An Energy-Efficient Data Collection Framework for Wireless Sensor Networks by Exploiting Spatiotemporal Correlation

Chong Liu, *Student Member, IEEE*, Kui Wu, *Member, IEEE*, and Jian Pei, *Senior Member, IEEE*

Abstract—Limited energy supply is one of the major constraints in wireless sensor networks. A feasible strategy is to aggressively reduce the spatial sampling rate of sensors, that is, the density of the measure points in a field. By properly scheduling, we want to retain the high fidelity of data collection. In this paper, we propose a data collection method that is based on a careful analysis of the surveillance data reported by the sensors. By exploring the spatial correlation of sensing data, we dynamically partition the sensor nodes into clusters so that the sensors in the same cluster have similar surveillance time series. They can share the workload of data collection in the future since their future readings may likely be similar. Furthermore, during a short-time period, a sensor may report similar readings. Such a correlation in the data reported from the same sensor is called temporal correlation, which can be explored to further save energy. We develop a generic framework to address several important technical challenges, including how to partition the sensors into clusters, how to dynamically maintain the clusters in response to environmental changes, how to schedule the sensors in a cluster, how to explore temporal correlation, and how to restore the data in the sink with high fidelity. We conduct an extensive empirical study to test our method using both a real test bed system and a large-scale synthetic data set.

Index Terms—Energy efficiency, data collection, spatiotemporal correlation, wireless sensor networks.

1 Introduction

A wireless sensor network may consist of a large number of sensor nodes, and each node is equipped with sensors, microprocessors, memory, wireless transceiver, and battery. Once deployed, the sensor nodes form a network through short-range wireless communication. They collect environmental surveillance data and send them back to the data processing center, which is also called the *sink node*.

An application domain where wireless sensor networks are broadly used is environmental data collection and surveillance. Imagine that scientists want to collect information about the evolving process of a particular phenomenon over a given region. They can deploy a wireless sensor network in the region, consisting of a large number of geographically distributed sensor nodes. Each sensor node periodically collects local measures of interest such as temperature and humidity and transmits them back to the sink node. Each sampling value is associated with the sensor's location. In the sink node, a snapshot of the region is obtained by assembling the received sampling values in the corresponding locations. Snapshots may be periodically generated, one for each time instant at a given time granularity. These snapshot sequences constitute a SpatioTemporal Data Map (STDMap) [3]. Building up an STDMap is actually the target of a lot of applications.

STDMaps could be considered as analogous to video recording of the monitored region. The playback quality of a piece of a video clip is determined by its time resolution (frame rate) and spatial resolution (the number of pixels). Similarly, the observation fidelity of an STDMap is decided by its spatiotemporal resolution as well. For a given requirement on the observation fidelity, a certain spatiotemporal resolution might suffice and, hence, it is unnecessary to ask all the sensor nodes to sample and report data as fast as they can. Since sampling and wireless communication consume much energy, energy consumption can be significantly reduced by turning off redundant sensor nodes. Fortunately, due to the pervasive existence of spatiotemporal correlation in the sampling data, data transmission can be reduced aggressively for energy saving without a large degradation of observation fidelity.

Spatial correlation usually exists among the readings of close sensor nodes, meaning that the measures from a sensor node may be predicted from that of its nearby sensor nodes with high confidence. Therefore, given a certain requirement on spatial accuracy, only part of the sensor nodes should be required to work for sampling and data transmission in order to save energy.

Unlike spatial correlation, temporal correlation exists in the time series from a single sensor node, meaning that the future readings of a sensor node can be predicted based on the previous readings from the same node. The correlation can be captured by mathematical models such as the linear model or the Auto Regressive Integrated Moving Average (ARIMA) model [8]. Therefore, the time series can be approximated by suitable mathematical models, and the number of model parameters is usually significantly less

- C. Liu and K. Wu are with the Department of Computer Science, University of Victoria, Victoria, British Columbia, Canada V8W 3P6. E-mail: {chongliu, wkui}@cs.uvic.ca.
- J. Pei is with the Department of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada, V5A 1S6. E-mail: jpei@cs.sfu.ca.

Manuscript received 9 May 2006; revised 23 Aug. 2006; accepted 25 Aug. 2006; published online 9 Jan. 2007.
Recommended for acceptance by C. Shahabi.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0121-0506.
Digital Object Identifier no. 10.1109/TPDS.2007.1046.

than the length of the whole series. Transferring the model parameters, instead of the raw time series, can significantly decrease the energy consumption on communication.

This paper aims to save energy in continuous data collection applications in Wireless Sensor Networks (WSNs) by exploiting the spatiotemporal correlation. To exploit the spatial correlation, we partition the sensor nodes with similar observations into a cluster. Within each cluster, the reading of any sensor node can be approximated by any other sensor nodes within an error bound. Therefore, the sensor nodes within a cluster can be scheduled to work alternatively to save energy. We model this clustering problem as a clique-covering problem and propose a greedy algorithm to solve it. Also, we propose a randomized scheduling algorithm to balance the energy consumption.

To exploit temporal correlation, we adopt the piecewise linear approximation technique, that is, approximating the time series with a sequence of line segments. In order to minimize the energy consumption on data transmission with certain accuracy at the sink node, we model the problem as an optimization problem, called the PLAMLiS (Piecewise Linear Approximation with Minimum number of Line Segments) problem. The PLAMLiS problem can be solved in polynomial time by being converted to a shortest path problem [18]. However, the optimal solution requires excessive computing and memory resources, which might be prohibitive for sensor nodes. Therefore, we propose a lightweight greedy algorithm to solve it. The greedy algorithm has the time complexity of $O(n^2 \log n)$, where n is the number of data items; it can be easily integrated into our Energy-Efficient Data Collection (EEDC) framework and work together with the dynamical clustering and the randomized scheduling algorithm, leading to further energy saving.

Our EEDC framework is aware of the spatiotemporal correlation and can achieve *adaptive* sampling in both time and spatial domains. Adaptive sampling can dynamically adjust the temporal and spatial sampling rate aligning with the phenomenon changes while a given accuracy requirement is always satisfied. When a phenomenon varies slowly in the time domain, the number of line segments required for approximation with certain accuracy decreases and, hence, less sampling data is transmitted to the sink node. Therefore, the actual temporal sampling rate decreases, and oversampling is avoided during the slow-changing period. Similarly, when a phenomenon varies slowly in the spatial domain and the observations are similar in a larger area, the size of each cluster increases and, hence, the total number of cluster decreases. Since the number of simultaneous working nodes is proportional to the number of the clusters, the actual spatial sampling rate decreases in this case.

In Section 2, we review related work and point out the differences between our method and other existing methods. In Section 3, we propose the EEDC framework. In Section 4, we explore spatial correlation, and in Section 5, we propose a sensor-scheduling scheme based on spatial correlation. In Section 6, we explore temporal correlation for further energy saving. In Section 7, we perform a comprehensive performance evaluation of the EEDC framework based on both a real test bed and a large synthetic data set. The paper is concluded in Section 8.

2 Related Work

WSNs are generally used in two types of applications: event detection or continuous data collection. The main task of event detection applications is to detect and report the occurrences of interesting events to the sink node. Each sensor device has a coverage area in which events will be detected with high confidence. Therefore, a high detection rate of interested events requires a high coverage rate. The sensor node does not transmit data to the sink unless events are detected. In this context, the data quality is directly determined by the coverage rate. Coverage-based scheduling methods [10], [26], [29] are widely used in event detection applications in WSNs.

Continuous data collection applications require sensor nodes to continuously sample local environmental measures such as temperature and humidity and transmit them back to the sink node. Each sample of an individual sensor node is associated with the location and time. Then, the sink node assembles all the received samples to reproduce the evolving process of certain physical phenomenon in the monitored area. In this context, the data quality is directly determined by the spatial and temporal resolution of these samples. In order to save energy, it is desirable that sensor nodes can adaptively adjust their spatial and temporal sampling rates, according to the changes of physical phenomenon. Adjusting the spatiotemporal sampling rate is a sensor-scheduling problem, that is, coordinating the on/off operation of distributed sensor nodes to ensure data quality.

Achieving energy efficiency is more challenging for continuous data collection than for event detection applications, due to a larger amount of data transmitted in the former. Most existing coverage-based scheduling methods assume the homogeneity of the monitored area, that is, the evolving pattern of phenomenon is the same in the whole area. They may be effective for event detection applications but are inapplicable in continuous data collection, for which any a priori assumption on the homogeneity of a phenomenon may be inaccurate. Spatial and temporal sampling rates must be adaptively adjusted in continuous data collection.

Several pioneering methods have been proposed to adjust the spatial sampling rate according to the statistic features of the monitored phenomenon. In [9], a linear model is proposed to capture the spatiotemporal correlation in sampling data from different sources. With this model, most sensor nodes can be turned off, and their readings can be estimated with certain accuracy by using the linear combination of data from working sensor nodes. However, in the real world, a lot of systems may not be linear. Furthermore, the method of choosing the right working nodes has not been discussed in [9].

An approach to adjusting the spatial sampling rate with the help of mobile sensor nodes is proposed in [1]. Mobility, combined with an adaptive algorithm, allows the system to get the most efficient spatiotemporal sampling rate to achieve the specified data accuracy. Mobility can also make the system respond quickly to unpredictable environmental changes.

To exploit temporal correction, piecewise linear approximation has been broadly studied in the database research community and is widely used to classify, cluster, and index time series. The basic idea is to approximate complex time series with pieces of line segments. Many algorithms have been proposed [7], [22], [14], [15], [25], [27]. These algorithms are supposed to execute on a server, which has enough computational resources for mining large time series online or offline. Unfortunately, sensor nodes in wireless sensor networks are very limited in computational and energy resources, making existing methods not effective.

Our work aims at energy-efficient continuous data collection by *dynamically* exploiting spatiotemporal correlation. Unlike other EEDC techniques, exploiting spatial

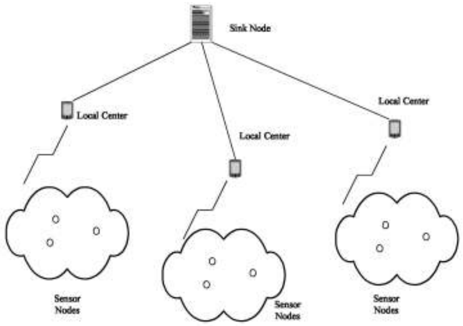


Fig. 1. EEDC framework in a hierarchical architecture.

correlation such as Slepian-Wolf coding [2], [4] and model-driven data acquisition [6] where the spatial correlation is assumed to be known a priori, our method does not make any assumption on the spatial correlation model. Unlike the research exploiting the spatial correlation in in-network aggregation [21], [24], [28], [30], where redundant data are aggregated in intermediate nodes on their way to the sink, our adaptive sampling approach utilizes the spatial correlation in the context of sensor scheduling. Our method determines data redundancy among sensor nodes in runtime to avoid the generation and transmission of redundant data. That is, *our method depresses redundant data in the first mile and does not transmit them into the network at all*. So, we can expect that the adaptive sampling approach can achieve much higher energy savings than the in-network data aggregation.

The novelty of our method is described as follows. First, our method is among the first several that exploit the spatiotemporal correlation in sensory data in the context of sensor scheduling rather than the in-network aggregation. Second, our method is purely data driven and does not make any assumption on the sensor nodes' sensing range or the homogeneity of the monitored area. Third, our method dynamically adjusts spatial and temporal sampling rates according to phenomenon changes. Last, we abort powerful but costly techniques on mining temporal correlation and focus on lightweight piecewise linear approximation to achieve a good balance among accuracy, optimization, and resource consumption.

3 The EEDC Framework

In this paper, we assume that all the sensor nodes are within a single-hop radio transmission to the sink node or to a local center. With this setting, we do not need to consider network partitioning due to sensor scheduling. As can be seen, the sink node may become a bottleneck for large-scale networks. When a network becomes very large, the EEDC framework can be easily extended to a hierarchical architecture, as shown in Fig. 1. A commonly used strategy is to divide the network into several subnetworks, with each depending on a local center for local data collection and long-distance radio transmission to the sink node [20]. In this case, the EEDC framework should be implemented in the local centers, and a local center could be considered as a local sink node. In the following, the sink node should be understood as a local center if the hierarchical network architecture is assumed.

Compared to other existing sensor-scheduling schemes that rely on sensing coverage and the geographic distribution of sensors [10], [26], [29], EEDC distinguishes itself in that it is data aware and makes scheduling decisions according to the spatiotemporal correlation of phenomena.

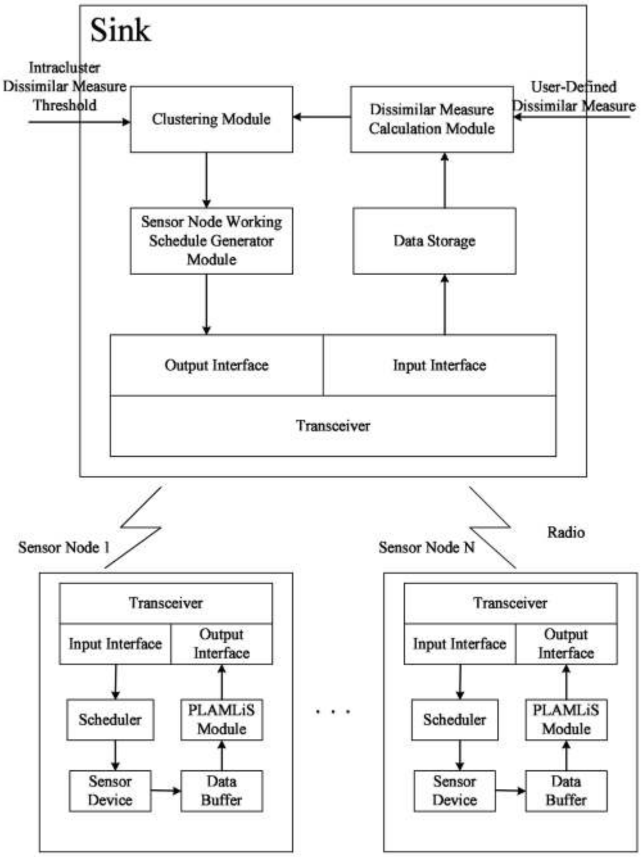


Fig. 2. EEDC framework.

Compared to sensor nodes, the sink node usually has a much larger memory and more powerful computing capability. Such an asymmetry between the sink node and the sensor nodes determines that a good design for data collection should not put heavy burdens on sensor nodes. Instead, the heavy duties should be assigned to the powerful sink node. Our EEDC framework follows this design principle and is shown in Fig. 2. As we can see, the functionalities in sensor nodes are much simpler than those in the sink node.

In a sensor node, the scheduler module simply extracts the working schedules received from the sink node and makes the sensor node work/sleep according to the schedule. In its working shift, it obtains the sensor readings periodically and puts the reading to the buffer. When the buffer is full, the PLAMLiS module takes the readings in the buffer as a time series, approximates it with the PLAMLiS algorithm, and sends the calculated line segments to the output interface.

In contrast, the sink node takes most workloads, including four main functional modules, as shown in Fig. 2:

1. *Data storage module*. It stores all sampling data received from the sensor nodes. This module records a time series for each sensor, which is fed into the dissimilarity measure module as input data.
2. *Dissimilarity measure module*. It calculates the pairwise dissimilarity measure of time series. The dissimilarity measure is application specific, and it is impossible to use a common dissimilarity measure to accommodate all application scenarios. As such, this module assumes that the dissimilarity measure is provided

by the user for a specific application scenario. We will introduce the details of the dissimilarity measure used in our experimental study in Section 4.2.

3. *Clustering module*. Given the dissimilarity computed by the dissimilarity measure module and a maximal dissimilarity threshold value max_dst, this module divides the sensor nodes into clusters such that the dissimilarity of any two sensor nodes within a cluster is less than max_dst. The details of the clustering algorithm will be discussed in Section 4.3.
4. *Sensor node working schedule generator*. It generates a working schedule for each sensor node based on the clusters obtained from the clustering module. The details of sensor scheduling will be discussed in Section 5.

With the EEDC framework, the data collection procedure in a sensor network could be divided into the following three phases:

1. *Data accumulation*. In this phase, each sensor node keeps sampling, processes the samples with the PLAMLiS algorithm when the data buffer is full, then transmits the approximated time series to the sink node. The sink node records the received time series for each sensor node. After collecting enough data, the sink node calculates the dissimilarity measure between any two time series. It terminates this phase whenever the dissimilarity measure among most of the collected time series remains roughly stable. Note that, in some applications, the dissimilarity measure among some sensor nodes may never become stable due to environmental noise or fast-changing phenomena. In this situation, it may be very hard, if not impossible, to explore data correlation for energy saving for these nodes. In other words, the sensors that do not exhibit any similarity to other sensors cannot use spatial correlation for energy saving and will not be considered in the following clustering phase.
2. *Clustering*. In this phase, the sink node uses a clustering algorithm to partition sensor nodes according to the dissimilarity measure calculated in the first phase. The output of the clustering algorithm is a set of clusters, and inside each cluster, the dissimilarity measure between two arbitrary sensor nodes is smaller than a given threshold value. Consequently, the whole field is divided into pieces of subregions, each of which is covered by a corresponding cluster of sensor nodes. The observation at any point in this subregion can be approximated by the observation of any sensor node within the cluster covering this subregion.
3. *Saving and dynamic clustering*. In this phase, the sink node sends out the decision of clusters to all sensor nodes and requires the sensor nodes within the same cluster to work alternatively to save energy. In the meantime, the sink node monitors large variations within a cluster and dynamically adjusts the cluster.

Note that, after clustering, the environment may change and, thus, the previous clusters may not be valid anymore. It is desirable to adaptively change the clusters in response to the changes of the spatial correlation. The details of reclustering will be discussed in Section 5 when we introduce the scheduling algorithm.

In terms of data restoration quality, if the approximation error is bounded by ϵ_1 in the PLAMLiS algorithm and by ϵ_2 in the clustering algorithm, the final approximation error of each reading at any point in the restored STDMap is upper bounded by $\epsilon_1 + \epsilon_2$. Our experimental results show that the approximation error is actually far below this upper bound in our experimental scenario. The details will be disclosed in Section 7.8. Note that the order of performing the PLAMLiS algorithm and clustering algorithm is interchangeable since we exploit the spatial correlation and temporal correlation orthogonally, and the final approximation error bound is the same regardless of the order. However, executing PLAMLiS first can save energy in the data accumulation phase.

4 Exploiting Spatial Correlation

4.1 Motivation and Methodology

The purpose of continuous data collection applications is to construct the STDMap by restoring the spatiotemporal evolving process of phenomenon in the field for further online or offline analysis. The restoration is based on a sequence of snapshots, each of which consists of local measures from geographically distributed sensor nodes. Therefore, in such applications, sensor nodes must report their local measures of interest such as temperature and light intensity to the sink node *continuously*. Also, the density of the measure points in a field should be sufficiently high so that the spatial distribution of the measures can be restored with high quality.

As discussed in Section 2, the coverage-based scheduling methods [10], [26], [29], which aim at event detection applications, are not suitable for continuous data collection applications. The inherent problem of coverage-based scheduling methods is that *they only rely on the static structure of the sensor networks but are unaware of the data reported by the sensor nodes*. Intuitively, the correlation between the data reported by the sensor nodes may help to reduce the spatial sampling rate of the sensor nodes substantially.

Example 1 (Intuition). The readings reported by a sensor node over time form a time series. Suppose that the time series of sensor nodes x, y, and z were very similar in the past. Thus, we may conjecture that the readings of x, y, and z would also likely be similar in the future. Therefore, instead of scheduling the three sensor nodes reporting data simultaneously, we can let two out of the three sensor nodes report at a time, and all the three take turns to report. Such a schedule has the following two advantages:

- *Energy saving*. Each sensor node saves 33.3 percent energy on reporting data.
- *Quality guarantee*. When the three sensors are still correlated, we can obtain the data with high quality and also save energy. On the other hand, even if the readings of one sensor node become dissimilar to the other two, we can still detect the divergence with a minor delay. Then, an updated schedule can be made based on the change.

Motivated by the intuition in Example 1, we develop a dynamic clustering and scheduling approach to solve the typical data collection problem in continuous data collection with wireless sensor networks. Our solution is to dynamically group sensor nodes into a set of disjoint clusters such that the sensor nodes within a single cluster have strong spatial correlation and, hence, great similarity in observations. Therefore, all the sensor nodes in a cluster can be treated equally, and at any time instant, only a small fraction of sensor nodes needs to be active, serving as the representatives for the whole cluster. All the rest of the sensor nodes can sleep without much degradation of observation fidelity. To balance the energy consumption, the sensor nodes within a cluster can share the workload equally.

The clustering operation is based on the dissimilarity measure of time series consisting of historical observations from individual sensor nodes. The degree of spatial correlation can be evaluated by the dissimilarity measure. For two locations with high spatial correlation, their corresponding time series are usually associated with a low dissimilarity measure. Therefore, in a very smooth subregion, the observed measure has only small changes within the subregion; that is, the difference between observations at any two locations within the subregion may be quite small. Hence, the working sensor nodes within this subregion could be sparse without losing the observation fidelity. In contrast, in a fast-changing subregion, the working sensor nodes should be dense. By setting an appropriate dissimilarity measure threshold value to distinguish similar nodes from dissimilar nodes, the spatial sampling rate will match the spatial variation of the observed physical phenomena. A smaller threshold value increases the spatial sampling rate and the observation fidelity, but it requires more energy consumption. In this sense, the dissimilarity measure threshold value constitutes a degree of freedom that could be tuned to balance the trade-off between observation fidelity and energy consumption.

4.2 Dissimilarity Measure

As described above, the time-ordered data sequence at each sensor node forms a time series, and the clustering algorithm is based on the dissimilarity between the time series of the sensor nodes. Nevertheless, the notion of dissimilarity of complex objects such as time series is application specific and task oriented, and defining dissimilarity is nontrivial [12]. However, in the context of continuous data collection, we believe that the magnitude and the trend of time series are of most concern, since they determine the general shape and the trend of the phenomena's evolving curve. Therefore, we define the following two metrics to evaluate the difference of the two time series.

Definition 1: magnitude m-dissimilarity. *Two time series $X\{x_1, x_2, \ldots, x_q\}$ and $Y\{y_1, y_2, \ldots, y_q\}$ are magnitude m-dissimilar if there is an $i (1 \leq i \leq q)$ such that $|x_i - y_i| > m$.*

Definition 2: trend t-dissimilarity. *Two time series $X\{x_1, x_2, \ldots, x_q\}$ and $Y\{y_1, y_2, \ldots, y_q\}$ are trend t-dissimilar if*

$$\frac{q_1}{q} < t,$$

where q_1 is the total number of pairs (x_i, y_i) in the time series that satisfy $\nabla x_i \times \nabla y_i \geq 0$, $\nabla x_i = x_i - x_{i-1}$, $\nabla y_i = y_i - y_{i-1}$, $1 \leq i \leq q-1$.

If we put geographical constraints on the dissimilarity measure of two sensor nodes, for instance, any two sensor nodes whose geographical distance is larger than a threshold value are considered to be dissimilar [13], the computational complexity of calculating pairwise dissimilarity can be greatly reduced. Thus, we constrain that the geographic distance between any two sensor nodes that are considered similar must be at most $gmax\_dist$, where $gmax\_dist$ is a given maximal distance threshold.

In general, we want the dissimilarity measure to have a straightforward and practical meaning. Assume that $gdst(S_x, S_y)$ denotes the geographical distance between sensor node S_x and sensor node S_y, and assume that $gmax\_dst$ is the user-defined threshold value for geographical distance. In our later experiments, we put two time series X and Y into different groups if 1) they are magnitude m-dissimilar, 2) they are trend t-dissimilar, or 3) $gdst(S_x, S_y)$ is greater than $gmax\_dst$.

4.3 Clustering Sensor Nodes

Given the pairwise dissimilarity values, we need a clustering algorithm to partition the sensor nodes into exclusive clusters such that, within each cluster, the pairwise dissimilarity measure of the sensor nodes is below a given intracluster dissimilarity threshold max_dst, which is defined by the tuple $(m, t, gmax\_dst)$ in our example. All sensor nodes in the same cluster are correlated. In each cluster, only as few as one sensor node needs to work at any time instant. Because of this reason, it is desirable to minimize the number of clusters to maximize the energy savings.

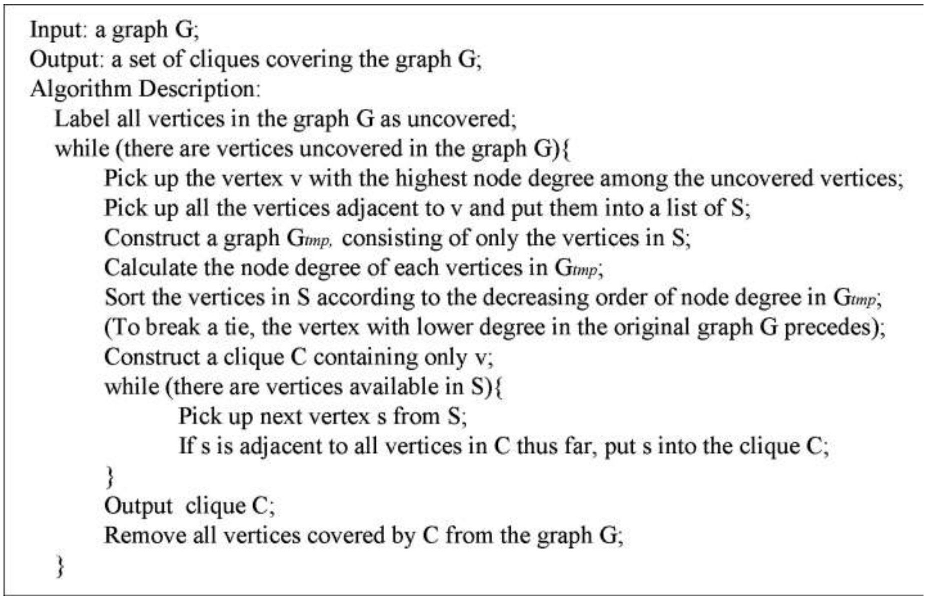
Interestingly, the above problem could be modeled as a clique-covering problem. We construct a graph G such that each sensor node is a vertex in the graph. An edge (u, v) is drawn if the dissimilarity measure between vertex u and vertex v is less than or equal to the given intracluster dissimilarity measure threshold max_dst. Clearly, a cluster is a clique in the graph. Then, the clustering problem is to use the minimum number of cliques to cover all vertices in the graph.

The clique-covering problem is proven NP-complete and does not even allow constant approximation [19], [31]. Hence, we adopt a greedy algorithm to obtain a rough approximation, as shown in Fig. 3. The basic idea of the algorithm is to heuristically find cliques that cover more vertices that have not been clustered. Heuristically, the vertices with larger degrees may have a better chance of appearing in larger cliques. Thus, the search starts from the vertex with the largest degree until all vertices are covered. The output of this algorithm is a set of cliques that cover all vertices.

5 Sensor Scheduling Based on Spatial Correlation

5.1 Randomized Intracluster Scheduling Method

Unlike the round-robin scheduling method, which maintains a single active sensor node within each cluster [17], we use multiple active sensor nodes per cluster at an instant. This has several advantages. First, multiple active sensor nodes can improve data reliability. When multiple active sensor nodes per cluster are used, the data can be restored as long as at least one packet out of the multiple active sensor nodes reaches the sink node. Second, it is possible that a cluster needs to be split into two or more clusters due to spatial correlation changes caused by the environmental changes. Multiple active sensor nodes can help to shorten the delay of detecting spatial correlation changes, making the system quickly respond to such changes within a single cluster.



```
Input: a graph G;
Output: a set of cliques covering the graph G;
Algorithm Description:
  Label all vertices in the graph G as uncovered;
  while (there are vertices uncovered in the graph G){
      Pick up the vertex v with the highest node degree among the uncovered vertices;
      Pick up all the vertices adjacent to v and put them into a list of S;
      Construct a graph $G_{tmp}$, consisting of only the vertices in S;
      Calculate the node degree of each vertices in $G_{tmp}$;
      Sort the vertices in S according to the decreasing order of node degree in $G_{tmp}$;
      (To break a tie, the vertex with lower degree in the original graph G precedes);
      Construct a clique C containing only v;
      while (there are vertices available in S){
            Pick up next vertex s from S;
            If s is adjacent to all vertices in C thus far, put s into the clique C;
      }
      Output  clique C;
      Remove all vertices covered by C from the graph G;
  }
```

Fig. 3. The greedy clique-covering algorithm.

Randomized Intracluster Scheduling Method. The time is divided into a sequence of time slots, each with a length of T. T is the time needed to collect q samples. (This is because we need q samples to evaluate the dissimilarity between the two sensor nodes.) For each time slot, the sensor node goes into work state with a probability λ. Note that λ varies among clusters. A big cluster should have a small λ and a small cluster should have a large λ in order to obtain the same detection delay for cluster split. Note that the detection delay is the interval between the time instant when an event occurs and the time instant when it is first detected by an active sensor node.

Randomization balances the workload distribution among all sensor nodes without introducing any communication overhead for coordination. Besides, by adjusting the value of λ, we can easily control the trade-off between energy saving and detection delay for cluster split.

The sink node calculates the pairwise dissimilarity of active sensor nodes within each cluster at the end of each time slot, after it successfully collects the latest q samples from each active sensor node. The current cluster should be split if the sink node finds that there is at least one active sensor node reporting a significantly different data.

5.2 Analysis on Detection Delay for Cluster Split

Theorem 1. *With the randomized intracluster scheduling method, the expectation of the detection delay for cluster split T_d is no greater than $T\left(\frac{1}{\lambda\left[1-(1-\lambda)^{n-1}\right]}-\frac{1}{2}\right)$, where n is the size of the cluster.*

Proof. We only consider the case that the environment changes make a current cluster split into two clusters. If we can detect the case that the current cluster must be split into two clusters, we can certainly detect the case of splitting the cluster into more clusters, since the latter case is simply the recurrence of the former one.

Suppose that n sensor nodes are split into two clusters C_1 and C_2 of size n_1 and n_2 $(n_1 + n_2 = n)$, respectively. Without losing generality, we assume that the split occurs at slot 1. Therefore,

$$\begin{aligned}&Pr\{split\ is\ detected\ at\ slot\ 1\}\\&= Pr\{at\ least\ one\ node\ is\ active\ in\ C_1\}*\\&\quad Pr\{at\ least\ one\ node\ is\ active\ in\ C_2\}\\&= [1-(1-\lambda)^{n_1}]\times[1-(1-\lambda)^{n_2}].\end{aligned}$$

Denote the above probability by p. We have

$$\begin{aligned}&Pr\{split\ is\ detected\ at\ slot\ 2\}\\&= Pr\{split\ is\ not\ detected\ at\ slot\ 1\}*\\&\quad Pr\{splitting\ is\ detected\ at\ slot\ 2\}\\&= (1-p)\times p.\end{aligned}$$

Similarly,

$$Pr\{split\ is\ detected\ at\ slot\ i\} = (1-p)^{i-1}\times p.$$

So,

$$\begin{aligned}T_d &= \sum_{i=1}^{\infty}\int_{(i-1)\times T}^{iT}\frac{t\times Pr\{splitting\ is\ detected\ at\ slot\ i\}}{T}dt\\&= T\left(\frac{1}{p}-\frac{1}{2}\right).\end{aligned}$$

T_d reaches its upper bound when p is minimum. Conditioned on $n_1 > 0$, $n_2 > 0$, and $0 < \lambda \leq 1$, p is minimum when n_1 or n_2 equals 1. That is, it is hardest to detect when only one sensor node deviates from the rest of the sensor nodes in a cluster.

Therefore,

$$T_d \leq T\left(\frac{1}{\lambda[1-(1-\lambda)^{n-1}]}-\frac{1}{2}\right).$$

□

Since the actual working time of each sensor node is proportional to λ, based on Theorem 1, we can set λ to the minimum value such that the specified split-detection delay can be met while the actual working time of each sensor node is minimized.

Note that, with the randomized intracluster scheduling scheme proposed above, it is possible that, in a particular time slot, all the sensor nodes in a cluster happen to sleep and, hence, the sink node cannot restore the readings in this time slot. A solution to this problem is to use the round-robin scheduling method in our previous work [17] together with the randomized intracluster scheduling method. The round-robin scheduling method evenly distributes the workload among the nodes in a cluster and guarantees at least one node active in any time slot. In the working time slot assigned by the round-robin scheduling, the sensor node must go into work state. For the rest of the time slots, the sensor node randomly goes into the sleep/work state with the probability of λ. In this case, the number of working nodes in any time slot is no less than that in the purely randomized intracluster scheduling. Therefore, the upper bound of the expectation of the detection delay for cluster split still holds for the joint scheduling scheme.

5.3 Energy Saving

With joint round-robin and randomized scheduling, it is obvious that, after the formation of clusters, for any sensor node within a cluster of size n, the average amount of data it reports to the sink node is only a fraction $\frac{\lambda(n-1)+1}{n}$ of the amount of data without scheduling.

Note that the amount of energy saving varies with the size of the cluster. For sensor nodes within a large cluster, a long lifetime can be expected due to a low workload. For sensor nodes within a small cluster, their workload is high and, thus, the lifetime is short, due to low data redundancy in the vicinity. In other words, they are critical nodes, and the absence of their measures results in a large information loss. In this case, there is no room for the scheduling algorithm to exploit spatial correlation for energy saving. The only solution to prolong the lifetime of these critical nodes is to deploy more sensor nodes in their vicinity to share their workload.

5.4 Dynamic Adjustment

Once the sink node detects that a cluster should be split, it asks all sensor nodes in the corresponding cluster to work simultaneously. Then, the clustering algorithm will be executed to regroup these sensor nodes into several clusters in response to local spatial correlation changes. It is obvious that the number of clusters will keep increasing, since there are only splitting operations in the above adjustment. In the worst case, most sensors in the network will be woken up to work simultaneously. To avoid this situation, the sink node can recluster the whole network when the current number of clusters becomes significantly larger than the number of clusters at the previous network-wide clustering.

When the spatial correlation within a subregion remains stable, the frequency of dynamic adjustment of clusters should be very low. We stress that *spatial correlation is quite stable for many applications even if the monitored phenomenon changes dramatically*. For instance, in the experimental results presented in Section 7.2, we changed light strength very quickly by tuning the dimmer of a desk lamp. The sampling data from the sensors within the same box remained similar no matter how fast we changed the light.

5.5 Data Restoration at the Sink

Based on the joint round-robin and randomized scheduling, for a cluster of size n, the number of working sensor nodes in any time slot varies between 1 and n with an expectation of $1+(n-1)\times\lambda$. The sink node uses the following restoration rule to restore the observation of sleeping nodes:

Restoration Rule. For any sleeping node, the observation at instant i, denoted as v_i, is restored as $\frac{min_i+max_i}{2}$, where max_i and min_i denote the maximum and minimum observation of the working sensor nodes within the same cluster at instant i, respectively.

Theorem 2. *If the cluster is not split, with the restoration rule given above, the upper bound of the restoration error at instant* i *is* $m-\frac{max_i-min_i}{2}$, *where* m *is the dissimilarity threshold value.*

Proof. If the cluster is not split, the difference of two observations from any pair of sensor nodes, no matter whether they are working or sleeping, should be less than m. Therefore, the upper bound and lower bound of observation from sleeping nodes at instant i is min_i+m and max_i-m, respectively. The length of this feasible region for the observation of a sleeping node is $min_i+m-\ (max_i-m)=2\times m+min_i-max_i$. Since we take the middle point of this feasible region as the restoration value, the difference between the actual value and restoration value is no more than half the length of the feasible region, that is, $\frac{2\times m+min_i-max_i}{2}=m-\frac{max_i-min_i}{2}$. □

Note that another way to restore the data is to use the average of all working nodes, but in the worst case, the error bound could be as large as m. We ignore the proof due to lack of space.

6 Exploiting Temporal Correlation

6.1 Motivation and Methodology

Typically, each sensor node measures the environment at a fixed interval, and the time-ordered sequence of samples constitutes a time series. However, transmitting all data from each sensor node back to the sink node is often prohibitive due to limited bandwidth and heavy energy consumption on data transmission. In order to reduce the volume of transmitted data from each individual sensor node, we approximate the observed time series within a given error bound. Our technique builds on the fact that, for most time series consisting of environmental measures such as temperature and humidity, linear approximation is simple and works well enough in a short time period. Therefore, we use the piecewise linear approximation technique to approximate the time series at each individual sensor node. If computational capacity permits, it is possible to adopt other approximation and compression techniques such as wavelet transformation. We did not explore complex approximation techniques in this paper since it is still unrealistic to perform complex calculation with the current sensor nodes.

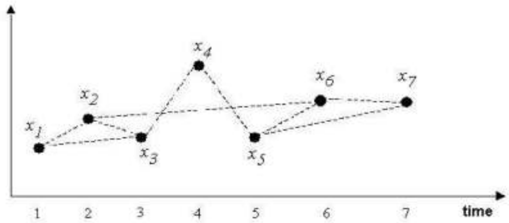


Fig. 4. An example of the piecewise linear approximation.

The sensor node maintains a fixed size of buffer to temporally store the latest sampling values. Once the buffer is full, the sensor node calculates the line segments approximating the original time series. Then, only the end points from every line segment, rather than the whole time series, will be transmitted to the sink node. Usually, each line segment represents more than two measures, so we can expect a significant decrease in data volume.

6.2 Problem Modeling

Piecewise linear approximation is a well-investigated research area [7], [14], [15], [22], [25], [27]. The idea is to use a sequence of line segments to represent the time series with a bounded approximation error. Since a line segment can be determined by only two end points, piecewise linear approximation can make the storage, transmission, and computation of time series more efficient.

Most existing piecewise linear approximation algorithms use a standard linear regression technique to calculate a line segment fitting the original data with the minimum mean squared error. Unlike these methods, we use a simpler method to reduce the computation complexity. In our method, the end points of each line segment must be the points in the time series and, hence, the linear fitting procedure is not necessary. For the observations that are not the end points of the selected line segments in the time series, their values are approximated by the corresponding points in the line segments. To facilitate understanding, Fig. 4 shows a possible scenario for the piecewise linear approximation, where three line segments (four values) (x_1, x_2), (x_2, x_6), and (x_6, x_7) could be used to approximate the time series.

The problem. For a given time series and a given error bound ϵ, find the minimum number of line segments to approximate the time series such that the difference between any approximation value and its actual value is less than ϵ. The end points of the line segments must be the points in the time series.

6.3 The PLAMLiS Algorithm

In [18], it is disclosed that the PLAMLiS problem can be solved in polynomial time. Assume that the time series consists of n points $\{x_1, x_2, \dots, x_n\}$. The basic idea is to build a graph with the data points as the vertices. An edge is established between x_i and $x_j (i < j)$ if the line segment (x_i, x_j) meets the error bound; that is, the difference between the approximation value of $x_k (i < k < j)$ and the actual value of x_k is not larger than the given error bound. Note that the approximation value of x_k is the intersection point of the line segment (x_i, x_j) and the vertical line $x = x_k$. The PLAMLiS problem is solved by searching the shortest path from x_1 to x_n in Fig. 4. Fig. 4 shows the graph and the shortest path (x_1, x_2, x_6, x_7) that constitutes the optimal solution.

In the worst case, the above algorithm has the time complexity of $O(n^3)$ and requires a space cost of $O(n^2)$. The calculation and storage complexity of the solution might be too high for the sensor nodes. Therefore, we propose a greedy algorithm to solve it. If the time series consists of n points, the greedy PLAMLiS algorithm can run in $O(n^2 \log n)$ time and takes $O(n)$ space.

The greedy algorithm converts the PLAMLiS problem into a set-covering problem as follows: Suppose that there is a time series X consisting of n points $\{x_1, x_2, \dots, x_n\}$. For each point x_i in the time series, associate it with point x_j $(j > i)$, which is farthest away from point x_i, and the line segment (x_i, x_j) meets the given error bound; that is, the difference between the approximation value of x_k $(i < k < j)$ and the actual value of x_k $(i < k < j)$ is not larger than the given error bound. Let F_i denote the subset consisting of all the points on this line segment $\{x_i, x_{i+1}, \dots, x_j\}$. Eventually, we have a set F with n subsets $(\{F_1, F_2, \dots, F_n\})$. The running time to construct the set F is $O(n^2 logn)$, and the memory needed is $O(n)$.

Now, the PLAMLiS problem is converted to the problem of picking up the least number of subsets from F, which covers all the elements in set X. This is the minimum set cover problem, which is proved NP-complete. We use the following well-known greedy algorithm to solve this problem. The basic idea is that, at each stage, pick the set that covers the largest number of remaining elements that are uncovered. The algorithm ends whenever all the elements are covered. This greedy algorithm can run in time $O(\sum_{S \in F} |S|)$ with a P(n)-approximation rate, where $P(n) = H(max|S| : S \in F)$ $(H(d) = \sum_{i=1}^{d} \frac{1}{i}$, which is the dth harmonic number). The space complexity is $O(n)$. In the PLAMLiS problem, since $|F| = n$ and $|S| \leq n$ $(S \in F)$, the running time is bounded by $O(n^2)$.

On the whole, the greedy PLAMLiS algorithm can run in $O(n^2 \log n)$ time and takes $O(n)$ memory space.

7 Comprehensive Performance Evaluation

7.1 Evaluation Methodology

In the following sections, we first evaluate the performance of EEDC when only spatial correlation is exploited. The evaluation is based on both a real test bed, as well as a large-scale synthetic data set. We then evaluate the performance of the PLAMLiS algorithm in exploring temporal correlation. Finally, we investigate the performance when spatial correlation and temporal correlation are jointly considered to further reduce energy consumption.

7.2 Experiment Setup of Testing Spatial Correlation

We experimentally test the EEDC framework based on MICA2 sensor nodes [5]. As illustrated in Fig. 5, we deploy 18 MICA2 sensor nodes in a 3×6 grid layout on a big table to sample the light intensity. The unit length of each grid is 1 foot. A desk lamp with a dimmer is the only light source in the room. We use several boxes with different sizes of holes on the top to divide the area into subregions with

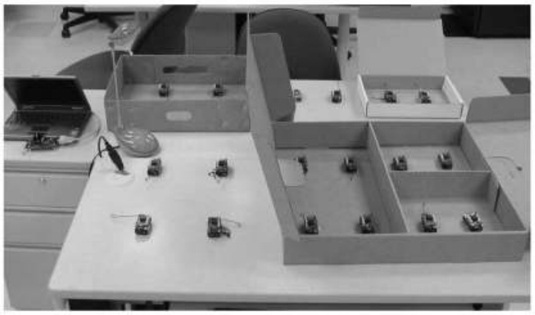


Fig. 5. The test bed.

different light intensity. The deployment of sensor nodes and the boxes are illustrated in Fig. 6. The monitored phenomenon is generated by varying the light intensity of the lamp. The onboard analog-to-digital converter (ADC) translates a light intensity raw reading into an integer value between 0 and 1,024. We implement the EEDC framework on the sink node and the MICA2 sensor nodes to collect the light intensity data. The sampling rate is 2 samples per second at the working sensors, and the data collection time is 10 minutes.

The purpose of this experiment is twofold. First, we need to verify the correctness of the proposed clustering algorithm. Second, since the energy saving should not be achieved at the cost of observation fidelity, we need to verify that the observation fidelity with EEDC is acceptable.

Although this experiment is not a real application, we remark that the experimental design may be representative for some real applications with sensor networks, for instance, the monitoring system for storage rooms in a grocery warehouse.

7.3 Experimental Results of Exploring Spatial Correlation

7.3.1 The Correctness of Clustering with EEDC

Although the clique-covering problem for a general graph is NP-complete, it is easy to know the optimal clique covers in our experiment since the knowledge of which sensor node belongs to which subregion is known a priori. We use the criteria in Section 4.2 to check the dissimilarity and set $m = 30$, $t = 95$ percent, and $gmax\_dst = 3$ feet. By calculating the pairwise dissimilarity measure, we get a graph as shown in Fig. 7, where a link between two nodes indicates that they are similar according to the above criteria. From this figure, we can see that all cliques consist of sensor nodes in the same box, which validates the effectiveness of the dissimilarity measure. The output of the clustering algorithm is illustrated in Fig. 8, which is apparently the optimal solution for this specific simple graph.

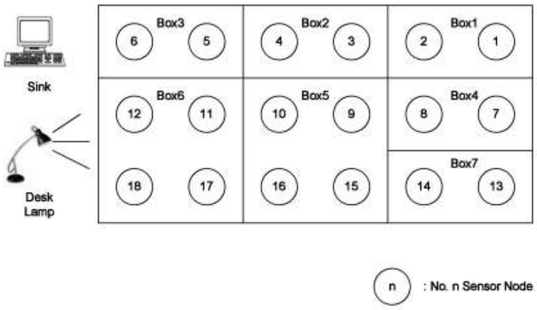


Fig. 6. The sensor nodes and the boxes.

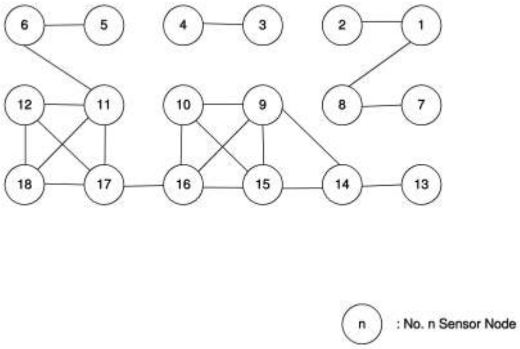


Fig. 7. The generated graph.

7.3.2 The Observation Fidelity with EEDC

The difference distortion measure has been broadly used in image compression to evaluate the fidelity of a reconstructed image against the original image [23]. The difference distortion measure σ^2 is given by

$$\sigma^2 = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N}(X_{ij} - Y_{ij})^2}{M \times N},$$

where X_{ij} is the jth actual sampling value from the ith sensor node, Y_{ij} is the jth restoration value of the ith sensor node at the sink, N is the total number of sensor nodes, and M is the total number of samples from each sensor node.

The absolute value of σ^2 is not meaningful without considering the degree of variation in magnitude. So, we normalize the difference distortion measure by the average variation of samples and used it as the measure of observation fidelity. Formally,

$$\sigma^2_{norm} = \frac{\sigma^2}{\frac{\sum_{i=1}^{N} Var(X_i)}{N}} = \frac{N\sigma^2}{\sum_{i=1}^{N} Var(X_i)},$$

where $Var(X_i)$ is the observation variation of the ith sensor node. Generally, a smaller σ^2_{norm} value indicates higher

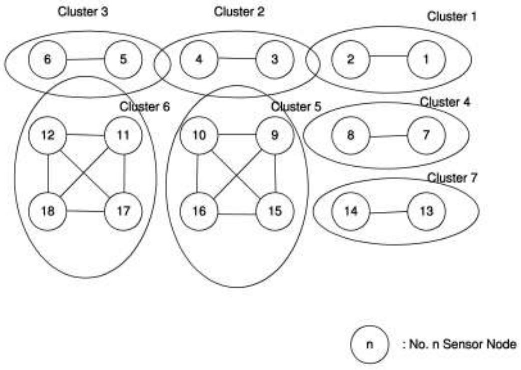


Fig. 8. The clustering result with the real data set.

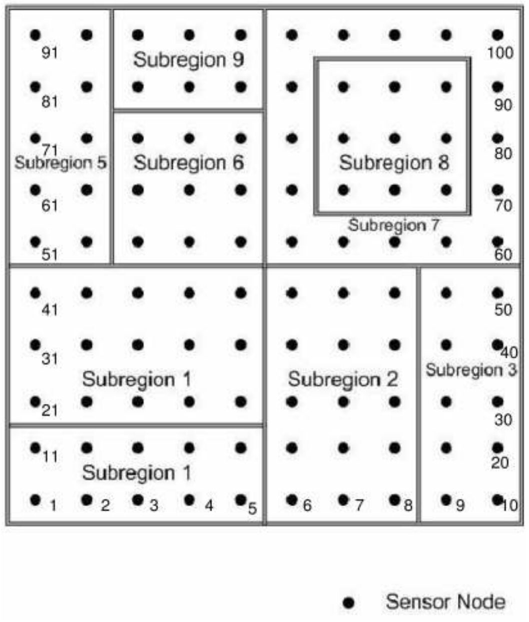


Fig. 9. The field with nine distinguished subregions.

observation fidelity. When we set $m = 30$, $t = 95$ percent, and $gmax\_dst = 3$ feet, the σ^2_{norm} of the light intensity in our test bed is equal to 0.0093, indicating that the data collected with EEDC has high fidelity.

7.3.3 Energy Saving

In this case study, at any time instant, only one sensor node in a cluster is scheduled to work. The 18 sensor nodes were grouped into seven clusters with EEDC, as shown in Fig. 8. By calculating $\frac{2*2*5+4*4*2}{18} \approx 3$, we can see that, without using EEDC, on the average, each sensor will spend three times more energy in sampling and data transmission.

7.4 Large-Scale Synthetic Data Generation

Due to the high system cost, we cannot afford an experiment with hundreds of sensor nodes. In order to further investigate the performance of EEDC with large-scale networks, we generate large traces of a spatially correlated data set based on a mathematical model proposed in [13]. We utilize the software toolkit provided in [13] to extract the model parameters from small-scale real data sets and generate large-scale synthetic data sets based on the model parameters. The toolkit has been validated by comparing the statistical features of the synthetic data set and the experimental data set [13].

Initially, we use our test bed in Section 7.2 to collect a small-size real data set. Then, we utilize the synthetic data generation toolkit [13] on the data set from each individual subregion to generate a larger data set for each individual subregion. As a result, a field consisting of nine distinguished subregions with 100 sensor nodes in a 10×10 grid layout is generated, as shown in Fig. 9.

7.5 Performance Results on Large-Scale Synthetic Data

7.5.1 The Correctness of Clustering with EEDC

Since we know which sensor node belongs to which subregion, it is easy to verify the correctness of the clustering algorithm. We set $m = 20$, $t = 95$ percent, and $gmax\_dst = 8$ distance units. The distant unit is defined as the distance between two neighboring sensor nodes in a row. By calculating the pairwise dissimilarity measure and performing the clustering algorithm, we obtain nine clusters, each for a subregion.

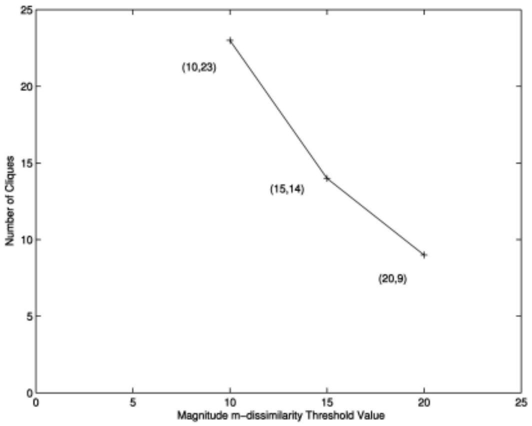


Fig. 10. Magnitude m-dissimilarity threshold versus the number of cliques.

7.5.2 The Observation Fidelity and Energy Saving with EEDC

By varying the value m in the magnitude m-dissimilarity and applying different intracluster scheduling methods, we collect a set of performance data, based on which Figs. 10, 11, and 12 are drawn. Fig. 10 demonstrates that, with the decrease of m, the number of cliques increases. This conforms to intuition, because a lower m value corresponds to a higher data resolution requirement. Note that the number of cliques is irrelevant to the intracluster scheduling methods. Therefore, Fig. 10 will not change under different scheduling methods.

Figs. 11 and 12 compare the performance of two different intracluster scheduling methods discussed in previous sections in terms of observation fidelity and energy saving. With the round-robin scheduling, only one sensor node

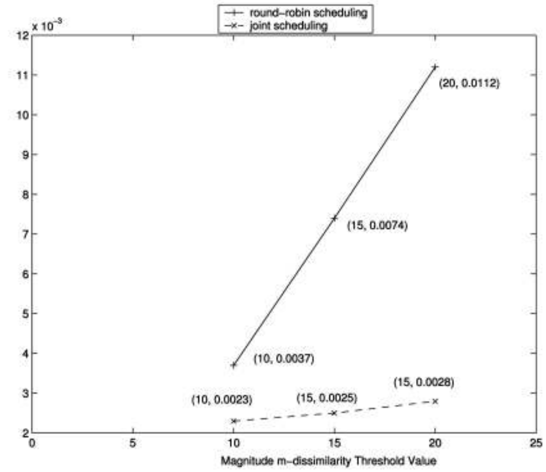


Fig. 11. Obsevation fidelity.

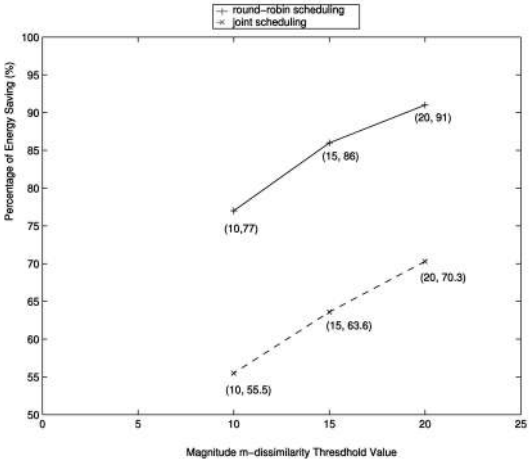


Fig. 12. Energy savings.

from a cluster is working at a time, whereas, with the joint round-robin and randomized scheduling, the multiple sensor nodes are working simultaneously within each cluster. Note that, in our experiment, the wakeup probability λ is chosen cluster by cluster such that the expectation of the split detection delay of a specific cluster $T_d \leq 4T$. The data restoration rule discussed in Section 5.5 is applied to both cases to restore the observations of sleeping nodes. Note that, with the round-robin scheduling, there is only one working node in each cluster at any given time; the sink node simply assumes that the sampling values of all sleeping sensor nodes in the same cluster are equal to that of the working node. Figs. 11 and 12 clearly demonstrate the trade-off between energy saving and observation fidelity: Compared with the round-robin scheduling, the joint round-robin and randomized scheduling can significantly improve the observation fidelity, but with a smaller energy savings.

7.6 Response to Spatial Correlation Changes

In order to verify that the spatial correlation changes can be detected by EEDC and demonstrate how EEDC responds to the changes, we simulate a scenario where an opaque object suddenly covers sensor nodes 10, 20, 30, 40, and 50 in subregion 3 and makes the covered area totally dark. Therefore, the readings from those covered sensor nodes should be totally different from those from the rest of the nodes, and a cluster split action is expected in response to significant changes in the spatial correlation.

In this scenario, the joint round-robin and randomized scheduling is used and the wakeup probability λ is set such that the expectation of the split detection delay of every cluster $T_d \leq 4T$. An active sensor node sends 10 packets in a working shift with a length of T. We monitor the number of packets sent back to the sink node from the 10 sensor nodes in subregion 3, which is illustrated in Fig. 13. Originally, the number of packets that is sent to the sink node per work shift oscillates around 30 packets. The lid comes in at time instant $3.5T$. The sink detects the changes in spatial correlation between $4T$ and $5T$ and instructs all the sensor nodes to wake up and send sampling at $5T$. The sink node gets enough data for reclustering at $6T$, executes the clustering algorithm to split the sensor nodes in subregion 3 into two clusters, and then sends the updated working schedules to the sensor nodes at $7T$. Afterward, the sensor nodes work according to the updated schedule, and the number of packets sent to the sink node oscillates around 40 packets.

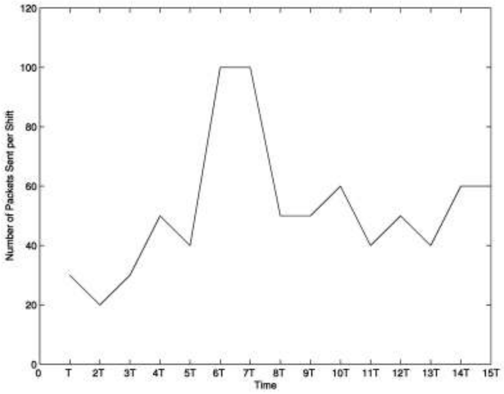


Fig. 13. Response to spatial correlation changes.

7.7 Performance Evaluation of Exploring Temporal Correlation

To evaluate the performance of the proposed PLAMLiS algorithm, we first apply the algorithm to an indoor temperature data set we collect on the Crossbow MICA2 platform. The sampling interval is set to 2 minutes, and 877 samples are taken in total.

By varying the error bound from 0.5 °C to 0.05 °C, we get Figs. 14, 15, 16, and 17. From these figures, we can see that the number of line segments generated by the PLAMLiS algorithm to fit the original data curve is significantly less than the number of points in the original data curve. When the error bound is set to 0.5, only 12 line segments are needed to transmit. Since each line segment is presented by two end points, only 24 points are transmitted with the PLAMLiS algorithm. Compared to 877 points, if the PLAMLiS algorithm is not used, this translates to a 97.3 percent data reduction in transmission.

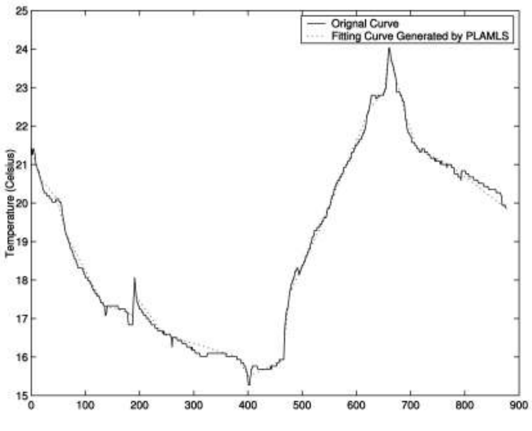


Fig. 14. $\text{Error bound} = 0.5$, number of line segments with $\text{PLAMLiS} = 12$.

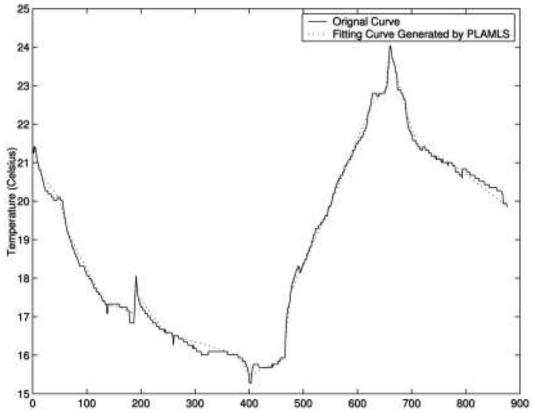


Fig. 15. Error bound $= 0.3$, number of line segments with PLAMLiS $= 23$.

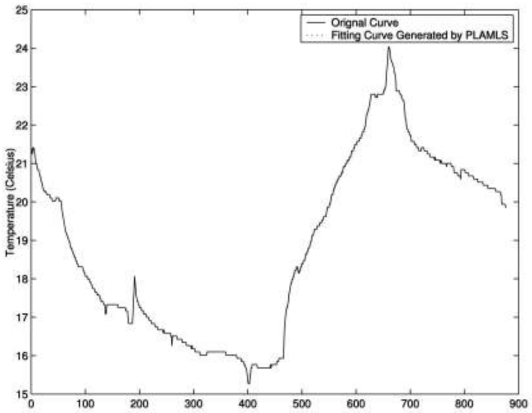


Fig. 17. Error bound $= 0.5$, number of line segments with PLAMLiS $= 158$.

With the PLAMLiS algorithm, the transmitted data increases with the decrease of error bound. However, as shown in Fig. 17, since even the error bound is as small as $0.05\,^{\circ}\mathrm{C}$, only 158 line segments need to be transmitted, which translates to a 64 percent data reduction in transmission.

We also apply the PLAMLiS to the light intensity data set we collect in our experimental test bed. Unlike the indoor temperature data set, the light intensity data set is not smooth at all. Even in this situation, Fig. 18 illustrates that only 58 line segments are needed to approximate the time series, which translates to a 42 percent reduction in transmission.

7.8 Performance of Jointly Exploring Spatial Correlation and Temporal Correlation

In this phase, both the PLAMLiS algorithm and the dynamical clustering algorithm are integrated into EEDC, as illustrated in Fig. 2.

7.8.1 Restoration Data Quality

Suppose that the error bound of the dynamical clustering algorithm is set to ϵ_1 (that is, $m = \epsilon_1$ in the magnitude m-dissimilarity), and the error bound of the PLAMLiS algorithm is set to ϵ_2. Since the temporal correlation and spatial correlation are exploited orthogonally, it is easy to see that, in the enhanced EEDC framework, the error bound of data restoration $\epsilon = \epsilon_1 + \epsilon_2$. That is to say, any restoration value in the sink node will not differ from its actual value more than $\epsilon_1 + \epsilon_2$.

However, given an error bound ϵ with the enhanced EEDC framework, we are faced with a question of assigning appropriate values to ϵ_1 and ϵ_2 such that $\epsilon_1 + \epsilon_2 = \epsilon$ and data transmission is minimized. The heuristic is that, for the sensor nodes having strong temporal correlation in their readings, ϵ_2 can be set large. For the sensor nodes having strong spatial correlation in their neighborhood, ϵ_1 can be set large. The method of obtaining the optimal values and dynamically adjusting these values is left as our future work.

7.8.2 Performance Evaluation

To illustrate the benefit of using the PLAMLiS algorithm in the EEDC framework, we evaluate the performance of EEDC with and without using the PLAMLiS algorithm. In all tests, round-robin scheduling is used for simplicity. In the EEDC framework without PLAMLiS, the error bound threshold value ϵ is equal to m in the magnitude m-dissimilarity measure. In the EEDC framework with PLAMLiS, the error bound threshold value $\epsilon = \epsilon_1 + \epsilon_2$, as discussed above. In all our test cases, ϵ_1 takes 75 percent of ϵ and ϵ_2 takes 25 percent.

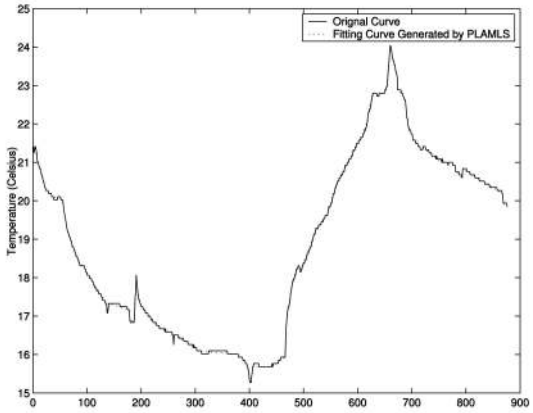


Fig. 16. Error bound $= 0.1$, number of line segments with PLAMLiS $= 56$.

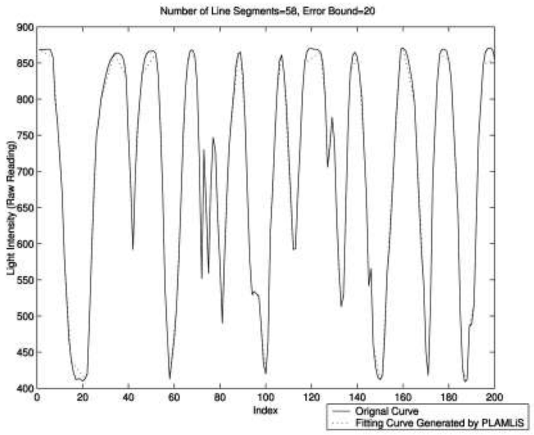


Fig. 18. Error bound $= 20$, number of line segments with PLAMLiS $= 58$.

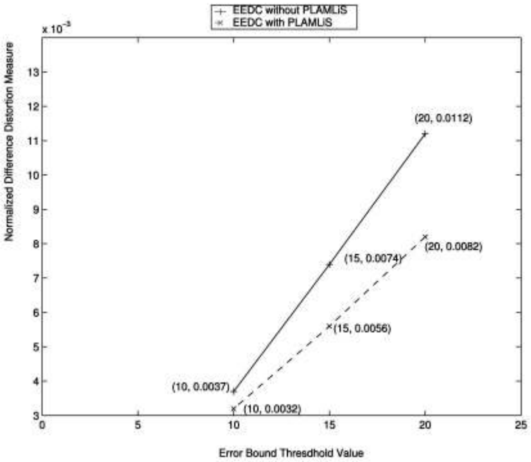


Fig. 19. Observation fidelity.

Figs. 19 and 20 clearly demonstrate that, after introducing the PLAMLiS algorithm, the data restoration accuracy is improved and the energy consumption decreases. This is because, given the same error bound ϵ, the actual error bound used for dynamical clustering ϵ_2 in the enhanced EEDC framework is smaller than that in the EEDC framework without PLAMLiS. Therefore, more cliques are generated when the enhanced EEDC framework is used, as illustrated in Fig. 21. This actually increases the spatial sampling rate and helps to improve the data restoration accuracy. The increase in energy consumption by using more clusters is offset by the energy savings with the PLAMLiS algorithm.

8 Conclusion

In this paper, we design an EEDC framework that is aware of the spatiotemporal correlation among the sensing data. Spatial correlation is exploited by dynamically grouping sensor nodes into clusters based on the dissimilarity measure of sampling data. With this method, the whole network is divided into several subregions, with each covered by a cluster of sensor nodes. Since the clusters are based on the features of sampling data, scheduling based on the clusters is much more accurate than scheduling based purely on the sensing range of sensor nodes. Temporal correlation is exploited by using the piecewise linear approximation technique to represent the sampling data and minimize the data transmission under a given bound on approximation accuracy. We discuss the details of all major components in the EEDC framework, including the calculation of dissimilarity, sensor clustering, sensor scheduling, data restoration, and the PLAMLiS algorithm.

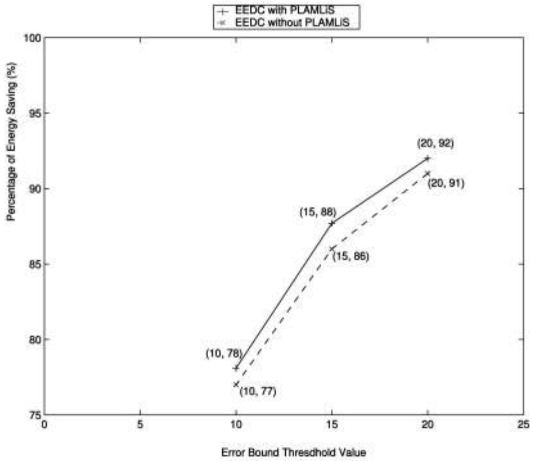


Fig. 20. Energy savings.

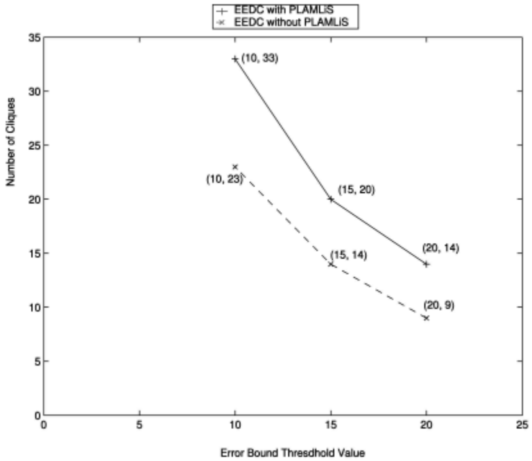


Fig. 21. Error bound threshold versus the number of cliques.

We thoroughly evaluate the performance of the EEDC framework using a real experiment based on MICA2 motes [5] and a large-scale synthetic data set. Experimental results demonstrate that the EEDC framework can effectively save energy without losing observation fidelity.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Foundation for Innovation (CFI).

References

[1] M. Batalin, G. Sukhatme, Y. Yu, M. Rahimi, G. Pottie, W. Kaiser, and D. Estrin, “Call and Response: Experiments in Sampling the Environment,” *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys ’04),* Nov. 2004.

[2] J. Chou, D. Petrovic, and K. Ramchandran, “A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks,” *Proc. IEEE INFOCOM ’03,* Mar. 2003.

[3] A. Coman, M.A. Nascimento, and J. Sander, “Exploiting Redundancy in Sensor Networks for Energy Efficient Processing of Spatiotemporal Region Queries,” *Proc. 14th ACM Conf. Information and Knowledge Management (CIKM ’05),* Nov. 2005.

[4] R. Cristescu, B. Beferull-Lonzano, and M. Vetterli, “On Network Correlated Data Gathering,” *Proc. IEEE INFOCOM ’04,* Mar. 2004.

[5] Crossbow Technology, Feb. 2005, http://www.xbow.com/Products/products.htm.

[6] A. Deshpande, C. Guestrin, S. Madden, J Hellerstein, and W. Hong, “Model-Driven Data Acquisition in Sensor Networks,” *Proc. 13th Int’l Conf. Very Large Data Bases (VLDB ’04),* Aug. 2004.

[7] D.H. Douglas and T.K. Peucker, “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature,” *Canadian Cartographer,* vol. 10, no. 2, pp. 112-122, Dec. 1973.

[8] G. Edward, P. Box, and G.M. Jenkins, *Time Series Analysis: Forecasting and Control.* Prentice Hall, 1994.

[9] F. Emekci, S.E. Tuna, D. Agrawal, and A.E. Abbadi, "BINOCULAR: A System Monitoring Framework," *Proc. First Workshop Data Management for Sensor Networks (DMSN '04),* Aug. 2004.

[10] C. Hsin and M. Liu, "Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithm," *Proc. Information Processing in Sensor Networks '04,* Apr. 2004.

[11] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02),* July 2002.

[12] H. Jagadish, A.O. Mendelzon, and T. Milo, "Similarity-Based Queries," *Proc. 14th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS '95),* May 1995.

[13] A. Jindal and K. Psounis, "Modeling Spatially-Correlated Sensor Network Data," *Proc. Sensor and Ad Hoc Comm. and Networks (SECON '04),* Oct. 2004.

[14] E. Keogh and P. Smyth, "A Probabilistic Approach to Fast Pattern Matching in Time Series Databases," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining,* Aug. 1997.

[15] E. Keogh and M. Pazzani, "An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback," *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining,* Aug. 1998.

[16] Y. Kotidis, "Snapshot Queries: Towards Data-Centric Sensor Networks," *Proc. 21st Int'l Conf. Data Eng. (ICDE '05),* Apr. 2005.

[17] C. Liu, K. Wu, and J. Pei, "A Dynamic Clustering and Scheduling Approach to Energy Saving in Data Collection from Wireless Sensor Networks," *Proc. Second IEEE Int'l Conf. Sensor and Ad Hoc Comm. and Networks,* Sept. 2005.

[18] C. Liu and K. Wu, "Optimal and Approximate Solutions to PLAMLiS Problem in Wireless Sensor Networks," Technical Report DCS-313-IR, Univ. of Victoria, Sept. 2005.

[19] C. Lund and M. Yannakakis, "On the Hardness of Approximating Minimization Problems," *J. ACM,* vol. 41, 1994.

[20] A. Mainwaring, J. Polastre, D. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. Workshop Wireless Sensor Networks and Application (WSNA '02),* Sept. 2002.

[21] S. Pattem, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," *Proc. Information Processing in Sensor Networks,* Apr. 2004.

[22] Y. Qu, C. Wang, and S. Wang, "Supporting Fast Search in Time Series for Movement Patterns in Multiples Scales," *Proc. Seventh Int'l Conf. Information and Knowledge Management,* Nov. 1998.

[23] K. Sayood, *Introduction to Data Compression.* Morgan Kaufmann, 1996.

[24] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Proc. Fifth Symp. Operating Systems Design and Implementation (OSDI '02),* Dec. 2002.

[25] H. Shatkay and S. Zdonik, "Approximate Queries and Representations for Large Data Sequences," *Proc. 12th IEEE Int'l Conf. Data Eng.,* Feb. 1996.

[26] D. Tian and N.D. Georganas, "A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks," *Proc. ACM Workshop Wireless Sensor Networks and Applications,* Oct. 2002.

[27] C. Wang and S. Wang, "Supporting Content-Based Searches on Time Series via Approximation," *Proc. 12th Int'l Conf. Scientific and Statistical Database Management,* July 2000.

[28] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks," *Proc. First Biennial Conf. Innovative Data Systems Research (CIDR '03),* Jan. 2003.

[29] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-Lived Sensor Networks," *Proc. 10th IEEE Int'l Conf. Network Protocols,* Nov. 2002.

[30] S. Yoon and C. Shahabi, "Exploiting Spatial Correlation Towards an Energy Efficient Clustered Aggregation Technique (CAG)," *Proc. IEEE Int'l Conf. Comm. 2005,* May 2005.

[31] D. Zuckerman, "NP-Complete Problems Have a Version That's Hard to Approximate," *Proc. Eighth IEEE Ann. Structure in Complexity Theory Conf.,* May 1993.

Chong Liu is a PhD candidate in the Computer Science Department, University of Victoria. His major research interests include energy-efficient node clustering, scheduling, and data retrieval in wireless sensor networks. He is a student member of the IEEE.

Kui Wu received the PhD degree in computing science from the University of Alberta, Canada, in 2002. He then joined the Department of Computer Science, University of Victoria, Canada, where he is currently an assistant professor. His research interests include mobile and wireless networks, sensor networks, network performance evaluation, and network security. He is a member of the IEEE.

Jian Pei received the PhD degree in computing science from Simon Fraser University, Canada, in 2002, where he is currently an assistant professor of computing science. His research interests can be summarized as developing effective and efficient data analysis techniques for novel data-intensive applications. Particularly, he is currently interested in various techniques of data mining, data warehousing, online analytical processing, and database systems, as well as their applications in bioinformatics, privacy preservation, and software engineering and education. His current research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the US National Science Foundation (NSF), IBM, Hewlett-Packard (HP), and the Canadian Imperial Bank of Commerce (CIBC). He is a member of the ACM and a senior member of the IEEE.