Predictive Modeling-Based Data Collection in Wireless Sensor Networks*

Lidan Wang and Amol Deshpande

Computer Science Department, University of Maryland,
A.V. Williams Building, College Park, MD 20742, USA
{lidan,amol}@cs.umd.edu

Abstract. We address the problem of designing practical, energy-efficient protocols for data collection in wireless sensor networks using predictive modeling. Prior work has suggested several approaches to capture and exploit the rich spatio-temporal correlations prevalent in WSNs during data collection. Although shown to be effective in reducing the data collection cost, those approaches use simplistic corelation models and further, ignore many idiosyncrasies of WSNs, in particular the broadcast nature of communication. Our proposed approach is based on approximating the joint probability distribution over the sensors using *undirected graphical models*, ideally suited to exploit both the spatial correlations and the broadcast nature of communication. We present algorithms for optimally using such a model for data collection under different communication models, and for identifying an appropriate model to use for a given sensor network. Experiments over synthetic and real-world datasets show that our approach significantly reduces the data collection cost.

1 Introduction

Wireless sensor networks (WSNs), comprising of tiny, radio-enabled sensing devices open up new opportunities to observe and interact with the physical world, and have been applied in domains ranging from patient health monitoring through the use of biomedical sensors to military applications such as battlefield surveillance [1]. In this paper, we address the problem of designing energy-efficient protocols for collecting all data observed by the sensor nodes in a wireless sensor network at an Internet-connected base station at a specified frequency [2,25,22,4]. The key issue in designing such data collection protocols is modeling and exploiting the strong spatio-temporal correlations present in most sensor networks (see Figure 1). In most sensor network deployments, especially in environmental monitoring applications, the data generated by the sensor nodes is highly correlated both in time (future values are correlated with current values) and in space (two co-located sensors are strongly correlated). Naive data collection protocols tend to be significantly suboptimal in the presence of such correlations. These correlations can usually be captured quite easily by constructing predictive models using either prior domain knowledge or historical data traces. However, because of the distributed nature of data generation in sensor networks, and the resource-constrained

---

nature of sensor nodes, traditional data compression techniques cannot be easily adapted to exploit such correlations.

The distributed nature of data generation has been well-studied in the literature under the name of *Distributed Source Coding* [26,30,31,27]. In their seminal work, Slepian and Wolf [26] prove that it is theoretically possible to encode the correlated information generated by distributed data sources (in our case, the sensor nodes) at the rate of their joint entropy *even if the data sources do not communicate with each other*. However this result is non-constructive, and constructive techniques are known only for a few specific distributions [23]. More importantly, these techniques require precise and perfect knowledge of the correlations. This may not be acceptable in practical sensor networks where deviations from the modeled correlations must be captured accurately (we use DSC to provide a lower bound on the data collection cost; see Section 2.2). Pattem et al. [22] and Chu et al. [4], among others, propose practical data collection protocols that exploit the spatio-temporal correlations while guaranteeing correctness; however, these protocols may exploit only a subset of the correlations, and further require the sensor nodes to communicate with each other (increasing the overall cost).

Sensor networks, especially wireless sensor networks, exhibit other significant peculiarities that make the data collection problem challenging. First, sensor nodes are typically computationally constrained and have limited memories. Hence, it may not be feasible to run sophisticated data compression algorithms on them. Second, the communication in wireless sensor networks is typically done in a broadcast manner – when a node transmits a message, all nodes within the radio range can receive the message. As we will see later, this enables many optimizations that would not be possible in a one-to-one communication model.

In this paper, we present an approach to exploit all the spatial correlations in the data by approximating the joint probability distributions using a subclass of undirected graphical models called *decomposable models*. We develop algorithms for performing data collection using such a model, and for choosing an appropriate decomposable model for a given sensor network. Our data collection protocols are also naturally able to exploit the broadcast nature of communication among wireless sensors. Finally, we present an extensive experimental study over several synthetic and real-world datasets, and demonstrate that the expressiveness of our data collection model leads to a significant reduction in the total transmission cost.

2 Background

We begin with presenting preliminary background on data compression in sensor networks, and discuss the prior approaches. We then present an overview of the class of decomposable models.

2.1 Notation and Preliminaries

We are given a sensor network with n nodes that continuously monitors a set of distributed attributes $\mathcal{X} = \{X_1, \cdots, X_n\}$, and generates a discrete data value vector
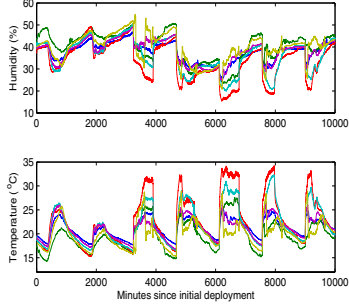
Fig. 1. A plot of several traces from the Intel Lab Dataset [18] shows the strong spatio-temporal correlations in the data
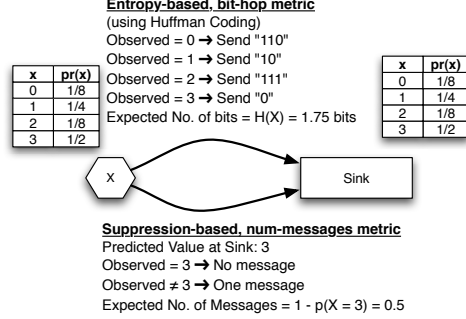
Fig. 2. Two extremes in the spectrum of communication model and data encoding options

$\mathbf{x}^t = \{x_1^t, \cdots, x_n^t\}$ at every time instance t[1]. Each attribute, X_i, may be an environmental property being sensed by the node (e.g., *temperature*), or it may be the result of an operation on the sensed values (e.g., in an anomaly-detection application, the sensor node may continuously evaluate a filter such as "$temperature > 100$" on the observed values). If the sensed attributes are continuous, we assume that an error threshold ϵ is provided and the readings are binned into intervals of size 2ϵ to discretize them. In this paper, we focus on optimal exploitation of spatial correlations at any given time t and drop the superscript in the rest of the paper; however we note that our ideas can be easily generalized to handle temporal correlations as well.

Predictive modeling-based approaches to data compression begin by building a predictive model over the sensor network attributes that is used to obtain a joint probability distribution (pdf) over the attributes. We denote this pdf by $p(X_1, ..., X_n)$.

We denote the communication graph of the sensor network by $\mathcal{G}_C = (\mathcal{X}, E)$, where E consists of the pairs of vertices that are within communication radius of each other. We denote by $d(X, Y)$ the minimum distance between X and Y in terms of number of hops. For simplicity, we assume all communication links to be perfect and identical[2], and consider two alternatives for computing communication costs:

(1) *bit-hop metric:* The total cost of sending a message containing n bits from X to Y is given by $n * d(X, Y)$. In practice, this can be approximated reasonably well by batching multiple messages together (at the cost of increasing latency).
(2) *num-messages metric:* The total cost of sending a message (that can contain at most 32 bytes) from X to Y is given simply by $d(X, Y)$. In other words, we only count the number of messages that are transmitted.

In many practical sensor network deployments, the cost of *receiving* a message at the sensor node can be quite high (sometimes as high as the *transmission* cost). For

---

[1] The time instances at which data is acquired depends on the application-specified frequency of data collection.

[2] Both these assumptions can be relaxed by assigning appropriate weights to the communication links and adjusting the cost metric formulas accordingly.

simplicity, in our analysis and algorithm descriptions, we assume that the cost of receiving a message at a sensor node is zero; we however present several experiments where we account for receiving cost as well.

The choice of cost metric is closely tied with how the data is encoded during data collection. We consider two extremes in the spectrum of possibilities:

Joint Entropy-Based Data Collection (bit-hop metric): Assuming that it is possible to compress the data optimally according to the joint pdf (e.g. using Huffman coding), the number of bits that need to be transmitted from a sensor node X (also called *source*) to the base station (called *sink*) is given by the *information entropy* of the distribution:

$$H_p(X) = \Sigma_x - p(x) \log(p(x))$$

where $p(X)$ denotes the probability distribution (pdf) over the attribute X.

If an approximation, $q(X)$, is used instead to compress, the number of bits transmitted is given by $H(p) + D(p||q)$, where $D(p||q)$, called *relative entropy*, is given by: $D(p||q) = \Sigma - p(x) \log(p(x)/q(x))$.

Suppression-Based Data Collection (num-messages metric): Full-scale data compression may not be feasible in a sensor network; hence prior work in this area has typically considered a suppression-based approach [21,25,4], where the base station uses the pdf to predict a value for the attribute X. The sensor node, which has access to the same distribution, also predicts the same value and only sends a message if the predicted value is different from the actual observed value. We denote the expected number of messages by $M_p()$, and note that:

$$M_p(X) = 1 - \max_x p(x)$$

Note that we assume here that only a single message is needed to update the base station with the correct values.

Figure 2 illustrates these two approaches for an example distribution. Our algorithms are invariant to the approach used for compression. However, we assume the ability to compute an analogous function to $H_p()$ or $M_p()$ for any distribution p. We use the former metric when analyzing the problem and for experiments on synthetic datasets, but use the latter, more practical, metric for our experiments on real datasets.

2.2   Predictive Modeling-Based Data Compression in Sensor Networks

Given a joint pdf over the sensor network attributes, the key problem in using it for data compression is the distributed nature of data generation. The natural way to use the joint pdf, $p(X_1, \ldots, X_n)$, would be to gather the sensed values at a central sensor node, and compress the data there. The data gathering cost, however, would typically dwarf any advantages gained by doing joint compression.

The prior research in this area has suggested several approaches that utilize a subset of correlations instead. One approach, called *Independent (IND)*, is to ignore the spatial correlations and to compress the data from each sensor node independently of the others (Figure 3 (i)). In other words, an approximate distribution $q_1(X_1, ..., X_n) = p(X_1)p(X_2)...p(X_n)$ is used for compression (where $p(X_i)$ denotes the marginal probability distribution of X_i, computed by summing over the remaining variables in $\mathcal{X}$).

The second approach, that we call *Clustering (CLSTR)* [22,4], is to group the sensor nodes into clusters, and to compress the data from the nodes in each cluster jointly. Figure 3 (ii) shows an example of this using three clusters $\{X_1\}, \{X_2, X_5\}, \{X_3, X_4\}$, which corresponds to using the distribution $q_2(X_1, ..., X_5) = p(X_1)p(X_2, X_5)p(X_3, X_4)$. In this approach, the intra-cluster spatial correlations are exploited during compression; however, the correlations across clusters are not utilized.

Several other approaches based on *routing driven compression* [22,24,6] have also been suggested. However, these approaches typically require joint compression and decompression of large numbers of data sources inside the network, and hence are not suited for resource-constrained sensor networks. We leave a detailed comparison of these approaches with our proposed approach to future work.

Distributed source coding (DSC), although not feasible in this setting for the reasons discussed earlier, can be used to obtain a lower bound on total communication cost as follows [7,6,27]. Let the sensor nodes be numbered in the increasing order of their distances from the base station (i.e., for all i, $d(X_i, sink) < d(X_{i+1}, sink)$). Then, the optimal scheme for using DSC is as follows: X_1 is compressed according to $p(X_1)$, and transmitted directly to the sink (incurring a total cost of $d(X_1, sink) \times H(X_1)$). X_2 is compressed according to $p(X_2|X_1)$ (since the sink already has the value of X_1, it is able to decode according this distribution). Note that, according to the distributed source coding theorem [26], sensor node X_2 does not need to know the actual value of X_1. Similarly, X_i is compressed according to $p(X_i|X_1 \ldots X_{i-1})$ and so on. The total communication cost incurred by this scheme is given by:

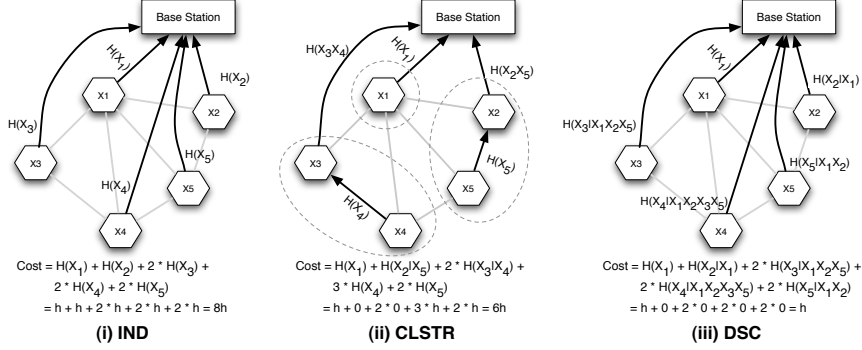$$DSC(p) = \Sigma_{i=1}^{n} d(X_i, sink) \times H_p(X_i|X_1, \ldots, X_{i-1})$$

Figure 3 (iii) shows this for our running example (note that X_5 is closer to sink than X_3 or X_4).

As we can see in Figure 3, if the spatial correlation is high, both IND and CLSTR would incur much higher communication costs than DSC. As an example, if $H(X_i) = h, \forall i$, and if $H(X_i|X_j) \approx 0, \forall i, j$ (ie., if the spatial correlations are almost perfect), the total communication costs of IND, CLSTR(as shown in the figure), and DSC would be $8h, 6h$, and h respectively.

2.3   Discussion: Factors Affecting Data Compression Quality

The difference between the data compression ratios achieved by DSC and other techniques can be attributed to two factors.

Approximation Loss: If a data collection scheme only uses a subset of the correlations, then even if the scheme was optimal (ie., was able to compress as well as DSC), more bits would have to be communicated than minimally needed. For the example setup in Figure 3, since IND assumes independence between the sensor nodes, the node X_2 must transmit $H(X_2)$ bits to the sink compared to $H(X_2|X_1)$ that DSC transmits; in fact, the difference between IND and DSC ($8h - h = 7h$), can be attributed entirely to Approximation Loss. Although CLSTR is able to exploit some of the spatial correlations, it does not exploit inter-cluster correlations. Since the clusters are typically small (for reasons discussed below), the Approximation Loss can be quite high for CLSTR as

Cost = H(X$_1$) + H(X$_2$) + 2 * H(X$_3$) +
2 * H(X$_4$) + 2 * H(X$_5$)
= h + h + 2 * h + 2 * h + 2 * h = 8h

(i) IND

Cost = H(X$_1$) + H(X$_2$|X$_5$) + 2 * H(X$_3$|X$_4$) +
3 * H(X$_4$) + 2 * H(X$_5$)
= h + 0 + 2 * 0 + 3 * h + 2 * h = 6h

(ii) CLSTR

Cost = H(X$_1$) + H(X$_2$|X$_1$) + 2 * H(X$_3$|X$_1$X$_2$X$_5$) +
2 * H(X$_4$|X$_1$X$_2$X$_3$X$_5$) + 2 * H(X$_5$|X$_1$X$_2$)
= h + 0 + 2 * 0 + 2 * 0 + 2 * 0 = h

(iii) DSC

Fig. 3. Illustration of three prior approaches to data commpression for a 5-node network (CLSTR uses 3 clusters $\{X_1\}, \{X_2, X_5\}, \{X_3, X_4\}$). If spatial correlations are perfect, total communication costs (using the *bit-hop* metric) for IND and CLSTR can be very high compared to the theoretical optimal DSC.

well. Of the $5h$ difference between CLSTR and DSC in Figure 3, $3h$ can be attributed to Approximation Loss.

Formally, let p denote the joint pdf that captures all spatial correlations in the network, and let q denote an approximation to p that captures a subset of those correlations. Let $DSC(p)$ denote the cost incurred by DSC when compressing according to the pdf p. Then the Approximation Loss for a data collection scheme that only exploits the correlations in q is given by: $DSC(q) - DSC(p)$.

Intra-source Communication: If two or more nodes are compressed jointly to exploit the spatial correlation, then the data from these nodes must be gathered at a single location. For the example shown in Figure 3, CLSTR communicates X_4 to X_3 to compress them jointly. We call this Intra-source Communication cost (the remaining $2h$ difference between CLSTR and DSC in Figure 3 can be attributed to intra-source communication).

By increasing the expressive power of the model used and thus capturing larger subsets of spatial correlations (for example, by increasing the cluster sizes), we can reduce the Approximation Loss, but the increase in the Intra-source Communication cost will typically outweigh the benefits (e.g. in Ken [4], the optimal cluster sizes were found to be < 4).

2.4 Decomposable Models and Junction Trees

In this paper, we propose using a subclass of *undirected probabilistic graphical models* [10], called *decomposable models* [8], to capture the spatial correlations and to perform data compression in a sensor network. Decomposable models capture and exploit the *conditional independences* in the data to compactly represent joint pdfs over a large number of variables. Two random variables X_1 and X_2 are conditionally independent of each other given X_3 iff:

$$p(X_1, X_2|X_3) = p(X_1|X_3)p(X_2|X_3)$$

Even though any two sensor nodes in a sensor network may be highly correlated with each other in isolation, given the values of other nodes in the network, many of these correlations almost entirely disappear. For instance, in an environmental monitoring application, a sensor node is typically independent of its non-neighbors given the values of its neighbors. Hence, by using an appropriate decomposable model to approximate the joint pdf, we can exploit most of the spatial correlations in a typical sensor network while keeping the Intra-source Communication cost low. As we will see in the next section, these models can also naturally utilize the broadcast nature of communication to further reduce the Intra-source Communication cost. Next, we provide a brief introduction to the class of decomposable models.

Given a set of variables, $\mathcal{X}$, a decomposable model, denoted $\mathcal{M}$, uses a graph, $\mathcal{G}_{\mathcal{M}}$, over $\mathcal{X}$ to encode the conditional independences among the variables. More precisely, a decomposable model satisfies the global Markov property with respect to $\mathcal{G}_{\mathcal{M}}$ [28]:

If two node sets A and B are separated by a third node set C, i.e., if removing the nodes in C and all the edges attached to the nodes in C results in the node sets A and B getting disconnected, then A and B are conditionally independent given C.

Further, the graph $\mathcal{G}_M$ must be *decomposable* (also called *chordal* or *triangulated*): every cycle of length greater than 3 must posses a *chord* – an edge joining two non-consecutive vertices of the cycle. Figure 4 shows two examples of decomposable graphs over 5 nodes. In the first graph, removing X_1 will separate the remaining vertices from each other; thus, we can say that X_2, X_3, X_4, X_5 are all conditionally independent of each other given X_1. In the second graph, if the edge (X_1, X_5) were missing, then it would not be chordal (since the 4-cycle $(X_1, X_2, X_5, X_4, X_1)$ would have no chord).
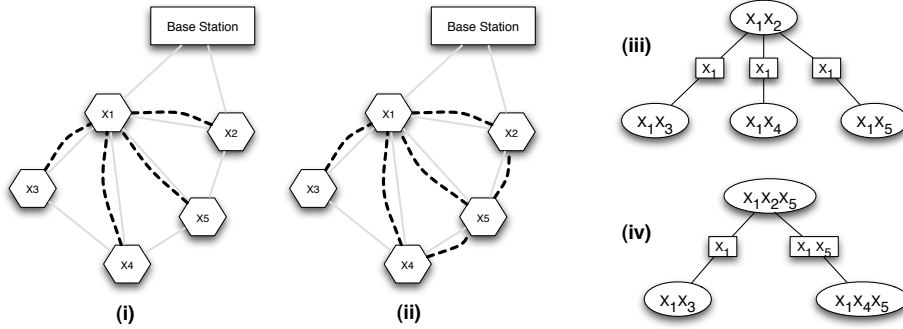
Fig. 4. (i, ii) Two example decomposable graphs superimposed on the communication network. Solid lines are network edges; dashed lines are model edges. (iii, iv) Junction trees for the two models rooted at cliques $X_1 X_2$ and $X_1 X_2 X_5$, respectively.

A compact and particularly useful representation of decomposable graphs is provided by *junction trees* (also known as *clique trees*) [3,16]. Briefly, given a decomposable graph $\mathcal{G}_{\mathcal{M}}$, a rooted junction tree $J_C(\mathcal{G}_{\mathcal{M}})$ is a tree whose vertex set consists of the *maximal cliques* of $\mathcal{G}_{\mathcal{M}}$, and whose root is the clique C. The edges in a junction tree are required to satisfy the following *clique-intersection property*:

For every pair C_i and C_j of cliques in $\mathcal{G}_\mathcal{M}$, the set $C_i \cap C_j$ is contained in every clique on the path connecting C_i and C_j in $J(\mathcal{G}_\mathcal{M})$.

We denote by $\mathcal{C}$ the set of all maximal cliques of $\mathcal{G}_\mathcal{M}$. Without loss of generality, we will assume that C_1 denotes the root of the junction tree. For a clique C, let $parent(C)$ denote the parent of the C, and let $S_C = C \cap parent(C)$ be the *separator* between the node and its parent (it is easy to see that S_C separates the vertex sets $parent(C) \setminus S_C$ and $C \setminus S_C$.). We denote by $\mathcal{S}$ the set of all separators. Although junction trees are not unique, all junction trees of a decomposable graph have the same set of separators.

Figure 4 shows one junction tree each for the two example decomposable graphs.

Approximating a Joint PDF Using a Decomposable Model: A decomposable graph, $\mathcal{G}_\mathcal{M}$, can be used to approximate a joint probability distribution, $p(X_1, \ldots, X_n)$, as follows. For a set of variables, $C \subset \mathcal{X}$, let $p(C)$ denote the the marginal probability distribution over the variables in C (computed by summing over the remaining variables in $\mathcal{X}$). Let $q_{\mathcal{G}_\mathcal{M}}(X_1, \ldots, X_n)$ be the probability distribution computed using the decomposable graph as follows:

$$q_{\mathcal{G}_\mathcal{M}}(X_1, \ldots, X_n) = \frac{\Pi_{C \in \mathcal{C}} p(C)}{\Pi_{S \in \mathcal{S}} p(S)} = \Pi_{C \in \mathcal{C}} p((C - S_C)|S_C) \qquad \text{(Equation(1))}$$

For example, for the junction tree shown in Figure 4 (iii), we get that:

$$q_1(X_1, \ldots, X_5) = p(X_1 X_2) p(X_3|X_1) p(X_4|X_1) p(X_5|X_1) \qquad \text{(Equation(2))}$$

We note that existence of such a closed form expression is perhaps the biggest advantage of using a decomposable graph over an arbitrary graph. Further, for any clique $C \in \mathcal{C}$, it is easy to see that q satisfies the following property: $q_{\mathcal{G}_\mathcal{M}}(C) = p(C)$. In other words, $q_{\mathcal{G}_\mathcal{M}}$ and p agree on the marginal distributions over the maximal cliques of $\mathcal{G}_\mathcal{M}$.

If the approximation quality was the sole concern, we would like to use a decomposable model that minimizes the relative entropy between $q_{\mathcal{G}_\mathcal{M}}$ and p, given by: $D(p||q_{\mathcal{G}_\mathcal{M}}) = (\Sigma_{C \in \mathcal{C}} H(C) - \Sigma_{S \in \mathcal{S}} H(S)) - H(\mathcal{X})$. This is also the commonly used metric in probabilistic modeling [8], and further will result in low Approximation Loss. However, as we will see in next section, when using such a model for data compression, we also need to be cognizant of the communication topology.

3 Using Decomposable Models for Data Collection in WSNs

A decomposable model typically captures a subset of the correlations present in the sensor network. In this section, we first consider the problem of designing data collection protocols for optimally exploiting those correlations for a given decomposable model. We then address the problem of choosing a decomposable model for a given sensor network.

3.1 Data Collection Using a Decomposable Model

Example. We begin with the example decomposable model shown in Figure 4 (i) and the corresponding junction tree (Figure 4 (iii)). For this model, Equation (2) provides us with the way to fully exploit the captured correlations as follows:
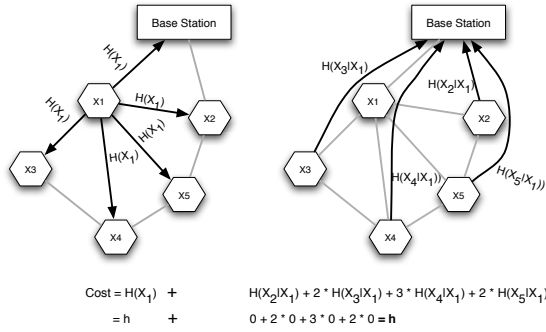
- For each of the cliques in the model, $\{X_1, X_2\}$, $\{X_1, X_3\}$, $\{X_1, X_4\}$, $\{X_1, X_5\}$, gather the values of the attributes in the clique at some sensor node (this can be different for each clique).
- Use the marginal probability distribution $p(X_1 X_2)$ to jointly compress X_1 and X_2 (the clique at the root), and send them to the sink.
- Let the observed value of X_1 be x_1. Use the distribution $p(X_3|X_1 = x_1)$ to compress and transmit the observed value of X_3. Since $X_1 = x_1$ is already known to the sink, it can decompress using the appropriate distribution.
- Similarly, use the distributions $p(X_4|X_1 = x_1)$ and $p(X_5|X_1 = x_1)$ to transmit the values of X_4 and X_5 respectively.

The total number of bits received by the sink can be shown to be exactly:
$$H(p) + D(p||q_1) = H(X_1 X_2) + H(X_3|X_1) + H(X_4|X_1) + H(X_5|X_1)$$
However, to be able to compute the total communication cost incurred during this protocol, we need to "place" the cliques at the sensor nodes (ie., decide which sensor nodes

Cost = $H(X_1)$ + $H(X_2|X_1) + 2 * H(X_3|X_1) + 3 * H(X_4|X_1) + 2 * H(X_5|X_1)$

= h   + $0 + 2 * 0 + 3 * 0 + 2 * 0 = $ **h**

Fig. 5. Data compression using the example decomposable model from Figure 4 (i)

to collect the data for each clique at). Figure 5 shows an example placement that optimally exploits the broadcast nature of the communication. In this case, we place the clique $(X_a X_1)$ at the sensor node X_a for $a \in \{2...5\}$. With one broadcast from node X_1 (at a cost of $H(X_1)$), the value of $X_1 = x_1$ will be known to each of the remaining nodes (including the sink).

The total communication cost in the second step after this broadcast is given by:
$$H(X_2|X_1) + 2H(X_3|X_1) + 2H(X_4|X_1) + 2H(X_5|X_1)$$

For the case of perfect correlations considered in Figure 3, the total cost for this model can be seen to be h as well (same as DSC). However, the cost would be higher if the receiving costs were non-zero (whereas the cost for DSC would remain h).

Given an arbitrary decomposable model $\mathcal{M}$ and a rooted junction tree for it, the data collection is done as follows:

- Place the cliques of $\mathcal{M}$ on to the sensor network nodes (Section 3.2).
- For each node X_i, let D_{X_i} denote the sensor nodes which have been assigned a clique that contains X_i.
- Find a *broadcast tree* to communicate X_i to the nodes in D_{X_i} (we use a *breadth-first search* algorithm for this in our implementation).
- At the sensor node that has been assigned a clique C, compress the values of the sensor nodes in $(C - S_C)$ according to the distribution $p((C - S_C)|S_C)$.

3.2   Clique Placement Algorithms

We first present an optimal algorithm for the case when the decomposable model graph ($\mathcal{G}_M$) is a subgraph of the communication graph ($\mathcal{G}_C$). We then consider the more general case, and show that it is NP-Hard. We then present an efficient heuristic that we use in our experimental study.

CASE: $\mathcal{G}_M$ is a subgraph of $\mathcal{G}_C$. In most sensor networks, geographically co-located sensors tend to exhibit stronger spatial correlations than sensors that are far away from each other. As a result, in many cases, the decomposable model graph may only contain the edges between neighboring sensor nodes. We present an optimal algorithm to solve this case below.

Consider a clique C_i in $\mathcal{M}$. Since we must gather together all sensor nodes in C_i at one location, we can either (1) transmit some $|C_i| - 1$ of these nodes to the remaining node, or (2) transmit all of them to another node not in C_i (combined $|C_i| + 1$ alternatives). However, all the nodes in C_i are within communication radius of each other (since the decomposable model graph is a subgraph of the communication graph). Hence, each of the sensor nodes whose value needs to transmitted only needs to broadcast its value once. In other words, multi-hop transmissions are not needed to get the values in C_i together at one location. Thus, we only need to make binary decisions for each node (whether to broadcast, or not)[3]. Given these decisions, the placement of cliques follows (assuming sufficient nodes broadcast their values).

Our dynamic programming-based algorithm uses the following observation: once the broadcast decisions for the nodes in a separator S_i are made, the decisions for the nodes in the subtree below S_i can be made independently of the decisions for the remaining nodes in the graph. Algorithm 1 shows the pseudo-code for the main recursive procedure. Briefly, the algorithm starts at the root of the junction tree C_1 ($ComputeOptimalCost(C_1, \phi)$), and tries each of the $|C_1| + 1$ alternatives, recursing down the junction tree for each of the alternatives. It is easy to see that the algorithm runs in time $O(n^3)$.

If the receiving costs are non-zero, then the number of different possible decisions for a separator S_i is $O(|S_i|2^{|S_i|})$ (since we not only have to decide which of the nodes in S_i will broadcast, but we also must decide which of the nodes in S_i will receive those values). Overall the complexity of the algorithm increases to $O(n^3 2^s)$, where s denotes the maximum separator size. Although it is exponential in the worst case, in practice, we expect the value of s to be quite small (< 3), and hence this algorithm is quite feasible even in that case.

CASE: $\mathcal{G}_M$ is not a subgraph of $\mathcal{G}_C$

Theorem 3.1. *The general case of the clique placement problem is NP-Hard.*

Proof Sketch: We reduce a variant of the *minimum connected dominating set* problem to the clique placement problem. Given a graph $G = (V, E)$ and a set of nodes $S \subset V$, we construct a clique placement problem as follows. The communication graph over

---

[3] Note that we assume here that only the transmission costs are counted, and that the cost of receiving a message is zero.

Algorithm 1. Procedure ComputeOptimalCost(C_i, bc)

Input: C_i: A clique in $\mathcal{M}$; $bc[X_j] = true$ if $X_j \in S_{C_i}$ is broadcast

Let $key = (C_i, bc)$;

if key exists in cache **return** $cached\ cost$;

Let $D_1, \ldots, D_k$ denote the children of C_i;

if *there exists $X \in S_{C_i}$ such that $bc[X] = false$* **then**

    /* All nodes in $C_i - S_i$ must be broadcast */

    $c = \Sigma_{Y \in C_i - S_i} H(Y) + H(C_i|S_i) \times d(X, sink)$;

    for $j = 1, \ldots, k$ **do**

        Construct a bit-vector bc_j of size $|S_{D_j}|$ and set all entries to $true$;

        if $X \in D_j$ **then** set $bc_j[X] = false$;

        $c = c + ComputeOptimalCost(D_j, bc_j)$;

    end

    Insert $\langle key, cost \rangle$ into cache;

    return c;

else

    /* We must try all possible placements for C_i. */

    for $X \in C_i - S_{C_i}$ **do**

        Let c_X denote the total cost assuming all nodes in C_i except X are broadcast;

        Compute c_X as above;

    end

    Compute c_{all} = the cost assuming all nodes in C_i are broadcast (C_i may be placed at a node $\notin C_i$);

    Insert $\langle key, \min(\min_X(c_X), c_{all}) \rangle$ into cache;

    return c_{min}

end

---

the sensor network is set to be G. For a node $X \notin S$, we set $H(X) = 0$. For $X \in S$, we set $H(X) = c$ for some constant c, and for each pair $(X, Y), X \in S, Y \in S$, we set $H(X|Y) = 0$. In other words, all nodes in S are perfectly correlated with each other. Further, we choose a node $A \in S$, and use a decomposable model with cliques $(A, X), X \neq A, X \in S$, and further choose an arbitrary junction tree for this model. It is easy to see that, for any junction tree, the optimal solution involves broadcasting A to all the other nodes in S. The problem of constructing the broadcast tree is identical to the problem of computing the *Steiner connected dominating set* for S, a problem that is known to be NP-Hard [11]. $\qquad\square$

We next present an efficient greedy heuristic that we use for solving the clique placement problem (Algorithm 2). Intuitively, Algorithm 2 starts off by placing all cliques as close to the sink as possible. Then, starting with the node closest to the sink, it makes local, cost-based decisions about whether to broadcast the value of each node away from the sink, into the sensor network (in effect, moving the cliques away from the sink).

Example. In our running example (Figure 4 (i) and (iii)), the four cliques $\{X_1, X_2\}$, $\{X_1, X_3\}, \{X_1, X_4\}, \{X_1, X_5\}$ would initially be placed at node X_1. The algorithm then checks if it would be beneficial to broadcast X_1 instead, which would result in placement of cliques as shown in Figure 5. After making the decision for X_1 (which is not changed afterwards), the algorithm then checks to see if X_2 should be set to broadcast and so on.

Algorithm 2. Heuristic Clique Placement Algorithm

Input: A decomposable model $\mathcal{M}$; a rooted junction tree of $\mathcal{M}$
Output: An assignment of cliques to the nodes in $\mathcal{G}_C$
Let bc denote current broadcast decisions ($bc[X] = true \implies X$ is broadcast);
Initialize $bc[X] = unknown$ for all nodes;
for $C \in \mathcal{M}$ **do**
    if $\exists X \in C$ such that $bc[X] \neq true$ **then**
        Let $X_r \in C$ be the node closest to the sink such that $bc[X_r] \neq true$;
    else
        Let $X_r \in C$ be the node farthest away from sink;
    Place C at X_r;
end
Let c denote the cost of the above clique placement;
Let X_i be the ith closest node to the sink;
for $i = 1, \ldots, n$ **do**
    Set $bc[X_i] = true$ and re-assign each clique currently placed at X_i as above;
    Let c_i be the new total cost;
    if $c_i < c$ **then** set $c = c_i$; **else** set $bc[X_i] = false$;
end

3.3   Choosing a Decomposable Model

The problem of finding an optimal decomposable model for a given data sample to minimize an error metric such as *Chi-squared error*, is known to be intractable [10], and heuristic algorithms are typically used for this purpose [8]. Although our metric (which accounts for the communication topology) is quite different from the Chi-squared error metric, we adapt a similar heuristic search procedure in our system. More specifically, we use a *forward stepwise selection* [8] algorithm to find an appropriate decomposable model. The algorithm starts with an *empty* decomposable model, i.e., a model with no edges. It then incrementally adds eligible edges in the order of their benefits until there is no improvement in the total expected communication cost. (An edge is said to be eligible if the model remains decomposable after adding it.) Algorithm 2 is used as a subroutine for evaluating the total expected communication cost of the model after adding a candidate edge in the incremental step.

To make the search problem tractable, we observe that disconnected components of the decomposable model do not influence the placement or junction tree decisions of each other. Hence, when a new edge is added, only the costs of the connected components that are affected by the addition need to be re-evaluated using Algorithm 2; a connected component is affected if the new candidate edge is incident on a vertex (or two vertices) in the component. We also memoize (cache) the total costs of all connected components encountered during search, as computed by Algorithm 2. Employing these two optimizations results in significantly reduced total execution time for the selection process. Due to space constraints, we omit a more detailed description of the algorithm.

4 Experiments

We conducted a comprehensive experimental study over several synthetic and real-world sensor network datasets. In this section, we present the results of that study.

Data Sets: Our first synthetic dataset (SYNTH-1, a 30-node network) is generated using a multivariate Gaussian distribution; each variable follows the standard normal distribution (with variance 1), and the covariance between attributes X_i and X_j is set to be a function of the distance between them, $c^{d(X_i, X_j)}$ (where c ($0 \leq c \leq 1$) denotes the correlation strength). The sensor nodes are placed randomly in a 20x20 square and have an average hop count of 8.5 to the sink (placed at (0, 0)).

For the second and third synthetic data sets (SYNTH-2 and SYNTH-3 , two 72-node networks), we use an analytical expression for computing the entropy for a precipitation data model (presented and used by Pattem et al. [22] in their study). The network topologies are generated by placing the nodes randomly within a 66x66 square and a 3x24 rectangle respectively, with average hop counts of 6.5 and 13.5 to the sink.

The first real-world data set, *Lab* [18], contains traces from 49 sensors deployed in the Intel Research Lab at Berkeley. The data contains roughly 23 days of recordings on light, humidity, temperature and voltage. We use the temperature readings between 9pm to 3am for our experiments. The data from first 15 days is used for training (for constructing the model and the pdfs), and the data from next 8 days is used for testing. Our second real-world data set, *Precipitation*, contains precipitation data in the states of Washington and Oregon collected during 1949-1994 [29]. Fifty stations are randomly selected from the deployment. We discretize the observed values into three categories: *light rain, medium rain*, and *heavy rain*. The initial two thirds of the data is selected as the training set, and the remaining data is used for testing.

Comparison Systems:
We compare the following data collection methods.

- NAIVE: No compression is done while collecting the data.
- IND (Section 2.2): Each node compresses its data independently of the others.
- CLSTR (Section 2.2): The clusters are chosen using the greedy algorithm presented in Chu et al. [4].
- KEN [4]: This is similar to above, except that no compression is performed while collecting the data for each cluster at the cluster-head (this will always perform worse than CLSTR).
- DECOMP: An appropriate decomposable model is chosen using the algorithms presented in Section 3, and is used for data collection.
- DSC: Where applicable, the theoretical lower bound is plotted (Section 2.2).

Methodology: We investigate the performance under different data and network characteristics including correlation, error threshold ϵ (for SYNTH-1 and Lab), network topology, and the sensor receiving costs. Unless otherwise mentioned, we set $\epsilon = 0.5$. To avoid model over-fitting, we limit the clique/cluster size $S \leq 3$ for DECOMP, KEN, and CLSTR. For the synthetic datasets, we restrict the models learned by DECOMP to be subgraphs of the communication network since the spatial correlations are strongest for neighboring nodes. We remove this restriction for the real-world data sets. For the
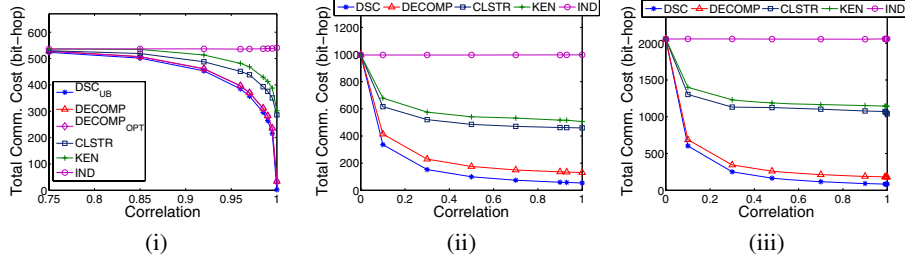
Fig. 6. Total data collection costs for (i) SYNTH-1, (ii) SYNTH-2, and (iii) SYNTH-3

synthetic datasets, we use the joint-entropy based data collection – for SYNTH-1, we estimate the entropy using the training dataset, whereas for SYNTH-2 and SYNTH-3, we use the analytical expressions presented in Pattem et al. [22]. For real-world datasets, we use suppression-based data collection.

Results: Synthetic Datasets. Figure 6 compares the effectiveness of the different schemes at reducing the total transmission cost on the three synthetic datasets for varying correlation characteristics. We also plot an estimate of the cost of DSC; for SYNTH-1, we can only compute an upper bound since accurate estimation of entropy over large sets of variables is not feasible. For SYNTH-1, we plot two graphs for DECOMP, one where we use the optimal clique placement algorithm and the other using the heuristic algorithm (Section 3.2). There is however almost no difference in the total transmission cost, and the two graphs overlap entirely. We use the heuristic algorithm in the rest of the section as it is much more efficient than the optimal algorithm.

Several facts become clear from these figures. At low correlations, the techniques perform fairly similarly; the intra-source communication cost outweighs the benefits of joint compression, and hence all techniques degenerate to IND. As the correlation strength increases, the total costs of the techniques that exploit the correlations decrease rapidly, with DECOMP performing much better than CLSTR or KEN. In fact, the total cost for DECOMP is very close to that of DSC; not only is the Approximation Loss of DECOMP very low, but, because it exploits broadcast communication, the Intra-source communication cost of DECOMP is also very low. We again note that, if the receiving costs are factored in, the performance of DECOMP is noticeably worse than DSC, although it is still much superior to CLSTR or KEN (see below).

Another interesting aspect is how network topologies affect the qualitative behaviors of the schemes. Comparing Figure 6(iii) with Figure 6(ii), we see that it is more expensive to transmit data in the deep network (Figure 6(iii)) since the average hop count is larger. Forming spatial cliques for doing in-network compressions becomes more attractive in such a network; if correlations are ignored, the Approximation Loss in such networks can be very high.

Figure 7(i) presents the effects of varying the user-defined error threshold ϵ (for the SYNTH-1 dataset, with correlation $c = 0.9$). As expected, the total cost decreases when ϵ is increased for all techniques. We note that the performance of DECOMP remains close to the upper bound on DSC for a wide range of error thresholds.
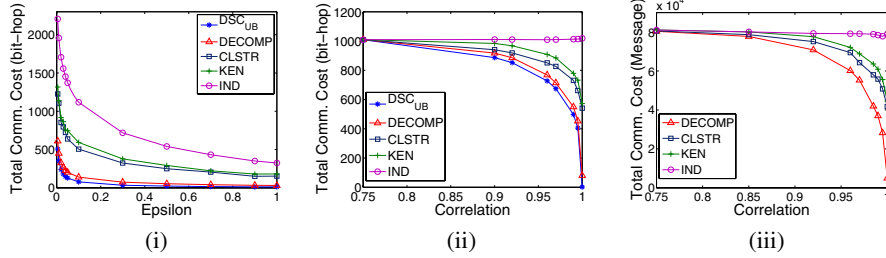
Fig. 7. (i) Effects of ϵ on the total cost for SYNTH-1, (ii) Impact of receiving cost for SYNTH-1, (iii) Total message-based transmission cost for SYNTH-1
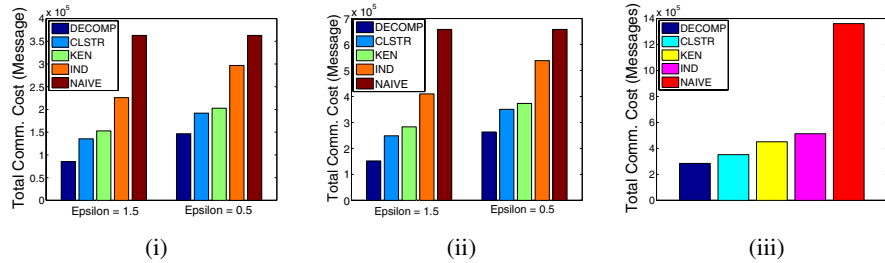
Fig. 8. The total communication costs for: (i) Lab dataset (over $\approx$ 1300 runs), (ii) Lab dataset including receiving costs, (iii) Precipitation dataset (over 5600 runs)

We next present the results from an experiment where the receiving cost was set to be the same as the transmission cost (as is common in many deployments). Figure 7(ii) shows the total communication cost with the receiving cost included for SYNTH-1. Similarly to Figure 6(i), both DECOMP and the clustering based methods stay at their upper bound, IND, till $c = 0.75$[4]. As we can see, the relative performance of DE-COMP, IND and CLSTR remains essentially unchanged; however, the relative cost of DECOMP increases slightly compared to DSC. This is because DECOMP exploits broadcast communication, which gets penalized when receiving costs are non-zero.

Finally, Figure 7(iii) presents the total cost in terms of the message-based metric for 500 test tuples in SYNTH-1. Messages, instead of bits, are used to quantify the communication costs. We note that the graph shows similar trends as the entropy-based metric (Figure 6(i)), with DECOMP resulting in much fewer total messages transmitted than the alternatives.

Results: Real-World Datasets. Figure 8(i) present the results for Lab's test traces when receiving costs are not considered. The results are for $\epsilon = 0.5$ and $\epsilon = 1.5$. DECOMP achieves the best performance in both cases. A small value of ϵ (i.e. $\epsilon = 0.5$) results in higher entropy and the total cost of all techniques increases. The increase in ϵ

---

[4] Although $c = 0.75$ might seem large, we note that the 30-node network resides in a 20×20 square, resulting in large pairwise node distances.

results in sharp drops in communication costs: DECOMP achieves a 41% drop in total cost, CLSTR has a 29% drop, and KEN has a 25% drop. More subtle, a small value of ϵ introduces higher variances in the quantized data, and hence the correlations across sensors are weaker. As a result, the relative performance of all modeling-based techniques, relative to IND, is slightly worse for small values of ϵ.

Introducing receiving costs (Figure 8(ii)) results in a higher total communication cost for all schemes. DECOMP continues to outperform the other methods, and the amount of performance differences for all modeling-based methods with respect to NAIVE stays relatively unchanged.

Finally, Figure 8(iii) plots the results of exact data collection for the precipitation data (i.e. $\epsilon = 0$). The spatial correlations in this data are not high, but as Figure 8(iii) shows, the modeling-based approaches significantly outperform Naive, and DECOMP achieves the lowest total communication cost among all four modeling-based approaches.

5   Related Work

Wireless sensor networks have been a very active area of research in recent years (see [1] for a survey). Due to space constraints, we only discuss some of the most closely related work on data collection in sensor networks here. Directed diffusion [15], Cougar [32], TAG [19], TinyDB [20], LEACH [14] are few of the general purpose data collection mechanisms that have been proposed in the literature. The focus of that work has been on designing protocols and/or declarative interfaces to collect data, and not on optimizing continuous data collection. Aside from the work by Pattem et al. [22] and Chu et al. [4], the BBQ system [9] also uses a predictive modeling-based approach to collect data from a sensor network. However, the BBQ system only provides probabilistic, approximates answers to queries, without any guarantees on the correctness. Scaglione and Servetto [24] also consider the interdependence of routing and data compression, but the problem they focus on (getting all data to all nodes) is different from the problem we address. Cristescu et al. [6] consider the problem of finding a near-optimal tree-based communication structure to minimize the total transmission cost; their approach is similar to routing driven compression (RDC) [24,22] and may require repeated compression and decompression over large numbers of data sources at the sensor nodes, which may make it unsuitable for resource-constrained sensor networks. In a seminal work, Gupta and Kumar [13] proved that the transport capacity of a random wireless network scales only as $O(\sqrt{n})$, where n is the number of sensor nodes. Although this seriously limits the scalability of sensor networks in some domains, in the kinds of applications we are looking at, the *bandwidth* or the *rate* is rarely the limiting factor; to be able to last a long time, the sensor nodes are typically almost always in sleep mode.

Several approaches not based on predictive modeling have also been proposed for data collection in sensor networks or distributed environments. Kotidis [17] and Gupta et al. [12] consider approaches based on using a representative set of sensor nodes to approximate the data distribution over the entire network. Constraint chaining [25] is a suppression-based exact data collection approach that monitors a minimal set of node and edge constraints to ensure correct recovery of the values at the base station.

More recently, Cormode et al. [5] have proposed a similar approach of using replicated predictive models to solve the problem of maintaining accurate quantile summaries over distributed data sources.

6   Conclusions

In this paper, we presented an approach that uses a subclass of undirected graphical models called *decomposable* models for continuous sensor data collection with accuracy guarantees. Compared to previous predictive modeling-based approaches, our approach is more effective at exploiting the spatial correlations in the data, and thus reducing the total communication cost incurred during the process. Our proposed approach also naturally exploits the broadcast nature of communication in sensor networks. An extensive experimental study using both synthetic and real-world data sets demonstrates the effectiveness of our approach.

There are several directions of future work that we are planning to pursue. We are developing more efficient algorithms that can scale to very large sensor networks, and that can efficiently exploit both spatial and temporal correlations. So far we have assumed that the sensor nodes do not fail; extending our protocols to function correctly in presence of such faults remains a challenge. Finally, although our approach performs very well compared to the lower bound provided by DSC, understanding the fundamental reasons behind the gap between the two and how we can bridge that gap remains an open question.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38 (2002)
2. Arici, T., Gedik, B., Altunbasak, Y., Liu, L.: PINCO: A pipelined in-network compression scheme for data collection in wireless sensor networks. In: IEEE Intl. Conf. on Computer Communications and Networks (2003)
3. Jean, R.S.: Blair and Barry Peyton. An Introduction to Chordal Graphs and Clique Trees. In: Graph Theory and Sparse Matrix Computation, pp. 1–29. Springer, New York (1993)
4. Chu, D., Deshpande, A., Hellerstein, J., Hong, W.: Approximate data collection in sensor networks using probabilistic models. In: ICDE. Proceedings of the International Conference on Data Engineering (2006)
5. Cormode, G., Garofalakis, M., Muthukrishnan, S., Rastogi, R.: Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In: SIGMOD (2005)
6. Cristescu, R., Beferull-Lozano, B., Vetterli, M., Wattenhofer, R.: Network correlated data gathering with explicit communication: Np-completeness and algorithms. IEEE/ACM Transactions on Networking 14(1), 41–54 (2006)
7. Cristescu, R., Beferull-Lozano, B., Vetterli, M.: Networked slepian-wolf: Theory and algorithms. In: EWSN, pp. 44–59 (2004)
8. Deshpande, A., Garofalakis, M., Jordan, M.: Efficient stepwise selection in decomposable models. In: UAI (2001)
9. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., Hong, W.: Model-driven data acquisition in sensor networks. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) Databases, Information Systems, and Peer-to-Peer Computing. LNCS, vol. 2944, Springer, Heidelberg (2004)

10. Edwards, D.: Introduction to Graphical Modeling. Springer, New York (1995)
11. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica 20(4), 374–387 (1998)
12. Gupta, H., Navda, V., Das, S., Chowdhary, V.: Efficient gathering of correlated data in sensor networks. In: MobiHoc (2005)
13. Gupta, P., Kumar, P.R.: The capacity of wireless networks. IEEE Transactions on Information Theory 46, 388–404 (2000)
14. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: HICSS 2000: Proceedings of the 33rd Hawaii International Conference on System Sciences, vol. 8, p. 8020 (2000)
15. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: ACM MobiCOM (2000)
16. Jensen, F.V., Jensen, F.: Optimal Junction Trees. In: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence, Seattle, Washington (July 1994)
17. Kotidis, Y.: Snapshot queries: Towards data-centric sensor networks. In: ICDE (2005)
18. Madden, S.: Intel lab data (2003),
    `http://db.csail.mit.edu/labdata/labdata.html`
19. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: A tiny aggregation service for ad-hoc sensor networks. In: OSDI (2002)
20. Madden, S., Hong, W., Hellerstein, J., Franklin, M.: TinyDB web page,
    `http://telegraph.cs.berkeley.edu/tinydb`
21. Olston, C., Loo, B., Widom, J.: Adaptive precision setting for cached approximate values. In: SIGMOD (2001)
22. Pattem, S., Krishnamachari, B., Govindan, R.: The impact of spatial correlation on routing with compression in wireless sensor networks. In: IPSN (2004)
23. Pradhan, S., Ramchandran, K.: Distributed source coding using syndromes (DISCUS): Design and construction. IEEE Trans. Information Theory (2003)
24. Scaglione, A., Servetto, S.: On the interdependence of routing and data compression in multi-hop sensor networks. In: Mobicom (2002)
25. Silberstein, A., Braynard, R., Yang, J.: Constraint-chaining: On energy-efficient continuous monitoring in sensor networks. In: SIGMOD (2006)
26. Slepian, D., Wolf, J.: Noiseless coding of correlated information sources. IEEE Transactions on Information Theory 19(4) (1973)
27. Su, X.: A combinatorial algorithmic approach to energy efficient information collection in wireless sensor networks. ACM Trans. Sen. Netw. 3(1), 6 (2007)
28. Whittaker, J.: Graphical Models in Applied Multivariate Statistics. Probability and Mathematical Statistics. Wiley, Chichester (1990)
29. Widmann, M., Bretherton, C.: 50 km resolution daily precipitation for the pacific northwest (2003), `http://www.jisao.washington.edu/data_sets/widmann`
30. Wyner, A.D., Ziv, J.: The rate-distortion function for source coding with side information at the decoder. IEEE Transactions on Information Theory (1976)
31. Xiong, Z., Liveris, A.D., Cheng, S.: Distributed source coding for sensor networks. IEEE Signal Processing Magazine 21, 80–94 (2004)
32. Yao, Y., Gehrke, J.: Query processing in sensor networks. In: CIDR (2003)