{e}$, which is equal to the global statistics vector $\vec{v}$ computed by the local statistics vectors transmitted by sites at certain times, and (ii) a delta vector $\Delta\vec{v}_i$, denoting the difference of the current local statistic vector from the last local statistic vector that S_i has transmitted. Based on these two quantities, S_i calculates its drift vector $\vec{u}_i = \vec{e} + \Delta\vec{v}_i$. Additional optimizations have been developed in the framework, such as the ability to *balance* only a portion of the network in case of violations. In that case, an additional *slack* vector needs to be maintained and added in the calculation of the drift vector.

The domain space $\mathcal{R}^d$ represents the potential locations of the global statistics vector at any time. Let all points in $\mathcal{R}^d$ where $f(\vec{v}) <= T$ be colored by the same color (i.e., white in Figure 1), while the remaining points be colored by a different color (i.e., green in Figure 1). Because the sites do not perform transmissions at each time period, the current global statistics vector $\vec{v}$ is not known to the sites. However, what is guaranteed is that $\vec{v}$ will always lie within the convex hull $Conv(\vec{u}_1, \ldots, \vec{u}_n)$ of the drift vectors and, thus, within the convex hull $Conv(\vec{e}, \vec{u}_1, \ldots, \vec{u}_n)$ of the drift vectors and the estimate vector. Thus, if $Conv(\vec{e}, \vec{u}_1, \ldots, \vec{u}_n)$ is *monochromatic* (i.e., either entirely below/equal to the threshold, or entirely above to the threshold), then all sites are certain about the color of the function $f()$, since this will coincide with the color of $f(\vec{e})$. Of course, each node cannot compute $Conv(\vec{e}, \vec{u}_1, \ldots, \vec{u}_n)$, since it is not aware of the current drift vectors of other sites. However, an important observation [22] is that if each site monitors the sphere $B(\vec{e}, \vec{u}_i)$ constructed with diameter the estimate vector and its own drift vector, then the union of these spheres covers the convex hull. Thus, it suffices for each node to simply monitor whether its sphere is monochromatic. If all the spheres are monochromatic, then the convex hull is also monochromatic and, thus, $f(\vec{v})$ has the same color as $f(\vec{e})$. Otherwise, nodes transmit their local statistics vectors, and a new estimate vector is computed and made known to all nodes.

IV. Problem Setup

In this paper we are interested in the following general problem setup. A base station monitors the pair-wise similarity of W-dimensional vectors of measurements collected at sensor nodes. While our techniques are not restricted to the origin of these W values, two likely alternatives are the following: (i) the vector contains the latest W readings regarding the same quantity; or (ii) the vector contains the current reading of W different quantities that the sensor monitors.

Each sensor S_i is asked to continuously compare its vector of measurements to a subset (which we will term as the *comparison neighborhood* (CN) of S_i) of the other nodes in the network. We make no assumption on whether S_i can directly communicate with all nodes in CN_i. We, thus, first discuss the general case, where each sensor may require a unicast (potentially multi-hop) communication with all nodes in its CN, and then discuss further potential optimizations that can be applied when all the nodes in CN_i are in direct communication with S_i. The in-between scenario where a node S_i is in direct communication with only a fraction of the nodes in CN_i can be trivially handled by partitioning the nodes in CN_i in two sets - the first set contains those nodes that are in direct communication with S_i, while the second set contains the remaining nodes - and applying the techniques developed for the appropriate scenario to each of these sets.

The requirement of our application is for the base station to know with certainty (assuming no message losses) whether each node is similar (or not) to the other nodes in its CN. If a node S_i detects that its similarity with a node $S_j \in CN_i$ has changed (i.e., S_i and S_j were dissimilar/similar up until the previous epoch, but they are now deemed similar/dissimilar), then (exactly) one of these nodes needs to notify the base station. Obviously, these notifications from all sensor nodes may use for their propagation an interconnect such as the aggregation tree [17].

Another interesting application involves monitoring whether the measurements in each node S_i have a required minimum support $minSupp$, expressed as the number of nodes in CN_i that are deemed similar to S_i. We show in Section VI-C that this case can also be easily handled in our framework.

V. Expressing Similarity Functions using the Geometric Approach

We now show how several interesting similarity functions (shown in Figure 2) can be transformed in a way that they can be used in the geometric approach.

Notation and Important Notes. Let us consider the similarity between two nodes with measurement vectors X and Y, correspondingly. Because some of the transformations that we need to perform do not treat X and Y in a symmetric way, we always assign X to the node (in the pair-wise test) with the lowest identifier (id), among the two nodes.

In order to use the geometric approach, each similarity function must be expressed as a general function over the

Similarity Metric	*Function based on vectors X, Y of two nodes*	*Transforming the Function Calculation into a Function over Average Quantities that each node may compute individually*	*Local Statistic Vectors X', Y' (may contain more elements than X, Y)*	*Value of function on any point Z*
$\|\| X - Y \|\|_\infty$	$dist(X,Y) = max_i \|X_i - Y_i\|$	$2\, max_i\left(\left\|\frac{X_i - Y_i}{2}\right\|\right)$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\, max_i(\|Z_i\|)$
$\|\| X - Y \|\|_1$	$dist(X,Y) = \sum_{i=1}^{W} \|X_i - Y_i\|$	$2\sum_{i=1}^{W}\left\|\frac{X_i - Y_i}{2}\right\|$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\sum_{i=1}^{W}\|Z_i\|$
$\|\| X - Y \|\|_2$	$dist(X,Y) = \sqrt{\sum_{i=1}^{W}(X_i - Y_i)^2}$	$\sqrt{4\sum_{i=1}^{W}\left(\frac{X_i - Y_i}{2}\right)^2}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\sqrt{\sum_{i=1}^{W} Z_i^2}$
$\|\| X - Y \|\|_k$	$dist(X,Y) = \sqrt[k]{\sum_{i=1}^{W}(\|X_i - Y_i\|)^k}$	$\sqrt[k]{2^k\sum_{i=1}^{W}\left(\left\|\frac{X_i - Y_i}{2}\right\|\right)^k}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \end{bmatrix}$ $Y' = \begin{bmatrix} -Y_1 \\ \dots \\ -Y_W \end{bmatrix}$	$2\sqrt[k]{\sum_{i=1}^{W}\|Z_i\|^k}$
Cosine Similarity	$cos(\theta(X,Y)) = \frac{XY}{\|X\|_2\,\|Y\|_2}$	$\frac{2\sum_{i=1}^{W}\left(\frac{X_i + Y_i}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}}{2\left(\frac{\|X\|_2 + \|Y\|_2}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \\ \|X\|_2^2 \\ \|X\|_2 \end{bmatrix}$ $Y' = \begin{bmatrix} Y_1 \\ \dots \\ Y_W \\ \|Y\|_2^2 \\ \|Y\|_2 \end{bmatrix}$	$\frac{2\sum_{i=1}^{W} Z_i^2 - Z_{W+1}}{2\,Z_{W+2}^2 - Z_{W+1}}$
Extended Jaccard Coefficient	$J(X,Y) = \frac{XY}{\|X\|_2^2 + \|Y\|_2^2 - XY}$	$\frac{2\sum_{i=1}^{W}\left(\frac{X_i + Y_i}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}}{2\frac{\|X\|_2^2 + \|Y\|_2^2}{2} - \left(2\sum_{i=1}^{W}\left(\frac{X_i + Y_i}{2}\right)^2 - \frac{\|X\|_2^2 + \|Y\|_2^2}{2}\right)}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \\ \|X\|_2^2 \end{bmatrix}$ $Y' = \begin{bmatrix} Y_1 \\ \dots \\ Y_W \\ \|Y\|_2^2 \end{bmatrix}$	$\frac{2\sum_{i=1}^{W} Z_i^2 - Z_{W+1}}{2\,Z_{W+1} - (2\sum_{i=1}^{W} Z_i^2 - Z_{W+1})}$
Correlation Coefficient	$corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$	$\frac{\left(\frac{2}{W}\sum_{i=1}^{W}(\frac{X_i + Y_i}{2})^2 - \frac{1}{W}\frac{\|X\|_2^2 + \|Y\|_2^2}{2}\right) - \left(2\left(\frac{E[X] + E[Y]}{2}\right)^2 - \frac{(E[X])^2 + (E[Y])^2}{2}\right)}{2\left(\frac{\sigma_X + \sigma_Y}{2}\right)^2 - \frac{\sigma_X^2 + \sigma_Y^2}{2}}$	$X' = \begin{bmatrix} X_1 \\ \dots \\ X_W \\ \|X\|_2^2 \\ E[X] \\ (E[X])^2 \\ \sigma_X \\ \sigma_X^2 \end{bmatrix}$ $Y' = \begin{bmatrix} Y_1 \\ \dots \\ Y_W \\ \|Y\|_2^2 \\ E[Y] \\ (E[Y])^2 \\ \sigma_Y \\ \sigma_Y^2 \end{bmatrix}$	$\frac{\left(\frac{2}{W}\sum_{i=1}^{W} Z_i^2 - \frac{1}{W} Z_{W+1}\right) - (2 Z_{W+2}^2 - Z_{W+3})}{2 Z_{W+4}^2 - Z_{W+5}}$

Fig. 2. Expressing common similarity functions between two nodes with W-dimensional measurement vectors X and Y using the geometric approach. The function must be expressed as a general function over the average of local statistics vectors X' and Y', with the elements of X' (Y') being computed based only on the elements of X (Y). X' (Y') may have a different dimensionality than X (Y).

average of local statistics vectors X' and Y', with the elements of X'/Y' being computed based only on the elements of X/Y. We need to emphasize the following points:

- X'/Y' may have a different dimensionality than X/Y.
- The dimensions of the sub-space that each node monitors (i.e., the spheres in Figure 1) correspond to the dimensionality of X'/Y', and not on the dimensionality of X/Y.
- During its monitoring, each node needs to calculate the value of the similarity function over any point Z that lies in its monitoring zone (i.e., in the sphere that it constructs). Any such point Z represents a potential position of the global statistics vector (which is computed as the average of local statistics vectors), and should not be confused with how the function is computed on either X/Y/X'/Y'. The last column of Figure 2 demonstrates how to compute the value of the function over any point Z.

Transformations for the L_∞, L_1, L_2 and L_k norms. Let us first consider the simplest case of the L_∞ norm, computed as the maximum difference $max_{i=1...W}\{|X_i - Y_i|\}$. Then:

$$\begin{aligned} L_\infty(X,Y) &= \max_{i=1...W}\{|X_i - Y_i|\} = 2\max_{i=1...W}\{|\frac{X_i - Y_i}{2}|\} \\ &= 2\max_{i=1...W}\{|\frac{X_i + (-Y_i)}{2}|\} \end{aligned}$$

Thus, by setting $X' = X$ and $Y' = -Y$, the overall $L_\infty(X,Y)$ can be computed as a function on the average vector $\frac{1}{2}(X' + Y')$. Please recall that, as mentioned earlier in this section, the local statistics vector Y' corresponds to the node in the pair-wise test with the highest id.

The transformations for the L_1, L_2 and L_k norms are similar in principle. We, thus demonstrate the transformation only for the L_2 norm.

$$\begin{aligned} L_2(X,Y) &= \sqrt{\sum_{i=1}^{W}(X_i - Y_i)^2} = \sqrt{2^2\sum_{i=1}^{W}(\frac{X_i - Y_i}{2})^2} \\ &= 2\sqrt{\sum_{i=1}^{W}(\frac{X_i - Y_i}{2})^2} \end{aligned}$$

Thus, similarly to the L_∞ case, by using $X' = X$ and $Y' = -Y$ as the local statistics vectors at the two nodes, the L_1, L_2 and L_k norms, can be evaluated as shown in Figure 2.

Transformations for the Cosine Similarity. In order to compute the cosine similarity of two vectors, we first need to express the inner product (which is equal to the summation of the products of corresponding vector elements) in a form that admits the geometric approach. Thus:

$$\begin{aligned} X \cdot Y &= \sum_{i=1}^{W} X_i Y_i = \frac{1}{2}\sum_{i=1}^{W} 2X_i Y_i \\ &= \frac{1}{2}(\sum_{i=1}^{W}(X_i + Y_i)^2 - (\sum_{i=1}^{W}(X_i)^2 + \sum_{i=1}^{W}(Y_i)^2)) \\ &= 2\sum_{i=1}^{W}(\frac{X_i + Y_i}{2})^2 - \frac{||X||_2^2 + ||Y||_2^2}{2} \end{aligned}$$

Using a similar transformation for the quantity at the denominator of the cosine similarity:

$$||X||_2||Y||_2 = 2(\frac{||X||_2 + ||Y||_2}{2})^2 \ - \frac{||X||_2^2 + ||Y||_2^2}{2}.$$

Therefore, the overall cosine similarity

$$cos(\theta(X,Y)) = \frac{XY}{||X||_2||Y||_2}$$

can be computed as a function of average quantities. Thus, by maintaining

$$X' = [X_1, \ldots, X_W, ||X||_2^2, ||X||_2]^T$$

(for the node with the lowest id in the comparison test), and

$$Y' = [Y_1, \ldots, Y_W, ||Y||_2^2, ||Y||_2]^T$$

(for the node with the highest id), the overall cosine similarity can be computed over the average of the local statistics vectors maintained at the nodes (Figure 2). Please note that in this case the dimensionality of the local statistics vectors is $W+2$ and, therefore, larger than the dimensionality of the measurements vector.

Transformations for the Extended Jaccard Coefficient. The Extended Jaccard contains two quantities that need to be transformed: (i) The inner product $X \cdot Y$, which is transformed in the same way as in the cosine similarity, and (ii) the term $||X||_2^2 + ||Y||_2^2$, which can be transformed to $2\frac{||X||_2^2+||Y||_2^2}{2}$. Thus, as shown in Figure 2, the Extended Jaccard Coefficient can be transformed in the required format by maintaining as local statistics vectors

$$X' = [X_1, \ldots, X_W, ||X||_2^2]^T$$

(for the node with the lowest id in the comparison test), and

$$Y' = [Y_1, \ldots, Y_W, ||Y||_2^2]^T$$

(for the node with the highest id).

Transformations for the Correlation Coefficient. The most difficult transformation involves the computation of the Correlation Coefficient: $corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$, computed based on the covariance of the two vectors of measurements, and their standard deviations. The denominator can be computed based on the same approach that we have demonstrated:

$$\sigma_X \sigma_Y = 2(\frac{\sigma_X + \sigma_Y}{2})^2 - \frac{\sigma_X^2 + \sigma_Y^2}{2}.$$

We then observe that the covariance $cov(X,Y)$ can be computed as: $cov(X,Y) = E[XY] - E[X]E[Y]$. The product $E[X]E[Y]$ can be transformed as:

$$E[X]E[Y] = 2(\frac{E[X] + E[Y]}{2})^2 - \frac{(E[X])^2 + (E[Y])^2}{2}.$$

For the term $E[XY]$ we obtain:

$$E[XY] \quad = \quad \frac{1}{W}\sum_{i=1}^{W} X_i Y_i$$

$$\overset{\substack{See\ inner \\ product}}{=} \quad \frac{2}{W}\sum_{i=1}^{W}(\frac{X_i + Y_i}{2})^2 - \frac{1}{W}\frac{||X||_2^2 + ||Y||_2^2}{2}$$

Based on the above transformations, as shown in Figure 2, the Correlation Coefficient can be transformed in the required format by maintaining as local statistics vectors

$$X' = [X_1, \ldots, X_W, ||X||_2^2, E[X], (E[X])^2, \sigma_X, \sigma_X^2]^T$$

(for the node with the lowest id in the comparison test), and

$$Y' = [Y_1, \ldots, Y_W, ||Y||_2^2, E[Y], (E[Y])^2, \sigma_Y, \sigma_Y^2]^T$$

(for the node with the highest id).

VI. Node Operation for Similarity Monitoring

We now propose different alternatives of node operation, so that each node can monitor with accuracy its similarity with any node in its CN set. For ease of presentation, we begin our discussion with a model of having just two nodes that want to compute their similarity. This model, on one hand, helps explain the various alternatives and, on the other hand, can be used for answering similarity queries between nodes that do not communicate directly with each other (the first scenario discussed in Section IV), while also being applicable to the second scenario (direct connectivity with all nodes in CN) as well.

Besides, simply applying the same pair-wise algorithms that we will present to all neighbors of a node, provides a solution to the monitoring task that we tackle in this paper. Each node, simply has to maintain statistics for each node in its CN set, and follow for each such node the process described in the two-node communication model.

A. A Two-Node Communication Model

In our initial model, we do not worry about whether the two nodes are in direct communication with each other. Thus, whenever we refer to a node S_i transmitting a message to S_j, this message may involve a multihop communication. Of course, it is more natural to expect that the similarity between the readings of two nodes will be useful when these sensors are placed nearby and can, thus, either communicate directly (pending any obstacle that may block their communication), or are reachable within a few hops.

All but (the last) one of the modes of operation that we propose in this paper do not require a fixed order (amongst the two nodes whose similarity is monitored) in which the sensor nodes will examine whether they have a local violation. However, for ease of presentation, let us simply assume that such an order, which need not be the same across all epochs (i.e., as in a case where nodes alternate their turn) has been established.

We need to note that a local violation occurs when a node deems that its local constraint is violated (i.e, when the sphere that it monitors is bi-chromatic). A global violation occurs when a node is certain that its similarity with another node has changed color. The node that detects a global violation notifies the base station.

The Simple Mode of Operation. Algorithm 1 presents the *Simple* mode of operation of each node. This mode of operation has no optimizations in its decisions and it is,

Algorithm 1 Node i: Operation under Simple Mode

Require: Threshold T, Similarity Function F

1: Maintain $\vec{v'}_i$: last transmitted local statistics vector
2: Maintain $\vec{e}$: Estimate vector
3: Maintain $\vec{v'}_j$: last received local statistics vector from S_j
4: **while** Asked to Monitor Similarity to S_j **do**
5: Obtain new measurements and form local statistics vector $\vec{v}_i$
6: Compute delta vector $\Delta\vec{v}_i = \vec{v}_i - \vec{v'}_i$
7: Compute drift vector $\vec{u}_i = \vec{e} + \Delta\vec{v}_i$
8: **if** Acting Second in Pair **then**
9: **if** MessageWait(S_j, $\vec{v'}_j$, $\vec{e'}$) == true **then**
10: **if** $\vec{e'}$ and $\vec{e}$ are bi-chromatic **then**
11: Notify base station about global violation
12: **end if**
13: Send local measurements vector to S_j
14: $\vec{v'}_i = \vec{v}_i$, $\vec{e} = \vec{e'}$
15: Continue
16: **end if**
17: **end if**
18: {Acting first, or acting second but did not receive a message}
19: localViolation = checkIfViolation($\vec{e}$, $\vec{u}_i$, F, T)
20: **if** localViolation == true **then**
21: Send local measurements vector to S_j
22: $\vec{v'}_i = \vec{v}_i$
23: MessageWait(S_j, $\vec{v'}_j$, $\vec{e'}$) {Will definitely arrive}
24: Compute $\vec{e} = \frac{1}{2}(\vec{v'}_j + \vec{v}_i)$
25: Continue {If global violation, S_j will send notification}
26: **end if**
27: **if** Acting first **then**
28: **if** MessageWait(S_j, $\vec{v'}_j$, $\vec{e'}$) == true **then**
29: goto 10
30: **end if**
31: **end if**
32: **end while**

Algorithm 2 Node i: MessageWait Subroutine

Require: Paired node S_j, last received local statistics vector $\vec{v'}_j$, vector $\vec{e'}$

1: Wait for message from S_j
2: **if** Message Arrived, containing vector of measurements **then**
3: Compute local statistics vector $\vec{v}_j$ of S_j
4: $\vec{v'}_j = \vec{v}_j$
5: Compute $\vec{e'} = \frac{1}{2}(\vec{v'}_j + \vec{v}_i)$
6: RETURN true
7: **end if**
8: RETURN false

thus, presented only as a baseline solution for estimating the similarity of two nodes S_i and S_j. One may consider this Simple mode as the most natural way of applying the geometric approach to our problem. However, while in prior work typically a coordinator node exists, in the Simple mode, as an optimization to reduce message transmissions, each node in a pair may act as a coordinator.

Both nodes at each epoch update their vectors of measurements (Line 5), and compute their delta and drift vectors (Lines 6-7). There are three cases in which S_i may receive a message from S_j:

- Lines 8-17: If S_i acts second and receives a message from S_j (local violation at S_j). Then S_i checks to see if a global violation has occurred. In order to accomplish this, it does not need to construct any spheres using its drift vector. Please note that S_i knows the measurement vector of S_j and also knows its own vector of measurements. Then S_i knows the exact value of the global statistics vector $\vec{e'}$ (Algorithm 2, Line 5), and can simply check whether the similarity function at the estimate vector $\vec{e}$ and $\vec{e'}$ are bi-chromatic. If so (Lines 10-12), then a global violation has occurred, and the base station will be notified by S_i.
- Lines 20-26: If S_i acts second, S_j has no local violation, but S_i detects a local violation (using Algorithm 3). Then, S_i will transmit its vector of measurements and will wait to receive the corresponding vector from S_j, in order to update its estimate vector.
- Lines 27-31: If S_i acts first, S_i detects no local violation, but S_j (which in this case acts second) detects a local violation. This case is identical to the first case that we described in this list.

Algorithm 3 Node i: checkIfViolation Subroutine

Require: estimate vector $\vec{e}$, drift vector $\vec{u}_i$, function F, threshold T

1: **if** $F(\vec{e}) > T$ **then**
2: Find min value testVal in sphere B($\vec{e}$, $\vec{u}_i$)
3: **else**
4: Find max value testVal in sphere B($\vec{e}$, $\vec{u}_i$)
5: **end if**
6: **if** $F(\vec{e})$ and testVal are monochromatic **then**
7: Return false
8: **end if**
9: Return true

The Autobalance Mode of Operation. In the Simple mode operation either both nodes will not detect any local violation and will remain silent, or they will both transmit their local measurement vectors to the other node. The latter is a significant drawback of the Simple mode, which we aim to improve upon with our Autobalance mode.

In order to ensure accurate monitoring, the geometric approach requires that (i) both nodes always use the same estimate vector, and (ii) the average of the drift vectors is equal to the global statistic vector (i.e., the true average of the local statistics vectors).

In the Autobalance mode, the first node (in the pair) that detects a local violation:

- Modifies its estimate vector to $\vec{e} + \frac{1}{2}\Delta\vec{v}$
- Transmits its measurement vector (after the transmission, $\Delta\vec{v} = 0$).
- Awaits for a potential message from the other node. If a message is received, it updates its estimate vector by incorporating *half* of the delta vector that it calculates for the pairing node.

On the other hand, the node that first receives a message, uses the following steps:

- Checks if the new global estimate vector is monochromatic with the estimate vector.
- Updates (note: the update must be done after the previous check) its estimate vector by incorporating *half* of the delta

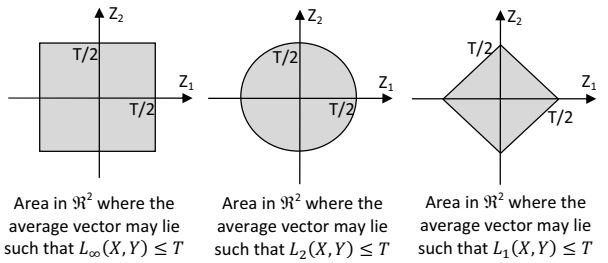

Fig. 3. Depicting the shaded areas where the global statistic vector may lie such that the L_∞, L_2 and L_1 distance between the two sensor nodes is not above the threshold T, for 2-dimensional vectors of measurements. The shaded areas are also convex in any dimension for these similarity measures.

vector that it calculates for the pairing node.

- If the first step signaled a global violation, the node notifies the base station, sets the estimate vector to the global statistics vector that it computed, and transmits its measurement vector to the pairing node. Otherwise, it continues using the estimate vector that it computed in the second step.

The above procedure is similar in principle to the techniques developed in [22], where a slack vector may be added to all the computed drift vectors, with the requirement that the slack vectors cancel out. It is simple to verify that our procedure distributes the delta vector of the node with the violation to both nodes (both nodes modify their estimate vector and, thus, their drift vector by half of this delta, while the transmitting node also zeroes its delta vector), thus resulting in zero overall change in the computation. To understand why this is the case, assume that S_i detected the local violation. Then the overall change in the drift vector of S_i is, $-\frac{1}{2}\Delta\vec{v}_i$ (the node zeroes its delta vector after the transmission, but half of that vector has been added to the estimate vector), while the overall change in the drift vector of S_j is $\frac{1}{2}\Delta\vec{v}_i$ (due to the modification of the estimate vector). Thus, the sum of the drift vectors before and after the autobalance operation remains the same.

In the Autobalance mode, the node that first receives a message during an epoch does not necessarily transmits its own measurement vector, unless it detects a global violation. We, thus, expect the Autobalance mode to significantly reduce the number of transmitted messages. However, since only the second message may be pruned in each epoch, this reduction may not exceed 50%, when compared to the Simple mode.

Exploiting the Convexity of Safe Zones. For the L_∞, L_1 and L_2 similarity functions, we may further reduce the number of transmitted messages, by exploiting the notions of Safe Zones (SZ) [14] and the convexity of the Safe Zones for the aforementioned similarity functions.

According to [14], "a node's SZ consists of the set of vectors which satisfy the local constraints, and as long as the vectors remain in their SZs, no communication is required". Based on the above definition, for the L_∞, L_1 and L_2 similarity functions we can define the gray safe zones that are depicted, for $W = 2$, in Figure 3. In higher dimensions, the corresponding safe zones are a hypercube (for L_∞), a sphere (for L_2) and the intersection of hyperplanes (for L_1). We note that the safe zones that we depict are convex. Thus, as long as the estimate vector and the drift vectors remain within the safe zones, then the global statistics vector (which is the average of the drift vectors) will also lie inside the safe zone. Exploiting this fact, we may modify Algorithm 3 in the following way: If $f(\vec{e}) \leq T$, then simply check if $f(\vec{u}) \leq T$ (i.e., do not find the maximum value within a sphere) in order to test whether a local violation exists. In the case where $f(\vec{e}) > T$, the existing technique depicted in Lines 1-2 of Algorithm 3 is followed. We denote as Convex the variation of the Simple mode that exploits the convexity of safe zones for L_∞, L_1 and L_2. We denote as *Convex Autobalance* the corresponding variation of the Autobalance mode.

B. Neighbors in Direct Communication.

In the case where all the neighbors of S_i are in direct communication with S_i, then any message transmission by S_i, regarding its measurements vector, can be performed using broadcast communication. Due to space constraints, we do not enumerate all possible combinations of the broadcast communication with the Simple, Convex, Autobalance and Convex Autobalance modes, but rather focus on the most efficient one, namely the *Proactive Broadcast Convex* mode. This mode is based on the *Convex Autobalance* mode.

In this mode, the nodes within each CN must be ordered, so that each node knows who acts before it, or not. Let us consider the operation of node S_i:

- S_i waits for messages from nodes that act before it.
- For each received message, S_i updates its estimate vector (due to autobalancing) with the corresponding neighbor, and checks if a global violation regarding that neighbor occurs. If yes, then S_i will later broadcast its measurement vector, so we skip the next step.
- S_i examines if a local violation exists regarding itself and nodes that act after S_i. S_i marks all such nodes with which it observes a local violation.
- If either of the steps 2,3 necessitate a transmission, S_i broadcasts its measurement vector. S_i will update (due to autobalancing) its estimate vector for all nodes: (i) that act after it, and (ii) for which a global violation was detected in Step 1. Please note that in this case S_i proactively modifies its estimate vector for nodes that act after it, even if it does not detect a local violation for their similarity.
- S_i then awaits messages regarding potential violations. It ignores received messages from any node S_j that act after S_i and which did not have a global violation with S_i (i.e., the received message was due to a violation that S_j had with another node). The latter is easy for S_i to determine, based on the received measurements vector from S_j.

C. Handling Minimum Support Queries

In the case of queries involving a required minimum support (i.e., number of nodes with similar measurements), the base station wants to be notified about any sensor node that did not

have (did have) the required minimum support at the previous epoch, but reaches (drops below) the required minimum support at the current epoch. However, this problem is trivial to handle in our framework, since each node S_i will only transmit a message to the base station at the end of the epoch (i.e., after the monitoring process for all neighbors has been completed for this epoch) if its current support follows the previous condition. Please recall that each node knows with absolute certainty whether its desired distance from any node in its CN is above/below the required threshold.

We actually expect minimum support queries to be more bandwidth efficient, since our initial algorithms notified the base station each time that their similarity status with any node in CN was modified. However, as we demonstrate in our experimental evaluation, the overall bandwidth savings are restricted by the fact that the vast majority of the transmitted messages are between pairs of nodes tested for similarity. Thus, even though the transmissions to the base station are reduced, these transmissions were relatively few, to begin with.

D. A Note on Message Losses

All the presented algorithms assume, similarly to previous work that uses the geometric approach, the reliable delivery of messages. However, in sensor network applications messages are often lost due to conflicts. We now discuss the consequences of message loss when using the geometric approach.

In case a message involving the notification to the base station is lost, the base station is not aware whether two nodes are similar or not. In the case of minimum support queries, the base station will not know whether a node's measurements have reached the required minimum support or not. This issue will be resolved in the next notification transmitted by the same node to the base station.

In versions of our algorithms that do not include the Autobalance mode, each node that first makes a transmission due to a local violation expects a reply. Thus, if a message is lost, the node will not receive a reply, will recognize this message loss and may resend its message, thus resolving this issue. On the other hand, in versions of our algorithms that include the Autobalance mode, it is possible that a node that exhibits a local violation and transmits a message will not receive a reply. In such a case, the node will not know whether this occurred because of a message loss, or because of a successful balancing operation at the pairing node (unless we use acknowledgments). In this case, the 100% guarantees of our algorithms are no longer valid, as the two corresponding pairing nodes will end up using different estimate vectors. Let us now consider when this problem will be resolved. Let X denote the node in the pair whose transmitted message was not received. The problem will only be resolved in either (i) the next epoch that X exhibits a local violation (when monitoring the same pair-wise similarity), if the message of X is not lost again, or (ii) the next epoch that the pairing node first detects a local violation and X detects a threshold violation, as X will then transmit its measurements vector to its pairing node.

E. Handling the Addition and Removal of Nodes

When nodes are added/removed from the network (for example, due to node failures), then each sensor simply needs to monitor its similarity to more/fewer sensor nodes. In most cases, this does not require any additional effort by the remaining sensor nodes. In the case of minimum support queries, the removal of sensor nodes may cause some nodes to notify the base station, as their minimum support may drop below the required threshold. The only minor complication that arises involves the broadcast case described in VI-B: any new node added in the network, after determining its comparison neighborhood CN_i, will have to pick/determine its order (when to act) within CN_i, so that it knows which other sensors act before/after it.

VII. Experiments

In our experiments we utilized two real world data sets. The first data set, termed Intel Lab Data, includes temperature, humidity and light measurements collected by motes in the Intel Research, Berkeley Lab [2]. We selected the measurements of the following nodes (in the specified order) that lie in nearby lab locations: 38, 39, 40, 41, 43, 37, 35, 36. In experiments where we vary the number of used nodes, any experiment containing K nodes, contains measurements from the K first nodes in the above list, for 30000 epochs. The second data set, termed as Weather Data, includes air temperature, relative humidity and solar irradiance measurements from the station in the University of Washington and for the year 2002[3]. We used these measurements to generate readings for up to 9 motes for a period of 2000 epochs. To avoid presenting experiments for the same quantity in both data sets, in the Intel Lab data set we present the results for the temperature and light data, while in the Weather data we present the results for the humidity and the solar irradiance data. Our simulator was written in Java.

Techniques and Parameter Settings. We compare the performance of our Simple, Convex, Autobalance, Convex Autobalance and Proactive Broadcast Convex techniques in different data sets, when we vary several parameters, such as the used threshold, the similarity function, the minimum support, the dimensionality of the measurements vector, or the number of the sensor nodes. In order to be able to test the Convex alternatives, we focused on the L_∞, L_1 and L_2 similarity functions. Moreover, these functions have a closed form solution that helps us determine in a simple way their minimum and maximum values within any sphere.

We also consider in our discussion (but do not depict in our graphs due to its high bandwidth consumption compared to our techniques), a baseline method, termed NAIVE, in which the sensor nodes transmit their measurement vectors to the base station at each epoch. To favor this NAIVE algorithm, we generated clusters of k sensor nodes (the default value of k was 5), all in direct communication to each other and to the base station. Even though the competitive algorithm

[2]Data available at: http://db.csail.mit.edu/labdata/labdata.html

[3]Data available at: http://wwwk12.atmos.washington.edu/k12/grayskies

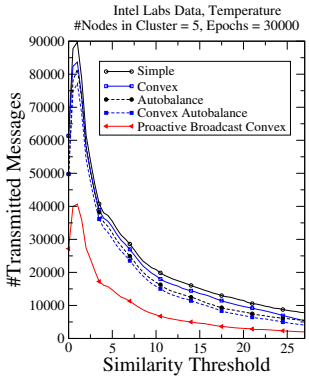

Fig. 4. Intel Lab: #Messages vs Threshold, Temperature, L_1

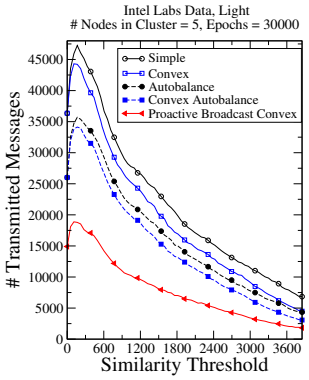

Fig. 5. Intel Lab: #Messages vs Threshold, Light, L_1

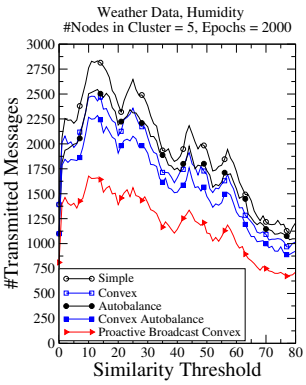

Fig. 6. Weather: #Messages vs Threshold, Humidity, L_1

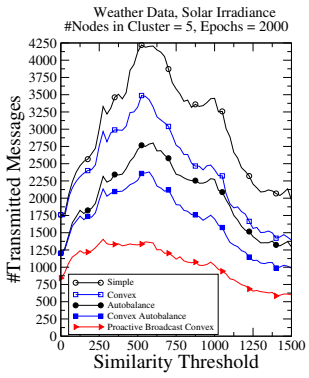

Fig. 7. Weather: #Messages vs Threshold, Solar Irradiance, L_1

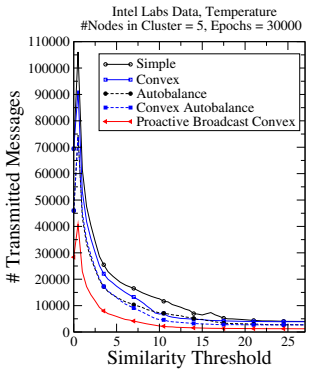

Fig. 8. Intel Lab: #Messages vs Threshold, Temperature, L_∞

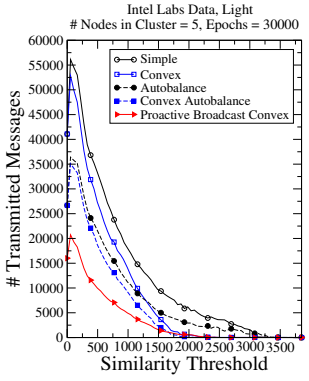

Fig. 9. Intel Lab: #Messages vs Threshold, Light, L_∞

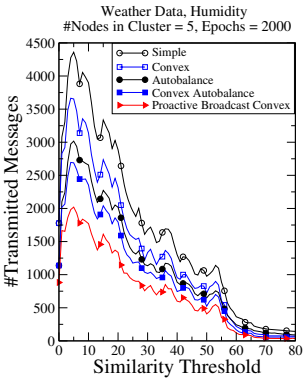

Fig. 10. Weather: #Messages vs Threshold, Humidity, L_∞

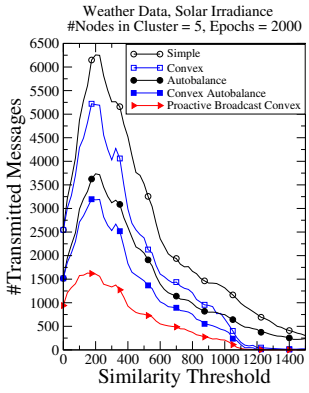

Fig. 11. Weather: #Messages vs Threshold, Solar Irradiance, L_∞

is termed as NAIVE, the setup that was provided favors it extremely, since at each epoch each node transmits its measurements vector over a single hop. Please also note that, as mentioned in Section II, recent approaches [6], [8] require each node to perform a transmission at each epoch and, thus, cannot perform better than our baseline NAIVE algorithm. Contrary to the NAIVE algorithm, our algorithms transmit to the base station only when a global violation is detected. Thus, our algorithms, contrary to NAIVE, would not be severely impacted in cases where the base station is several hops away from the sensor nodes. Table II depicts the default values used for our parameters.

A. *Varying the Monitored Function*

In Figures 4-7 we demonstrate the total number of transmitted messages of our techniques in our data sets when varying the used threshold, with the L_1 similarity function. The cluster contained 5 nodes, which means that the total number of transmitted messages for NAIVE (not depicted) was 150,000 in the Intel Lab data, and 10,000 in the Weather data. We make two interesting observations:

• Our techniques can provide significant bandwidth savings compared to NAIVE. For example, by using a modest 5-degree threshold for L_1 in Figure 4, the Simple mode reduces the number of messages by a factor of 3.3 (compared to NAIVE), while our Broadcast mode achieves about a 10 fold reduction. When considering these improvements, recall that we have selected a setup that significantly favors the NAIVE algorithm.

• In the controlled environment of the Intel Lab, the maximum number of transmitted messages occurs in very small threshold values, which implies small differences on the sensor measurement vectors. This is encouraging, as it seems less likely that one would pick such low threshold values in order to test whether the readings of some sensors are "abnormal". On the other hand, the Weather data set contains measurements from outdoor sensors, thus resulting in significant differences between their measurements. This explains why the maximum number of transmitted messages in the Weather data set is at higher threshold values. Please note that using the geometric approach, a node is less likely to have a local violation if its estimate point is far away from the threshold surface, inde-

TABLE II
PARAMETERS AND DEFAULT VALUES

Parameter	Default Value
k: cluster size	5
W: vector dimensionality	3
minSupp: minimum Support	unspecified
#epochs	Intel Lab: 30000, Weather:2000

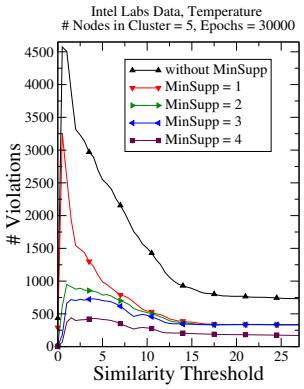

Fig. 12. Intel Lab: #Violations vs Threshold, varying minimum support, L_2, Proactive Broadcast Convex

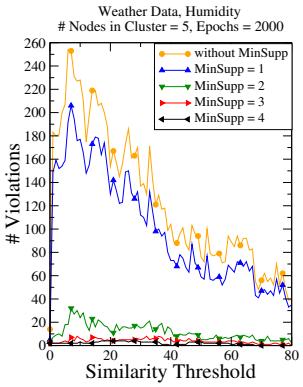

Fig. 13. Weather: #Violations vs Threshold, varying minimum support, L_2, Proactive Broadcast Convex

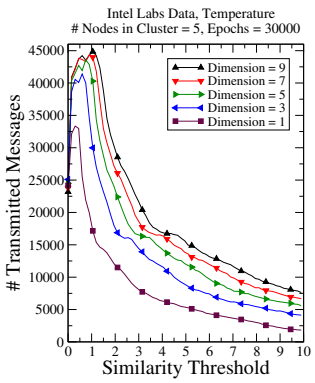

Fig. 14. Intel Lab: #Messages vs Threshold, varying Dimension, L_2, Proactive Broadcast Convex

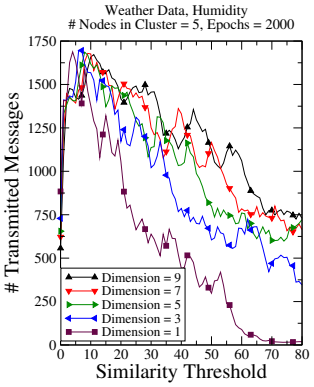

Fig. 15. Number of Messages vs Threshold, varying Dimension, L_2, Proactive Broadcast Convex

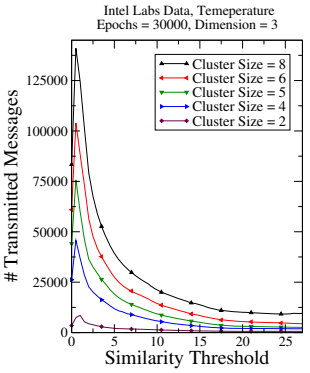

Fig. 16. Number of Messages vs Threshold, for different Cluster Size, L_2 Convex Autobalance

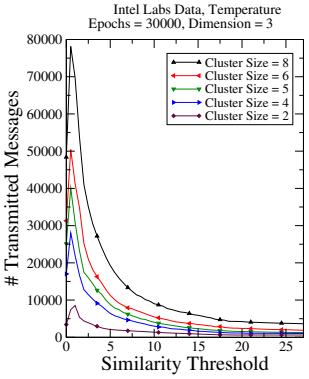

Fig. 17. Number of Messages vs Threshold, for different Cluster Size, L_2 Proactive Broadcast Convex

pendently on which side of the surface (above, or below the threshold) it belongs to. Thus, with very low (high) threshold values, nodes often construct monochromatic spheres with values entirely above (below) the desired threshold, which, in turn, results in fewer local violations.

In Figures 8-11 we depict the performance (for the same data sets) when using the L_∞ similarity function. The performance is similar to the L_1 case, with L_∞, on one hand, generating more messages in very small threshold values but, on the other hand, exhibiting a very rapid decrease in the transmitted messages as the threshold increases.

B. Minimum Support Queries

We now investigate whether the benefits of our algorithms improve even more when we specify a desired minimum support value. In Section VI-C we argued why this may be the case. In Figures 12 and 13 we depict the number of global violations (please note the difference, compared to all the other depicted figures) for our Proactive Broadcast Convex mode of operation and for the Intel Lab/Temperature and Weather/Humidity data, respectively. We make three important observations:

- The number of global violations in the modes of operation that are not depicted are similar, for the same values of the threshold and the minimum support. Since these alternative modes of operation modify their estimate vectors in different ways (i.e., more or less aggressively), the number of global violations cannot be exactly the same. Thus, since the number of violations is about the same in all modes of operation, the benefits of the techniques (compared to each other) stem from reducing the communication within each pair of nodes.
- The number of global violations is only a small fraction of the total number of transmitted messages. Thus, our techniques are less sensitive than NAIVE when the base station is placed further away from the cluster.
- Specifying a minimum support reduces the number of violations in all cases. We need to note that the pair-wise communication is not impacted (only the notification of the base stations is impacted), which implies that the overall decrease in the transmitted messages is modest.

C. Varying the Dimensionality W

In Figures 14 and 15 we depict the number of transmitted messages for our Proactive Broadcast Convex mode of operation under the L_2 distance, and for the Intel Lab/Temperature and Weather/Humidity data, respectively, when varying the dimensionality of the measurement vector. In this case, the L_2 function accumulates the sum of more terms. It is, thus, expected that the shape of the graphs will gradually shift towards higher threshold values (for example, assuming the same absolute vector elements in all dimensions, the L_2 function increases with the number of dimensions).

D. Varying the Cluster Size

We now investigate how our techniques are influenced when increasing the cluster size. For a cluster with k nodes, $\binom{k}{2}$ similarity relations need to be monitored. Thus, we expect an increase in the number of transmitted messages as the size of the cluster increases. Please note that the NAIVE technique scales linearly with the cluster size (i.e., for a cluster of 8 nodes in the Intel Lab data, $240,000$ messages

are required). Figures 16-17 depict the scalability for the Intel Labs/Temperature data, and for our Convex Autobalance and Proactive Broadcast Convex modes of operation. Even when increasing the cluster size, our techniques maintain important savings over the NAIVE approach, with the Proactive Broadcast Convex mode being able to scale better.

VIII. Conclusions

In this paper we proposed a novel framework for outlier detection in sensor networks, based on the geometric approach. We demonstrated that several common similarity functions used for outlier detection can be transformed and expressed in a way that allows the applicability of the geometric approach. We then proposed a general framework for outlier detection and suggested multiple modes of operation for the sensor nodes. Appropriate optimizations, such as the Autobalance mode, the exploitation of the convexity of safe zones, and the potential for broadcast communication were also discussed and evaluated. Furthermore, our framework can easily be used for minimum support queries. Perhaps most importantly, our framework allows each sensor node to accurately monitor its similarity to other nodes, while resulting in bandwidth consumption that is merely a fraction of the communication cost that a centralized approach would require.

For our future work, we will investigate how additional similarity functions can be incorporated in our framework. Moreover, for some similarity functions it may be hard for the sensor node to accurately find the maximum/minimum value within the local sphere that it constructs. In such cases, examining the value of the function over a set of points belonging to a grid (which lies within the sphere) is a feasible alternative, but may not always result in perfect accuracy. Perhaps, a more promising direction involves asking the base station (which may have additional computational resources) to compute, when possible, an appropriate convex safe zone, which the sensors will then utilize. We plan to explore these directions in the future.

Acknowledgment

We would like to thank Daniel Keren, Izchak Sharfman, Assaf Schuster, Minos Garofalakis, Vasilis Samoladas and Nikos Giatrakos for their helpful discussions and suggestions.

References

[1] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating Aggregates on a Peer-to-Peer Network. Technical report, Stanford, 2003.
[2] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *ICDCS* , 2006.
[3] M. Chatterjee, S. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. *Journal of Cluster Computing(Special Issue on Mobile Ad hoc Networks)*, 5, 2002.
[4] J. Chen, S. Kher, and A. Somani. Distributed Fault Detection of Wireless Sensor Networks. In *DIWANS*, 2006.
[5] J. Considine, M. Hadjieleftheriou, F. Li, J. Byers, and G. Kollios. Robust approximate aggregation in sensor data management systems. *ACM Trans. Database Syst.*, 34(1):1–35, 2009.
[6] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE*, 2009.
[7] E. Elnahrawy and B.Nath. Cleaning and querying noisy sensors. In *WSNA*, 2003.
[8] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. TACO: Tunable Approximate Computation of Outliers in Wireless Sensor Networks. In *SIGMOD*, 2010.
[9] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *HICSS*, 2000.
[10] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative Support for Sensor Data Cleaning. In *Pervasive*, 2006.
[11] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. A pipelined framework for online cleaning of sensor data streams. In *ICDE*, 2006.
[12] S. Jeffery, M. Garofalakis, and M. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, 2006.
[13] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *FOCS*, 2003.
[14] D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape sensitive geometric monitoring. *IEEE Transactions on Knowledge and Data Engineering (to appear)*.
[15] N. Khoussainova, M. Balazinska, and D. Suciu. Towards Correcting Input Data Errors Probabilistically using Integrity Constraints. In *MobiDE*, 2006.
[16] Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. In *ICDE*, 2005.
[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for ad hoc Sensor Networks. In *OSDI*, 2002.
[18] S. Meng, K. Xie, G. Chen, X. Ma, and G. Song. A kalman filter based approach for outlier detection in sensor networks. In *CSSE* , 2008.
[19] M. Otey, A. Ghoting, and S. Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Min. Knowl. Discov.*, 12(2-3), 2006.
[20] M. Qin and R. Zimmermann. VCA: An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks. *J.UCS*, 13(1), 2007.
[21] G. Sagy, D. Keren, I. Sharfman, and A. Schuster. Distributed threshold querying of general functions by a difference of monotonic representation. *Proc. VLDB Endow.*, 4, November 2010.
[22] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD*, 2006.
[23] I. Sharfman, A. Schuster, and D. Keren. Aggregate threshold queries in sensor networks. In *IPDPS*, 2007.
[24] I. Sharfman, A. Schuster, and D. Keren. Shape sensitive geometric monitoring. In *PODS*, 2008.
[25] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *MobiHoc*, 2007.
[26] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware Routing in Mobile Ad Hoc Networks. In *MobiCom*, 1998.
[27] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, 2006.
[28] Y.-J. Wen, A. M. Agogino, and K.Goebel. Fuzzy Validation and Fusion for Wireless Sensor Networks. In *ASME*, 2004.
[29] X. Xiao, W. Peng, C. Hung, and W. Lee. Using SensorRanks for In-Network Detection of Faulty Readings in Wireless Sensor Networks. In *MobiDE*, 2007.
[30] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3), 2002.
[31] O. Younis and S. Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. In *INFOCOM*, 2004.
[32] D. Zeinalipour, P. Andreou, P. Chrysanthis, G. Samaras, and A. Pitsillides. The Micropulse Framework for Adaptive Waking Windows in Sensor Networks. In *MDM*, 2007.
[33] K. Zhang, S. Shi, H. Gao, and J. Li. Unsupervised outlier detection in sensor networks using aggregation tree. In *ADMA* , 2007.
[34] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *International Journal of IEEE Communications Surveys and Tutorials*, 12(2), 2010.
[35] Y. Zhuang and L. Chen. In-network Outlier Cleaning for Data Collection in Sensor Networks. In *Clean DB Workshop*, 2006.
[36] Y. Zhuang, L. Chen, S. Wang, and J. Lian. A Weighted Moving Average-based Approach for Cleaning Sensor Data. In *ICDCS*, 2007.