Dimensionality Reduction in Phenomenon-Aware Stream Query Processing

Mohamed H. Ali *mali@microsoft.com*
Microsoft Corporation
Redmond, 98052, USA

Ashish Bindra *abindra@uw.edu*
Institute of Technology
University of Washington Tacoma
Tacoma, 98402, USA

Ankur M. Teredesai *ankurt@uw.edu*
Institute of Technology
University of Washington Tacoma
Tacoma, 98402, USA

Abstract

Geographically co-located sensors are exposed to the same environmental conditions and, hence, are part of the same environmental phenomena. Phenomenon-aware stream query processing reduces the workload on the sensor's query processor by subscribing each standing query to, and only to, a subset of sensors that participate in the phenomena of interest to that query. In the case of sensors that generate readings with a multi-attribute schema, phenomena may develop across the values of one or more attributes. Ideally, a phenomenon-aware query processing system detects and tracks phenomena across all sensory attributes (all dimensions). However, such multi-attribute or multi-dimensional phenomenon-aware systems do not scale with respect to the number of dimensions.

In this paper, we present a novel n-dimensional Phenomenon Detection and Tracking mechanism (termed as nd-PDT) over n-ary sensor readings. Then, we perform dimensionality reduction from n to n' by dropping dimensions where no meaningful phenomena are detected. More interestingly, we reduce the dimensionality from n' to n'' (n''<n') by detecting various forms of functional dependencies amongst the phenomenon dimensions. We then enhance this dimensionality reduction as a continuous process that dynamically determines the number of monitored dimensions to be anywhere between 1 and n dimensions. Experimental analysis explores the trade-offs of reduced-dimensionality PDT in terms of processing cost and query accuracy.

Keywords: Data Streams, Phenomenon-aware, Query Processing, DSMS, Sensors Networks.

1. INTRODUCTION

With the advances in sensor network technology and with the growing ubiquity of sensor deployments, large amounts of sensor data is readily gathered; yet processing this data efficiently and accurately still poses significant challenges. We believe that exploring redundancies and functional dependencies among these amounts of data can help mitigate such challenges particularly in the sensor network query processing domain. Consider the fact that although sensor data exhibits tremendous variations across the entire regions of a sensor network, sensor readings tend to be similar across geographically co-located sensors over a period of time. Hence, the notion of a phenomenon, as defined in [1][2][3][4] is a group of near-by sensors that persistently read similar values over a period of time. The Challenges in the domain of

Phenomenon aware query processing are: (a) the continuous detection and tracking of phenomena as they appear, move, and disappear from the sensor field, and (b) the optimization of subsequent user queries using the knowledge of detected phenomena. The two goals are achieved by implementing two components that reside at the core of a phenomenon-aware system: the Phenomenon Detection and Tracking (PDT) module and the phenomenon-aware optimizer, respectively.

The PDT module continuously monitors correlations amongst readings arriving from near-by sensors. A phenomenon is said to be observed once a group of sensors continue generating similar values over a period of time. The phenomenon-aware optimizer then assesses the interest of a user-given query in all detected phenomena. If a query is interested in a specific phenomenon, the query is remotely deployed over (and only over) the sensors that participate in that phenomenon. If the query is interested in multiple phenomena, the query is deployed over the union of sensors participating in these phenomena.

In most cases, the number of sensors that participate in various phenomena is significantly less than the total number of sensors in the sensor field. Also, the number of phenomena that are of particular interest to query Qi, is a subset of all observed phenomena. Therefore, query Qi is deployed only over a small subset of sensors where interesting data is believed to exist. This phenomenon-guided query deployment reduces the number of queries being processed by each sensor that would have, otherwise, processed every single standing query within the system.

The PDT module has been presented in literature in [1][3][4], while phenomenon-aware query optimization has been presented in [2]. Currently, all previously conducted research assumes a single-attribute sensor schema, where sensors read single value types, e.g., temperature sensors, light intensity sensors, humidity sensors, etc. Even if the sensor's schema is a multi-attribute schema, phenomenon detection (and its subsequent optimizations) is carried over a single attribute at a time. This severely limits the applicability of phenomenon aware query processors.

In this paper, we alleviate this problem by proposing a phenomenon aware query processing system that handles sensors that generate multi-attribute readings, where phenomena may develop across some or all of the attributes. Meanwhile, there are hundreds or thousands of standing queries that are featured by conjunctive predicates over the values of some or all of these attributes. The naive approach is to deploy *n* independent PDT systems, one per attribute (or dimension). Then, the phenomenon-aware query optimizer deploys query Qi over sensor Sj if there is consensus or quorum, among all dimensions, that sensor Sj is of interest to query Qi. Note that while there is some cost to monitor a dimension in the PDT process the benefit is that it results in early filtering of some sensors that are irrelevant to query Qi. However the cost of phenomena detection over a large number of dimensions may not justified by the increased benefit of reducing the query deployment to a smaller subset of sensors. The challenge then is to find the right set of dimensions that balance the PDT cost and the gained reduction in the query deployment map.

By developing an *n* dimensional phenomenon-aware system, we highlight three important observations. First, for a given domain, meaningful phenomena tend to appear along a subset of the dimensions. If we have n dimension, only n' dimensions (where $n' < n$) exhibit correlations across sensor readings and, hence, result in detected phenomena. Second, the appearance/disappearance of phenomena across different dimensions does not seem to be random. Instead, whenever a phenomenon is detected in one dimension, other dimensions tend to report phenomena simultaneously or within a short period of time. Relating this observation to real life, we understand that whenever a fire strikes a sensor field, there will be an observable phenomenon over the ‘temperature’ dimension, accompanied by another simultaneous phenomenon over the 'light intensity' dimension. The same thing happens whenever a phenomenon disappears from the sensor field. Third, behavioral changes in the environment readings are not only expected but they can also be frequent enough to invalidate the efficiency of the chosen set of monitored dimensions for phenomenon-aware optimizations.

The above three observations shape the research conducted in the context of this paper and our contributions can be summarized as follows:

1. We enhance phenomenon-aware query processors with the ability to detect phenomena across multiple dimensions.

2. We present a two-step dimensionality reduction technique. The first step filters-out the dimensions where no interesting phenomena are detected and the second step then eliminates functionally dependent dimensions.

3. We further extend the multi-dimensional PDT and make it dynamic such that it can handle behavioral changes in the streaming environment by continuously adjusting the monitored set of dimensions. Dimensions are added and removed dynamically to balance between the cost of phenomenon detection across multiple dimensions and gained benefit of deploying queries over smaller subsets of sensors.

4. We experimentally evaluate and compare the multi-dimensional phenomenon-aware query processor and its reduced-dimensionality variants in terms of processing costs and query accuracy and present the results.

The remainder of this paper is organized as follows. Section 2 provides the background on phenomenon-aware systems. Section 3 outlines the issues involved in designing multidimensional phenomenon-aware query processing systems. Section 4 focuses on the proposed dimensionality reduction techniques and presents the system's ability to be adaptive and responsive to changes in the behavior of the underlying stream sources. Section 5 provides an experimental analysis of the proposed stream query processor along with its reduced dimensionality and adaptive variants. An overview of the related work is given in Section 6. Section 7 concludes the paper.

2. BACKGROUND

This section summarizes the basic definitions and the key concepts of phenomenon-aware stream query processing. More specifically, we present a definition for a phenomenon and overview the architecture of phenomenon-aware systems. Then, we discuss the cost-benefits trade-off of such systems.

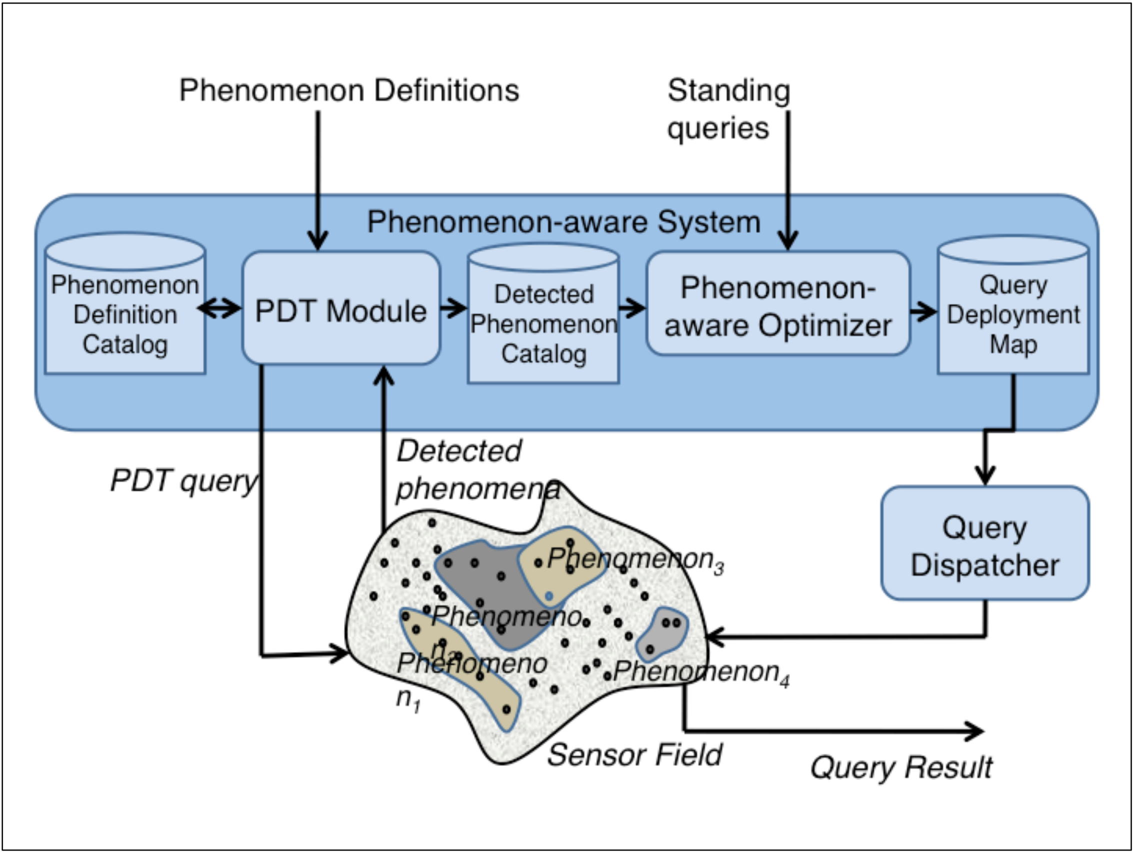

Figure 1: Architecture of a phenomenon-aware system.

A phenomenon is a group of near-by sensors that persistently generate similar behaviors over a period of time [1][4]. Definition 1 provides a high level definition of a phenomenon that captures the notion of similarity among sensors. The notion of similarity has been extensively studied by several research groups, yet under different terminologies, e.g., phenomena, isobars, homogenous regions, deformable 2D objects, etc. ([5][10][11][15][16][19]). Section 6 overviews and compares amongst some of these similarity-based techniques.

Definition 1: In a sensor network SN, a phenomenon takes place only when a set of sensors S in SN report similar reading values more than α times within a time window w.

Definition 2: A phenomenon Pτ at time instant τ is a binary tuple (Rτ ,Bw), where Rτ is the bounding region of phenomenon Pτ at time instant τ and Bw is the representative behavior of phenomenon Pτ over the most recent time window of size w, S.T. stream Si Є Rτ , Prob(lBw (Si) – Bwl ≥ ε) ≤ α.

Definition 2 (as presented in [1]) provides a formal definition of phenomenon. Based on Definition 2, a phenomenon is associated with a time instant τ because a phenomenon may change its location (Rτ) and behavior (Bw) over time. Also, the representative behavior of a phenomenon Bw is captured over a window of time (w) to ensure its persistency and to avoid the effect of noise. A stream source Si that lies in the phenomenon region Rτ should report a behavior (Bw(Si)) that is similar to the phenomenon representative behavior Bw with high probability (i.e., Prob(lBw (Si) – Bwl ≥ ε) ≤ α). Bw, the phenomenon representative behavior, captures the intrinsic features of the underlying phenomenon, e.g., values, frequencies, and trends of tuples contributing to the phenomenon.

Figure 1 illustrates the architecture of phenomenon-aware systems. Phenomenon-aware systems receive, as input, a set of phenomenon definitions. These definitions are registered and stored in a special system catalog called phenomenon definition catalog. The first major component in phenomenon aware systems is the phenomenon detection and tracking module (PDT), which translates a phenomenon definition into a continuous PDT query. The continuous PDT query is distributed over the sensor network to be executed in-network. The outcome of this query is a set of detected phenomenon on the format of specified by Definition 2. The detection algorithms, both the centralized and distributed versions, are presented in [3]. Detected phenomena are stored in the detected phenomenon catalog and are updated continuously in response to the continuous evaluation of PDT query.

The phenomenon-aware optimizer, which is the second major component of the system, assesses the similarity between the expected query result and the phenomena stored in the detected phenomenon catalog. The optimizer binds each query to a subset of the detected phenomena that are believed to contain sensor readings of interest to the query. Equivalently, the optimizer binds each query to (and only to) the sensors that participate in the query's subset of interesting phenomena. These selected sensors are termed the query's working set of sensors. Then, the query dispatcher deploys each query remotely to (and only to) the sensors that belong to the query's working set, and hence, achieves an efficient query deployment map. The phenomenon-aware optimizer is presented in [2].

Note that the execution of the PDT modules comes at the cost of taking some processing cycles from the sensor's scarce resources to continuously detect and track phenomena. However, the detected phenomena are used by the phenomenon-aware optimizer to efficiently deploy the query over a smaller subset of sensors. Therefore, the remaining CPU cycles of the sensor will be invested efficiently in processing queries (and only queries) that are expected to be satisfied by the sensor's readings. The phenomenon-aware query processing has been implemented in the context of the Nile [11] data stream management system developed at Purdue University.

3. MULTI-DIMENSIONAL PHEMOMENON-AWARE QUERY PROCESSINGS

Phenomenon-aware query processing has proved its efficiency in the context of single attribute sensor readings, i.e., the schema of incoming sensor tuples consist of a single field. Experiments have been conducted over heat sensors, light sensors, humidity sensors, etc. However, saying that the concept of phenomenon-aware query processing established its position and consolidated its basic concepts in the single-attribute domain, research ambitions have been directed towards sensors with multi-attribute schemas. Moreover, thanks to technology advances, most commercial sensors in the market generate a multi-attribute schema. Motivated by this, we focus on the concept of multi-attribute or multi-dimensional phenomenon-aware query processing in this paper. Examples of sensors that generate temperature, humidity, light, speed, acceleration, and other different reading attributes in the same tuples include, but are not limited to, [22] and [23].

In a multi-dimensional domain of sensor readings, similarity (and consequently, phenomena) can be expected to develop along any dimension. Given an n_d dimensional sensor reading and given n_q standing queries, a sketch for a naive multi-dimensional phenomenon-aware system that deploys queries over their interesting sensors (a.k.a. the query's working set of sensors) is as follows:

```
Algorithm: Naive Multi-dimensional Phenomenon-aware Query Processing
Input:
  • Given multi-dimensional sensor readings with dimensionality of $n_d$
  • Given $n_q$ standing query with conjunctive predicates over sensor readings
Output:
  • Provide an efficient query deployment map that deploys query $Q_i$ over Sensor $S_i$
    such that Probability
    $P(Si.value \in Qi.QueryResult) > \varepsilon, where\ \varepsilon\ is\ a\ preset\ threshold.$
Steps:
  1. For every dimension run an independent PDT module to detect phenomena across
     every dimension
  2. For every query, the query working set is the intersection of the query's working sets
     across "all" dimensions.
  3. Deploy query on its workings set of sensors
```

The algorithm above suggests that we run n_d independent *PDT* modules (one module per dimension), where n_d is the dimensionality of the sensor reading (Step1). Then for every query, we compute the query's working set over every dimension independently. Then the query's working set is the intersection of the query's workings set from all dimensions (Step 2). This step is interpreted as follows: a sensor belongs to the query's working set if there is a consensus among all dimensions (or at least a quorum) that this sensor generates data of interest to the query.

n_d	Number of PDT monitored dimensions
n_q	Number of standing queries at the sensor's query processor
n_{input}	Number of input events
$T_{acquisition}$	Acquisition time: total time taken to acquire n_{input} events from the environment and enqueue them at the sensor's input buffers
$T_{processing}$	Processing time: total time taken to process n_{input} events for n_q standing queries
$t_{processing}$	Processing time for a single tuple per single query
T_{PDT}	The cost of a single PDT measured in time units
Sel_i	Selectivity of query number *i*

Table 1: Summary of the costing formula symbols.

To better understand a multi-dimensional phenomenon-aware system and to build the correct expectations for the performance of such systems, we devote the remainder of this section to formalize the expected behavior of the system in terms of its throughput. Table 1 provides a summary of the symbols used in these costing equations.

$$Thruput_{acquisition} = \frac{n_{input}}{\mathrm{T}_{acquisition}} \qquad (1)$$

$$Thruput_{input} = \frac{n_{input} \times n_q}{\mathrm{T}_{acquisition}} \tag{2}$$

$$Thruput_{output} = \frac{n_{input} \times \sum_i^{n_q} Sel_i}{\mathrm{T}_{acquisition} + \mathrm{T}_{\text{processing}}} \tag{3}$$

$$\mathrm{T}_{\text{processing}} = n_{input} \times n_q \times t_{processing} + n_d \times T_{PDT} \tag{4}$$

$$Thruput_{output} = \frac{n_{input} \times \sum_i^{n_q} Sel_i}{\mathrm{T}_{acquisition} + (n_{input} \times n_q \times t_{processing} + n_d \times T_{PDT})} \tag{5}$$

We calculate three different types of throughput. (1) The *acquisition throughput*, which is the total number of tuples acquired by the sensor per time unit (Equation 1). (2) The *input throughput*, which is the acquisition throughput multiplied by the number of standing queries (Equation 2). The input throughput represents the total number of tuples that need to be processed by the sensor's query processor. (3) The *output throughput*, which is the number of output tuples coming out of the sensor's query processor. Equation 3 shows that the output throughput equals number of input events (after imposing the selectivity of each standing query) divided by the total time (acquisition time and processing time). Higher output throughput indicates that the sensor's processing cycles are invested efficiently to direct input tuples to the right queries, i.e., queries that are interested in those readings.

Equation 4 captures the processing time as: time spent in queries (i.e., the time taken to process a single tuple times the number of input tuples times the number of standing queries) and time spent in the *PDT* modules, which is considered an overhead for phenomenon-aware query processors. Substituting the $T_{processing}$ in Equation 3, we get Equation 5.

$$n_q \propto \frac{1}{F_1(n_d)} \tag{6}$$

$$Sel_i \propto F_2(n_d) \tag{7}$$

Equation 6 and 7 show that the number of standing queries per sensors (n_q) decreases, and meanwhile, each standing query achieves a higher selectivity (Sel_i), as we increase the number of monitored *PDT* dimensions (n_d). By substituting for n_q and Sel_i in Equation 5, we find that the output throughput increases due to the higher selectivity gained by increasing the number of *PDT* monitored dimensions. However, the output throughput decreases as we increase the number of *PDT* monitored dimensions due to the T_{PDT} cost itself; which may be prohibitive and may dominate the sensor's processing cycles. Hence, the selection of the right set of dimensions to be monitored significantly impacts the output throughput and achieves a balance between the T_{PDT} cost and the gained selectivity. Section 4 focuses on how to select the right set of dimensions to monitor the appearance and disappearance of phenomena.

4. DIMENSIONALITY REDUCTION

In Section 3 we discussed the importance of the n_d (the number of monitored dimensions) tuning parameter. More precisely, the efficiency of a multi-dimensional phenomenon-aware system relies on the careful choice of the *monitored dimension set*: (1) how many dimensions to monitor and (2) which dimensions out of the full set of dimensions to select. In this section, we address the choice of the *monitored dimension set* (*MDS*) as follows:

- **Step1:** Given a *MDS* of size n_d, what is the optimal choice of another *MDS'* of size n_d *-1*

- **Step 2**: Given an initial *MDS* of size n_d*.initial*, what is the optimal choice of a desired *MDS* of size n_d*.desired*

Step 1 is a single step dimensionality reduction process that reduces n_d by a single dimension. Step 2 specifies the stopping criterion to stop the dimensionality reduction process. The remainder of this section presents two algorithms to perform step 1. Section 4.1 presents an algorithm that eliminates dimensions with no (or "few") phenomena that are of interest to standing queries and introduces a concrete measure for what we mean by "few" phenomena. Section 4.2 eliminates dimensions that are functionally dependent on other dimensions. Section 4.3 presents the stopping criterion for the dimensionality reduction process.

4.1 Eliminating dimensions with no phenomena

Running the *PDT* modules on dimensions that have no phenomena at all or that have few phenomena incur additional overhead on the sensor's CPU without any significant reduction in the query deployment map. We quantify the effectiveness of running a *PDT* module on a given dimension in terms of the sensor's expected throughput. We make use of two input parameters that are computed continuously by the PDT module while the phenomenon detection and tracking process is in progress:

- Sensor Participation Ratio (*SPR*): the average number of phenomena a single sensor participates in divided by the total number of phenomena.
- Tuple Participation Ratio (*TPR*): the average number of tuples that participate in a phenomenon divided by the total number of tuples.

Also, based on the systems workload, we assume the availability of the following parameters:

- Query-Per-Phenomenon (*QPP*): the average of number of queries interested in each given phenomenon.
- Phenomenon-Per-Query (*PPQ*): the average of number of phenomena that each query is interested in.

SPR reflects the number of phenomenon a sensor will be monitoring, while QPR represents how many queries are interested in each phenomenon, multiplying *SPR* by *QPR* gives the number of queries that are expected to run on a single sensor. Thus,

$$n_q = SPR \times QPP \qquad (8)$$

The expected selectivity of each query is the number of phenomena that are of interest to the query multiplied by average percentage of tuples that participate in a phenomenon.

$$Sel_i = TPR \times QPR \qquad (9)$$

We substitute for n_q and Sel_i using equations (8) and (9) in equation (5) to evaluate the expected throughput for each dimension. Then, we eliminate the dimension with the least expected throughput. Note that for the correctness of the equations above, there are two implicit assumptions: (1) We assume that sets of queries that are interested in two different phenomena are disjoint. This assumption allows us to assume that the overall selectivity is the summation of all individual query selectivities. While this assumption is not hundred percent true in practice, it simplifies the model without much distortion to the expected behavior. Accounting for overlap in the query's interesting set of phenomena is possible but omitted in this paper for brevity. (2) We assume that every query is interested in one or more phenomena. This allows us to simplify and exclude the cases where some queries are interested in sensor values that are not part of any phenomena.

4.2 Eliminating functionally dependent dimensions

It is common in multi-attribute sensors that an increase/decrease in one attribute's readings implies an increase/decrease in another attribute's readings. Generally speaking, the trend of one attribute readings is related to the trend of another attribute readings by a functional correlation *F* (i.e., $Attr_1.value = F(Attr_2.value)$). For example, the increase in the readings of the temperature attribute, in case of a fire, is usually accompanied by an increase in the carbon monoxide (smoke) detector attribute readings.
Such correlations have been previously exploited to develop model based approximate query processors [24][25]. For simplicity, we limit the correlation function *F* to monitor correlations to follow a linear relationship on the form of:

$Attr_1.value = a \times Attr_2.value + b$, where x and y are dimensions and a, b are constants. (10)

We assume that linear correlation is sufficient for a wide range of applications. However, extending the model to higher orders of correlation is also straightforward. Since, deriving the functional dependence between attributes is not the focus of this paper, we assume that if such dependencies exist they can be identified and expressed within the phenomenon detection and tracking process. In the remainder of this section, we answer two interesting questions: First, given two correlated dimensions, which dimension is to be removed to increase efficiency? Second, given that we removed the dimension $Attr_1$ (say) which is correlated with $Attr_2$, how the phenomenon-aware query optimizer could use $Attr_1$ to filter the query predicates against $Attr_2$.
Between two correlated dimensions, we eliminate one dimension or the other based on the throughput equation (Equation 5). We evaluate the throughput of the system assuming that $Attr_1$ has been removed. We repeat the throughput evaluation assuming that $Attr_2$ has been removed. Then, we decide to eliminate the attribute whose removal maximizes throughput. The intuition behind such a decision is that, given two correlated dimensions that are expected to detect a similar set of phenomena, we remove the dimension whose phenomenon detection cost is higher, or the dimension with tuples that satisfy fewer query predicates.

Given a query predicate on Attr1 on the form of $Attr_1.Value_{Left} < Attr_1 < Attr_1.value_{Right}$, where $Attr_1.value = a \times Attr_2.value + b$ and where $Attr_1$ has been eliminated from the system. Figure 2 shows that the predicates can be transformed to the other dimension. Then, the query interest in the detected phenomena is carried over to the retained dimension. More specifically, we calculate the values of $Attr_2.Value_{Left}$ and $Attr_2.value_{Right}$ using Equation 11. Then, we transform the predicate into $Attr_2.Value_{Left} < Attr_2 < Attr_2.value_{Right}$, where ε is the correlation error

$$Attr_2 = \frac{Attr_1 - b}{a} \tag{11}$$

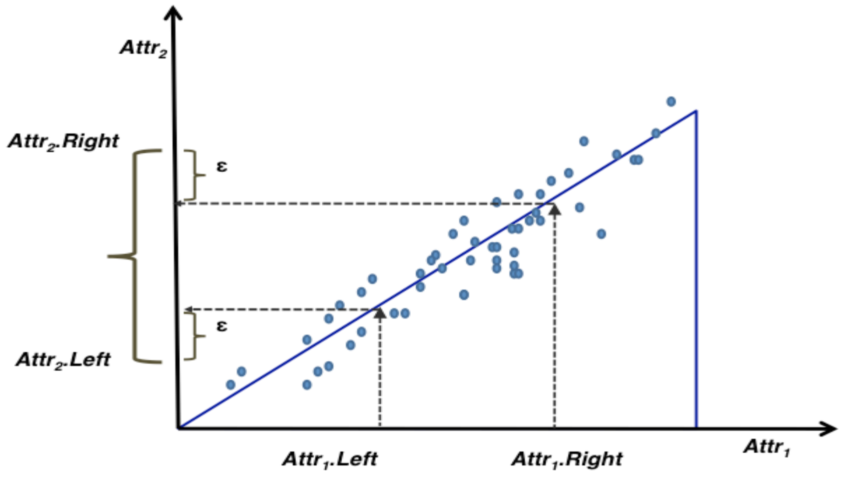

Figure 2: Transforming predicates across correlated dimensions

4.3 The stopping criterion for dimensionality reduction

As we discussed earlier, iteratively determining the correlated dimensions and increasing the system throughput is the central idea. Unfortunately, it is not trivial to correlate all dimensions with each other to produce an optimal set. Hence we suggest the following strategy to determine the stopping criterion for determining the optimal size of the monitored set.

Step 1: Start with *d* dimensions. Measure the throughput in Equation 5.

Step 2: Reduce one dimension ($d_{new}=d-1$) where no phenomena are found or where correlation is detected then measure throughput again.

Step 3: If the$\Delta thruput = Thruput_{output}(d) - Thruput_{output}(d-1) < \varepsilon, where\ \varepsilon \approx 0,$ then stop iterating, otherwise perform step 2 again.

This strategy of deriving an optimal set is easy to implement and meets the needs of incrementally reducing the dimension set. Note that we suggest performing step1 through step 3 in two phases: *Phase 1*: we remove all dimensions with no phenomena or with few phenomena as described in Section 4.1. *Phase 2*: remove all correlated dimensions as described in Section 4.2. The rationale behind this is to avoid the correlation cost between dimensions that have a limited number of interesting phenomena.

5. EXPERIMENT

In this section, we evaluate the performance of the proposed Multi-dimensional Phenomenon-aware Stream Query Processor. We first discuss the performance metrics, the experimental parameters, and other background information regarding reproducibility of our experiments. Then, we discuss the experimental results.

The performance metric used throughout this analysis is *accuracy*, where we define accuracy θ as,

$$\theta = \frac{\varphi}{\gamma} \tag{10}$$

Where γ is the number of "*exact"* output events that are generated by an imaginary system with infinite resources and φ is the number of "*actual"* output events generated by our proposed system. Typically, $\varphi < \gamma$ since our system will eventually fail to cope with all events once we increase the sampling rate to a level that exceeds the sensors' processing capabilities. With respect to throughputs as described in section 4, accuracy can be expressed as:

$$\theta = Thruput_{output} / Thruput_{output_{under\ infinite\ procesing\ resources}} \tag{11}$$

A sensor node has limited processing cycles to sample the environment, process the sampled input and satisfy the standing queries, within a given time period. As the number of queries increases or as the rate of incoming samples increases, a sensor will not have the capacity to process every sample against every query. Furthermore, the storage space (buffers) is also restrictive, and queries are time sensitive, which results in data samples being dropped before newer ones can be processed. Thus, accuracy tends to decrease as the number of registered queries or the sampling rates increase.

5.1 Experimental Setup

We measure and compare the accuracy (θ) across different implementation strategies. These strategies are summarized in Table 2. Each sensor in our simulation is multi-attribute or multidimensional. Each sensor samples its environment at a sampling rate that is fixed over the course of a single experimental run. The data set is generated via a multi-modal normally distributed random number generator for each attribute. Our sensors have a *fixed size circular buffers* into which new readings are inserted. If the buffer is full when a new tuple is added the oldest tuple is dropped from the buffer.

Strategy	**Description**
NoPDT	Naïve implementation without phenomenon aware optimization. Every query is dispatched over every sensor.
SinglePDT	PDT across a single attribute.
DoublePDT	PDT across exactly two attributes.
TriplePDT	PDT across exactly three attributes.
CorPDT	PDT with correlation detection and dimensionality reduction across other attributes.

Table 2: Strategies considered in the experimental setup

Parameter	**Default Value**
Sampling Rate	0.04 samples per millisecond
Number of Queries	20000
Number of Sensors	5000
Selectivity	15%

Dimensions (Attributes)	3

Table 3: Simulation Parameter Settings

Our queries are assumed to be long running range queries and are randomly generated with a given selectivity. Recall that the output throughput (equation 5), increases by decreasing the processing cost and increasing selectivity. For our experiments, both the number of queries and the selectivity are preset to the values described in Table 3. All our queries are range queries in conjunctive normal form:

$$(a_1 \geq x_1 \wedge a_1 \leq \langle x_1 + s \rangle) \wedge (a_2 \geq x_2 \ \wedge a_2 \leq \langle x_2 + s \rangle) \wedge \ldots \wedge (a_n \geq x_n \ \wedge a_n \leq \langle x_n + s \rangle)$$

Where a_i represents an attribute, x_i represents a value in the domain of the attribute, *s* represents the range of the query, and *(x_i + s)* is always less than or equal to the maximum value in the attribute's domain.

Our simulator is implemented in Java 1.6 and all our experiments run on a machine with a 2.4 GHz Intel Core 2 Duo and 4 GB of RAM.

5.2 The Effect of the Sampling Rate

In this experiment, we study the accuracy of all five strategies under a variable sampling rate. Within each strategy, we expect accuracy to decrease with an increase in the sampling rate. Figure 3 shows that the output accuracy decreases as we increase the sampling rate due to the scarcity of processing cycles. The results indicate that single PDT and DoublePDT perform better than NoPDT. However, as we increase the number of PDTs (TriplePDT), most of the sensor node's processing cycles are spent on the phenomenon detection process and maintenance of the PDTs. Consequently, there are not enough cycles left to spend on the actual queries and the output accuracy decreases substantially. The strategy of maintaining a correlated PDT seems to be by far the most efficient; thereby supporting the motivation behind the proposed approach. As we keep increasing the sampling rate, accuracy drops across all strategies due to the sensor's limited processing cycles in front of the high rate input stream.

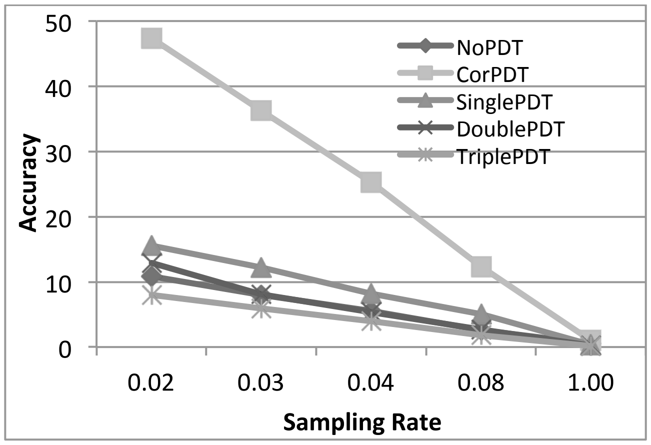

Figure 3: The effect of sampling rate on query accuracy.

5.3 The Effect of the Number of Standing Queries

In this experiment, we study the effect of increasing the number of standing queries on the measured accuracy. Similar to effect of increasing the sampling rate, we expect accuracy to decrease as we increase the number of standing queries. The processing cycles become scarce as they get divided amongst the standing queries. This behavior is depicted in Figure 4. Again, correlated PDT outperforms other strategies.

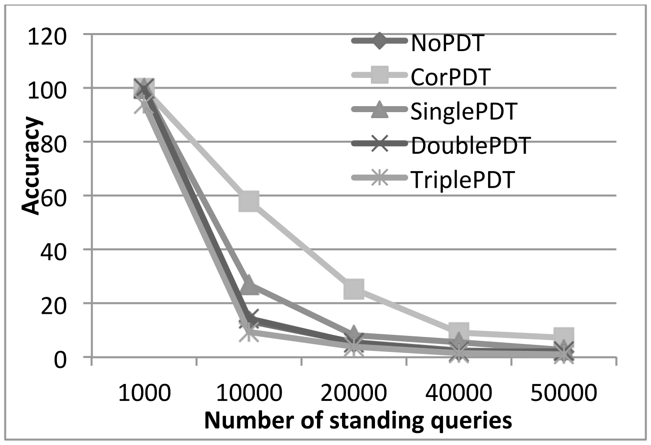

Figure 4: The effect of the number of standing queries on accuracy

5.4 The Effect of the number of Sensors

In this experiment, we vary the number of sensors. Since each sensor is an independent processing unit, we do not expect accuracy to be affected with the change in the number of sensors. Figure 5 shows how accuracy stays mostly constant with marginal variation in response to the increase in the number of sensors.

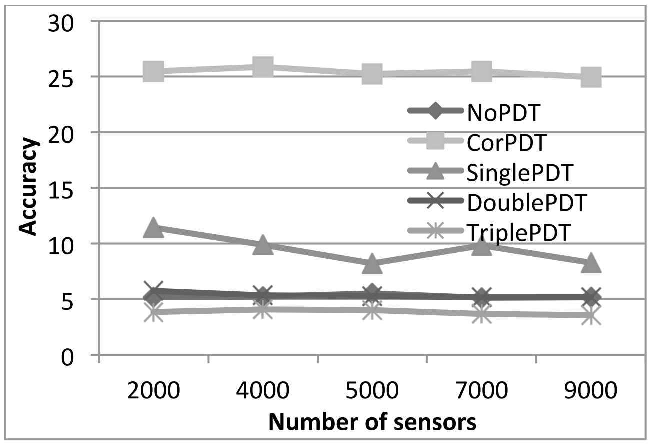

Figure 5: The effect of the number of sensors on accuracy

5.5 The Effect of the Range Query Selectivity

In this experiment, we vary the selectivity of the range query predicates and examine its effect on the output accuracy. It is interesting to note that the selectivity effect is more pronounced for CorPDT. This is because the performance gains of CorPDT come from transforming the query predicate from one dimension (the monitored dimension) to other dimensions (the correlated dimensions) with the help of the detected and interpolated correlation function. As the range of the query predicates gets bigger, the transformation seems to get diluted and to become less selective over the correlated dimensions. Figure 6 shows the effect of selectivity on the output accuracy for all strategies.

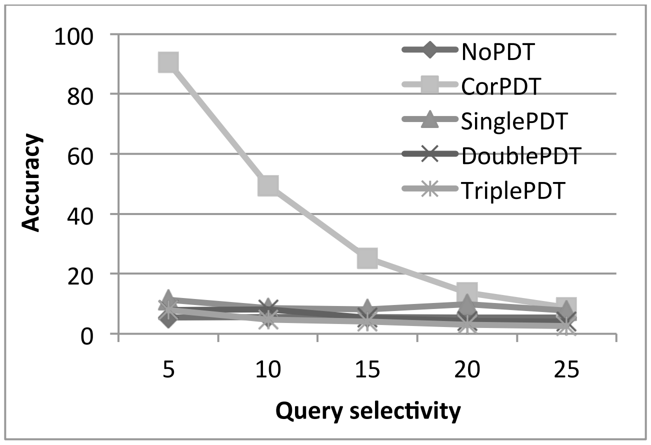

Figure 6: Effect of selectivity on accuracy

5.6 Adaptivity and Responsiveness to Drifts in Stream Behavior and Correlation

PDT-aware optimizations select the correct set of dimensions based on the observed behavior of the incoming streams over a window of time. In particular, the intrinsic properties of the detected phenomena along each dimension plus the correlation across different dimensions are key factors in the selected dimensions. However, and as an example from our heat, light intensity and smoke detector sensors, every change in the light intensity does not have to mean a fire and does not have to show an increase in temperature. Also, every increase in temperature does not have to be accompanied by sensing an increased level of smoke. Correlation amongst attributes is expected to changes over time.

In this section, our goal is to compare an adaptive CorPDT against a static CorPDT in circumstances where the correlation among attributes is non-static. An adaptive CorPDT is defined to be a CorPDT that reconsiders the observed correlation every *n* units of time (*n* = 30 seconds in this experiment). In other words, the correlation among attributes is reevaluated while the system is running. Figure , 8 and 9 shows the effect of the sampling rate, the number of queries and the selectivity on both adaptive and non-adaptive CorPDTs. Even though NoPDT does not depend on correlation, NoPDT is included in the figures as a reference point.

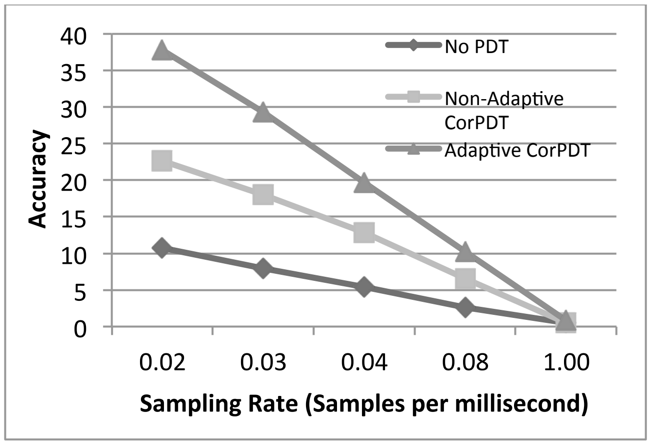

Figure 7: The effect of the sampling rate on accuracy

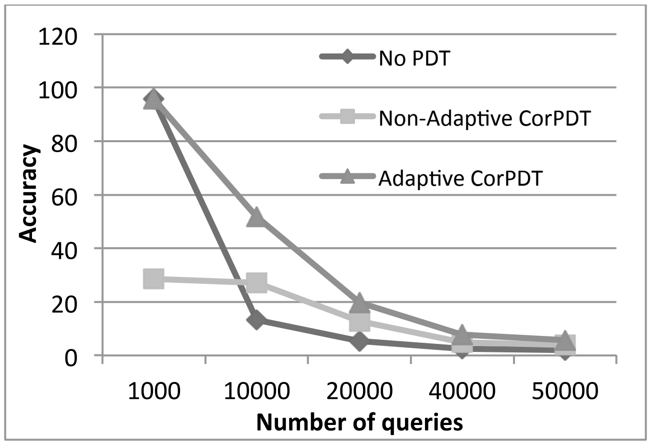

Figure 8: The effect of the number of queries on accuracy

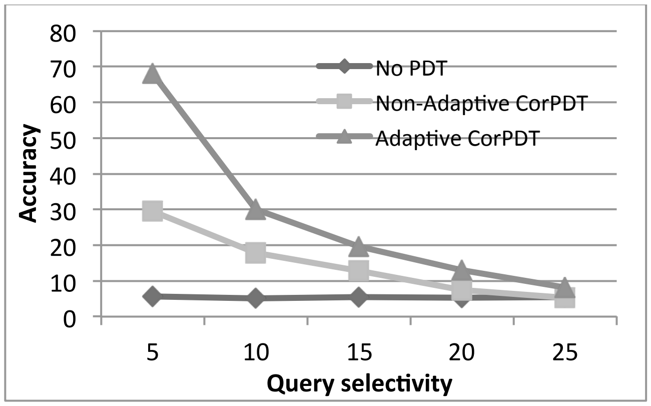

Figure 9: The effect of selectivity on accuracy

6. RELATED WORK

The advances in the sensor technology, coupled with wide availability of GPS embedded devices, directed the research focus of the GIS community towards spatio-temporal data stream processing. There has been an ongoing effort to share and explore the generated data streams and amortize the sensor deployment cost efficiently among sensor owners through an open and scalable infrastructure, called *SenseWeb* [10]. Out of several research directions, we consider the areas of (1) continuous query processing of spatio-temporal streams, (2) similarity detection across geosensor fields, and (3) efficient algorithms for object and region tracking to be most relevant to our work. Note that the phenomenon-aware systems are tracking-based systems that track similarity-based regions through continuous query processing and optimization over spatio-temporal streams.

Examples of continuous query processing over data streams generated by GPS-enabled sensors and mobile objects include [14] and [18]. More specifically, object tracking in sensor networks using continuous queries has been addressed by several research groups. For example, [3][9] and [12] propose *join* algorithms over a sliding window that can be used to track moving objects in a sensor field. We distinguish between two types of tracking: object tracking and region tracking. Object tracking (e.g., [9] and [12] tracks a single object in the sensor field. Region tracking (e.g., [1][2][4][6][7][13][16]) tracks the movements of regions that show similar behavior. Phenomenon detection falls under the second category (i.e., region tracking).

A framework to track phenomena inside a DSMS is proposed in [1], [3] and [4]. Detection of boundaries that separate homogeneous regions of sensors is investigated in [19]. In [7], streams of sensor data that have approximately the same value are grouped into continuous regions called *isobars*. The work in [13] identifies an aggregate picture of a sensor network's conditions/states that enables online monitoring of evolving phenomena. All these methods demonstrate the capacity to successfully estimate phenomenon. Our efforts extend these to include a multi-dimensional phenomenon index to keep track of which sensors participate in each query. Given a database of object trajectories, [16] refers to a set of objects, which move together as a cluster. In [8], the authors focus on topological changes (e.g., region merging/splitting, and hole formation/elimination) in areas of high activity during the evolution of a field to monitor geographical phenomena. The efforts in [17] bring up added value for the notion of similarity by geographically mining for the similarity among users based on their location histories. In [6] the notion of phenomena in geosensor networks is abstracted to 2D objects and presents an in-network energy-efficient algorithm based on the concept of deformable curves to incrementally track spatiotemporal changes of the object. The work in [5] extends the notion of similarity among sensor values to similarity among the trajectories of moving objects. The emphasis of our work is distinct in the sense that we develop a generic framework where any of these notions of similarity can be plugged in to provide additional throughput.

Taking into account the spatial properties and the network topology while detecting various notion of similarity, a wide spectrum of energy-efficient techniques have been proposed to transmit sensor reading over the network. For example, [21] proposes a prediction based strategy to reduce power consumption by focusing on regions where moving objects are expected to appear. To optimize for the tracking process, [20] reconfigures a tree-like communication structure of a sensor network dynamically. The work in [15] makes use of similarity among sensors to divide the responsibility of data transmission among sensors that are geographically co-located in the same cluster.

7. CONCLUSION

Developing a query processor for aggregating geographically distributed information using continuous queries over data streams poses significant challenges due to the efficiency issues in widespread deployment of hundreds of continuous queries over thousands of sensor nodes to monitor tens of dimensions. In a geographically distributed sensor network, processes that exhibit similarity in behavior over time are termed as phenomena. We can obtain significant efficiency gains by developing systems capable of detecting and tracking phenomena occurring in multiple dimensions. Queries can then be deployed intelligently to run on limited nodes that participate in the phenomenon of interest. Moreover, functional correlations between dimensions can be exploited to further improve the performance of such query processors.

In this paper, we presented a novel *n*-*d*imensional *P*henomenon *D*etection and *T*racking mechanism that reduces both the number of sensors and the number of dimensions over *n*-ary sensor readings on which a continuous query is deployed. We performed dimensionality reduction from n to n' by dropping dimensions where no meaningful phenomena were detected. We next used functional correlation to reduce the dimensionality further, from n' to n'' by detecting various forms of functional dependencies amongst the phenomenon dimensions. We then enhance the performance of this dimensionality reduction by making it an adaptive continuous process that dynamically determines the number of monitored dimensions. We addressed the design issues for each of these advances and developed the metrics that can be used to judge the utility of the proposed system. Experiments and results demonstrate that the multi-dimensional phenomenon-aware stream query processing system significantly outperforms a basic stream query processing system and provides enhanced efficiency gains compared to a single dimensional phenomenon-aware system. The adaptive components of the proposed system further enhance its utility in comparison to existing efforts.

8. REFERENCES

[1] Mohamed H. Ali, Mohamed F. Mokbel, Walid G. Aref, Ibrahim Kamel. “Detection and Tracking of Discrete Phenomena in Sensor-Network Databases”. In SSDBM, 2005.

[2] Mohamed H. Ali, M. F. Mokbel, and W. G. Aref. Phenomenon-aware stream query processing. In MDM, 2007.

[3] Mohamed H. Ali, Walid G. Aref, Ibrahim Kamel: Scalability Management in Sensor-Network PhenomenaBases. In SSDBM, 2006.

[4] Mohamed H. Ali, Walid G. Aref, Raja Bose, Ahmed K. Elmagarmid, Abdelsalam Helal, Ibrahim Kamel, Mohamed F. Mokbel: NILE-PDT: A Phenomenon Detection and Tracking Framework for Data Stream Management Systems. In VLDB 2005.

[5] Goce Trajcevski, Hui Ding, Peter Scheuermann, Roberto Tamassia, Dennis Vaccaro: Dynamics-aware similarity of moving objects trajectories. In GIS, 2007.

[6] Guang Jin, Silvia Nittel: Tracking deformable 2D objects in wireless sensor networks. In GIS. 2008.

[7] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In IPSN, 2003.

[8] Jixiang Jiang, Michael F. Worboys: Detecting basic topological changes in sensor networks by local aggregation. In GIS, 2008.

[9] L. Golab and M. T. Ozsu. Processing sliding window multi-joins in continuous queries over data streams. In VLDB, 2003.

[10] Liqian Luo, Aman Kansal, Suman Nath, Feng Zhao: Sharing and exploring sensor streams over geocentric interfaces. In GIS, 2008.

[11] M. A. Hammad, M. F. Mokbel, Mohamed H. Ali, and et al. Nile: A query processing engine for data streams. In ICDE, 2004.

[12] M. A. Hammad, W. G. Aref, and A. K. Elmagarmid. Stream window join: Tracking moving objects in sensor-network databases. In SSDBM, 2003.

[13] M. Halkidi, V. Kalogeraki, D. Gunopulos, D. Papadopoulos, D. Zeinalipour-Yazti, and M. Vlachos. Efficient online state tracking using sensor networks. In MDM, 2006.

[14] Mohamed F. Mokbel, Walid G. Aref: SOLE: scalable on-line execution of continuous queries on spatio-temporal data streams. VLDBJ. 17(5): 971-995, 2008.

[15] Mohamed H. Ali, Walid G. Aref, Cristina Nita-Rotaru: SPASS: scalable and energy-efficient data acquisition in sensor databases. In MobiDE, 2005.

[16] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In SSTD, 2005.

[17] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, Wei-Ying Ma: Mining user similarity based on location history. In GIS 2008.

[18] R. Nehme and E. Rundensteiner. SCUBA: Scalable Cluster-Based Algorithm for Evaluating Continuous Spatio-Temporal Queries on Moving Objects. In EDBT, 2006.

[19] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In IPSN, 2003.

[20] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In INFOCOM, 2004.

[21] Y. Xu, J.Winter, andW.-C. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In MDM, 2004.

[22] Crossbow, Inc. Wireless sensor networks. http://www.xbow.com

[23] A. Helal, H. Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: A programmable pervasive space. Cover Feature, IEEE Computer, 38(3):64–74, March 2005.

[24] A. Desphande, C. Guestrin, W. Hong, and S. Madden. Exploiting correlated attributes in acquisitional query processing. Technical report, Intel-Research, Berkeley, 2004.

[25] Cui, X , Zhao, B and Li, Q, "Exploiting Data Correlation for Multi-Scale Processing in Sensor Networks", Proc. INFOSCALE, 2007, pp.6-8.