In-Network Approximate Computation of Outliers with Quality Guarantees

Nikos Giatrakos[a,*], Yannis Kotidis[b], Antonios Deligiannakis[c], Vasilis Vassalos[b], Yannis Theodoridis[a]

[a]*Department of Informatics, University of Piraeus, Central Building, 80 Karaoli & Dimitriou St., GR-18534 Piraeus, Greece*
[b]*Department of Informatics, Athens University of Economics and Business, 76 Patission St., GR-10434 Athens, Greece*
[c]*Department of Electronic and Computer Engineering, Technical University of Crete, University Campus., GR-73100 Chania, Greece*

Abstract

Wireless sensor networks are becoming increasingly popular for a variety of applications. Users are frequently faced with the surprising discovery that readings produced by the sensing elements of their motes are often contaminated with outliers. Outlier readings can severely affect applications that rely on timely and reliable sensory data in order to provide the desired functionality. As a consequence, there is a recent trend to explore how techniques that identify outlier values based on their similarity to other readings in the network can be applied to sensory data cleaning. Unfortunately, most of these approaches incur an overwhelming communication overhead, which limits their practicality. In this paper we introduce an in-network outlier detection framework, based on locality sensitive hashing, extended with a novel boosting process as well as efficient load balancing and comparison pruning mechanisms. Our method trades off bandwidth for accuracy in a straightforward manner and supports many intuitive similarity metrics. Our experiments demonstrate that our framework can reliably identify outlier readings using a fraction of the bandwidth and energy that would otherwise be required.

Keywords: Sensor Network, Outlier, Locality Sensitive Hashing, Similarity

1. Introduction

Pervasive applications are increasingly supported by networked sensory devices that interact with people and themselves in order to provide the desired services and functionality. Because of the unattended nature of many applications and the inexpensive hardware used in the construction of the sensors, sensor nodes often generate imprecise individual readings due to interference or failures [1]. Sensors are also often exposed to severe conditions that adversely affect their sensing elements, thus yielding readings of low quality. For example, the humidity sensor on the popular MICA mote is very sensitive to rain drops [2].

The development of a flexible layer that will be able to detect and flag outlier readings, so that proper actions can be taken, constitutes a challenging task. Conventional outlier detection algorithms [3, 4] are not suited for our distributed, resource-constrained environment of study. First, due to the limited memory capabilities of sensor nodes, in most sensor network applications, data is continuously collected by motes and maintained in memory for a limited amount of time. Moreover, due to the frequent change of the data distribution, results need to be generated continuously and computed based on recently collected measurements. Furthermore, a central collection of sensor data is not feasible nor desired, since it results in high energy drain, due to the large amounts of transmitted data. Hence, what is required are continuous, distributed and in-network approaches that reduce the communication cost and manage to prolong the network lifetime.

One can provide several definitions of what constitutes an outlier, depending on the application. For example in [5], an outlier is defined as an observation that is sufficiently far from most other observations in the data set. However, such a definition is inappropriate for physical measurements (like noise or temperature) whose absolute

*Corresponding author. Tel.: (+30) 210 4142449; fax: (+30) 210 4142264
Email addresses: ngiatrak@unipi.gr (Nikos Giatrakos), kotidis@aueb.gr (Yannis Kotidis), adeli@softnet.tuc.gr (Antonios Deligiannakis), vassalos@aueb.gr (Vasilis Vassalos), ytheod@unipi.gr (Yannis Theodoridis)

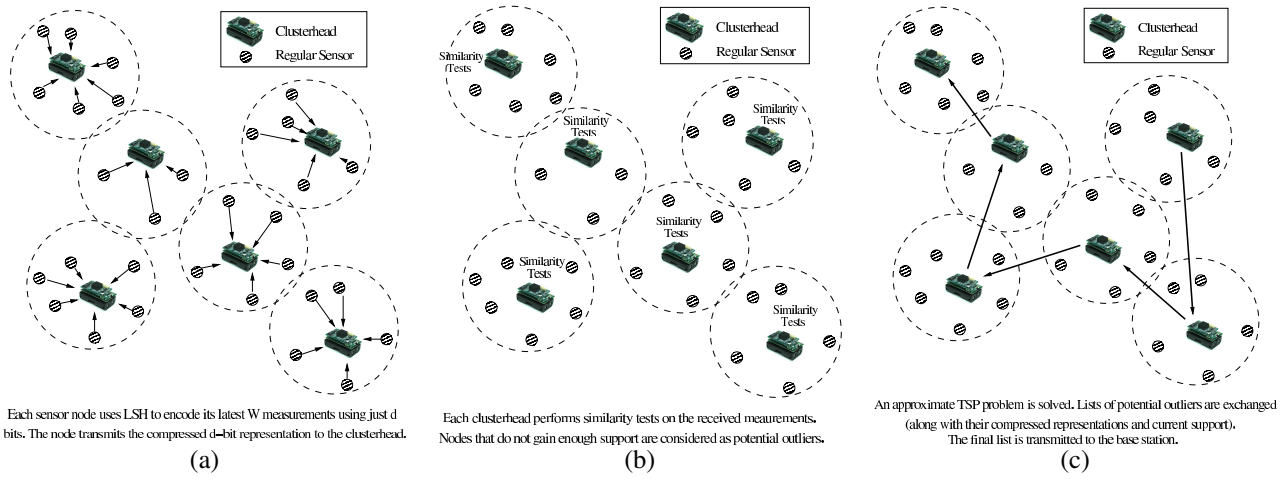


Figure 1: Main Stages of the TACO Framework

values depend on the distance of the sensor from the source of the event that triggers the measurements. Moreover, in many applications, one cannot reliably infer whether a reading should be classified as an outlier without considering the recent history of values obtained by the nodes. Thus, in our framework we propose a more general method that detects outlier readings taking into account the recent measurements of a node, as well as spatial correlations with measurements of other nodes.

Similar to recent proposals for processing declarative queries in wireless sensor networks, our techniques employ an *in-network processing* paradigm that fuses individual sensor readings as they are transmitted towards a *base station*. This fusion, dramatically reduces the communication cost, often by orders of magnitude, resulting in prolonged network lifetime. While such an in-network paradigm is also used in proposed methods that address the issue of data cleaning of sensor readings by identifying and, possibly, removing outliers [6, 2, 1, 7], none of these existing techniques provides a straightforward mechanism for controlling the burden of the nodes that are assigned to the task of outlier detection.

An important observation that we make in this paper is that existing in-network processing techniques cannot reduce the volume of data transmitted in the network to a satisfactory level and lack the ability of tuning the resulting overhead according to the application needs and the accuracy levels required for outlier detection. Please note that it is desirable to reduce the amount of transmitted data in order to also significantly reduce the energy drain of sensor nodes. This occurs not only because radio operation is by far the biggest culprit in energy drain [8], but also because fewer data transmissions also result in fewer collisions and, thus, fewer re-transmissions by the sensor nodes.

In this paper we present a novel outlier detection scheme termed TACO (TACO stands for Tunable Approximate Computation of Outliers). TACO [9] adopts two levels of hashing mechanisms. The first is based on locality sensitive hashing (LSH) [10], which is a powerful method for dimensionality reduction [10, 11, 12]. We first utilize LSH in order to encode the latest W measurements collected by each sensor node as a bitmap of $d \ll W$ bits. This encoding is performed locally at each node. The encoding that we utilize trades accuracy (i.e., probability of correctly determining whether a node is an outlier or not) for bandwidth, by simply varying the desired level of dimensionality reduction and provides tunable accuracy guarantees based on the d parameter mentioned above. Assuming a clustered network organization [13, 14, 15, 16], motes communicate their bitmaps to their clusterhead, which can estimate the similarity amongst the latest values of any pair of sensors within its cluster by comparing their bitmaps, and for a variety of similarity metrics that are useful for the applications we consider. Based on the performed similarity tests, and a desired minimum support specified by the posed query, each clusterhead generates a list of *potential* outlier nodes within its cluster. At a second (inter-cluster) phase of the algorithm, this list is then communicated among the clusterheads, in order to allow potential outliers to gain support from measurements of nodes that lie within other clusters. This process is sketched in Figure 1.

The second level of hashing (omitted in Figure 1) adopted in TACO's framework comes during the intra-cluster communication phase. It is based on the hamming weight of sensor bitmaps and provides a pruning technique (re-

garding the number of performed bitmap comparisons) and a load balancing mechanism alleviating clusterheads from communication and processing overload. We choose to discuss this load balancing and comparison pruning mechanism separately, for ease of exposition, as well as to better exhibit its benefits.

The contributions of this work can be summarized as follows:

1. We present TACO, an outlier detection framework that trades bandwidth for accuracy in a straightforward manner. TACO supports various popular similarity measures used in different application areas. Examples of such measures include, but are not limited to, the cosine similarity, the correlation coefficient and the Jaccard coefficient.
2. We present an extensive theoretical study on the trade offs occurring between bandwidth and accuracy during TACO's operation.
3. We subsequently devise a boosting process that provably improves TACO's accuracy under no additional communication costs.
4. We devise novel load balancing and comparison pruning mechanisms, which alleviate clusterheads from excessive processing and communication load. These mechanisms result in a more uniform, intra-cluster power consumption and prolonged network unhindered operation, since the more evenly spread power consumption results in an infrequent need for network reorganization.
5. We present a detailed experimental analysis of our techniques for a variety of data sets and parameter settings. Our results demonstrate that our methods can reliably compute outliers, while at the same time significantly reducing the amount of transmitted data, with average recall and precision values exceeding 80% and often reaching 100%. It is important to emphasize that the above results often correspond to bandwidth consumptions that are lower than what is required by a simple continuous aggregate query, using a method like TAG [8]. We also demonstrate that TACO may result in prolonged network lifetime, up to a factor of 3 in our experiments. We further provide comparative results with the recently proposed technique of [2] that uses an equivalent outlier definition and supports common similarity measures.[1] Overall, TACO appears to be more accurate up to 10% in terms of the F-Measure metric while resulting in lower bandwidth consumption.

This paper proceeds as follows. Initially, in Section 2 we present related work, while Section 3 introduces our basic framework. In Sections 4 and 5 we analyze TACO's operation in detail. Our load balancing and comparison pruning mechanisms are described in Section 6, while Section 7 demonstrates how a variety of similarity measures can be utilized by TACO. In Section 8 we elaborate on interesting extensions to TACO that are capable of further reducing the communication cost. Section 9 presents our experimental evaluation, while Section 10 includes concluding remarks.

2. Related Work

The emergence of sensor networks as a viable and economically practical solution for monitoring and intelligent applications has prompted the research community to devote substantial effort to define and design the necessary primitives for data acquisition based on sensor networks [8, 17]. Different network organizations have been considered, such as using hierarchical routes (i.e., the aggregation tree [18, 19]), cluster formations [13, 14, 15, 16], or even completely ad-hoc formations [20, 21, 22]. Our framework assumes a clustered network organization. Such networks have been shown to be efficient in terms of energy dissipation, thus resulting in prolonged network lifetime [15, 16].

Sensor networks can be rather unreliable, as the commodity hardware used in the development of the motes is prone to environmental interference and failures. As a result, substantial effort has been devoted to the development of efficient outlier detection techniques that manage to pinpoint motes exhibiting extraordinary behavior [23]. The authors of [1, 24] introduce a declarative data cleaning mechanism over data streams produced by the sensors. Similarly, the work of [25] 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 [26] 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 [27] makes use of a weighted

[1] We emphasize that a fair comparison between two techniques that detect outliers is only possible if both compared techniques can support the same definition of what constitutes an outlier.

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 [28] while outliers are determined utilizing the Dynamic Time Warping (*DTW*) distance of neighboring motes' values. A different approach is presented in [29], where Pairwise Markov Networks are used as a tool to derive a subset of motes sufficient to infer the values obtained by the whole network. However, this technique requires an energy draining learning phase. In other related work, [30] 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 [31].

In [32], the authors discuss a framework for cleaning input data errors using integrity constraints, while in [33, 34] unsupervised outlier detection techniques are used to report the top-k values that exhibit the highest deviation in a network's global sample. Amongst these techniques, of particular interest is the technique of [33], as it is flexible with respect to the outlier definition. However, in contrast to our framework, it provides no means of directly controlling the bandwidth consumption, thus often requiring comparable bandwidth to centralized approaches for outlier detection [33].

In [35], a probabilistic technique for cleaning RFID data streams is presented. The framework of [2] is used to identify and remove outliers during the computation of aggregate and group-by queries posed to an aggregation tree [36, 8]. Its definition of what constitutes an outlier, based on the notion of minimum support and the use of recent history, is adopted in this paper by our framework. It further demonstrates that common similarity metrics such as the correlation coefficient and the Jaccard coefficient can capture the types of dirty data encountered by sensor network applications. Similarly to TACO, the PAO framework [37] operates on top of clustered network organisations and attempts to restrain communication costs during outlier identification by detecting trends on mote measurements and applying linear-regression based compression. In [5] 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 [38] where a distributed outlier detection approach for dynamic data sets is presented. However, in cases where the motes observe physical quantities (such as noise levels, temperature) the absolute values of the readings acquired depend, for example, on the distance of the mote from the cause of the monitored event (i.e., a passing car or a fire respectively). Thus, correlations among readings in space and time are more important than the absolute values, used in [38, 5].

The algorithms in [33, 2, 1, 34, 37] provide no easily tunable parameters in order to limit the bandwidth consumed while detecting and processing outliers. On the contrary, our framework has a direct way of controlling the number of bits used for encoding the values observed by the motes. While [2] takes a best effort approach for detecting possible outliers and [1] requires transferring all data to the base station in order to accurately report them, controlling the size of the encoding allows our framework to control the accuracy of the outlier detection process.

The work in [6, 7] 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 [2]. Furthermore, the framework of [7] requires a *correlation network* to be maintained, while our algorithms can be implemented on top of commonly used clustered network organizations.

In Section 8 we extend TACO by a message suppression strategy that fledges bandwidth consumption preservation. Message suppression schemes in sensor networks for continuous aggregate queries have been studied in [39, 40, 41]. Our work differs in that we do not suppress raw measurements but extracted, compact bitmap representations instead.

The Locality Sensitive Hashing (LSH) scheme used in this work was initially introduced in the rounding scheme of [42] to provide solutions to the MAX-CUT problem. Since then, LSH has been adopted in similarity estimation [10, 43], clustering [44] or indexing techniques for set value attributes [45]. Additionally, LSH has also been fostered in approximate nearest neighbor (NN) queries [12] while [46] introduces a novel hash-based indexing scheme for approximate NN retrieval that, unlike [12], can be applied to non-metric spaces as well. Eventually, the recent work of [47] extends the Random Hyperplain Projection LSH scheme [10] by automatically detecting correlations, thus computing embeddings tailored to the provided data sets.

Similarity Metric	Calculation of Similarity
Cosine Similarity	$cos(\theta(u_i, u_j)) = \frac{u_i \cdot u_j}{\|u_i\| \cdot \|u_j\|} \Rightarrow \theta(u_i, u_j) = arccos \frac{u_i \cdot u_j}{\|u_i\| \cdot \|u_j\|}$
Correlation Coefficient	$corr(u_i, u_j) = \frac{cov(u_i, u_j)}{\sigma_{u_i} \sigma_{u_j}} = \frac{E(u_i u_j) - E(u_i)E(u_j)}{\sqrt{E(u_i^2) - E^2(u_i)} \sqrt{E(u_j^2) - E^2(u_j)}} = = \frac{\sum_{\ell=1}^{W} (u_{i\ell} - E(u_i)) \cdot (u_{j\ell} - E(u_j))}{\sqrt{\sum_{\ell=1}^{W} (u_{i\ell} - E(u_i))^2} \cdot \sqrt{\sum_{\ell=1}^{W} (u_{j\ell} - E(u_j))^2}}$
Jaccard Coefficient	$J(u_i, u_j) = \frac{\|u_i \cap u_j\|}{\|u_i \cup u_j\|}$
Extended Jaccard Coefficient	$T(u_i, u_j) = \frac{u_i \cdot u_j}{\|u_i\|^2 + \|u_j\|^2 - u_i \cdot u_j}$
Euclidean Distance	$dist(u_i, u_j) = \sqrt{\sum_{\ell=1}^{W} (u_{i\ell} - u_{j\ell})^2}$

Table 1: Computation of some supported similarity metrics between vectors u_i, u_j containing the latest W measurements of nodes S_i and S_j.

3. Basic Framework

3.1. Outlier Definition

As in [2], we do not aim to compute outliers based on a mote's latest readings but, instead, take into consideration its most recent measurements. In particular let u_i denote the latest W readings obtained by node S_i. Then, given a similarity metric sim:$R^W \rightarrow [0, 1]$ and a *s*imilarity threshold Φ we consider the readings by motes S_i and S_j similar if

$$sim(u_i, u_j) > \Phi. \tag{1}$$

In TACO, we classify a mote as an outlier if its latest W measurements are not found to be similar with the corresponding measurements of at least *minSup* other motes in the network. The parameter *minSup*, thus, dictates the minimum support (either in the form of an absolute, uniform value or as a percentage of motes, i.e per cluster) that the readings of the mote need to obtain by other motes in the network, using Equation 1. By allowing the user/application to control the value of *minSup*, our techniques are resilient to environments where spurious readings originate from multiple nodes at the same epoch, due to a multitude of different, and hence unpredictable, reasons. Our framework can also easily incorporate additional *witness criteria* based on non-dynamic grouping characteristics (such as the node identifier or its location), in order to limit, for each sensor, the set of nodes that are tested for similarity with it. For example, one may not want sensor nodes located in different floors to be able to witness each other's measurements.

3.2. Supported Similarity Metrics

The definition of an outlier, as presented in Section 3.1, is quite general to accommodate a number of intuitive similarity tests between the latest W readings of a pair of sensor nodes S_i and S_j. Examples of such similarity metrics include the *cosine similarity*, the *correlation coefficient* and the *Jaccard coefficient* [10, 2]. The *Euclidean distance* and the *extended Jaccard coefficient* of standardized value vectors are supported as well. Table 1 demonstrates the formulas for computing the aforementioned metrics over the two vectors u_i, u_j containing the latest W readings of sensors S_i and S_j, respectively[2].

It is important to emphasize that our framework is not limited to using just one of the metrics presented in Table 1. On the contrary, as it will be explained in Section 4.1, any similarity metric satisfying a set of common criteria may be incorporated in our framework.

3.3. Network Organization

We adopt an underlying network structure where motes are organized into clusters (shown as dotted circles in Figure 1). Queries are propagated by the base station to the clusterheads, which, in turn, disseminate these queries to sensors within their cluster.

[2] $E(.)$, σ and $cov(.)$ in the table stand for mean, standard deviation and covariance, respectively.

Various algorithms [13, 14, 15, 16] have been proposed to clarify the details of cluster formation, as well as the clusterhead election and substitution (rotation) during the lifetime of the network. All these approaches have been shown to be efficient in terms of energy dissipation, thus resulting in prolonged network lifetime. The aforementioned algorithms differ in the way clusters and corresponding clusterheads are determined, though they all share common characteristics since they primarily base their decisions on the residual energy of the sensor nodes and their communication links.

An important aspect of our framework is that the choice of the clustering algorithm is orthogonal to our approach. Thus, any of the aforementioned algorithms can be incorporated in our framework. An additional advantage of our techniques is that they require no prior state at clusterhead nodes, thus simplifying the processes of clusterhead rotation and re-election.

3.4. Operation of the Algorithm

We now outline the various steps involved in our TACO framework. These steps are depicted in Figure 1.

Step 1: Data Encoding and Reduction. At a first step, the sensor nodes encode their latest W measurements using a bitmap of d bits. In order to understand the operation of our framework, the actual details of this encoding are not important (they are presented in Section 4). What is important is that:

- As we will demonstrate, the similarity function between the measurements of any pair of sensor nodes can be evaluated using their encoded values, rather than using their uncompressed readings.
- The used encoding trades accuracy (i.e., probability of correctly determining whether a node is an outlier or not) for bandwidth, by simply varying the desired level of dimensionality reduction (i.e., parameter d mentioned above). Larger values of d result in increased probability that similarity tests performed on the encoded representation will reach the same decision as an alternative technique that would have used the uncompressed measurements instead.

After encoding its measurements, each sensor node transmits its encoded measurements to its clusterhead.

Step 2: Outlier Detection at the Cluster Level. Each clusterhead receives the encoded measurements of the sensors within its cluster. It then performs similarity tests amongst all pairs of sensor nodes that may witness each other (please note that the posed query may have imposed restrictions on this issue), in order to determine nodes that cannot reach the desired support level and are, thus, considered to be outliers at a cluster level.

Step 3: Intercluster Communication. After processing the encoded measurements within its cluster, each clusterhead has determined a set of potential outliers, along with the support that it has computed for each of them. Some of these potential outliers may be able to receive support from sensor nodes belonging to other clusters. Thus, a communication phase is initiated where the potential outliers of each clusterhead are communicated (along with their current support) to other clusterheads in which their support may increase. Please note that depending on the restrictions of the posed queries, only a subset of the clusterheads may need to be reached. The communication problem is essentially modeled as a TSP problem, where the origin is the clusterhead itself, and the destination is the base station.

The extensible definition of an outlier in our framework enables the easy application of semantic constraints on the definition of outliers. For example, we may want to specify that only movement sensors trained on the *same location* are allowed to witness each other, or similarly that only readings from vibration sensors attached to identical engines in a machine room are comparable. Such static restrictions can be easily incorporated in our framework (i.e., by having clusterheads maintain the corresponding information, such as location and type, for each sensor id) and their evaluation is orthogonal to the techniques that we present in this paper.

4. Data Encoding and Reduction

In this section we provide the definition and properties of the Locality Sensitive Hashing schemes. We further investigate the details of a particular, the Random Hyperplain Projection, LSH scheme which serves as the basis for incorporating in TACO most of the similarity measures presented in Table 1 (the details of the latter issue are included in the current section as well as Section 7).

Symbol	Description
S_i	the $i-th$ sensor node
u_i	the value vector of node S_i
W	tumble size (length of u_i)
$\theta(u_i, u_j)$	the angle between vectors u_i, u_j
X_i	the bitmap encoding produced after applying LSH to u_i
d	bitmap length
$D_h(X_i, X_j)$	the hamming distance between bitmaps X_i, X_j
Φ	similarity threshold used
Φ_θ	threshold based on angle similarity
Φ_{D_h}	threshold based on hamming distance similarity
minSup	the minimum support parameter

Table 2: Notation used

4.1. Definition and Properties of LSH

A *Locality Sensitive Hashing scheme* is defined in [10] as a distribution on a family F of hash functions that operate on a set of objects, such that for two objects u_i, u_j:

$$P_{h\epsilon F}[h(u_i) = h(u_j)] = sim(u_i, u_j)$$

where $sim(u_i, u_j)\epsilon[0, 1]$ is some similarity measure. In [10] the following necessary properties for existence of an LSH family function for given similarity measures are proved:

Lemma 1. *For any similarity function $sim(u_i, u_j)$ that admits an LSH function family, the distance $1 - sim(u_i, u_j)$ satisfies the triangle inequality.*

Lemma 2. *Given an LSH function family F corresponding to a similarity function $sim(u_i, u_j)$, we can obtain an LSH function family F′ that maps objects to {0, 1} and corresponds to the similarity function $\frac{1+sim(u_i,u_j)}{2}$.*

Lemma 3. *For any similarity function $sim(u_i, u_j)$ that admits an LSH function family, the distance $1 - sim(u_i, u_j)$ is isometrically embeddable in the hamming cube.*

As a result, the above lemmas simultaneously summarize the conditions any candidate similarity measure should satisfy so as to be incorporated in our outlier detection framework.

4.2. Data Reduction at the Sensor Level

In our setting, TACO applies LSH to the value vectors of physical quantities sampled by motes. It can be easily deduced that LSH schemes have the property of dimensionality reduction while preserving similarity between these vectors. Dimensionality reduction can be achieved by introducing a hash function family such that (Lemmas 2,3) for any vector $u_i\epsilon R^W$ consisting of W sampled quantities, $h(u_i) : R^W \rightarrow [0, 1]^d$ with $d \ll W$.

In what follows we first describe an LSH scheme for estimating the cosine similarity between motes (please refer to Table 1 for the definition of the cosine similarity metric).

Theorem 1. *[Random Hyperplane Projection (RHP) [10, 42]]*
Assume we are given a collection of vectors defined on the W dimensional space. We choose a family of hash functions as follows: We produce a spherically symmetric random vector r of unit length from this W dimensional space. We define a hash function h_r as:

$$h_r(u_i) = \begin{cases} 1 & ,if\ r \cdot u_i \geq 0 \\ 0 & ,if\ r \cdot u_i < 0 \end{cases}$$

For any two vectors $u_i, u_j\epsilon R^W$:

$$P = P[h_r(u_i) = h_r(u_j)] = 1 - \frac{\theta(u_i, u_j)}{\pi}\square \tag{2}$$

Equation 2 can be rewritten as:

$$\theta(u_i, u_j) = \pi \cdot (1 - P) \tag{3}$$

Note that Equation 3 expresses angle similarity as the product of the potential range of the angle between the two vectors (π), with the probability of equality in the result of the hash function application (P). Thus, after repeating a stochastic procedure using d random vectors r, the final embodiment in the hamming cube results in [48]:

$$D_h(X_i, X_j) = d \cdot (1 - P) \tag{4}$$

where $X_i, X_j \epsilon [0, 1]^d$ are the bitmaps (of length d) produced and $D_h(X_i, X_j) = \sum_{\ell=1}^{d} |X_{i\ell} - X_{j\ell}|$ is their hamming distance.

Hence, we finally derive:

$$\frac{\theta(u_i, u_j)}{\pi} = \frac{D_h(X_i, X_j)}{d} \tag{5}$$

Equation 5 provides the means to compute the angle (and thus the cosine similarity) between the initial value vectors based on the hamming distance of their corresponding bitmaps. The exact details of bitmap comparison with respect to outlier detection will be provided in the next section.

Given two vectors u_i, u_j on the R^W space, and a desired similarity metric, one can deterministically compute the similarity of the two vectors. However, the transition from the continuous (R^W) space to the hamming cube ($[0, 1]^d$) results in imprecision, since the estimation of the angle similarity depends on the selection of the spherically symmetric random vectors. We now determine the number of bits required in each encoding so that the resulting scheme achieves an $(\epsilon, \delta) - approximation$ of the actual similarity, where ϵ denotes a tolerance on the distortion of $\frac{\theta(u_i,u_j)}{\pi}$ and δ is the probability with which ϵ may be exceeded.

Theorem 2. *To estimate $\frac{\theta(u_i,u_j)}{\pi}$ with precision ϵ and probability at least $1 - \delta$, sensor nodes need to produce bitmaps of $O(\ell log(2/\delta)/(2\epsilon^2))$ length.*

PROOF. Assume a pair of bitmaps X_i, X_j, produced by nodes S_i, S_j, each consisting of d bits. $(X_{i_1}, X_{j_1}), \ldots, (X_{i_d}, X_{j_d})$ denotes corresponding positions of the two bitmaps.

As already mentioned (Theorem 1), using LSH on bitmap production at the sensor level ensures that bits at the same position of X_i, X_j are designed to be equal with a probability proportional to the angle similarity of the initial vectors. Nonetheless, bits at different positions are produced using independent, random vectors r. As a result, checks at each position k during the hamming distance calculation can be considered as independent random variables Y_k which yield 1, if the corresponding bits differ, and 0 otherwise.

This yields $Y_1, \ldots, Y_d$ random variables, with $Y_i = 0$ or $Y_i = 1$. Obviously, $\sum_{i=1}^{d} \frac{Y_i}{d} = \frac{D_h(X_i,X_j)}{d}$, the expectation of which is $\frac{\theta(u_i,u_j)}{\pi}$ (Equation 5). Hoeffding's inequality [49] states that for any $\epsilon > 0$:

$$Pr[|\frac{D_h(X_i, X_j)}{d} - \frac{\theta(u_i, u_j)}{\pi}| \geq \epsilon] \leq 2e^{-2d\epsilon^2} \tag{6}$$

Let the right side of the inequality be δ ($0 < \delta < 1$). Then, one can see that the estimation $\frac{D_h(X_i,X_j)}{d}$ obtained using bitmaps X_i, X_j is beyond $\pm\epsilon$ of the initial value $\frac{\theta(u_i,u_j)}{\pi}$ with probability $\leq \delta$. Hence, $\epsilon = \sqrt{\ell n(2/\delta)/(2d)}$ and $d = \ell n(2/\delta)/(2\epsilon^2)$ which completes the proof. □

So far, we discussed the general operational aspects of our framework (Section 3). Moreover, we formally presented the preliminaries of LSH schemes and looked into the primitives for RHP application on mote values (Section 4). In the upcoming section we bind these together and primly analyze the details of the outlier identification process throughout its various stages.

5. Detecting Outliers with TACO

We now present the operation of our TACO framework in detail. As discussed in Section 1, outlying values often cannot be reliably deduced without considering the correlation of a sensor's recent measurements with those of other sensor nodes. Hereafter, we propose a generic technique that takes into account the aforementioned parameters providing an energy efficient way to detect outlying values.

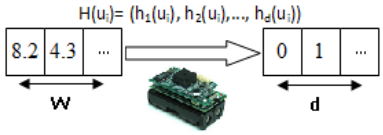


Figure 2: LSH application to mote's value vector

5.1. Running Example and Query Format

In what follows, we will use as a running example the case where

- the angle similarity between two vectors u_i and u_j is chosen; and
- the angle similarity threshold Φ_θ is set equal to Φ.

Thus, in our running example $sim(u_i, u_j) = \theta(u_i, u_j)$, and $\Phi_\theta = \Phi$. In Section 7 we demonstrate the way that other similarity measures, included in Table 1, can be utilized in TACO.

We now describe the syntax of the queries supported by TACO. Query sections enclosed in "[...]" denote optional sections that may be omitted. The meaning of each query section will be explained shortly.

Assume that the sensor network's base station poses a query of the form:

SELECT $c.S_i$
FROM Clusterheads c
WHERE $c.Support_{S_i} < minSup$
USING
TUMBLE_SIZE=W
ENCODING_BITS=d SEEDED_BY seed
TESTS(sim(), Φ)
[CHECKS_ON <specifications on sim() tests>]
[BOOSTING_GROUPS=μ]

This query is executed continuously in the network until it is explicitly terminated by the base station. The *seed* parameter in the USING compartment is encapsulated in the query so as to enable every mote in the network to produce the same set of random vectors r, which will be utilized during the LSH encoding. In addition, in the CHECKS_ON line of the query, users are able to declare a set of non dynamic specifications regarding restrictions on motes that can witness each other as discussed in Section 3.4. As an additional example, users may specify that motes within a certain radius can witness each other for similarity, irrespectively of whether they are assigned to the same cluster, as they are expected to sense similar conditions. Eventually, the BOOSTING_GROUPS specification regards the application of our proposed boosting process which will be presented in Section 5.6. As already mentioned, the CHECKS_ON and the BOOSTING_GROUPS lines (surrounded by "[...]") are optional. In case the CHECKS_ON specification is omitted, users allow comparisons of any pair of motes in the network, while the absence of BOOSTING_GROUPS declaration is equivalent to a $\mu = 1$ choice.

The rest of the query parameters have already been discussed in Section 3. The previously presented query is broadcasted to the sensor network having clusterheads disseminating the corresponding information within their cluster and towards other peers.

5.2. TACO at Individual Motes

Query reception at sensor nodes triggers a sampling procedure. Recalling Section 4.2, W recent measurements form a mote's tumble [50]. Sending a W-dimensional value vector as is, exacerbates the communication cost, which is an important factor that impacts the network lifetime. TACO thus applies LSH in order to reduce the amount of transmitted data. In particular, after having collected W values, each mote applies d $h_r()$ functions on it so as to derive a bitmap of length d (Figure 2), where the ratio of d to the size of the W measurements determines the achieved

Algorithm 1 TACO at Individual Motes

Require: Event of Query Reception
1: compute h_r Functions(W, d, *seed*)
2: {«Sampling»}
3: **repeat**
4: call getData()
5: **if** dataReady() **then**
6: $u_i \leftarrow newSample$
7: **end if**
8: **until** Tumble of W size is formed
9: compute$X_i(u_i, h_r$ functions) {X_i denotes the LSH bitmap of the node}
10: IntraclusterMessage.send(S_i,X_i)
11: {Go to «Sampling»}

Algorithm 2 TACO's Intra-cluster Processing at Clusterhead C_i

Require: Bitmap Reception from Motes in Cluster
1: **for all** pairs of motes (S_i,S_j) **do**
2: **if** $D_h(X_i, X_j) \leq \Phi_{D_h}$ **then**
3: $Support_{S_i} \leftarrow Support_{S_i} + 1$
4: $Supports_{S_j} \leftarrow Supports_{S_j} + 1$
5: **end if**
6: **end for**
7: **for all** S_is in Cluster **do**
8: **if** $Support_{S_i} < minSup$ **then**
9: $PotOut_{C_i} \leftarrow < S_i, X_i, Supports_{S_i} >$
10: **end if**
11: **end for**

reduction. The derived bitmap is then transmitted to the corresponding clusterhead. The whole process is sketched in Algorithm 1. We additionally note that in case of severe memory constraints, motes do not need to pre-compute and locally store the random vectors (Line 1 of Algorithm 1), but instead sensor nodes are capable of generating those vectors on the fly using the common seed [47].

5.3. Intra-Cluster Processing

In the next phase, each clusterhead is expected to report outlying values. To do so, it would need to compare pairs of received vectors in order to determine their similarity based on Equation 1 and the Φ_θ similarity threshold. On the contrary, the information that reaches the clusterhead is in the form of compact bitmap representations. Note that Equation 5 provides a way to express the angle similarity in terms of the hamming distance and also the similarity threshold $\Phi_{D_h} = d \cdot \frac{\Phi_\theta}{\pi}$. Thus, clusterheads can obtain an approximation of the initial similarity by examining the hamming distance between pairs of bitmaps.

Hence, after the reception of intracluster messages by motes in its cluster, every clusterhead proceeds according to Algorithm 2. If the hamming distance between two tested bitmaps is lower than or equal to Φ_{D_h}, then the two initial vectors will be considered similar, and each sensor in the pair will be able to witness the measurements of the other sensor, thus being able to increase its support by 1 (Lines 1-6). At the end of the procedure, each clusterhead determines a set of potential outliers (*PotOut*) within its cluster, and extracts a list of triplets in the form $\langle S_i, X_i, support \rangle$ containing for each outlier S_i its bitmap X_i and the current support that X_i has achieved so far (Lines 7-11).

5.4. Inter-Cluster Processing

Local outlier lists extracted by clusterheads take into account both the recent history of values and the neighborhood similarity (i.e., motes with similar measurements in the same cluster). However, this list of outliers is not final, as the posed query may have specified that a mote may also be witnessed by motes assigned to different clusterheads. Thus, an *inter-cluster communication* phase must take place, in which each clusterhead communicates information (i.e., its produced triplets) regarding its local, potential outlier motes that do not satisfy the *minSup* parameter. In addition, if the required support is different for each sensor (i.e., *minSup* is expressed as a percentage of nodes in the

Algorithm 3 TACO's Inter-cluster Processing at Clusterhead C_i

```
1: while more motes in PotOut_{C_i} do
2:     InterclusterMessage.send(<S_i, X_i, Support_{S_i}> ∈ PotOut_{C_i})
       {towards the next TSP hop}
3: end while
   {Intercluster Message Reception Handling}
1: if InterclusterMessage.receive(<S_i, X_i, Support_{S_i}>) then
2:     for all S_j s in the current cluster do
3:         if D_h(X_i, X_j) ≤ Φ_{D_h} then
4:             Support_{S_i} ← Support_{S_i} + 1
5:         end if
6:     end for
7:     if Support_{S_i} < minSup then
8:         InterclusterMessage.send(<S_i, X_i, Support_{S_i}>)
           {towards the next TSP hop}
9:     end if
10: end if
```

cluster), then the desired $minSup$ parameter for each potential outlier also needs to be transmitted. Please note that the number of potential outliers is expected to only be a small portion of the total motes participating in a cluster.

During the intercluster communication phase (sketched in Algorithm 3) each clusterhead thus transmits its potential outliers to those clusterheads where its locally determined outliers may increase their support (based on the restrictions of the posed query - first three lines of Algorithm 3). This is achieved by using a circular network path computed by solving a TSP problem that has as origin the clusterhead, as endpoint the base station, and as intermediate nodes those sensors that may help increase the support of this clusterhead's potential outliers. The TSP can be computed either by the basestation after clusterhead election, or in an approximate way by imposing GPSR [51] to aid clusterheads make locally optimal routing decisions. However, note that such a greedy algorithm for TSP may result in the worst route for certain point - clusterhead distributions [52].

Any item in the $PotOut_{C_i}$ set of potential outliers of clusterhead C_i received by a clusterhead C_j is compared to local sensor bitmaps and the support parameter of nodes within $PotOut_{C_i}$ is increased appropriately upon a similarity occurrence (Lines 1-6 in Intercluster Message Reception Handling of Algorithm 3). In this phase, upon a successful similarity test, we do not increase the support of motes within the current cluster (i.e., the cluster of C_j), since at the same time the potential outliers produced by C_j have been correspondingly forwarded to neighboring clusters in search of additional support. Any potential outlier that reaches the desired *minSup* support is excluded from the list of potential outliers that will be forwarded to the next clusterhead (Lines 7-9 of Intercluster Message Reception Handling).

Eventually, motes that do not manage to accumulate adequate witnesses are identified as outliers. The corresponding information that reaches the base station comes in the form of the S_i of each node that shows abnormal behavior. Nevertheless, the base station may additionally require inspection of the sampled values obtained by these motes. An efficient way to derive estimations of those values is by having clusterheads forwarding the bitmap X_i, on par with the identifier of the nodes, to the query source. Subsequently, sampled values' estimations can be extracted by applying a reverse LSH process [47]. Of course, a need for exact mote sample acquisition introduces an additional step of directly querying the pinpointed outlier sensors.

5.5. Analysis

The similarity tests that take place during TACO's intra- and intercluster processing are approximate in nature, as their decisions rely on hamming distance tests on pairs of mote bitmaps, instead of the angle similarity of initial mote vectors. In this section, we elaborate on the quality guarantees provided by TACO with respect to the aforementioned similarity estimations for the given Φ_θ threshold. We initially discuss some intuition on these quality guarantees and present graphical analysis results, when different parameters are varied. We then provide an analysis based on the use of Chernoff bounds.

We first demonstrate how one could estimate the expected error rate of the performed checks. Recall that for any pair of vectors u_i, u_j, the probability P that the corresponding bits in their bitmap encoding are equal is given

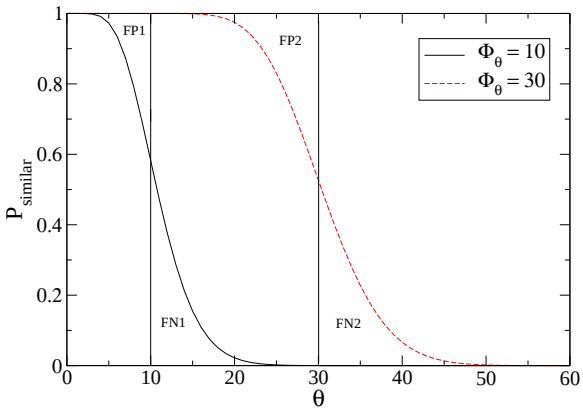


Figure 3: Probability $P_{similar}$ of judging two bitmaps as similar, depending on the angle (θ) of the initial vectors and for two different thresholds Φ_θ (W=16, reduction ratio=1/4).

by Equation 2. Thus, the probability of satisfying the similarity test, via LSH manipulation can be expressed by the cumulative function of a binomial distribution:

$$P_{similar} = \sum_{i=0}^{\Phi_{D_h}} \binom{d}{i} P^{d-i} \cdot (1-P)^i \tag{7}$$

As an example, Figure 3 plots the value of $P_{similar}$ as a function of $\theta(u_i, u_j)$ (recall that P is a function of the angle between the vectors) for two different values of Φ_θ. The area FP_1 above the line on the left denotes the probability of classifying any two vectors as dissimilar, even though their *theta* angle is less than the threshold (false positive). Similarly, the area FN_1 denotes the probability of classifying the encodings as similar, when the corresponding initial vectors are not (false negative). The areas denoted as FP_2 and FN_2 correspond to the same probabilities for an increased value of Φ_θ. We can observe that the method is more accurate (i.e., leads to smaller areas for false positive and negative detections) for more strict definitions of an outlier implied by smaller Φ_θ thresholds. In Figure 4 we depict the probability that TACO correctly identifies two similar ($\theta = 5, \Phi_\theta = 10$) vectors as similar, varying the length d of the bitmap. As expected, using more LSH hashing functions (i.e., choosing a higher value of d), increases the probability of resulting in a successful test.

We can, therefore, estimate the expected false positive and false negative rate in similarity tests as a fraction of those regions over the whole graph area i.e. $\frac{FP}{\pi}, \frac{FN}{\pi}$, with:

$$\begin{cases} FP = \Phi_\theta - \int_0^{\Phi_\theta} P_{similar}(\theta) d\theta \\ FN = \int_{\Phi_\theta}^{\pi} P_{similar}(\theta) d\theta \end{cases}$$

Additionally, in case some rough, prior knowledge of the probability density function df of $\theta(u_i, u_j)$ exists, we are able to derive more accurate FP, FN estimations. In our previous discussion, we assumed that every $\theta(u_i, u_j)$ value may appear with equal probability. $df(\theta)$ information, however, provides more precise information about the angle values frequency. As a result, we can incorporate this knowledge in our estimation by substituting $P_{similar}(\theta)$ with $df(\theta)P_{similar}(\theta)$ in the presented integrals.

We proceed by demonstrating that the probability of incorrect similarity estimation of two vectors u_i, u_j decreases exponentially with the difference $|\theta(u_i, u_j) - \Phi_\theta|$.

Theorem 3. *For any $\theta(u_i, u_j) > 0$ and $\epsilon = |\frac{\theta(u_i,u_j)-\Phi_\theta}{\theta(u_i,u_j)}|$, clusterheads perform a correct similarity test for u_i, u_j by means of $D(X_i, X_j)$ with probability at least $1-\delta$, where $\delta = e^{-\frac{(\theta(u_i,u_j)-\Phi_\theta)^2}{\theta(u_i,u_j)} \frac{d}{2\pi}}$.*

PROOF. First, please note that for $\theta(u_i, u_j) = 0$, our scheme will always return the same bitmaps for u_i and u_j, thus always correctly classifying them as similar.

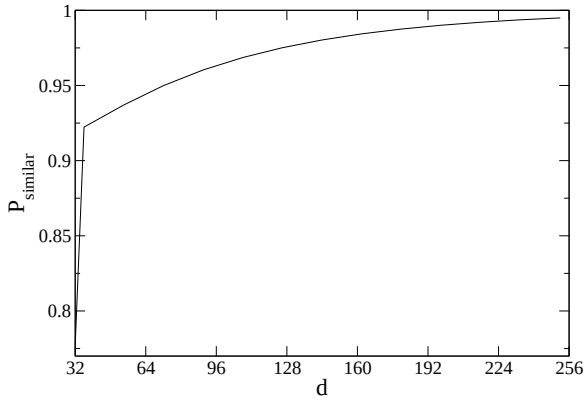


Figure 4: Probability $P_{similar}$ of judging two bitmaps (of vectors that pass the similarity test) as similar, depending on the number of bits d used in the LSH encoding (W=16, θ=5, Φ_θ=10)

Let Y_i be a random variable that takes the value of 0 if the bits at the $i-th$ position of the bitmaps X_i, X_j (received by a clusterhead) agree and 1 otherwise. We can then introduce a new random variable $Y = \sum_{i=1}^{d} Y_i$. Substituting $\sum_{i=1}^{d} Y_i$ with $D_h(X_i, X_j)$, we obtain: $Y = D_h(X_i, X_j)$. Since, by Equation 5, $E[Y_i] = \frac{\theta(u_i,u_j)}{\pi} \Rightarrow E[Y] = d\frac{\theta(u_i,u_j)}{\pi}$.

We prove the theorem for the case when the initial value vectors are dissimilar ($\theta(u_i, u_j) > \Phi_\theta$). The proof for the reverse case is symmetric. The proof will be based on the use of the Chernoff bound [53]. Let $\epsilon = |\frac{\theta(u_i,u_j)-\Phi_\theta}{\theta(u_i,u_j)}|$. Thus, for the case of $\theta(u_i, u_j) > \Phi_\theta$, $\Phi_\theta = \theta(u_i, u_j)(1-\epsilon)$.

$$Pr[E[Y] \leq d\frac{\Phi_\theta}{\pi}] = Pr[D_h(X_i, X_j) \leq d\frac{\theta(u_i, u_j)(1-\epsilon)}{\pi}] = Pr[d\frac{\theta(u_i, u_j)}{\pi} - D_h(X_i, X_j) \geq \epsilon d\frac{\theta(u_i, u_j)}{\pi}] \leq e^{-\epsilon^2 d\frac{\theta(u_i,u_j)}{2\pi}}$$

$$\Rightarrow Pr[\frac{D_h(X_i, X_j)}{d}\pi \leq \Phi_\theta] \leq e^{-\frac{(\theta(u_i,u_j)-\Phi_\theta)^2}{\theta(u_i,u_j)}\frac{d}{2\pi}} \quad (8)$$

The bound of Inequality 8 is a strictly increasing function of $\theta(u_i, u_j)$ in the interval $[0, \Phi_\theta]$ and a strictly decreasing function in the interval $[\Phi_\theta, \pi]$. Thus, the probability of incorrect estimation using TACO decreases (exponentially) with $|\theta(u_i, u_j) - \Phi_\theta|$. This concludes our proof.

□

5.6. Boosting TACO Encodings

We note that the process described in Sections 5.3, 5.4 can accurately compute the support of a mote in the network (assuming a reliable communication protocol that resolves conflicts and lost messages). Thus, if the whole process was executed using the initial measurements (and not the LSH vectors) the resulting list of outliers would be exactly the same with the one that would be computed by the base station, after receiving all measurements and performing the calculations locally. The application of LSH however results in imprecision during pair-wise similarity tests. We previously presented how this imprecision can be bounded in a controlable manner. We also noted (Figure 4) that increasing the size of the bitmaps produced by motes, improves TACO's accuracy. Nevertheless, larger bitmaps imply higher energy consumption. To avoid extra communication burden, we propose an alternative technique to achieve improved accuracy.

Assume that a clusterhead has received a pair of bitmaps X_i, X_j, each consisting of d bits. We split the initial bitmaps X_i, X_j in μ groups (X_{i_1}, X_{j_1}), (X_{i_2}, X_{j_2}), ..., (X_{i_μ}, X_{j_μ}), such that X_i is the concatenation of X_{i_1},...,X_{i_μ}, and similarly for X_j. Each of X_{i_κ} and X_{j_κ} is a bitmap of n bits such that $d = \mu \cdot n$. For each group g_κ we obtain an estimation θ_κ of angle similarity using Equation 5 and, subsequently, an answer to the similarity test based on the pair

of bitmaps in the group. We then provide as an answer to the similarity test, the answer provided by the majority of the μ similarity tests.[3]

Two questions that naturally arise are: (i) Does the aforementioned partitioning of the hash space help improve the accuracy of successful classification?; and (ii) Which value of μ should one use? Let us consider the probability of correctly classifying two similar vectors in TACO (the case of dissimilar vectors is symmetric). In our original (unpartitioned) framework, the probability of correctly determining the two vectors as similar is $P_{similar}(d)$, given by Equation 7. Thus, the failure probability of incorrectly classifying two similar vectors as dissimilar is $P_{wrong}(d) = 1 - P_{similar}(d)$.

By separating the initial bitmaps to μ groups, each containing $\frac{d}{\mu}$ bits, one can view the above classification as using μ independent Bernoulli trials, which each return 1 (similarity) with a success probability of $P_{similar}(\frac{d}{\mu})$, and 0 (dissimilarity), otherwise. Let Y denote the random variable that computes the sum of these μ trials. In order for our classification to incorrectly classify the two vectors as dissimilar, more than half of the μ similarity tests must fail. The average number of successes in these μ tests is $\overline{Y} = \mu \times P_{similar}(\frac{d}{\mu})$. A direct application of the Chernoff bounds gives that more than half of the Bernoulli trials can fail with a probability $P_{wrong}(d, \mu)$ at most: $P_{wrong}(d, \mu) \leq e^{-2\mu(P_{similar}(\frac{d}{\mu})-\frac{1}{2})^2}$. Given that the number of bits d and, thus, the number of potential values for μ is small, we may compare $P_{wrong}(d, \mu)$ with $P_{wrong}(d)$ for a small set of μ values and determine whether it is more beneficial to use this boosting approach or not. We also need to make two important observations regarding the possible values of μ: (i) The number of possible μ values is further restricted by the fact that our above analysis holds for μ values that provide a (per group) success probability > 0.5 and (ii) Increasing the value of μ may provide worse results, as the number of used bits per group decreases. We explore this issue further in our experiments.

5.7. Discussion

The robustness of TACO's outlier definition, as well as the tuning capabilities that it provides, render the framework straightforwardly applicable to a wide variety of application classes. The setting of TACO's parameters is in direct relation to the application context. In particular, the first of these parameters regards the window size W which can be tuned depending on the application's desire to base its decisions on short- or long-term observations. The second parameter refers to the length d of the LSH bitmaps, which affects the number of transmitted bits and for which we have extensively analyzed (Sections 4,5) its impact on the accuracy of our techniques. The desired similarity threshold (Φ) and the level of support ($minSup$) are essentially those values that determine sensitive or more relaxed outlier definitions referring to neuralgic or ordinary deployments of TACO, respectively.

A popular deployment field where wireless sensor networks suit themselves in, relates to habitat monitoring applications [54]. As an example, consider a monitoring application that aims at investigating bird breeding conditions with motes placed in nests. Since nest temperatures may affect monitored behaviors, such mote samples are considered of particular utility. The apparition of outlying temperature measurements of nodes near nests may attract researchers' interest to further look into caused reactions. As scientists usually base their investigation on long term observations [54], large window sizes may be utilized. In TACO's setting, W values between 24-32 measurements can be applicable, with $\Phi_\theta = 30$ degrees and $minSup$ values of 4 motes withing a radius of $\sim$10 meters near bird nests. To prolong the scientific observation interval and thus the lifetime of the whole sensor network, d values can be squeezed to yield data reduction ratios of $[1/8, 1/16]$.

As another example, consider motes as machine particles in industrial applications where controllers need to pinpoint machines that exhibit high vibrations indicating malfunctioning conditions. Their aim is to timely diagnose those machines and intervene to prevent the production process to be ceased due to permanent casualty. Consequently, sampling rates are high while accuracy is of importance to avoid false negative or positive alarms. The first requirement may lead to selecting smaller window sizes of $W = 16$ measurements, while the second requirement premises d values ensuring moderate (e.g. 1/2 or 1/4) reduction ratios and sensitive similarity definitions i.e., $\Phi_\theta = 10$ degrees.

The above are representative examples of TACO's adoption and parameter tuning. Of course, the framework itself is not limited to the discussed scenarios but can serve as an outlier detection tool in any deployment field.

[3] Ties are resolved by taking the median estimate of θ_ks.

6. Load Balancing and Comparison Pruning

In our initial framework, clusterhead nodes are required to perform data collection and reporting, as well as bitmap comparisons. As a result, clusterheads are overloaded with extra communication and processing costs, which entails larger energy drain, when compared to other nodes. In order to avoid draining the energy of clusterhead nodes, the network structure will need to be frequently reorganized (by electing new clusterheads). While protocols such as HEED [16] limit the number of messages required during the clusterhead election process, this election process still requires bandwidth. In this section, we tackle with this issue and describe efficient mechanisms provided by TACO for limiting clusterheads' load.

6.1. Leveraging Additional Motes for Outlier Detection

In order to limit the overhead of clusterhead nodes, we extend our framework by incorporating the notion of *bucket nodes*. *Bucket nodes* (or simply buckets) are motes within a cluster the presence of which aims at distributing communication and processing tasks and their associated costs. Besides selecting the clusterhead nodes, within each cluster the election process continues to elect additional B bucket nodes. This election process is easier to carry out by using the same algorithm (i.e., HEED) that we used for the clusterhead election.

After electing the bucket nodes within each cluster, our framework determines a mechanism that distributes the outlier detection duties amongst them. Our goal is to group similar bitmaps in the same bucket so that the comparisons that will take place within each bucket produce just a few local outliers. To achieve this, we introduce a second level of hashing.

Proposition 1. *Let $W_h(X_i) = \sum_{\ell=1}^{d} X_{i\ell}$ be the hamming weight of bitmap X_i with $0 \leq W_h(X_i) \leq d$. For any pair of bitmaps X_i and X_j, it holds that $D_h(X_i, X_j) \geq |W_h(X_i) - W_h(X_j)|$.*

PROOF. For any pair of bitmaps X_i, X_j:

$$|W_h(X_i) - W_h(X_j)| = |\sum_{\ell=1}^{d} X_{i\ell} - \sum_{\ell=1}^{d} X_{j\ell}| = |\sum_{\ell=1}^{d} (X_{i\ell} - X_{j\ell})| \overset{\text{triangle inequality}}{\leq} \sum_{\ell=1}^{d} |X_{i\ell} - X_{j\ell}| = D_h(X_i, X_j)$$

□

Colorrary 1. *If $|W_h(X_i) - W_h(X_j)| > \Phi_{D_h}$, then the bitmaps X_i and X_j cannot witness each other.*

Our second level of hashing takes into consideration Colorrary 1 to hash highly dissimilar bitmaps to different buckets. More precisely:

- Consider a partitioning of the hash key space to the elected buckets, such that each hash key is assigned to the $\lfloor \frac{W_h(X_i)}{\frac{d}{B}} \rfloor$-th bucket. Motes with similar bitmaps will have nearby Hamming weights, thus hashing to the same bucket with high probability.
- Please recall that encodings that can support each other in our framework have a Hamming distance lower or equal to Φ_{D_h}. In order to guarantee that a node's encoding can be used to witness any possible encoding within its cluster, this encoding needs to be sent to all buckets that cover the hash key range $\lfloor \frac{\max\{W_h(X_i) - \Phi_{D_h}, 0\}}{\frac{d}{B}} \rfloor$ to $\lfloor \frac{\min\{W_h(X_i) + \Phi_{D_h}, d\}}{\frac{d}{B}} \rfloor$. Thus, the value of B determines the number of buckets to which an encoding must be sent. Larger values of B reduce the range of each bucket, but result in more encodings being transmitted to multiple buckets. In our framework, we select the value B (whenever at least B nodes exist in the cluster) by setting $\frac{d}{B} > \Phi_{D_h} \Longrightarrow B < \frac{d}{\Phi_{D_h}}$. As we will shortly show, this guarantees that each encoding will need to be transmitted to at most one additional bucket, thus avoiding hashing the measurements of each node to multiple buckets.
- The transmission of an encoding to multiple bucket nodes ensures that it may be tested for similarity with any value that may potentially witness it. Therefore, the support that a node's measurements have reached is distributed in multiple buckets needing to be combined.

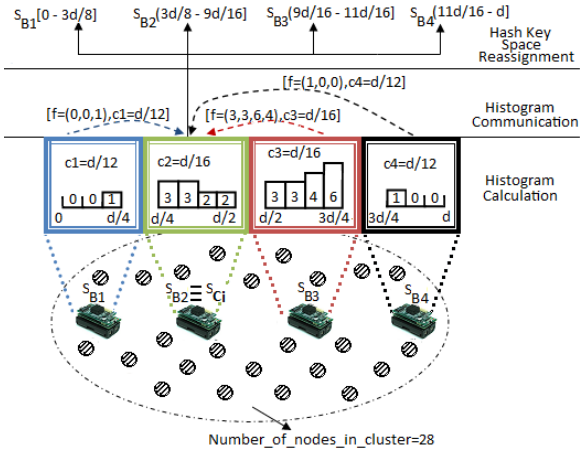


Figure 5: Exemplary (bottom-up) demonstration of the 3 phases of load balancing

- Moreover, we must also make sure that the similarity test between two encodings is not performed more than once. Thus, we impose the following rules: (a) For encodings mapping to the same bucket node, the similarity test between them is performed only in that bucket node; and (b) For encodings mapping to different bucket nodes, their similarity test is performed only in the bucket node with the lowest id amongst these two. Given these two requirements, we can thus limit the number of bucket nodes to which we transmit an encoding to the range $\lfloor \frac{\max\{W_h(X_i)-\Phi_{D_h},0\}}{\frac{d}{B}} \rfloor$ to $\lfloor \frac{W_h(X_i)}{\frac{d}{B}} \rfloor$. The above range for $B < \frac{d}{\Phi_{D_h}}$ is guaranteed to contain at most 2 buckets.

Thus, each bucket reports the set of outliers that it has detected, along with their support, to the clusterhead. The clusterhead performs the following tests:

- Any encoding reported to the clusterhead by at least one, but not all bucket nodes to which it was transmitted, is guaranteed not to be an outlier, since it must have reached the required support at those bucket nodes that did not report the encoding.
- For the remaining encodings, the received support is added, and only those encodings that did not receive the required overall support are considered to be outliers.

6.2. Load Balancing Among Buckets

Despite the fact that the introduction of bucket nodes does alleviate clusterheads from comparison and message reception load, it does not guarantee by itself that the portion of load taken away from the clusterheads will be equally distributed between buckets. In particular, we expect that motes sampling ordinary values of measured attributes will produce similar bitmaps, thus directing these bitmaps to a limited subset of buckets, instead of equally utilizing the whole arrangement. In such a situation, an equi-width partitioning of the hash key space to the bucket nodes is obviously not a good strategy. On the other hand, if we wish to determine a more suitable hash key space allocation, we require information about the data distribution of the monitored attributes and, more precisely, about the distribution of the hamming weight of the bitmaps that original value vectors yield. Based on the above observations, we can devise a load balancing mechanism that can be used after the initial, equal-width partitioning in order to repartition the hash key space between bucket nodes. Our load balancing mechanism fosters simple equi-width histograms and consists of three phases: a) *histogram calculation* per bucket, b) *histogram communication* between buckets, and c) *hash key space reassignment*.

During the *histogram calculation* phase, each bucket locally constructs equi-width histograms by counting $W_h(X_i)$ frequencies belonging to bitmaps that were hashed to them. The range of histogram's domain value is restricted to the hash key space portion assigned to each bucket. Obviously, this phase takes place side by side with the normal operation of the motes. We note that this phase adds minimum computation overhead since it only involves increasing by one the corresponding histogram bucket counter for each received bitmap.

In the *histogram communication* phase, each bucket communicates to its clusterhead (a) its estimated frequency counts attached, and (b) the width parameter c that it used in its histogram calculation. From the previous partitioning of the hash key space, the clusterhead knows the hash key space of each bucket node. Thus, the transmission of the width c is enough to determine (a) the number of received bars/values, and (b) the range of each bar of the received histogram. As a result, the clusterhead can easily reconstruct the histograms that it received.

The final step involves the adjustment of the hash key space allocation that will eventually provide the desired load balance based on the transmitted histograms. Relying on the received histograms, the clusterhead determines a new space partitioning and broadcasts it to all nodes in its cluster. The aforementioned phases can be periodically (but not frequently) repeated to adjust the bounds allocated to each bucket, adapting the arrangement to changing data distributions. Figure 5 depicts an example of the load balancing procedure. To simplify the figure, in this example we assume that the second bucket node is also the clusterhead.

The mechanisms described in this section better balance the load among buckets and also refrain from performing unnecessary similarity checks between dissimilar pairs of bitmaps, which would otherwise have arrived at the clusterhead. This stems from the fact that hashing bitmaps based on their hamming weight ensures that dissimilar bitmaps are hashed to different buckets. We experimentally validate the ability of this second level hashing technique to prune the number of comparisons in Section 9.5.

7. TACO under Other Supported Similarity Measures

We have already noted the ability of our framework to encompass a wide variety of popular similarity measures. So far, in our running example, we showed TACO's function using the angle (and, thus, the cosine similarity) between sensor value vectors. In this subsection we provide a detailed discussion on how the other measures presented in Table 1 can be incorporated in TACO.

Correlation Coefficient. Let $E(u_i)$ notate the mean value and σ_{u_i} the standard deviation of vector u_i. Moreover for mote value vectors u_i, u_j we denote $u_i^* = u_i - E(u_i)$, $u_j^* = u_j - E(u_j)$.

Proposition 2. *The correlation coefficient (*corr*) can be used as a similarity measure in TACO by using the same family of hashing functions as with the cosine similarity in the Random Hyperplane Projection LSH scheme.*

PROOF. To prove the proposition it suffices to show that some kind of equivalence between the two measures (i.e., cosine similarity and correlation coefficient) exists. In particular,

$$corr(u_i, u_j) = corr(u_i^*, u_j^*) = cos(\theta(u_i^*, u_j^*)) \tag{9}$$

holds. We now provide a simple proof for the previous equation, starting from the first part, and based on the observation that $E(u_i^*) = E(u_j^*) = 0$, while also $\sigma_{u_i^*} = \sigma_{u_i}$ and $\sigma_{u_j^*} = \sigma_{u_j}$:

$$\begin{aligned} corr(u_i^*, u_j^*) &= \frac{E(u_i^* u_j^*) - E(u_i^*)E(u_j^*)}{\sigma_{u_i^*}\sigma_{u_j^*}} = \frac{E((u_i - E(u_i))(u_j - E(u_j))) - E(u_i^*)E(u_j^*)}{\sigma_{u_i^*}\sigma_{u_j^*}} = \\ &= \frac{E(u_i \cdot u_j - u_j \cdot E(u_i) - u_i \cdot E(u_j) + E(u_i) \cdot E(u_j))}{\sigma_{u_i}\sigma_{u_j}} = \\ &= \frac{E(u_i \cdot u_j) - E(u_j \cdot E(u_i)) - E(u_i \cdot E(u_j)) + E(u_i) \cdot E(u_j)}{\sigma_{u_i}\sigma_{u_j}} = \frac{E(u_i \cdot u_j) - E(u_i) \cdot E(u_j)}{\sigma_{u_i}\sigma_{u_j}} \end{aligned}$$

which equals $corr(u_i, u_j)$. Furthermore:

$$cos(\theta(u_i^*, u_j^*)) = \frac{u_i^* \cdot u_j^*}{\|u_i^*\| \cdot \|u_j^*\|} = \frac{\frac{1}{W}\sum_{\ell=1}^{W} u_{i\ell}^* \cdot u_{j\ell}^*}{\frac{1}{W}\sqrt{\sum_{\ell=1}^{W} u_{i\ell}^{*2}} \cdot \sqrt{\sum_{\ell=1}^{W} u_{j\ell}^{*2}}} = \frac{E(u_i^* u_j^*)}{\sigma_{u_i^*}\sigma_{u_j^*}} = corr(u_i^*, u_j^*)$$

□

In other words, an outlier detection query may specify a similarity threshold Φ_{corr} based on the correlation coefficient for u_i, u_j. Since $corr(u_i, u_j) = corr(u_i^*, u_j^*)$, Φ_{corr} also holds for u_i^*, u_j^*. Additionally, because $corr(u_i^*, u_j^*) = cos(\theta(u_i^*, u_j^*))$, Φ_{corr} can be transformed into a threshold for the hamming distance between bitmaps using Equation 5. Hence, after the collection of u_is, motes produce and apply LSH on u_i^*s so as to obtain appropriate bitmaps preserving the angle and, subsequently, the *corr*-similarity of the initial value vectors. The rest of the outlier detection process presented in the previous sections remains unaffected.

Euclidean Distance of Standardized Vectors. A popular similarity measure often used in distance based outlier identification is the Euclidean distance. Nonetheless, this measure relies on absolute values to determine the similarity of u_i, u_j while we have previously reasoned that in our setting the emphasis should be set on the correlations of the motes measurements in space and time rather than on the absolute sampled values. Nevertheless, we are able to adjust the Euclidean distance so as to serve our purposes by considering standardized vectors.

Let $u_i' = \frac{u_i - E(u_i)}{\sigma_{u_i}}, u_j' = \frac{u_j - E(u_j)}{\sigma_{u_j}}$ and $dist(u_i', u_j')$ the Euclidean distance between u_i', u_j' which is calculated by $dist(u_i', u_j') = \sqrt{\sum_{\ell=1}^{W}(u_{i\ell}' - u_{j\ell}')^2}$.

Proposition 3. *The Euclidean distance* dist() *of standardized mote vectors can capture correlations among sensor readings and is incorporated in TACO by using the same family of hashing functions as with the cosine similarity in the Random Hyperplane Projection LSH scheme.*

PROOF. Initially, we ought to show that the Euclidean distance of standardized vectors can capture existing correlations between motes' values. In fact, $corr(u_i, u_j) = 1 - \frac{dist^2(u_i', u_j')}{2W}$ holds. Observe that upon standardizing mote value vectors, their mean is zero and their standard deviation is 1. As a result, $corr(u_i', u_j')$ is reduced to $\frac{1}{W}\sum_{\ell=1}^{W} u_{i\ell}' u_{j\ell}'$ and simple calculations yield that $corr(u_i, u_j) = corr(u_i', u_j')$. Moreover, $dist^2(u_i', u_j') = \sum_{\ell=1}^{W}(u_{i\ell}' - u_{j\ell}')^2 = \sum_{\ell=1}^{W} u_{i\ell}'^2 + \sum_{\ell=1}^{W} u_{j\ell}'^2 - 2\sum_{\ell=1}^{W} u_{i\ell}' u_{j\ell}' = 2W - 2W corr(u_i', u_j')$. Combining the previous pair of equivalences leads to the discussed $corr(u_i, u_j) = 1 - \frac{dist^2(u_i', u_j')}{2W}$ equality.

Since the correlation coefficient captures the correlation among vectors and we exposed a formula connecting it with $dist(u_i', u_j')$, the Euclidean distance of standardized vectors can also capture potential interrelations. It remains to exhibit that $dist(u_i', u_j')$ is encompassed by the Random Hyperplane Projection scheme which is derived in a straightforward manner according to Equation 9: $corr(u_i, u_j) = cos(u_i^*, u_j^*) = 1 - \frac{dist^2(u_i', u_j')}{2W}$. □

Summarizing, it suffices for the user query to place a threshold for $dist(u_i', u_j')$ which is transformed into an equivalent Φ_{corr}. That point forward, TACO operates in exactly the same way as with the *corr* similarity measure choice.

Extended Jaccard Coefficient of Standardized Vectors. Similarly, the Extended Jaccard (or Tanimoto) Coefficient of u_i', u_j' expressed by the ratio $T(u_i', u_j') = \frac{u_i' \cdot u_j'}{\|u_i'\|^2 + \|u_j'\|^2 - u_i' \cdot u_j'}$ is commutatively supported in TACO by means of their Euclidean distance. In particular, given a specific $\Phi_{Tanimoto} \in [0, 1]$ the similarity test $T(u_i', u_j') \geq \Phi_{Tanimoto}$ can be performed checking the condition: $dist(\frac{\Phi_{Tanimoto}+1}{2\Phi_{Tanimoto}} u_i', u_j') \leq \frac{\sqrt{(\Phi_{Tanimoto}+1)^2 - 4\Phi_{Tanimoto}^2}}{2\Phi_{Tanimoto}} \sqrt{W}$ instead [55].

Jaccard Coefficient. Jaccard coefficient is another measure that can be used in applications that require motes sample discrete quantities such as types of objects in the network realm, spatial features etc. In [45] the authors introduce a mechanism for transforming sets of values into dimensionally reduced bitmaps with preserved Jaccard similarity. To achieve that, they use minwise independent permutations and simplex codes. Here, we provide a summary of the main results of [45] for clarity and discuss the way the MinHash scheme [56, 57] utilized there can be embodied in the outlier detection procedure. Moreover, we extend the results of [45] by providing a mechanism to determine appropriate bitmap sizes upon utilizing that scheme in TACO's context.

Let $\pi()$ denote a random permutation on $u_i \epsilon N^W$ (assuming elements of value sets are labeled by Natural numbers) and $min\{\pi(u_i)\} = min\{\pi(u_{i\ell}) | u_{i\ell} \epsilon u_i\}$. According to the min hashing scheme [56, 57, 45]:

$$Pr[min\{\pi(u_i)\} = min\{\pi(u_j)\}] = J(u_i, u_j)$$

After using k random permutations the resulted signatures are expected to agree in $k \cdot J(u_i, u_j)$ values. Taking one step further, signatures can be embedded in the hamming space utilizing error correcting codes. Error correcting codes (ECCs) have the property of transforming the b-lengthed binary representation of each signature element to bitmaps of fixed $(b+s)/2$ distance, for some $s > 0$. The final bitmap is produced by concatenating the ECC outcomes. Eventually, the following equivalence can be proved [45]:

$$\frac{D_h(X_i, X_j)}{d} = \frac{1 - J(u_i, u_j)}{2} \tag{10}$$

with bitmap size $d = (b + s) \cdot k$ and $0 \leq D_h(X_i, X_j) \leq d/2$.

In [57] the authors dictate an appropriate value for k to control the number of false positives/false negatives during similarity tests using the signatures with given $\Phi_{Jaccard}$ threshold. Nonetheless, the transition to the hamming space results in additional imprecision. Signature elements are chosen as numbers of fixed precision which determines the length b of their binary representation. On the other hand, normally, when ECCs are used to correct bit errors in communication channels, the value of s is chosen on the basis of the number of errors that an ECC is able to amend. Nevertheless, in the current utilization no such criterion may be applied as our goal is different and regards accurate similarity preservation. Additionally, appropriate k values dictated by [57] premise that the similarity between value vectors is lower bounded by some constant number. Obviously, such an assumption cannot be guaranteed in our setting. In other words, the bounds provided in [57] are not sufficient for the current context since they do not take into consideration the length of the binary representation of the signature elements and the chosen ECC to determine the value of k and subsequently control the imprecision of the Jaccard coefficient based similarity test using Equation 10.

Notice that bits at corresponding positions of bitmaps X_i, X_j are not independent as they are produced based on the chosen error correcting code specifications and signature element binary formats. For this reason the boosting process of Section 5.6 cannot be used for this kind of bitmaps as we cannot freely partition them into groups. However, the k groups, each of $b+s$ bit length, inside a bitmap are independent since they originate from different signature elements. We exploit this fact to prove the following theorem.

Theorem 4. *Given the choice of signature element universe and the specifications of the chosen ECC (that is, given $b + s$), to estimate $\frac{1-J(u_i,u_j)}{2}$ with precision $\frac{\epsilon}{b+s}$ and probability at least $1 - \delta$ the number of signature elements k should be set to $O(log(2/\delta)(b + s)^2/(8\epsilon^2))$.*

PROOF. Assume a pair of bitmaps X_i, X_j produced by applying min hashing and a chosen ECC to sets u_i, u_j correspondingly. Let $Y_1, Y_2, \ldots, Y_k$ be independent random variables with $Y_i = 0$ or $Y_i = (b + s)/2$. The average of the sum of these variables is $\sum_{i=1}^{k} \frac{Y_i}{k} = \frac{D_h(X_i,X_j)}{k}$ and the expectation of the previous average, derived by Equation 10, is $(b + s)\frac{1-J(u_i,u_j)}{2}$. Utilizing Hoeffding's inequality [49] for some $\epsilon > 0$:

$$Pr[|\frac{D_h(X_i, X_j)}{k} - (b + s)\frac{1 - J(u_i, u_j)}{2}| \geq \epsilon] \leq 2e^{-\frac{8k\epsilon^2}{(b+s)^2}}$$

Substituting $(b + s) \cdot k$ with d:

$$Pr[|\frac{D_h(X_i, X_j)}{d} - \frac{1 - J(u_i, u_j)}{2}| \geq \frac{\epsilon}{b + s}] \leq 2e^{-\frac{8k\epsilon^2}{(b+s)^2}}$$

Setting the right side of the inequality equal to δ and performing simple calculations, completes the proof. □

The above theorem provides the means to determine the value of k and simultaneously incorporates the effect of the ECC choice (the value of s) in the desired precision of the estimation. After determining the overall d value, the following theorem elaborates on the accuracy of the similarity test performed at the clusterheads' level.

Theorem 5. *For any $J(u_i, u_j) < 1$ and $\epsilon = \frac{|J(u_i,u_j)-\Phi_{Jaccard}|}{1-J(u_i,u_j)}$, clusterheads perform a correct similarity test for u_i, u_j by means of $D(X_i, X_j)$ with probability at least $1 - \delta$, where $\delta = e^{-\frac{d(J(u_i,u_j)-\Phi_{Jaccard})^2}{2(1-J(u_i,u_j))}}$.*

A proof can be obtained as in Theorem 3 and is omitted. Note again that $J(u_i, u_j) = 1$ leads to identical bitmaps and the probability of incorrect similarity test decreases exponentially only this time with the $|J(u_i, u_j) - \Phi_{Jaccard}|$ difference. Upon motes transform initial sets of values to bitmaps, the outlier detection process using the Jaccard coefficient is quite analogous to the case of cosine similarity with Equation 10 (instead of Equation 5) as the main tool.

We further note [10] that there exist popular similarity metrics that do not accept an LSH scheme. For instance, Lemma 1 implies that the $Dice(u_i, u_j) = \frac{2|u_i \cap u_j|}{|u_i|+|u_j|}$ and $Overlap(u_i, u_j) = \frac{|u_i \cap u_j|}{min(|u_i|,|u_j|)}$ coefficients do not admit an LSH scheme since they do not satisfy the triangle inequality.

8. Extensions

An obvious way to further decrease communication costs during the intra- and intercluster processing of TACO is by suppressing mote messages when bitmaps are not altered in a number of successive tumbles. In particular, let X_i^{last} denote the last bitmap that mote S_i transmitted to the clusterhead (or bucket node) and X_i^{new} the bitmap produced in the current tumble. When $D_h(X_i^{last}, X_i^{new}) = 0$, S_i does not need to communicate any information to the clusterhead. This reduces the burden of intracluster communication. In the intercluster processing, as far as X_i is not modified, should it happen to be included in $PotOut_{C_i}$, X_i^{last} needs to be transmitted only once. Please notice that the result of similarity tests where S_i participates will not be affected at all.

Relaxing the previous condition, we may allow motes suppress their messages in case $D_h(X_i^{last}, X_i^{new}) \leq f$, where $0 \leq f \leq d$. As a consequence, additional savings in bandwidth consumption are yielded, however, with the make-weight of distorting the result of the tests which S_i takes place in. Despite this fact, we can still guarantee that a portion of these tests cannot be affected. Hereafter, we outline the cases for which no distortion in S_i's similarity test outcomes is introduced.

Assume that motes S_i, S_j are to be compared during the intra- or intercluster processing and S_i suppressed its message in the current tumble while S_j did not. The result of the similarity test will rely on $D_h(X_i^{last}, X_j^{new})$. Due to the fact that the hamming distance possesses the property of satisfying the triangle inequality and bearing that $D_h(X_i^{last}, X_i^{new}) \leq f$:

$$|D_h(X_i^{last}, X_j^{new}) - f| \leq D_h(X_i^{new}, X_j^{new}) \leq D_h(X_i^{last}, X_j^{new}) + f \tag{11}$$

Consequently:

$$(D_h(X_i^{last}, X_j^{new}) + f \leq \Phi_{D_h}) \vee ((D_h(X_i^{last}, X_j^{new}) \geq f) \wedge (D_h(X_i^{last}, X_j^{new}) - f \geq \Phi_{D_h})) \models \text{undistorted test.}$$

Provided that both sensor nodes S_i, S_j suppress their messages, $D_h(X_i^{last}, X_j^{last})$ is taken into consideration so as to decide their similarity. In this case:

$$|D_h(X_i^{last}, X_j^{last}) - f| \leq D_h(X_i^{last}, X_j^{new}) \leq D_h(X_i^{last}, X_j^{last}) + f \tag{12}$$

Combining inequalities (11),(12), for the case of pairs of motes that mutually suppress their messages, we overall obtain:

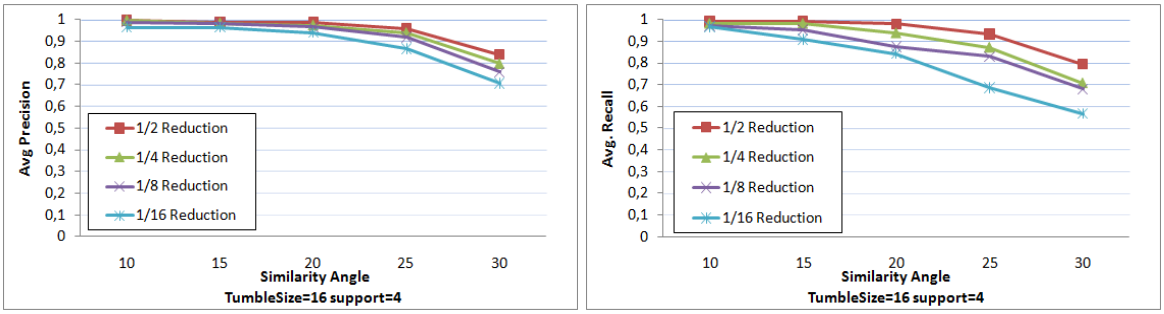
$$(D_h(X_i^{last}, X_j^{last}) + 2f \leq \Phi_{D_h}) \vee ((D_h(X_i^{last}, X_j^{last}) \geq 2f) \wedge (D_h(X_i^{last}, X_j^{last}) - 2f \geq \Phi_{D_h})) \models \text{undistorted test.}$$

In any other case, depending on the actual changes of the corresponding hamming distance, the adoption of the message suppression strategy may cause alteration (compared to the utilization of $D_h(X_i^{new}, X_j^{new})$) in the result of a test or not. Obviously, increasing the value of f provides increased communication savings but also weakens the guarantees on the distortion of similarity test outcomes. On the other hand, smaller fs produce tighter upper and lower bounds in the presented inequalities (11),(12) while yielding more moderate bandwidth consumption preservation.

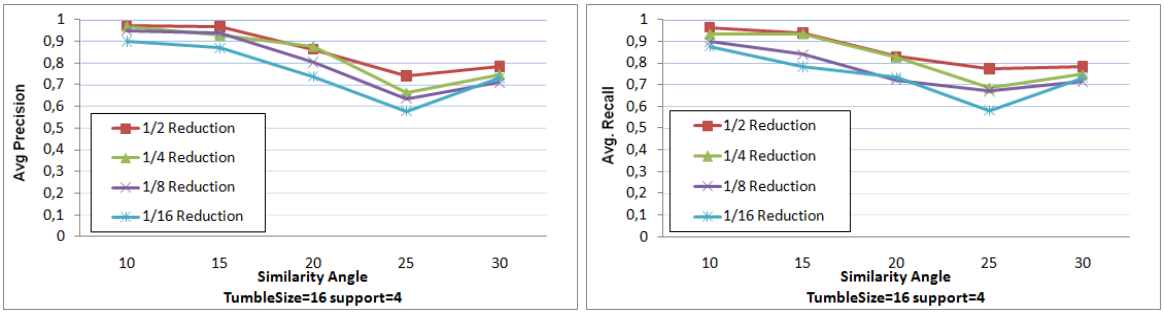
9. Experiments

9.1. Experimental Setup

In order to evaluate the performance of our techniques we implemented our framework on top of the TOSSIM network simulator [58]. Since TOSSIM imposes restrictions on the network size and is rather slow in simulating



(a) Intel.Temperature Precision, Recall vs Similarity Angle



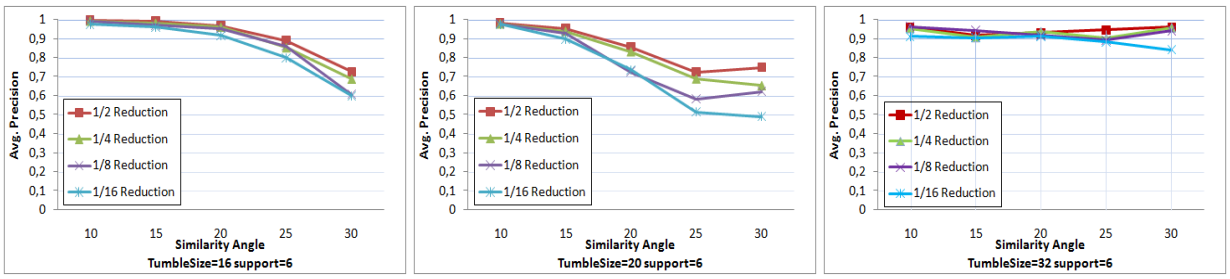
(b) Intel.Humidity Precision,Recall vs Similarity Angle

Figure 6: Average Precision, Recall in Intel Data Set

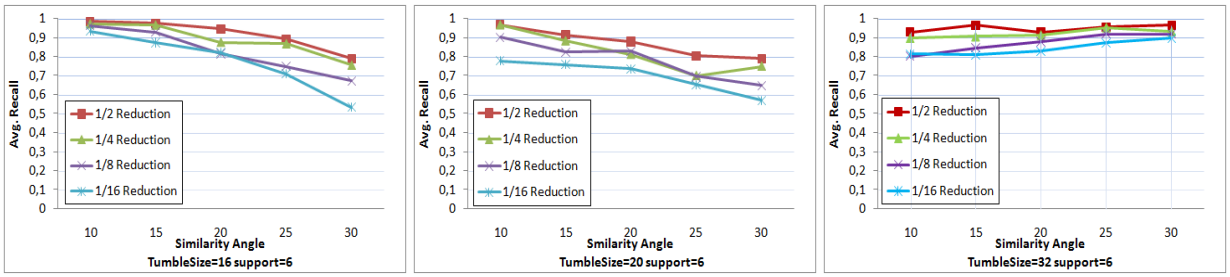
experiments lasting for thousands of epochs, we further developed an additional lightweight simulator in Java and used it for our sensitivity analysis, where we vary the values of several parameters and assess the accuracy of our outlier detection scheme. The TOSSIM simulator was used in smaller-scale experiments, in order to evaluate the energy and bandwidth consumption of our techniques and of alternative methods for computing outliers. Through these experiments we examine the performance of all methods, while taking into account message loss and collisions, which in turn result in additional retransmissions and affect the energy consumption and the network lifetime.

In our experiments we utilized two real world data sets. The first, termed Intel Lab Data, includes temperature and humidity measurements collected by 48 motes for a period of 633 and 487 epochs, respectively, in the Intel Research, Berkeley lab [59]. The second, termed Weather Data, includes air temperature, relative humidity and solar irradiance measurements from the station in the University of Washington and for the year 2002 [60]. We used these measurements to generate readings for 100 motes for a period of 2000 epochs. In both data sets we increased the complexity of the temperature and humidity data by specifying for each mote a 6% probability that it will fail dirty at some point. We simulated failures using a known deficiency [2] of the MICA2 temperature sensor: each mote that fails-dirty increases its measurement (in our experiment this increase occurs at an average rate of about 1 degree per epoch), until it reaches a MAX_VAL parameter. This parameter was set to 100 degrees for the Intel lab data set and 200 degrees for the Weather data (due to the fact that the Weather data set contains higher average values). To prevent the measurements from lying on a straight line, we also impose a noise up to 15% at the values of a node that fails dirty. Additionally, each node with probability 0.4% at each epoch obtains a spurious measurement which is modeled as a random reading between 0 and MAX_VAL degrees. Finally, for solar irradiance measurements, we randomly injected values obtained at various time periods to the sequence of readings, in order to generate outliers.

We need to emphasize that increasing the complexity of the real data sets actually represents a worst-case scenario for our techniques. It is easy to understand that the amount of transmitted data during the intracluster communication phase is independent of the data sets' complexity, since it only depends on the specified parameter d that controls the dimensionality reduction. On the other hand, the amount of data exchanged during the intercluster phase of our framework depends on the number of generated outlier values. Thus, the added data set complexity only increases the transmitted data (and, thus, the energy consumption) of our framework. Despite this fact, we demonstrate that our



(a) Weather Precision: Temperature, Humidity and Solar Irradiance vs Similarity Angle



(b) Weather Recall: Temperature, Humidity and Solar Irradiance vs Similarity Angle

Figure 7: Average Precision, Recall in Weather Data Set

techniques can still manage to drastically reduce the amount of transmitted data, in some cases even below what a simple aggregate query (i.e., MIN, MAX or SUM) would require under TAG [8].

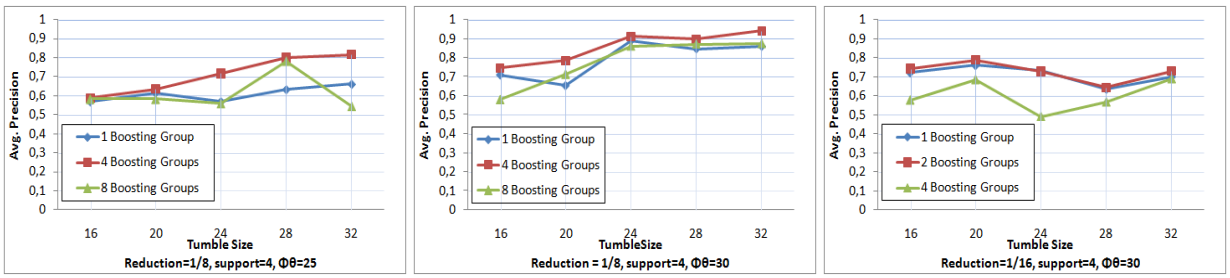
In the Intel Lab and Weather data sets we organized the sensor nodes in four and ten clusters, correspondingly. Please note that we selected a larger number of clusters for the Weather data set, due to the larger number of sensor nodes that appear in it. The sensor nodes were organized in clusters using the HEED algorithm.

9.2. Sensitivity Analysis

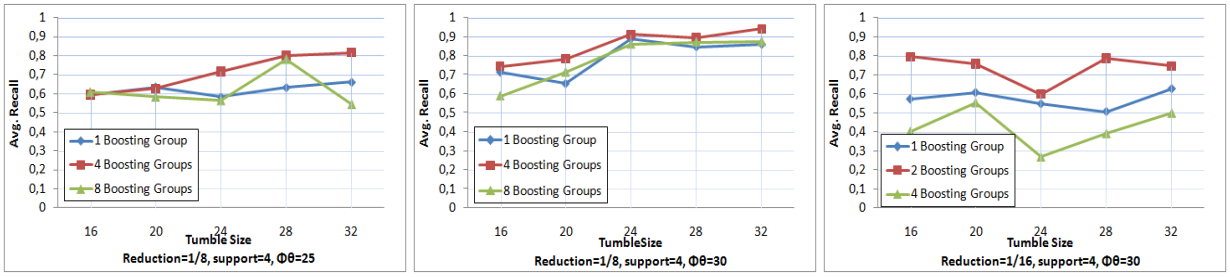
We first present a series of sensitivity analysis experiments using our Java simulator in order to explore a reasonably rich subset of the parameter space. To evaluate the accuracy of TACO in the available data sets we initially focus on the precision and recall metrics. In a nutshell, the precision specifies the percentage of reported outliers that are true outliers, while the recall specifies the percentage of outliers that are reported by our framework. The set of true outliers was computed offline (i.e. assuming all data was locally available), based on the selected similarity metric and threshold, specified in each experiment. The goal of these experiments is to measure the accuracy of the TACO scheme and of the boosting process, and to assess their resilience to different compression ratios.

We used different tumble sizes ranging between 16 and 32 measurements and Φ_θ thresholds between 10 and 30 degrees. Moreover, we experimented with a reduction ratio up to 1/16 for each (W, Φ_θ) combination. In the Intel Lab data sets we found little fluctuations by changing the *minSup* parameter from 3-5 motes, so henceforth we consider a fixed *minSup*=4 (please recall that there are 48 motes in this data set). Due to a similar observation in the Weather data set, *minSup* is set to 6 motes. All the experiments were repeated 10 times. Figures 6 and 7 depict the accuracy of our methods presenting the average precision and recall for the used data sets, for different similarity angles and reduction ratios. To acquire these, we obtained precision and recall values per tumble and calculated the average precision, recall over all tumbles in the run. Finally, we proceeded by estimating averages over 10 repetitions, using a different random set of hash functions in each iteration.

As it can be easily observed, in most of the cases, motes producing outlying values can be successfully pinpointed by our framework with average precision and recall $> 80\%$, even when imposing a 1/8 or 1/16 reduction ratio, for similarity angles up to 20 degrees. The TACO scheme is much more accurate when asked to capture strict, sensitive definitions of outlying values, implied by a low Φ_θ value. This behavior is expected based on our formal analysis (see Figure 3 and Equation 2). We also note that the model may slightly swerve from its expected behavior depending on



(a) Boosting Precision: Intel.Humidity (Φ_θ=25 & 30 degrees for 1/8 Reduction) and Intel.Temperature (Φ_θ=30 degrees for 1/16 Reduction) vs Tumble Size



(b) Boosting Recall: Intel.Humidity (Φ_θ=25 & 30 degrees for 1/8 Reduction) and Intel.Temperature (Φ_θ=30 degrees for 1/16 Reduction) vs Tumble Size

Figure 8: Boosting Application on Intel datasets

the number of near-to-threshold outliers (those falling in the areas FP, FN in Figure 3) that exist in the data set. That is, for instance, the case when switching from 25 to 30 degrees in the humidity data sets of Figures 6 and 7.

Obviously, an improvement in the final results may arise by increasing the length d of each bitmap (i.e., consider more moderate reduction ratios, Figure 4). Another way to improve performance is to utilize the boosting process discussed in Section 5.6. All previous experiments were ran using a single boosting group during the comparison procedure. Figure 8 depicts the improvement in the values of precision and recall for the Intel humidity and temperature datasets as more groups are considered and for a variety of tumble sizes (the trends are similar for the other data sets, which are omitted). Points in Figure 8 corresponding to the same tumble size W use bitmaps of the same length, so that the reduction ratio is 1/8 (Intel.Humidity) and 1/16 (Intel.Temperature), but differ in the number of groups utilized during the similarity estimation. It can easily be deduced that using 4 boosting groups in the humidity dataset is the optimal solution for all the cited tumble sizes, while the 4 group line tends to ascend by increasing the W parameter. This comes as no surprise since the selection of higher W values results in larger (but still 1/8 reduced) bitmaps, which in turn equip the overall comparison model with more accurate submodels. Moreover, notice that using 8 groups may provide worse results since the number of bits per group becomes smaller, thus resulting in submodels that are prone to produce low quality similarity estimations. For the same reason, in the Intel temperature dataset shown in Figure 8, where the imposed reduction ratio is 1/16, employing 2 boosting groups provides better results compared to the 4 boosting group case. In the latter plot, deviations upon switching between tumble sizes may appear to be steeper (i.e. for tumble size 24), since the 1/16 reduction causes larger discontinuities when transiting from the continuous space to the hamming cube. Notice that in the Intel temperature dataset, the application of boosting has a marginal effect on the average precision (<+4%). On the contrary, average recall values are skyrocketed with the improvement reaching a 30% percentage.

Taking one step further we extracted 95% confidence intervals for each tumble across multiple data sets. We omit the corresponding graphs, however, we note TACO exhibits little deviations (±0.04) from its average behavior in a tumble in all of the data sets.

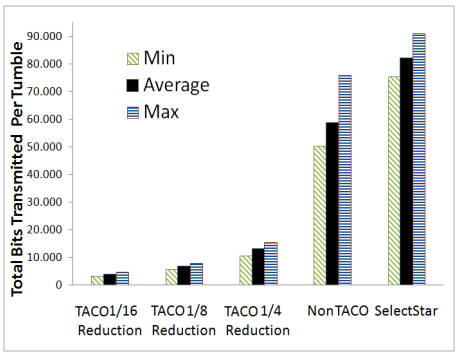


Figure 9: Total Bits Transmitted per approach

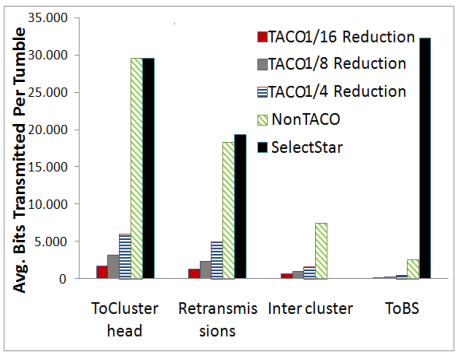


Figure 10: Transmitted bits categorization

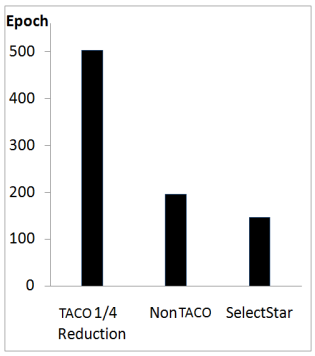


Figure 11: Average Lifetime

9.3. Performance Evaluation Using TOSSIM

Due to limitations in the simulation environment of TOSSIM, we restricted our experimental evaluation to the Intel Lab data set. We used the default TOSH_DATA_LENGTH value set to 29 bytes and applied 1/4, 1/8 and 1/16 reduction ratios to the original binary representation of tumbles containing W=16 values each.

We measured the performance of our TACO framework against two alternative approaches. The first approach, termed as *NonTACO*, performed the whole intra- and inter-cluster communication procedure using the initial value vectors of motes "as is". In the TACO and NonTACO approaches, motes producing outlying values were identified in-network, following precalculated TSP paths, and were subsequently sent to the base station by the last clusterhead in each path. In the third approach, termed as *SelectStar*, motes transmitted original value vectors to their clusterheads and, omitting the intercluster communication phase, clusterheads forwarded these values as well as their own vector to the base station.

Besides simply presenting results involving these three approaches (TACO, NonTACO and SelectStar), we further seek to analyze their bandwidth consumption during the different phases of our framework. This analysis yields some interesting comparisons. For example, the number of bits transmitted during the intracluster phase of NonTACO provide a *lower bound* for the bandwidth consumption that a simple continuous aggregate query (such as MAX, MIN or SUM query) would require under TAG for all epochs, as this quantity: (a) Simply corresponds to transmitting the data observations of each sensor to one-hop neighbors (i.e., the clusterheads), and (b) Does not contain bandwidth required for the transmission of data from the clusterheads to the base station. Thus, if TACO requires fewer transmitted bits than the intracluster phase of NonTACO, then it also requires less bandwidth than a continuous aggregate query.

Note that in our setup for TACO, during the first tumble, the base station broadcasts a message encapsulating the parameters (W,d, seed etc) of the query. The overhead of transmitting these values is included in the presented graphs.

Figure 9 depicts the average, maximum and minimum number of total bits transmitted in the network in a tumble for the TACO (with different reduction ratios), NonTACO and SelectStar approaches. Comparing, for instance, the performance of the middle case of 1/8 Reduction and the NonTACO executions, we observe that, in terms of total transmitted bits the reduction achieved by TACO is on the average 1/9 per tumble, thus exceeding the imposed 1/8 reduction ratio. The same observation holds for the other two reduction ratios. This comes as no surprise, since message collisions entailing retransmissions are more frequent with increased message sizes used in NonTACO, augmenting the total number of bits transmitted. Furthermore, comparing these results with the SelectStar approach exhibits the efficiency of the proposed inter-cluster communication phase for in-network outlier identification. The achieved reduction ratio of TACO 1/8 Reduction, when compared to the SelectStar approach is, on average 1/12, with a maximum value of 1/15. This validates the expected benefit derived by TACO.

Figure 10 presents a categorization of the average number of bits transmitted in a tumble. For each of the approaches, we categorize the transmitted bits as: (1) ToClusterhead: bits transmitted to clusterheads during the intracluster communication phase; (2) Intercluster: bits transmitted in the network during the inter-cluster communication phase (applicable only for TACO and NonTACO); (3) ToBasestation: bits transmitted from clusterheads towards the base station; (4) Retransmissions: additional bits resulting from message retransmission due to lossy communication channels or collisions. In Figure 10, please notice that the bits classified as Intercluster are always less than those in the

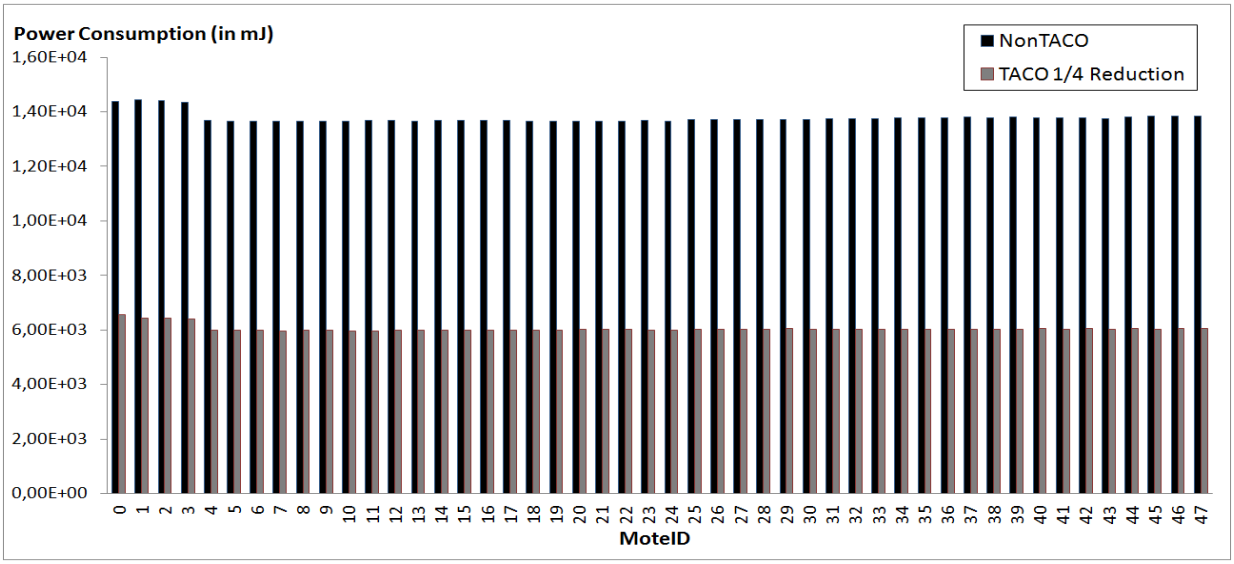


Figure 12: Power Consumption vs. MoteID

ToClusterhead category. Moreover, the total bits of TACO (shown in Figure 9), are actually less than what NonTACO requires in its intracluster phase (Figure 10), even without including the corresponding bits involving retransmissions during this phase (73% of its total retransmission bits). Based on our earlier discussion, this implies that TACO under collisions and retransmissions is able to identify outlier readings at a fraction than what even a simple aggregate query would require.

Also, we used PowerTOSSIM [61] to acquire power measurements yielded during simulation execution. Figure 12 provides an additional quantitative representation of the energy savings provided by our framework, presenting the power consumption in motes for the TACO using a reduction ratio of 1/4, and NonTACO approaches (motes 0-3 in the Figure are clusterheads). To keep the graph readable we omit the SelectStar approach, which had a much larger energy drain. Overall, the TACO application reduces the power consumption up to a factor of 1/2.7 compared to the NonTACO approach. The difference between the selected reduction ratio (1/4) and the corresponding power consumption ratio (1/2.7) stems from the fact that motes need to periodically turn on/off their radio to check whether they are recipients of any transmission attempts. This fact mainly affects the TACO implementation since in the other two approaches, where more bits are delivered in the network, the amount of time that the radio remains turned on is indeed devoted to message reception. We leave the development of a more efficient transmission/reception schedule, tailored for our TACO scheme as future work.

As a final exhibition of the energy savings provided by our framework, in Figure 11 we used the previously extracted power measurements to plot the average network lifetime for motes initialized with 5000 mJ residual energy. Network lifetime is defined as the epoch on which the first mote in the network totally drains its energy. Obviously, network lifetime is proportional to the energy savings provided by the TACO approach compared to the other techniques.

9.4. TACO vs Hierarchical Outlier Detection Techniques

In the previous sections we experimentally validated the ability of our framework to tune the amount of transmitted data while simultaneously accurately predicting outliers. On the contrary, existing in-network outlier detection techniques, such as the algorithm of [2, 5] cannot tune the amount of transmitted information. Moreover, these algorithms lack the ability to provide guarantees since they both base their decisions on partial knowledge of recent measurements received by intermediate nodes in the hierarchy from their descendant nodes.

In this subsection, we perform a comparison to the recently proposed algorithm of [2], which we will term as *Robust*. We use *Robust* as the most representative example to extract comparative results related to accuracy and bandwidth consumption since it uses an equivalent outlier definition and bases its decisions on common similarity

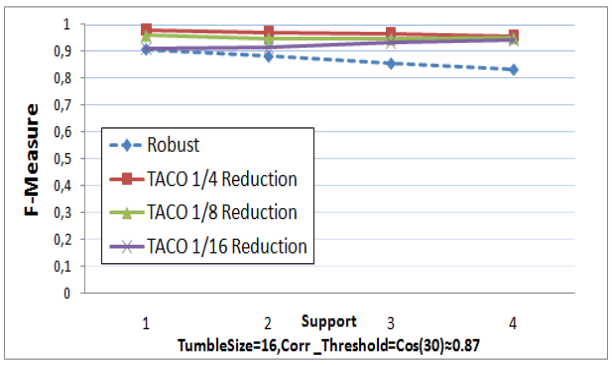


Figure 13: Intel.Temperature TACO vs Robust Accuracy varying *minSup*

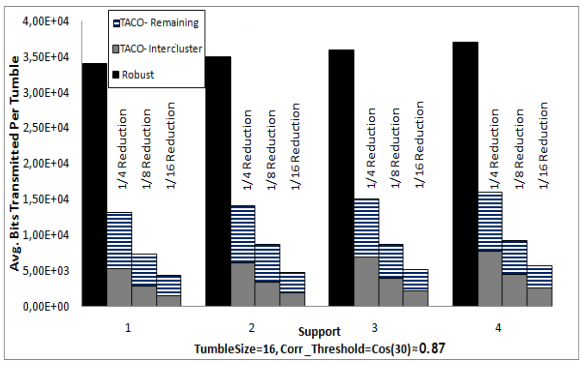


Figure 14: Intel.Temperature TACO vs Robust transmitted bits varying *minSup*

measures. As in the previous subsection, we utilized the Intel Lab data set in our study, keeping the TACO framework configuration unchanged.

In order to achieve a fair comparison, the Robust algorithm was simulated using a tree network organization of three levels (including the base station) with a $CacheSize = 24$ measurements. Note that such a configuration is a good scenario for Robust since most of the motes that can witness each other often share common parent nodes. Thus, the loss of witnesses as data ascend the tree organization is reduced. Please refer to [2] for further details.

In the evaluation, we employed the correlation coefficient-*corr* (see Table 1) as a common similarity measure equivalent to the cosine similarity as mentioned in Section 7. We chose to demonstrate results regarding the temperature measurements in the data set. However, we note that the outcome was similar for the humidity data and proportional for different Φ_{corr} thresholds. Figure 13 depicts the accuracy of Robust compared to TACO with different reduction ratios varying the *minSup* parameter. To acquire a holistic performance view of the approaches, we computed the *F-Measure* metric as *F-measure=2/(1/Precision+1/Recall)*. Notably, TACO behaves better even for the extreme case of 1/16 reduction, while Robust falls short up to 10%. To complete the picture, Figure 14 shows the average bits transmitted by motes in the two different settings. Notice that the stacked bars in the TACO approach form the total number of transmitted bits which comprises the bits devoted to intercluster communication (*TACO-Intercluster*) and those termed as *TACO-remaining* for the remainder. The increment of the *minSup* parameter in the graph correspondingly causes an increment in the *TACO-Intercluster* bits as more motes do not manage to find adequate support in their cluster and subsequently participate in the intercluster communication phase. TACO ensures less bandwidth consumption with a ratio varying from 1/2.6 for a reduction ratio of 1/4, and up to 1/7.8 for 1/16 reduction.

9.5. Bucket Node Exploitation

In order to better perceive the benefits derived from bucket node introduction, Table 3 summarizes the basic features ascribed to network clusters for different numbers B of bucket nodes. The table provides measurements regarding the average number of comparisons along with the average number of messages resulting from multi-hashed bitmaps. Moreover, it presents the average number of bitmaps received per bucket for different cluster sizes and Φ_θ thresholds. Focusing on the average number of comparisons per tumble (Comparisons in the Table), this significantly decreases as new bucket nodes are introduced in the cluster. From this point of view, we have achieved our goal since, as mentioned in Section 6.1, not only bucket nodes do alleviate the clusterhead from comparison load, but also the hash key space distribution amongst them preserves the redundant comparisons.

Studying the number of multi-hash messages (Multihash Messages in the Table) and the number of bitmaps received per bucket (Bitmaps Per Bucket) a trade-off seems to appear. The first column regards a message transmission cost mainly charged to the regular motes in a cluster, while the second involves load distribution between buckets. As new bucket nodes are adopted in the cluster, the Multihash Messages increases with a simultaneous decrease in Bitmaps Per Bucket. In other words, the introduction of more bucket nodes causes a shift in the energy consumption from clusterhead and bucket nodes to regular cluster motes. Achieving appropriate balance, aids in maintaining uniform energy consumption in the whole cluster, which in turn leads to infrequent network reorganization.

		Φ_θ					
		10			20		
Cluster Size	Buckets	Comparisons	Multihash Messages	Bitmaps Per Bucket	Comparisons	Multihash Messages	Bitmaps Per Bucket
12	1	66.00	0	12	66	0	12
	2	38.08	0.90	6.45	40.92	1.36	6.68
	4	24.55	7.71	3.65	30.95	8.88	4.08
24	1	276.00	0	24	276	0	24
	2	158.06	1.62	12.81	171.80	2.76	13.38
	4	101.10	14.97	7.27	128.63	17.61	8.15
36	1	630	0	36	630	0	36
	2	363.64	2.66	19.33	394.97	4.30	20.15
	4	230.73	22.88	10.88	291.14	26.28	12.19
48	1	1128	0	48	1128	0	48
	2	640.10	3.14	25.57	710.95	5.85	26.93
	4	412.76	30.17	14.49	518.57	34.64	16.21

Table 3: The effect of Bucket Nodes Introduction (W=16, d=128)

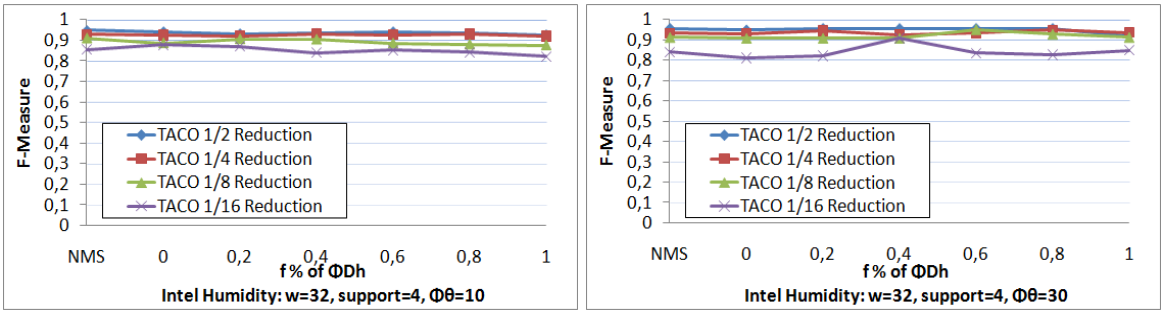
9.6. Message Suppression

Eventually, we study the effect of the message suppression strategy, introduced in Section 8, with respect to the accuracy it attributes to TACO as well as the reduction it yields on the amount of communicated data within the sensor network setting. Figure 15 plots corresponding experimental results for different reduction ratios utilizing the Intel Lab Humidity data as a representative example since the other datasets exhibit analogous behavior. We measured TACO's accuracy in terms of the F-measure metric (Section 9.4) and computed the total number of transmitted messages throughout the network operation varying the f value (which is expressed as a percentage of the respective Φ_{D_h} in the Figure) and the posed Φ_θ threshold. Particularly, we choose to present results where we set the tumble size equal to 32 since, given the default TOSH_DATA_LENGTH value of 29 bytes, the aforementioned size entails that in a single tumble motes are required to transmit 3 messages for 1/2 reduction, 2 messages for 1/4 reduction and a single message otherwise during both the intra- and inter-cluster communication. Consequently, we are able to acquire a well formed picture of the bandwidth consumption preservation provided under different circumstances.

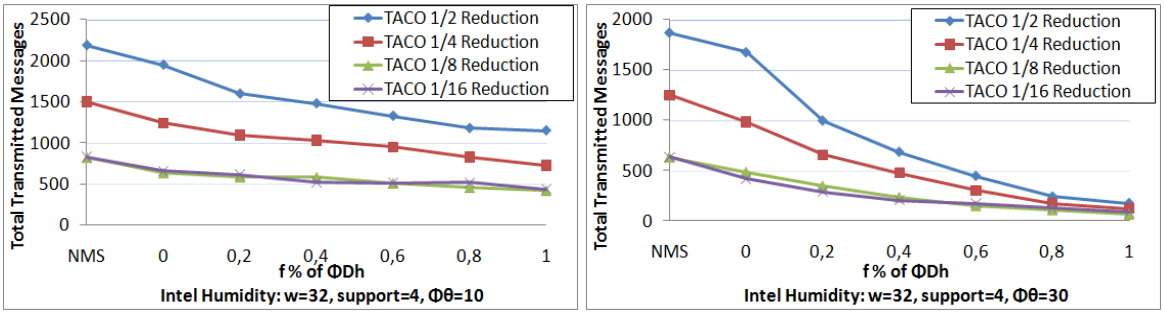
Based on Figure 15(a), it can be observed that the accuracy of the framework under message suppression mostly remains unaffected for reduction ratios up to 1/8. On the other hand, in Figure 15(b) the decrease of message transmissions reaches a factor of 1/2, and 1/11 for Φ_θ =10 and 30 degrees, respectively. The sign *NMS* in the horizontal axis of the graphs expresses the case where No Message Suppression is applied. Moreover, notice that due to the fact that f is declared as a percentage of Φ_{D_h} the greater the angle threshold, the larger the number of suppressed messages. To sum up the above discussion, we mention that message suppression is proven to equip TACO with significantly increased communication savings without precluding the accurate outlier identification as it does not exhibit high deviations from its *NMS* behavior.

10. Conclusions and Future Work

In this paper we presented TACO, a framework for detecting outliers in wireless sensor networks. Our techniques exploit locality sensitive hashing as a means to compress individual sensor readings and use a novel second level hashing mechanism to achieve intracluster comparison pruning and load balancing. TACO is largely parameterizable, as it bases its operation on a small set of intuitive application defined parameters: (i) the length of the LSH bitmaps (d), which controls the level of desired reduction; (ii) the number of recent measurements that should be taken into account when performing the similarity test (W), which can be fine-tuned depending on the application's desire to put more or less emphasis to past values; (iii) the desired similarity threshold (Φ); and (iv) the required level of support for non-outliers. Given the fact that TACO is not restricted to a monolithic definition of an outlier but, instead, it supports a number of intuitive similarity tests, the application can specialize and fine-tune the outlier detection process by choosing appropriate values for these parameters. We have also presented novel extensions to the basic TACO scheme that boost the accuracy of computing outliers. Our framework processes outliers in-network, using a novel intercluster communication phase. Our experiments demonstrated that our framework can reliably identify outlier readings using



(a) TACO's Accuracy under Message Suppression



(b) TACO's Number of Transmitted Messages under Message Suppression

Figure 15: Message Suppression on Intel Humidity datasets

a fraction of the bandwidth and energy that would otherwise be required, resulting in significantly prolonged network lifetime.

In our future work, we plan to further extend TACO so as to render it capable of efficiently handling multidimensional outlier definitions under different notions of suitable similarity measures. In addition, we are heading our efforts towards more generic scenarios where a variety of sensor data stream aspects such as time, location, measurement type [62] need to be considered for similarity estimation.

References

[1] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, J. Widom, Declarative Support for Sensor Data Cleaning, in: Pervasive, 2006.

[2] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, A. Delis, Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks, in: ICDE, 2009.

[3] S. D. Bay, M. Schwabacher, Mining distance-based outliers in near linear time with randomization and a simple pruning rule, in: KDD, 2003.

[4] A. Ghoting, S. Parthasarathy, M. Otey, Fast Mining of Distance-Based Outliers in High-Dimensional Datasets, in: SIAM, 2006.

[5] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos, Online Outlier Detection in Sensor Data Using Non-Parametric Models, in: VLDB, 2006.

[6] J. Chen, S. Kher, A. Somani, Distributed Fault Detection of Wireless Sensor Networks, in: DIWANS, 2006.

[7] X. Xiao, W. Peng, C. Hung, W. Lee, Using SensorRanks for In-Network Detection of Faulty Readings in Wireless Sensor Networks., in: MobiDE, 2007.

[8] S. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong, TAG: A Tiny Aggregation Service for ad hoc Sensor Networks, in: OSDI Conf., 2002.

[9] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, Y. Theodoridis, TACO: Tunable Approximate Computation of Outliers in Wireless Sensor Networks, in: SIGMOD, 2010.

[10] M. Charikar, Similarity estimation techniques from rounding algorithms, in: STOC, 2002.

[11] P. Indyk, Dimensionality reduction techniques for proximity problems, in: SODA, 2000.

[12] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: STOC, 1998.

[13] M. Chatterjee, S. Das, D. Turgut, WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks, Journal of Cluster Computing(Special Issue on Mobile Ad hoc Networks) 5.

[14] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-Efficient Communication Protocol for Wireless Microsensor Networks, in: HICSS, 2000.

[15] M. Qin, R. Zimmermann, VCA: An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks, J.UCS 13 (1).

[16] O. Younis, S. Fahmy, Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach, in: INFOCOM, 2004.
[17] Y. Yao, J. Gehrke, The Cougar Approach to In-Network Query Processing in Sensor Networks, SIGMOD Record 31 (3).
[18] S. Singh, M. Woo, C. S. Raghavendra, Power-aware Routing in Mobile Ad Hoc Networks, in: MobiCom, 1998.
[19] D. Zeinalipour, P. Andreou, P. Chrysanthis, G. Samaras, A. Pitsillides, The Micropulse Framework for Adaptive Waking Windows in Sensor Networks, in: MDM, 2007.
[20] M. Bawa, H. Garcia-Molina, A. Gionis, R. Motwani, Estimating Aggregates on a Peer-to-Peer Network, Tech. rep., Stanford (2003).
[21] D. Kempe, A. Dobra, J. Gehrke, Gossip-Based Computation of Aggregate Information, in: FOCS, 2003.
[22] Y. Kotidis, Snapshot Queries: Towards Data-Centric Sensor Networks., in: ICDE, 2005.
[23] Y. Zhang, N. Meratnia, P. Havinga, Outlier detection techniques for wireless sensor networks: A survey, International Journal of IEEE Communications Surveys and Tutorials 12 (2).
[24] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, J. Widom, A pipelined framework for online cleaning of sensor data streams, in: ICDE, 2006.
[25] E. Elnahrawy, B.Nath, Cleaning and querying noisy sensors, in: WSNA, 2003.
[26] S. Meng, K. Xie, G. Chen, X. Ma, G. Song, A kalman filter based approach for outlier detection in sensor networks, in: CSSE , 2008.
[27] Y. Zhuang, L. Chen, S. Wang, J. Lian, A Weighted Moving Average-based Approach for Cleaning Sensor Data, in: ICDCS, 2007.
[28] Y. Zhuang, L. Chen, In-network Outlier Cleaning for Data Collection in Sensor Networks, in: Clean DB Workshop, 2006.
[29] F. Chu, Y. Wang, D. S. Parker., C. Zaniolo, Data cleaning using belief propagation, in: IQIS, 2005.
[30] Y.-J. Wen, A. M. Agogino, K.Goebel, Fuzzy Validation and Fusion for Wireless Sensor Networks, in: ASME, 2004.
[31] B. Sheng, Q. Li, W. Mao, W. Jin, Outlier detection in sensor networks, in: MobiHoc, 2007.
[32] N. Khoussainova, M. Balazinska, D. Suciu, Towards Correcting Input Data Errors Probabilistically using Integrity Constraints, in: MobiDE, 2006.
[33] J. Branch, B. Szymanski, C. Giannella, R. Wolff, H. Kargupta, In-network outlier detection in wireless sensor networks, in: ICDCS , 2006.
[34] K. Zhang, S. Shi, H. Gao, J. Li, Unsupervised outlier detection in sensor networks using aggregation tree, in: ADMA , 2007.
[35] S. Jeffery, M. Garofalakis, M. Franklin, Adaptive Cleaning for RFID Data Streams, in: VLDB, 2006.
[36] J. Considine, M. Hadjieleftheriou, F. Li, J. Byers, G. Kollios, Robust approximate aggregation in sensor data management systems, ACM Trans. Database Syst. 34 (1) (2009) 1–35.
[37] N. Giatrakos, Y. Kotidis, A. Deligiannakis, PAO: Power-efficient Attribution of Outliers in Wireless Sensor Networks, in: DMSN, 2010.
[38] M. Otey, A. Ghoting, S. Parthasarathy, Fast distributed outlier detection in mixed-attribute data sets, Data Min. Knowl. Discov. 12 (2-3).
[39] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Hierarchical In-Network Data Aggregation with Quality Guarantees, in: EDBT, 2004.
[40] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Bandwidth Constrained Queries in Sensor Networks, The VLDB Journal.
[41] M. A. Sharaf, J. Beaver, A. Labrinidis, P. K. Chrysanthis, TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation, in: MobiDE, 2003.
[42] M. Goemans, D. Williamson, Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming, J. ACM 42 (6).
[43] W. Dong, M. Charikar, K. Li, Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces, in: SIGIR, 2008.
[44] D. Ravichandran, P. Pantel, E. Hovy, Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering, in: ACL, 2005.
[45] A. Gionis, D. Gunopulos, N. Koudas, Efficient and tunable similar set retrieval, in: SIGMOD, 2001.
[46] V. Athitsos, M. Potamias, P. Papapetrou, G. Kollios, Nearest Neighbor Retrieval Using Distance-Based Hashing, in: ICDE, 2008.
[47] K. Georgoulas, Y. Kotidis, Random Hyperplane Projection using Derived Dimensions, in: MobiDE, 2010.
[48] G. Xue, Y. Jiang, Y. You, M. Li, A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme, in: UPGRADE, 2007.
[49] M. Garofalakis, J. Gehrke, R. Rastogi, Querying and mining data streams: you only get one look a tutorial, in: SIGMOD, 2002.
[50] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, S. Zdonik, Monitoring streams: a new class of data management applications, in: VLDB, 2002, pp. 215–226.
[51] B. Karp, H. Kung, GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, in: MOBICOM, 2000.
[52] G. Gutin, A. Yeo, A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP, Discrete Applied Mathematics 117 (2002) 81–86.
[53] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, C. Yang, Finding interesting associations without support pruning, in: ICDE, 2000.
[54] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, J. Anderson, Analysis of wireless sensor networks for habitat monitoring, Kluwer Academic Publishers, 2004, pp. 399–423.
[55] T. Kristensen, Transforming tanimoto queries on real valued vectors to range queries in euclidian space, Journal of Mathematical Chemistry 48 (2010) 287–289.
[56] E. Cohen, Size-estimation framework with applications to transitive closure and reachability, J. Comput. Syst. Sci. 55 (3) (1997) 441–453.
[57] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, C. Yang, Finding Interesting Associations without Support Pruning, in: ICDE, 2000.
[58] P. Levis, N. Lee, M. Welsh, D. Culler, TOSSIM: Accurate and scalable simulation of entire TinyOS applications, in: SenSys, 2004.
[59] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong, Model-Driven Data Acquisition in Sensor Networks, in: VLDB, 2004.
[60] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Compressing Historical Information in Sensor Networks, in: ACM SIGMOD, 2004.
[61] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, M. Welsh, Simulating the Power Consumption of Large-Scale Sensor Network Applications., in: Sensys, 2004.
[62] J. Sun, S. Papadimitriou, P. Yu, Window-based tensor analysis on high-dimensional and multi-aspect streams, in: ICDM, 2006.