Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 304198, 14 pages
http://dx.doi.org/10.1155/2015/304198

Research Article

Adaptive Filter Updating for Energy-Efficient Top-k Queries in Wireless Sensor Networks Using Gaussian Process Regression

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

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 29 Yudao Street, Qinhuai District, Nanjing 210016, China*
[2]*State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Avenue, Qixia District, Nanjing 210093, China*

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

Received 10 March 2015; Revised 22 May 2015; Accepted 30 May 2015

Academic Editor: Mauro Conti

Adopting filtering mechanism of dynamic filtering windows installed on sensor nodes to process top-k queries is an important research direction in wireless sensor networks. The mechanism can reduce transmissions of redundant data by utilizing filters. However, existing algorithms based on filters consume a vast amount of energy due to filter updating. In this paper, an energy-efficient top-k query technique based on adaptive filters is proposed. Due to updating filters consuming a large amount of energy, an algorithm named FUGPR based on Gaussian process regression to process top-k queries is provided for saving energy. When the filters change, the sensor readings are predicted to calculate the updating costs of filters; then FUGPR decides whether the filters need to be updated or not. Thus, the energy consumption for updating filters is decreased. Experimental results show that our approach can reduce energy consumption efficiently for updating filters on two distinct real datasets.

1. Introduction

With the improved recognition to the physical world and the rapid development on technologies such as electron and wireless communication, wireless sensor networks have been applied to many areas such as military, medical treatment, and environment surveillance. The bright future of wireless sensor networks has attracted the attention of so many scholars. In wireless sensor networks, sensor nodes are energy-limited which are powered by batteries. In addition, various kinds of sensors generate a large amount of data. If all the data is sent to the base station, much energy will be consumed and sensors will run out of energy soon.

For large quantity of data generated by wireless sensors, users are always interested in max or min k objects among them. People are often interested in the k maximum or minimum values in a wireless sensor network. A top-k query which returns k sensor nodes with the highest readings can meet the requirements above. Top-k queries have been widely used in wireless sensor networks [1]; for instance, it can be used to effectively monitor environmental and ecological changes. Assume that there are many bird feeders placed in a forest, each of which includes a sensor node. These sensor nodes can detect weight changes to count the number of birds landing at the feeder; thus ornithologists can estimate the number of birds in different areas and choose k areas to observe living habits of birds. Yet, some sensor nodes frequently become the top-k queries candidate sets. After several runs of top-k queries, some failed nodes will emerge, because these nodes frequently perceive and transmit data. According to the top-k query semantic of the example above, the k best sensor nodes often become failed nodes. After that, the top-k query results can not reflect the true distribution of the birds in the forest due to the failed nodes whose energy is exhausted.

To answer top-k queries in wireless sensor networks, a typical solution is by making the use of filters. Filters are broadcasted into the network and used by individual sensor node to decide whether its value is relevant to compute the query result. FILA (filter based monitoring approach) [2, 3] is an energy-efficient approach among these algorithms. The basic idea of FILA is to install a filter at each child sensor

node, which avoids redundant data transmitting to parent nodes or base station. However, the approach consumes much energy on updating filters. Eager filter update policy in FILA updates windows parameters immediately once sensor readings violate the filters while lazy update policy cannot guarantee the updated filters need not be reupdated to the old ones in next epochs. By predicting future sensor readings, we can decide whether it is necessary to update current filters for saving energy in a long period. Mai et al. [4, 5] proposed DAFM algorithm according to FILA. The algorithm predicts the next sensor readings of sensor nodes by linear regression model. Then, the base station decides whether to update filtering windows based on predicted benefits. When sensor data satisfies complicated distributions and varies widely over time, the predicted performance of linear regression is unsatisfactory and the performance of DAFM algorithm gets worse. These issues are essential to top-k queries in wireless sensor networks and are investigated in this paper. The contributions of this paper are summarized as follows:

(i) We adopt a powerful Gaussian process regression (GPR) model to predict next s steps' sensor readings based on historical sensor data. The GPR model is suitable for sensor data due to the temporal correlations. Our proposed filter updating based on Gaussian process regression (FUGPR) approach focuses on reevaluation when filters need updating. FUGPR introduces prediction mechanism to achieve less energy consumption compared to eager filter update and lazy update policies in FILA approach. Also, comparing to DAFM approach, our FUGPR approach guarantees smaller mean squared error thus can perform effectively.

(ii) Instead of *one*-step-forward prediction of DAFM approach, we adopt adaptive s step prediction based on prediction errors. To select suitable s values based on specified sensor data, FUGPR outperforms existing approaches.

(iii) Extensive experiments are conducted to evaluate proposed FUGPR approach by using real data traces. The results show that our FUGPR approach outperforms DAFM, FILE-e, and FILA-l in terms of both energy consumption and network lifetime under single hop and multihop network configurations.

The rest of this paper is organized as follows. In Section 2, we discuss related work. Typical filtering methods for top-k query processing are introduced in Section 3 and our proposed FUGPR algorithm is provided in Section 4. In Section 5, extensive simulations are conducted to show the efficiency and accuracy of the proposed method. Finally, we conclude this paper in Section 6.

2. Related Work

A naive implementation for monitoring top-k query is to use a centralized approach in which all sensor readings are periodically collected by the base station, which then computes the top-k result set directly. The most predominant algorithm for top-k processing is the Threshold Algorithm (TA) [6]. Along with TA algorithm, Cao and Wang [7] developed a three-phase protocol (TPUT) which decreases the number of remote accesses in large distributed networks. Theobald et al. [8] introduced a family of approximate top-k algorithms with probabilistic guarantees for multidimensional data. By extending TPUT algorithm, Michel et al. [9] proposed the KLEE algorithm to provide approximate answers which provides an adaptive framework to allow trade-off efficiency against result quality and network bandwidth. Zeinalipour-Yazti et al. [10] proposed UB-K and UBLB-K algorithms that return upper/lower bounds instead of exact answers. Also, approximate evaluation of top-k queries has also been proposed by Silberstein et al. [11] based on sampling approaches.

In the area of wireless sensor networks, energy-efficient query processing has always been an important issue. A wireless sensor network can be viewed as a distributed network which consists of lots of energy-limited sensor nodes, and communication cost is the main energy consumption. Therefore, there is no doubt that the centralized approach will consume extra energy because of the transmission of massive data. Filtering and aggregation are two main strategies at present that cope with each other to process top-k queries in wireless sensor networks. In order to reduce the communication cost in data collection, Madden et al. [12] proposed an in-network aggregation technique, known as TAG, which keeps unavailable data from transmission compared with centralized algorithms. Similar to TAG, Cougar [13] and Kspot [14] also employ a centralized optimizer to coordinate sensor nodes in an energy-efficient manner. Chen et al. propose QF (Quantile Filter) [15] approach which treats sensing values and its sensor as a point. The goal of algorithm is to decrease energy consumption and prolong the lifetime of network which is not only minimizing the total energy consumption, but also consuming less energy on each node. Recently, they develop online algorithms for answering time-dependent top-k queries with different values of k through the dynamic maintenance of a materialized view that consists of historical top-k results [16]. Liu et al. [17] propose a new cross pruning (XP) aggregation framework for top-k query in wireless sensor networks. There is a cluster-tree routing structure to aggregate more objects locally and a broadcast-then-filter approach in the framework. In addition, it provides an in-network aggregation technique to filter out redundant values which enhance in-network filtering effectiveness. Abbasi et al. [18] proposed MOTE (model-based optimization technique) approach based on assigning filters on nodes by model-based optimization. Nevertheless, it is an *NP*-hard problem on how to get optimal filter settings for top-k set.

However, these approaches incur unnecessary updates in the network and are not really energy-efficient. Olston et al. [19] propose the range caching mechanism for approximate query processing. In range caching mechanism, the base station caches a value range for each sensor node and computes a tentative result based on cashed value ranges. The cashed values are refreshed only when the new values on the sensor

nodes violate the ranges. Also, Olston et al. [20] use filters to bound error in query result over distributed streams. An adaptive scheme is proposed to reduce the communication cost by precision adjustment at each individual source. Inspired by the filter based ideas [19, 20], Wu et al. [2, 3] propose FILA approach for top-k monitoring, which is to install a filter on each sensor node to filter out unnecessary data not contributed to final results. Reevaluation and filter setting are two critical aspects that ensure the correctness and effectiveness. When the new filters are different from the old ones maintained by sensor nodes themselves, base station needs to update them. But when sensing values on nodes vary widely, base station needs updating filters to related nodes frequently which leads to large scales of updating cost and makes the performance of algorithm worse. Mai et al. [4, 5] propose DAFM approach which aims to reduce the communication cost of sending probe messages in reevaluation process as well as the transmission cost in filter updating in FILA. DAFM approach predicts the next readings of sensor nodes to determine these values are in or out of filtering windows by linear regression models. To some extent, this approach decreases the cost of filter updating. However, the sensed values are affected by various factors; the performance is worse when adopting linear regression models to predicate future sensor values. In our previous work [21], time series models are adopted for predicting next sensor values for unnecessary filter updates. However, this approach is more suitable for specific sensor data, that is, time series data distribution and not energy-efficient for non time series sensor data.

3. Filter Based Top-k Query Methods

We first give a formal definition in Section 3.1. Then, Section 3.2 provides filter based top-k query methods and discusses the limitations of existing methods.

3.1. Problem Definition. We consider there are N sensor nodes in a wireless sensor network labeled $1, 2, \ldots, N$ which compose a set I, $I = \{1, 2, \ldots, N\}$. Each sensor node n_i ($n_i \in I$) measures the local physical phenomenon v_i (e.g., `temperature`, `voltage`, `light`, or `humidity`) at a fixed sampling rate. We consider a top-k query that requests the list of sensor nodes R with the highest readings, $R = \langle n_1, n_2, \ldots, n_k \rangle$ for $\forall i < j$, $v_i \geq v_j$, and $\forall l \neq i$ ($i = 1, 2, \ldots, k$), $v_l \leq v_k$. The results are maintained by base station and returned to users finally. The goal in this paper is to prolong the lifetime of wireless sensor networks by minimizing the overall energy consumption.

3.2. Filter Based Top-k Query Processing

3.2.1. An Overview of Filter Based Methods. To reduce network traffic in data collection, TAG [12] as an in-network aggregation technique has been proposed. The topology of the sensor network can be assumed as a spanning tree rooted at the base station. In TAG solution for monitoring top-k queries, every intermediate node of the spanning tree collects all values from its children and forwards the k largest of these values to the base station. Although TAG achieves a reduction of the number of transmitted values by aggregation, the number of messages remains high. Consider the situation that the final k largest values fall in one branch of the spanning tree at some level; the readings of other branches do not need to be forwarded. Therefore, TAG incurs unnecessary transmissions in the network and is not energy-efficient.

Range caching method was first provided by Olston et al. [19, 20] and later utilized by Wu et al. [2, 3] to process top-k queries in wireless sensor networks. The base station cashes a value range α for the value at each sensor node. If sensed value is v_i for sensor node n_i, we take $u_i = v_i + \alpha/2$, $l_i = v_i - \alpha/2$ as the upper bound and lower bound of the range. A sensor node is updated with the base station only when the new value is beyond the range of the previously reported value. However, it is obvious that range caching method is simple to process top-k queries. However, the ranges (we can take a range as a filter) of distinct sensor nodes are overlapped and top-k queries could only be executed after the base station collects all the values of the sensor nodes with the overlapped ranges. The procedure will consume much energy so range caching method is not energy-efficient. Beyond this, the range value α is difficult to be decided.

To overcome the defect of range caching method, Wu et al. [2, 3] proposed FILA to process top-k queries in sensor networks. At the beginning of FILA, the base station collects the readings from all sensor nodes and sorts the sensor readings to obtain the initial top-k result set. Then the base station calculates a filter $[l_i, u_i]$ for each sensor node N_i and sends it the corresponding sensor node for installation. At the next top-k query processing period, if the new reading of sensor node N_i is within the filter $[l_i, u_i]$, then no update is sent to the base station. Otherwise, if the new reading violates the filter, then an update is sent to the base station. The base station should reevaluate the top-k result and adjust the filters of relevant sensor nodes.

(1) Filter Setting. At the beginning of FILA method, the base station has obtained the sorted readings of all the sensor nodes $\{v_1, v_2, \ldots, v_N\}$ and $v_1 \geq v_2 \geq \cdots \geq v_N$. In FILA, each node in the top-k results has a separate filter while all the remaining non-top-k nodes share a common one. At any time, we only need $k + 1$ filters. In addition, to ensure the correctness of FILA method, each filter $[l_i, u_i]$ should cover their current readings but does not overlap with any other filter. A feasible filter setting $[l_1, u_1], [l_2, u_2], \ldots, [l_k, u_k]$, $[l_{k+1}, u_{k+1}]$ must satisfy $v_1 \leq u_1$, $v_{i+1} \leq u_{i+1} \leq l_i \leq v_i$, $(1 \leq i \leq k)$ and $l_{k+1} \leq v_N$.

To maximize the filtering capability, the upper bound of the top-1 node's filter u_1 is set to $+\infty$ while the lower bound of the non-top-k's node filter l_{k+1} is set to $-\infty$ and u_{i+1} is set equal to l_i for all the nodes except them. In FILA, two filter setting strategies, uniform and skewed settings, are provided, respectively. In this paper, we adopt uniform filter setting for simplicity. That is, $u_i + 1$ and l_i are set at the midpoint of two sensor readings which is shown as below and Figure 1

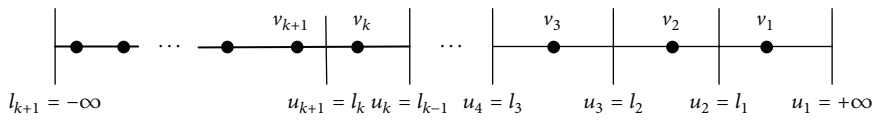


Figure 1: Filter setting.

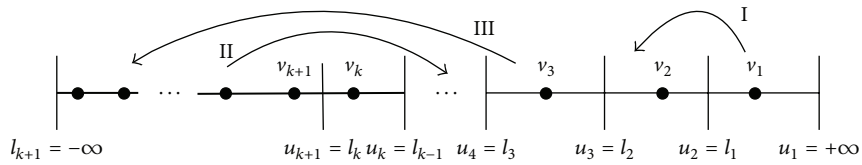


Figure 2: Three situations of windows update.

illustrates the filter setting of the sensor nodes for a top-k query. Consider

$$u_1 = +\infty$$
$$v_{i+1} = l_i = \frac{(v_i + v_{i+1})}{2}, \quad (1 \le i \le k) \tag{1}$$
$$l_{k+1} = -\infty.$$

(2) Query Reevaluation. In FILA, if there is at least one sensor reading sent to the base station for filter violation, the top-k results become undecided at the base station. Then the base station should probe some related sensor node(s) to reevaluate the top-k results.

When the updated reading overlaps with the filter of any other sensor node, there are three situations for this update.

Internal Update. An update originated from a top-k node jumps into the filter of another top-k node. For example, as shown in Figure 2(I), v_1 jumps into $[l_2, u_2]$.

Join Update. An update from a non-top-k node jumps over the critical bound and falls into the filter of a top-k node. For example, as shown in Figure 2(II), v_{k+1} jumps into $[l_{k-1}, u_{k-1}]$.

Leave Update. An update from a top-k node jumps over the critical bound and falls into the filter of non-top-k nodes. For example, as shown in Figure 2(III), v_3 jumps into $[l_{k+1}, u_{k+1}]$.

It is obvious that if an update is an internal or a join one, then only the relevant top-k node whose filter covers the updated sensor reading needs to be probed to reevaluate the top-k result. Otherwise, a leave update from a node N_i may have to probe all non-top-k nodes to look for the new top-kth node. This introduces high energy consumption.

(3) Filter Updating. When sensor readings pass their filters, the filter settings of corresponding sensor nodes need to be recomputed and the base station will send the new filtering windows to the nodes for installation. There are two approaches for updating the filter of each node n_i in FILA.

Eager Filter Update. If a new filtering window is different from the old one, then the new filter computed by the base station is immediately sent to the node n_i to replace the old one.

Lazy Filter Update. If a new filtering window $[l_i', u_i']$ fully contains the old one $[l_i, u_i]$, that is, $[l_i, u_i] \subset [l_i', u_i']$, then the base station delays the filter update until node n_i's reading violates the old one.

When top-k result set changes, the filters of the sensor nodes will also change. We demonstrate above two updating policies using the example shown in Figure 3. In Figure 3, we only consider four sensor nodes n_i, n_{i+1}, n_{i+2}, and n_{i+3}. When the readings v_{i+1}, v_{i+2} change, the four filters will also change. According to the eager filter update policy, the base station needs to send all the four filters to the sensor nodes. In contrast, according to the lazy filter update policy, due to the fact that new filter windows of nodes n_i, n_{i+3} contain the old ones, that is, $[l_i, u_i] \subset [l_i', u_i']$, $[l_{i+3}, u_{i+3}] \subset [l_{i+3}', u_{i+3}']$, the base station only needs to send the filters of nodes n_{i+1}, n_{i+2} while delaying the filter update of nodes n_i and n_{i+3}. We can see that eager filter update consumes much energy for updating the filters of all sensor nodes even when only one sensor reading violates the filter. The lazy filter update consumes less energy because the sensor nodes with the new filter windows containing the old ones remain unchanged.

We can see that lazy filter update policy is naive and not energy-efficient. Due to delay of the filter updating to some sensor nodes, the gap between adjacent filters will occur as illustrated in Figure 4 which decreases the filtering capability. In addition, lazy filter update is a suboptimal method because the total number of filtering updates remains in a high level and consumes much energy though the number of updates is reduced for one time filter update.

3.2.2. Defects of FILA. To some extent, FILA avoids redundant data from transmission and saves energy. However, there is massive unnecessary energy consumption. Assume that after sampling in epoch t, the filtering windows are shown in Figure 5 [5]. In Figure 5(a), at epoch t_i, before sampling top-3 has the value of $\{v_1, v_4, v_3\}$, and the result set is $\{v_1, v_3, v_4\}$ after sampling as shown in Figure 5(b); that is,

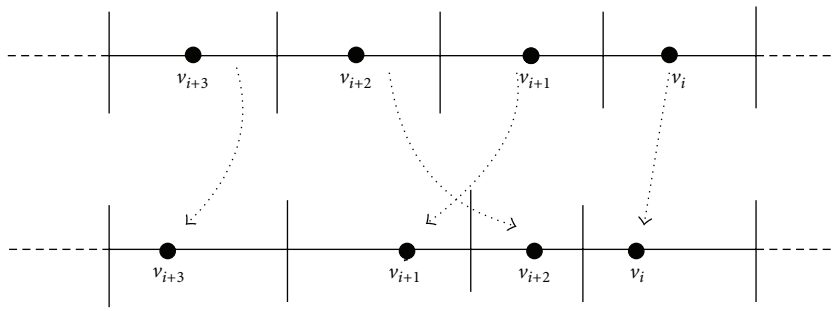


Figure 3: Changes of filter window.

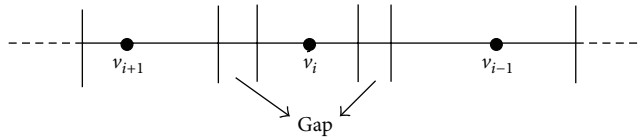


Figure 4: Existence of the gap between windows.

the value v_3 jumps out its filter and falls into the filtering window installed on v_4. The base station needs to adjust the filtering windows for relevant nodes which are shown as Figure 5(c). However, after sampling in epoch t_{i+1} as shown in Figure 5(d), the value v_3 changes again and jumps into the frontal window. In FILA, fluctuation of data will cause frequent updates of the filter windows which will incur large amount of communication cost. Mai et al. [5] proposed an updating algorithm based on prediction which determines whether to update the filters according to the possible cost on updating. The algorithm decreases the cost on updating the filters to some extent. However, this approach is limited by the prediction performance of linear regression model.

In addition, from Figure 5, the values on the non-top-k vary in a small range; their filtering windows are useful to filter out irrelevant sensor nodes. However, the upper bound of non-top-k nodes is determined by the values of ith and $(i+1)$th sensor nodes; when the value which is ranked at kth varies, the filters on non-top-k will be affected directly. For a wireless sensor network generally consists of large number of sensor nodes, k is usually small in a top-k query while a large number of sensor nodes fall into non-top-k node set. If the algorithm updates the filters once reevaluating the top-k results, it will consume more energy on updating the filters of non-top-k nodes.

In this paper, we adopt Gaussian process regression to alleviate the burden of filter update in FILA. When the filters change, the sensor readings are predicted by Gaussian process regression to calculate the updating costs of filters. An algorithm for top-k processing under FILA framework named FUGPR is provided. FUGPR reduces the cost of filter updating when fluctuation of sensor readings occurs and prolongs the lifetime of the wireless sensor network.

4. Filter Updating Algorithms

FUGPR is efficient top-k processing method based on predicted benefits of Gaussian process regression [22, 23]. Gaussian processes let the data "speak" more clearly for themselves. Gaussian processes extend multivariate Gaussian distributions to infinite dimensionality. Formally, a Gaussian process generates data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution. Under Bayesian linear regression framework, a Gaussian process predictor calculates posteriors from priors over functions rather than from priors over parameters [24]. Gaussian process regression has been widely exploited in many research topics, such as human motion estimation and time series prediction [25]. Recently, Gaussian process approach has been adopted for predicting of monitoring models over distributed data streams [26] and real-time sensor data processing in wireless sensor networks [27].

In FUGPR, each node in the top-k list has a separate filter while all the remaining non-top-k nodes share a common one. This setting is along with the framework of FILA. The $k+1$ filters must not overlap with one another. If the reading of a sensor node at next epoch falls in the filter, the reading will not be sent to the base station. Otherwise the violated reading needs to be sent to the base station. Due to limited storage of sensor nodes (e.g., MICAz and telosB only have 4 KB and 10 KB RAM, resp. [28]), each sensor node only preserves current reading and violated readings from its child nodes. The maximum number of readings stored for the sensor nodes are the ones connecting directly with the base station which is decided by the number of layers of the spanning tree (which is a value not less than 100). For typical sensor nodes such as MICAz and telosB, the storage is enough to preserve these readings. The base station will reevaluate

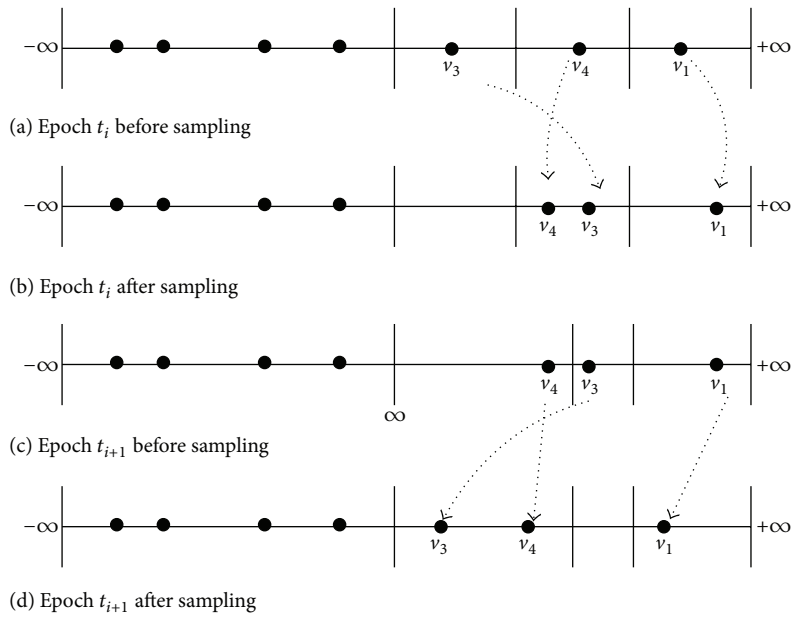


Figure 5: Frequent updating of the filters.

the filter setting of all the nodes based on the violated readings. According to the analysis in Section 3.2, the reevaluation process consumes much energy for filter updating. Instead, we utilize the historical sensor readings of each node to construct Gaussian process models to predict further values of each sensor reading. Then, the costs by utilizing the old filters as well as the new filters are evaluated which decides whether or not to update filters. The predict benefits of FUGPR can save much energy for avoiding updating filters frequently.

4.1. Gaussian Process Regression for Prediction. We predict next epochs' readings of a sensor node on their temporal correlations. The very simple example is to predict next epoch's sensor value based on current reading. For l step prediction, we assume $x_i^{(l)}$ is current reading and $y_i^{(l)}$ the predicting value of next epoch. We can get a dataset H_D, $H_D = \{(x_i^{(l)}, y_i^{(l)}),\ l = 1, 2, \ldots, m,\ y_i^{(l)} = x_i^{(l+1)}\}$, which can be seen as the input of the Gaussian process regression. However, it is obvious that this method will lead to big error margin for predicting next epoch's sensor readings based on current reading is far from enough.

To facilitate the calculation, we only focus on energy consumption resulting from communication and omit other aspects of energy consumption. The assumption is reasonable for the communication of wireless sensor networks is main aspect of energy consumption. The total amount of energy spent in sending a message with m bytes of content is given by $\alpha_s + \beta_s m$, where α_s and β_s are the per-message and per-byte sending costs, respectively. And the total amount of energy consumption in receiving a message with m bytes of content is given by $\alpha_r + \beta_r m$, where α_r and β_r are the per-message and per-byte receiving costs, respectively. As an example, typical values for MICA2 motes, receiving cost is defined analogously, with typical values of α_r and β_r roughly 60% less than their sending counterparts. To calculate quantitatively for our FUGPR, we set $\alpha_r = 0.4\alpha_s$, $\beta_r = 0.4\beta_s$ and the values of α_s, β_s, α_r, and β_r are displayed in Table 1, respectively [29].

Table 1: Typical values for energy calculation.

Symbol	Value
α_s	0.645 mJ
β_s	0.0144 mJ/byte
α_r	0.258 mJ
β_r	0.00576 mJ/byte

From Table 1, we can learn that the energy consumption of communication is higher than data transmission. In order to reduce the number of communications, each transmission the sensor node sends more bytes for saving energy. In FUGPR, readings fallen in their corresponding filter need not be sent to the base station while the violated sensor readings along with their past p readings are sent to the base station together. For each sensor node, the p value is learned from the historical data at the base station. The p value can not be a large number because transmitting too much data will also consume much energy. In this paper, the p value varies from 1 to 5. We choose the p value with the minimum squared error (MSE) for the Gaussian process regression in FUGPR. At the base station, for each sensor node we use Gaussian process regression model on historical dataset H_D. The input is the p sensor readings $\mathbf{X}(t)$ and the output is the predicting value $y(t)$. Let $\mathbf{X}^t = \begin{bmatrix} x^t & x^{t-1} & \cdots & x^{t-p+1} \end{bmatrix}$ and $y^t = x^{t+1}$.

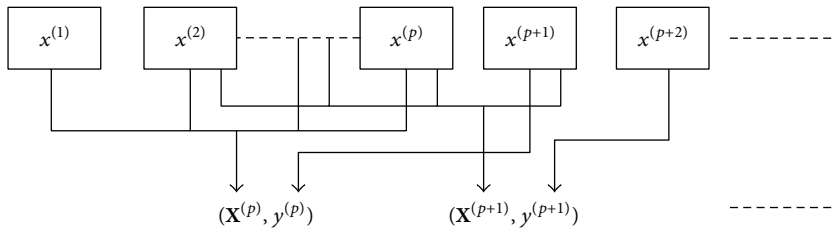


Figure 6: Construction of the training dataset.

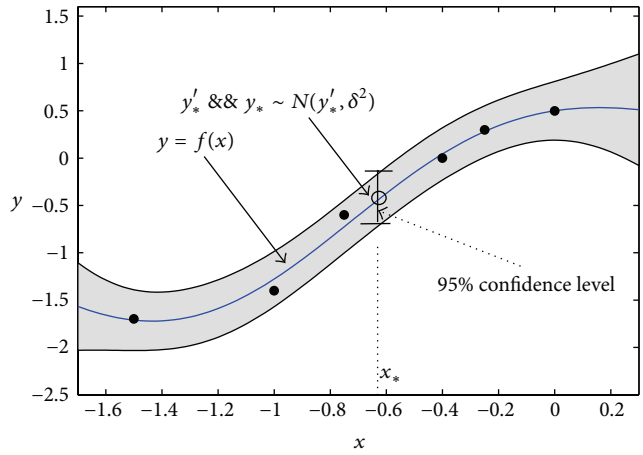


Figure 7: An example for Gaussian process regression.

The historical dataset H_D satisfies $H_D = \{(\mathbf{X}^{(l)}, y^{(l)}), l = p, p+1, \ldots, p+m-1\}$. As illustrated in Figure 6, the $p+m$ data values compose m regression points as shown in Figure 6.

To each sensor node, we use several past reading as the input and the reading at next epoch as the output; we can obtain the following equation through a Gaussian noise model:

$$y = f(X) + N\left(0, \sigma^2\right). \tag{2}$$

Users often expect the underlying function $f(X)$ to be linear and a least-squares method can be exploited to fit the straight line. In DAFM, Mai et al. [5] adopted linear regression method to predict next epochs' sensor values. Typical linear regression models for prediction have the following formula:

$$y = b + \theta X + \varepsilon, \tag{3}$$

where ε is random error satisfying Gaussian distribution $N(0, \sigma^2)$ and the parameter b, θ can be estimated by the least-squared method.

Due to various sensor readings from real applications in wireless sensor networks, the prerequisites of linear regression method are not satisfied. Besides linear regression models, other analogical models may use the principles of model selection to choose among the various possibilities for we suspect $f(X)$ may also be quadratic, cubic, or even nonpolynomial. Instead of claiming $f(X)$ relates to some specific model, a Gaussian process extends multivariate Gaussian distributions to infinite dimensionality and lets the data stand for themselves. In geostatistics, Gaussian processes are known as "Kriging," but the input space in the literature only concentrates two or three dimensions while Gaussian processes consider more general input spaces.

We illustrate Gaussian process regression as shown in Figure 7. For simplicity, we consider one-dimensional input space. The solid points are real sensor readings, the solid line indicates an estimation of prediction values for the real sensor readings, and the shaded region denotes 95% confidence intervals along with the regression line. If current sensor reading of a sensor node is x_*, the sensor value y_* of the next epoch along with the predicted value y'_* predicted by Gaussian process regression satisfies a Gaussian distribution. Consider

$$y_* \sim N\left(y'_*, \sigma^2\right). \tag{4}$$

4.1.1. Gaussian Process Regression. A Gaussian process is a collection of random variables, any finite number of which has a joint Gaussian distribution. In general, a Gaussian process is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$. Usually, for notational simplicity we will take the mean function to be zero. The covariance function $k(x, x')$ represents the influence of different sensor readings.

If there are m items in the training dataset H_D, a $m \times m$ positive definite covariance matrix can be constructed by $k(x, x')$ as follows:

$$K = \begin{bmatrix} k\left(x^1, x^1\right) & k\left(x^1, x^2\right) & \cdots & k\left(x^1, x^m\right) \\ k\left(x^2, x^1\right) & k\left(x^2, x^2\right) & \cdots & k\left(x^2, x^m\right) \\ \vdots & \vdots & \ddots & \vdots \\ k\left(x^m, x^1\right) & k\left(x^m, x^2\right) & \cdots & k\left(x^m, x^m\right) \end{bmatrix}. \quad (5)$$

When a new sensor reading x_* arrives, the covariance matrix K_* of previous m data items and K_{**} of itself are shown as follows:

$$K_* = \left[k\left(x_*, x_1\right) \; k\left(x_*, x_2\right) \; \cdots \; k\left(x_*, x_m\right)\right], \quad (6)$$

where $K_{**} = k(x_*, x_*)$. The estimation of sensor readings $\mathbf{x}$, $\mathbf{x} = \left[x^{(1)} \; x^{(2)} \; \cdots \; x^{(m)}\right]$ is $\mathbf{y}$, $\mathbf{y} = \left[y^{(1)} \; y^{(2)} \; \cdots \; y^{(m)}\right]$. The joint function of $\mathbf{y}$ and y_* is also a multivariate Gaussian function. Consider

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right). \quad (7)$$

According to Bayesian rules, we can obtain

$$y_* \mid \mathbf{y} \sim N\left(K_* K^{-1}\mathbf{y}, K_{**} - K_* K^{-1} K_*^T\right). \quad (8)$$

The mean value and variance of y_* are $K_* K^{-1}\mathbf{y}$ and $K_{**} - K_* K^{-1} K_*^T$, respectively. That is,

$$y_* \sim N\left(y_*' = K_* K^{-1}\mathbf{y}, K_{**} - K_* K^{-1} K_*^T\right). \quad (9)$$

When given a new sensor reading x_* and a confidence level $1 - \delta$, the confidence interval is $y_*' \pm z\delta/2\sigma$.

4.1.2. Training Gaussian Process Regression Models. How well does the selection of the covariance function decide the reliability of our regression process? The procedure mainly lies on the characteristics of the training sensor readings. In the Gaussian process literature, a popular choice is the squared exponential (SE) covariance function which is also adopted in our paper. Consider

$$k\left(x_p, x_q\right) = \sigma_f^2 \exp\left(-\frac{1}{2}\left(x_p - x_q\right)^T P^{-1}\left(x_p - x_q\right)\right). \quad (10)$$

When folding the noise covNoise $= \sigma_n^2 \delta(x_p, x_q)$ into $k(x_p, x_q)$, the SE covariance function will be

$$k\left(x_p, x_q\right) = \sigma_f^2 \exp\left(-\frac{1}{2}\left(x_p - x_q\right)^T P^{-1}\left(x_p - x_q\right)\right) + \sigma_n^2 \delta\left(x_p, x_q\right). \quad (11)$$

In the above equation, if $P = \begin{pmatrix} \lambda^{-2} & & \\ & \ddots & \\ & & \lambda^{-2} \end{pmatrix}$, the covariance function is named covSEiso, that is, squared exponential covariance function with isotropic distance measure. If x_p, x_q are d-dimensional column vectors, $x_p - x_q = [r_1, r_2, \ldots, r_d]^T$, and $P = \begin{pmatrix} l_1^2 & & \\ & \ddots & \\ & & l_d^2 \end{pmatrix}$, the covariance function is named covSEard, that is, squared exponential covariance function with automatic relevance determination (ARD) distance measure.

After covariance function is chosen, we call its parameters hyperparameters. Take the covariance function covSEard + covNoise, for example, hyperparameters set $\theta = \{l, \sigma_f, \sigma_n\}$, $l = \{l_1, l_2, \ldots, l_d\}$. Our maximum a posteriori estimate of θ occurs when $p(\theta \mid X, y)$ is at its greatest value. Consider

$$p\left(\theta \mid y, X\right) = \frac{p\left(y \mid X, \theta\right) p\left(\theta\right)}{p\left(y \mid X\right)}. \quad (12)$$

Assuming we have little prior knowledge about what θ should be, Bayes's theorem tells us this corresponds to maximizing $p(y \mid X, \theta)$ given by

$$\arg\max_\theta p\left(\theta \mid X, y\right) = \arg\max_\theta p\left(y \mid X, \theta\right). \quad (13)$$

For $y \sim N(0, K)$, we obtain

$$p\left(y \mid X, \theta\right) = \frac{1}{(2\pi)^{n/2}\left|K\right|^{1/2}} \exp\left(-\frac{1}{2} y^T K^{-1} y\right). \quad (14)$$

The log-style $\log p(y \mid X, \theta) = L(\theta)$ of the equation is

$$L\left(\theta\right) = -\frac{1}{2} y^T K^{-1} y - \frac{1}{2}\log\left|K\right| - \frac{n}{2}\log 2\pi. \quad (15)$$

Simply run the favorite multivariate optimization algorithm (e.g., conjugate gradients, Nelder-Mead simplex) on this equation and a pretty good choice for θ will be found.

4.2. Optimization Strategies of Filter Updating. If the new reading of one sensor node is without its filters, then an update is sent to the base station. Now, the update is jump into other nodes' filter; the top-k results become undecided at the base station. Therefore, the base station should probe some related sensor node(s) to reevaluate the top-k results. These nodes which should send their new readings to the base station need to wrap up the p readings of current and past epochs and then utilize Gaussian process regression for prediction to ascertain the means and variances of future readings. For these sensor nodes whose readings are not updated, the means and variances of their future readings still maintain the recent ones.

Assume the new reading of a node n_i at epoch t_{n+1} is v_i, the predicted new reading is v_i^{pre}, and its variance is σ_i^2; then $v_i \sim N(v_i^{\text{pre}}, \sigma_i^2)$. Assume the probability function of the approximate distribution of the new reading is $f_i(x)$. Let $F_i^{\text{old}} = [l_i^{\text{old}}, u_i^{\text{old}}]$ and $F_i^{\text{new}} = [l_i^{\text{new}}, u_i^{\text{new}}]$ denote the node n_i's old filter and newly computed filter, respectively. Then the probability of the real reading of n_i falling in F_i^{old} is

$$P_i^{\text{old}} = \int_{l_i^{\text{old}}}^{u_i^{\text{old}}} f_i\left(x\right) dx. \quad (16)$$

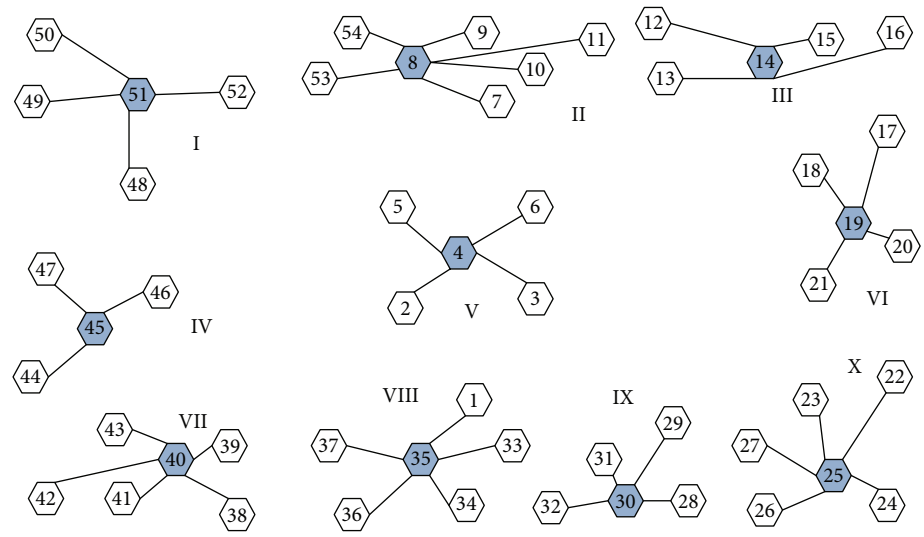


Figure 8: Sensor node topology of Intel Berkeley Research Lab.

And the probability of the real reading of n_i falling in F_i^{new} is

$$P_i^{\text{new}} = \int_{l_i^{\text{new}}}^{u_i^{\text{new}}} f_i(x)\, dx. \tag{17}$$

Let r_i be the average number of recent epochs that n_i's readings do not violate n_i's filter. Assume that we have run for 100 epochs. For the convenience of description, we only consider the sensor node n_i from epochs 80th to 100th. Assume that n_i's reading violates its filters at epochs 84th, 91th, 94th, and 100th. Then, the numbers of epochs that n_i's reading does not violate its filters are $91 - 84 - 1 = 6$ epochs, $94 - 91 - 1 = 2$ epochs, and $100 - 94 - 1 = 5$ epochs. So, the average number of epochs of nonviolation is $r_i = (6+2+5)/3$.

If the base station needs updating the filters, the cost of using the new filters at the future s epochs is referred to as cost_{new}. Otherwise, the cost of using the old filters is denoted as cost_{old}. If $\text{cost}_{\text{new}} < \text{cost}_{\text{old}}$, then the base station sends updated filters; otherwise, all the sensor nodes maintain the old filters. cost_{new} and cost_{old} are calculated as follows:

$$\begin{aligned} \text{cost}_{\text{new}} &= |N_u| + |N| \times s - \sum_{n_i \in I} P_i^{\text{new}} \times r_i \\ \text{cost}_{\text{old}} &= |N| \times s - \sum_{n_i \in I} P_i^{\text{old}} \times r_i. \end{aligned} \tag{18}$$

In formula (18), $|N_u|$ denotes the number of sensor nodes which need to update their filters; $|N|$ denotes the number of nodes in the wireless sensor networks. The result of cost_{new} and cost_{old} for comparison has nothing to do with $|N| \times s$. Hence, we do not need to calculate $|N| \times s$ and consider the specific value of s.

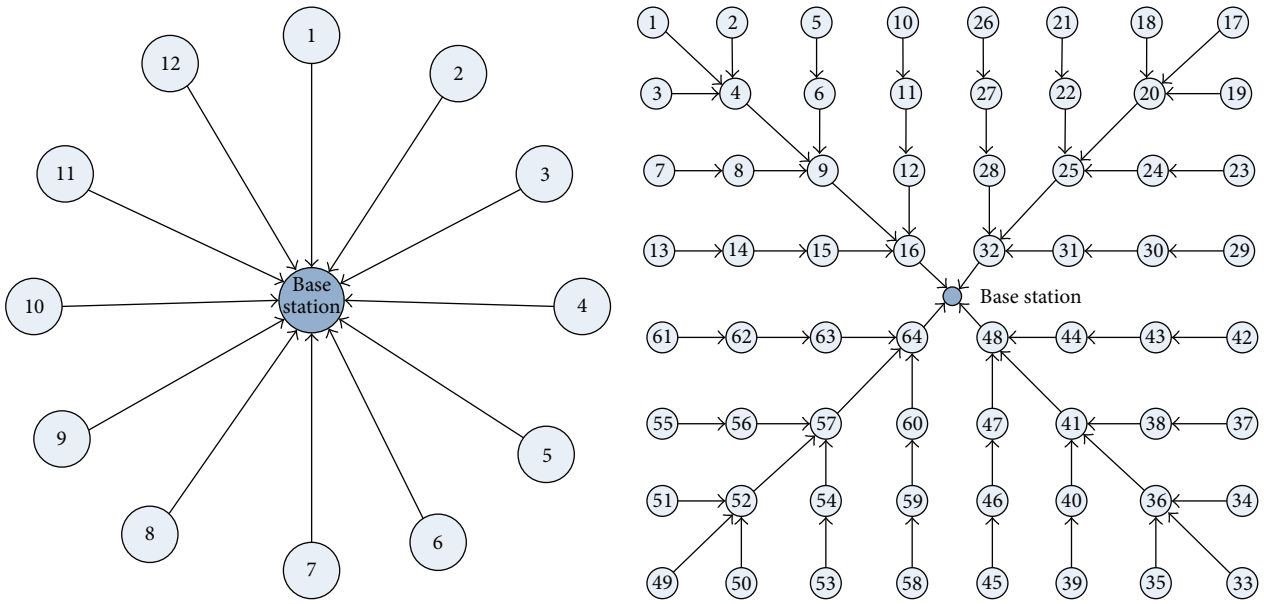
5. Experiments

5.1. Experiments Settings. We use two real datasets: Intel Lab data and LEM data to conduct our experiments.

Intel Lab Data. The set contains information about data collected from 54 sensors deployed in the Intel Berkeley Research Lab [30]. We partition the sensor network into some nonoverlapping regions, making the sensors with positions close to each other in the same area. In this paper, we do not research in depth in how to partition regions more reasonably and only partition the sensor network based on location information by using the k-means algorithm. Taking an example of the Intel Berkeley Research Lab wireless sensor network, we partition it into 10 regions. Figure 8 shows the result of region partitioning, where the dark nodes denote the initial center points. We adopt `temperature`, `humidity`, and `voltage` data on March 1, 2004, as experimental data and the data are collected every 31 seconds. We evaluate the performance of the proposed algorithm FUGPR by MATLAB simulations.

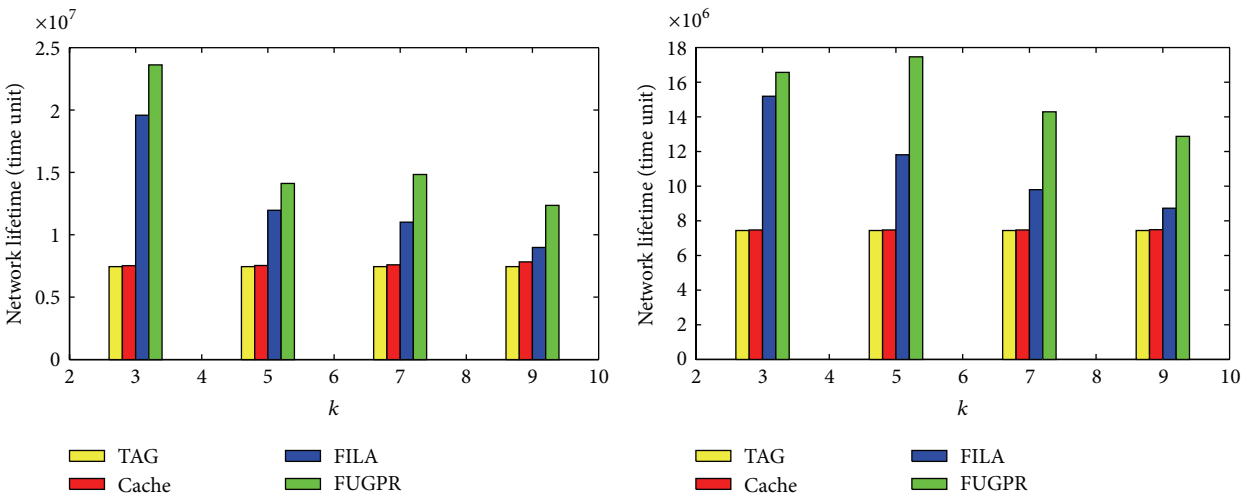
LEM Data. The dataset is collected from the Live from Earth and Mars project at the University of Washington [31]. We adopt `temperature`, `dew point`, and `sea level pressure` collected from December 1, 2012, to December 1, 2013, as experimental data. There are 509,174 records in total. The dataset has missing values in some epochs and we have filled the missing values with the average of the readings at the prior and subsequent epochs. Each attribute of the dataset has 509,174 readings. We extract many subsets, and each subset contained 3,000 readings. The subsets are used to simulate the physical phenomena in the immediate surrounding of different sensor nodes.

We simulated a single hop network and two multihop networks. The single hop network has 12 sensor nodes as shown in Figure 9(a). The two multihop networks contain $8 \times 8 = 64$ and $12 \times 12 = 144$ sensor nodes, respectively. The network consisting of 84 sensor nodes is shown as Figure 9(b), and we number the nodes. The network layout of 12×12 is similar to 8×8, is omitted for simplicity. To simulate the spatial correlation of sensor readings, the subset starting at successive time is assigned to neighboring nodes in the simulated networks.

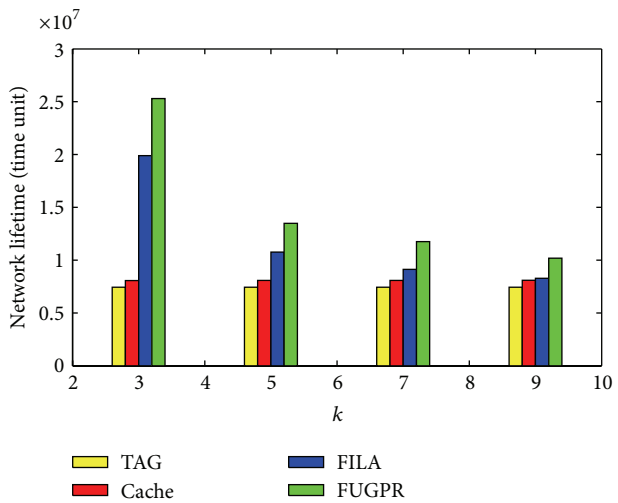


(a) Single hop network (b) 8 × 8 multihop network

Figure 9: Network layout.



(a) Temperature (b) Humidity



(c) Voltage

Figure 10: Comparison of energy consumption of Intel Berkeley Research Lab sensor network.

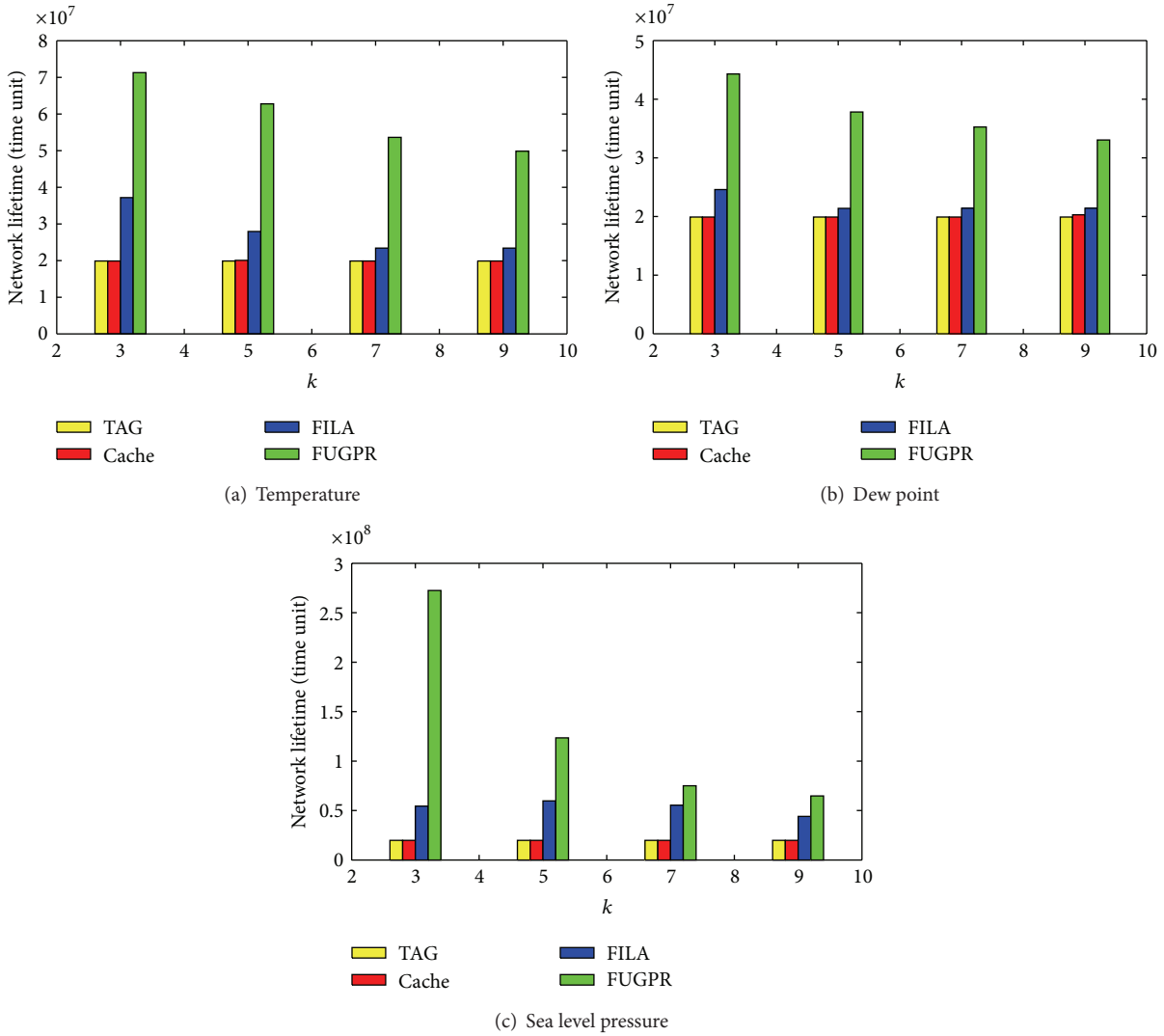


Figure 11: Comparison of energy consumption of a single hop network.

5.2. Experimental Results. We compare our FUGPR with several typical methods, TAG, range caching, and FILA. TAG involves several basic aggregation methods which can be seen as the standards of energy consumption. In range caching method, if two filters are overlapped, the base station may request sensor values to decide final top-k results. In our experiments, the filter window value α for range caching method varies from 0.3 to 3.2 and energy consumptions under different α values are calculated. The α value with the minimum energy consumption is chosen as the final filter window value in our experiments. For FILA method, we adopt lazy filter update policy for our experiments. For our FUGPR method, we use the GPML MATLAB toolbox (http://www.gaussianprocess.org/gpml/) for prediction by our proposed Gaussian process regression models.

Figure 10 shows the comparison of energy consumption for TAG, range caching, FILA, and FUGPR after 1000 runs of top-k queries under Intel Berkeley Lab data: `temperature`, `humidity`, and `pressure`. We can see that FILA and FUGPR adopt filter mechanism which avoids redundant data transmitted to the base station, thus saving much energy. FUGPR optimizes the filter updating procedure which reduces the energy consumption for frequent filter updating. In Figure 10, with the increase of the k values, FUGPR is much better than FILA for energy consumption. The reason is that there will be many violated sensor readings needed to be sent to base station which requires filter reevaluation in FILA method. In contrast, our FUGPR does not need to do much filter updating operations under prediction benefits by adopting Gaussian process regression models. FUGPR utilizes the predicted sensor values to evaluate the costs of using updated filters and old ones in next s epochs, namely, cost_{new} and cost_{old}. The small one will be adopted and no necessary filter updates can be avoided in FUGPR.

Figure 11 shows the comparison of energy consumption for TAG, range caching, FILA, and FUGPR after 1000 runs of top-k queries for single hop sensor network. FILA and FUGPR can filter many sensor readings, especially when

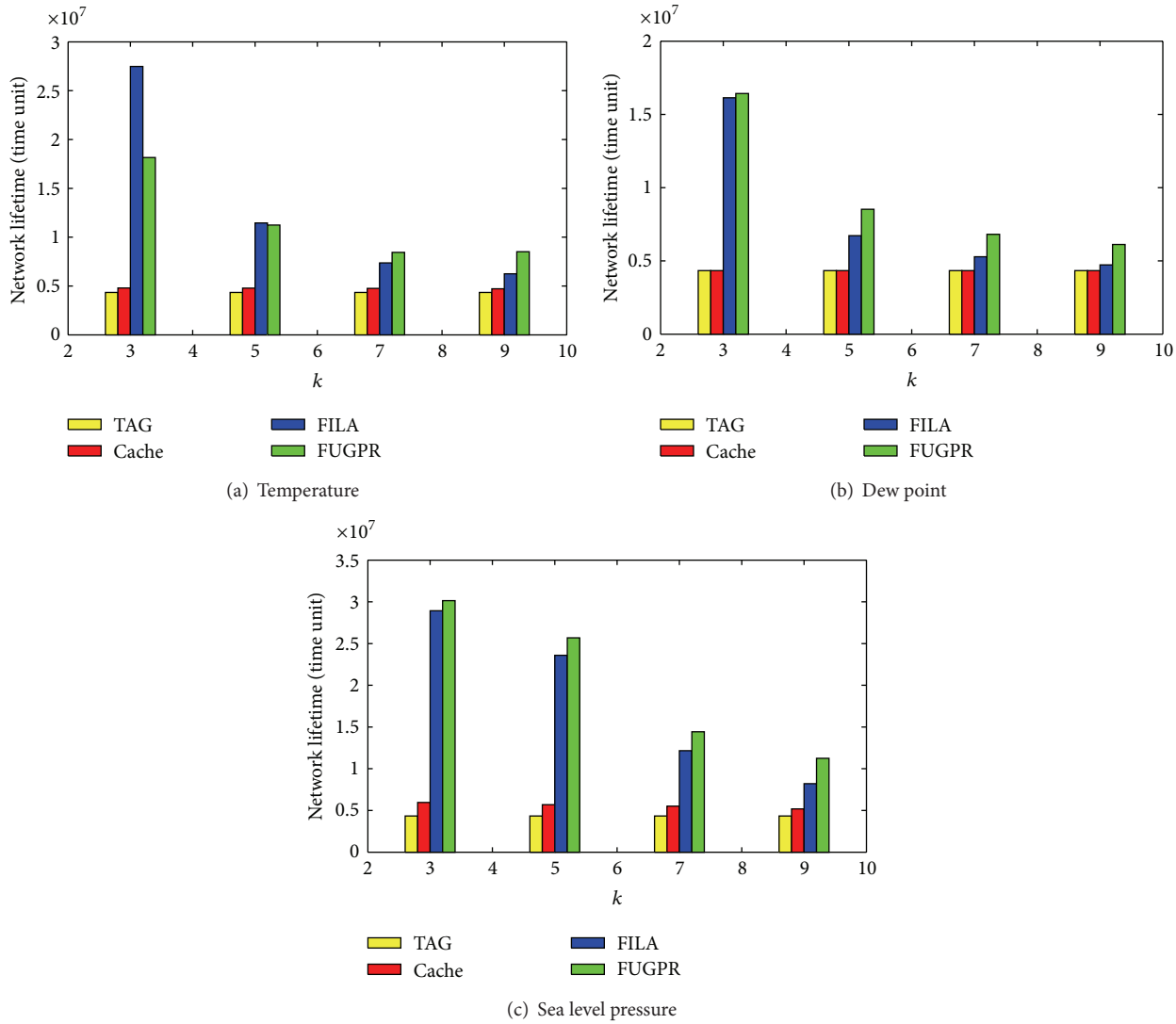


Figure 12: Comparison of energy consumption of 8 × 8 multihop network.

k is small. When k increases (>7), we can see that FILA consumes much energy. For the `dew point` data, when k is 9, the energy consumption of FILA is even larger than TAG method. The reason is that the fluctuation of data causes the energy consumption of requesting sensor node readings by the base station and updating the filters exceeds the energy consumption saved by filter benefits. In contrast, our FUGPR method is prior for all the circumstances.

Figures 12 and 13 show the comparison of energy consumption for TAG, range caching, FILA, and FUGPR after 1000 runs of top-k queries for two multihop sensor networks, 8 × 8 and 12 × 12, respectively. The same situation occurs for FILA method and our proposed FUGPR method is prior to the other three methods.

6. Conclusion

This paper combines Gaussian process regression models into FILA method and proposes FUGPR method for top-k query processing in wireless sensor networks. The prediction benefits of the Gaussian process regression models reduce the cost for filter updating caused by fluctuation of sensor values. Experimental results show that under two distinct real datasets, for example, Intel Berkeley Research Lab data, LEM data, and simulated single hop and multihop sensor network structures, our proposed FUGPR is prior to TAG, range caching, and FILA methods and thus reduces the energy consumption and prolongs the lifetime of the sensor networks.

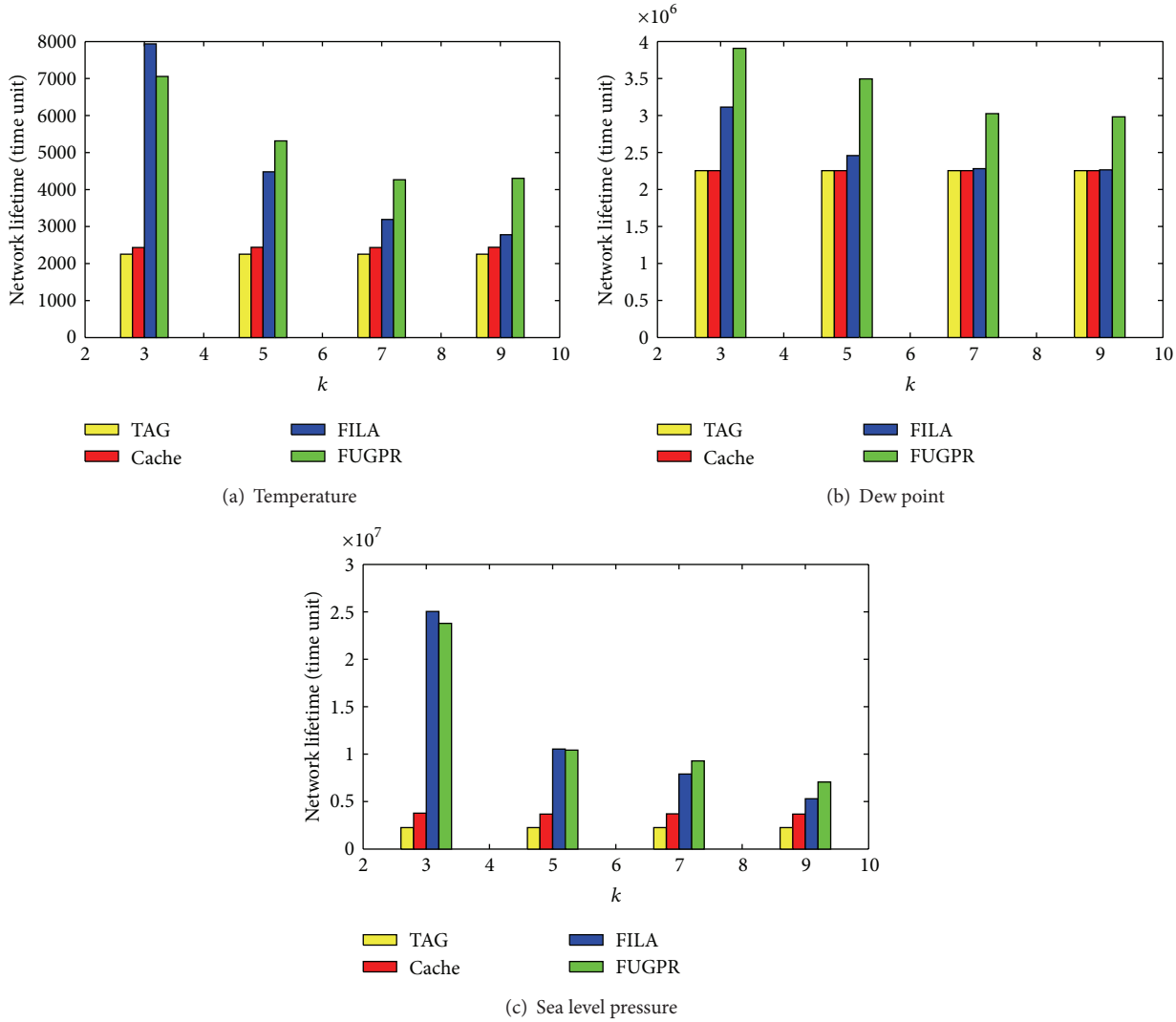


Figure 13: Comparison of energy consumption of 12 × 12 multihop network.

Conflict of Interests

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

Acknowledgment

This work is partially supported by the Fundamental Research Funds for the Central Universities of China under Grant no. NS2015095.

References

[1] M. Dylla, I. Miliaraki, and M. Theobald, "Top-k query processing in probabilistic databases with non-materialized views," in *Proceedings of the 29th International Conference on Data Engineering (ICDE '13)*, pp. 122–133, IEEE Computer Society, Washington, DC, USA, April 2013.

[2] M. Wu, J. Xu, X. Tang, and W.-C. Lee, "Monitoring top-k query in wireless sensor networks," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*, p. 143, April 2006.

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

[4] M. H. Thanh, K. Y. Lee, Y. W. Lee, and M. H. Kim, "Processing top-k monitoring queries in wireless sensor networks," in *Proceedings of the 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM '09)*, pp. 545–552, June 2009.

[5] H. T. Mai, Y. W. Lee, K. Y. Lee, and M. H. Kim, "Distributed adaptive top-k monitoring in wireless sensor networks," *Journal of Systems and Software*, vol. 84, no. 2, pp. 314–327, 2011.

[6] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '01)*, pp. 102–113, ACM, New York, NY, USA, May 2001.

[7] P. Cao and Z. Wang, "Efficient Top-K query calculation in distributed networks," in *Proceedings of the 23rd Annual ACM*

Symposium on Principles of Distributed Computing (PODC '04), pp. 206–215, ACM, New York, NY, USA, July 2004.

[8] M. Theobald, G. Weikum, and R. Schenkel, "Top-k query evaluation with probabilistic guarantees," in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB '04)*, VLDB Endowment, pp. 648–659, 2004.

[9] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: a framework for distributed top-k query algorithms," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, pp. 637–648, September 2005.

[10] D. Zeinalipour-Yazti, S. Lin, and D. Gunopulos, "Distributed spatio-temporal similarity search," in *Proceedings of the 15th ACM Conference on Information and Knowledge Management (CIKM '06)*, pp. 14–23, ACM, November 2006.

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

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pp. 131–146, Boston, Mass, USA, December 2002.

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

[14] P. Andreou, D. Zeinalipour-Yazti, P. K. Chrysanthis, and G. Samaras, "Power efficiency through tuple ranking in wireless sensor network monitoring," *Distributed and Parallel Databases*, vol. 29, no. 1-2, pp. 113–150, 2011.

[15] B. Chen, W. Liang, R. Zhou, and J. X. Yu, "Energy-efficient top-k query processing in wireless sensor networks," in *Proceedings of the 19th International Conference on Information and Knowledge Management and Co-located Workshops (CIKM '10)*, pp. 329–338, ACM, New York, NY, USA, October 2010.

[16] B. Chen, W. Liang, and J. X. Yu, "Energy-efficient top-k query evaluation and maintenance in wireless sensor networks," *Wireless Networks*, vol. 20, no. 4, pp. 591–610, 2014.

[17] X. Liu, J. Xu, and W.-C. Lee, "A cross pruning framework for top-k data collection in wireless sensor networks," in *Proceedings of the 11th IEEE International Conference on Mobile Data Management (MDM '10)*, pp. 157–166, May 2010.

[18] A. Abbasi, A. Khonsari, and N. Farri, "MOTE: efficient monitoring of top-k set in sensor networks," in *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC '08)*, pp. 957–962, IEEE, Marrakech, Morocco, July 2008.

[19] C. Olston, B. T. Loo, and J. Widom, "Adaptive precision setting for cached approximate values," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '01)*, pp. 355–366, ACM, New York, NY, USA, May 2001.

[20] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 563–574, New York, NY, USA, June 2003.

[21] J. Zheng, H. Zhang, B. Song, H. Wang, and Y. Wang, "Prediction-based filter updating policies for top-k monitoring queries in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 696978, 11 pages, 2014.

[22] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[23] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems*, vol. 8, pp. 514–520, MIT Press, 1996.

[24] C. Williams, "Prediction with gaussian processes: from linear regression to linear prediction and beyond," in *Learning in Graphical Models*, M. Jordan, Ed., vol. 89 of *NATO ASI Series*, pp. 599–621, Springer, Dordrecht, The Netherlands, 1998.

[25] S. Brahim-Belhouari and A. Bermak, "Gaussian process for nonstationary time series prediction," *Computational Statistics & Data Analysis*, vol. 47, no. 4, pp. 705–712, 2004.

[26] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster, "Prediction-based geometric monitoring over distributed data streams," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '12)*, pp. 265–276, ACM, New York, NY, USA, May 2012.

[27] M. A. Osborne, S. J. Roberts, A. Rogers, and N. R. Jennings, "Real-time information processing of environmental sensor network data using Bayesian Gaussian processes," *ACM Transactions on Sensor Networks*, vol. 9, no. 1, article 1, 2012.

[28] B. Cody-Kenny, D. Guerin, D. Ennis, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes," in *Proceedings of the 4th ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks (PM2HW2N '09)*, pp. 25–30, ACM, October 2009.

[29] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: on energy-efficient continuous monitoring in sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '06)*, pp. 157–168, ACM, New York, NY, USA, June 2006.

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

[31] Live from earth and mars data from the university of Washington, http://www-k12.atmos.washington.edu/k12/.