Compressed Data Aggregation: Energy Efficient and High Fidelity Data Collection

Liu Xiang, *Student Member, IEEE,* Jun Luo, *Member, IEEE,* and Catherine Rosenberg, *Fellow, IEEE*

***Abstract*—We focus on *wireless sensor networks* (WSNs) that perform data collection with the objective of obtaining the whole data set at the sink (as oppose to a function of the data set). In this case, energy efficient data collection requires the use of data aggregation. Whereas many data aggregation schemes have been investigated, they either compromise the fidelity of the recovered data or require complicated in-network compressions. In this paper, we propose a novel data aggregation scheme that exploits *compressed sensing* (CS) to achieve both recovery fidelity and energy efficiency in WSNs with arbitrary topology. We make use of *diffusion wavelets* to find a sparse basis that characterizes the spatial (and temporal) correlations well on arbitrary WSNs, which enables straightforward CS-based data aggregation as well as high fidelity data recovery at the sink. Based on this scheme, we investigate the *minimum energy compressed data aggregation* problem. We first prove its NP-completeness, and then propose a mixed integer programming formulation along with a greedy heuristic to solve it. We evaluate our scheme by extensive simulations on both real datasets and synthetic datasets. We demonstrate that our compressed data aggregation scheme is capable of delivering data to the sink with high fidelity while achieving significant energy saving.**

***Index Terms*—Wireless sensor networks (WSNs), data collection, data aggregation, compressed sensing (CS), diffusion wavelets, energy efficiency**

I. Introduction

Energy efficiency of data collection is one of the dominating issues of *wireless sensor networks* (WSNs). It has been tackled from various aspects since the outset of WSNs. This includes, among others, energy conserving sleep scheduling (e.g., [35], [20]), topology control (e.g., [26], [21]), mobile data collectors (e.g., [28], [39], [29]), and data aggregation[1] (e.g., [23]). While the first three approaches (and many others) focus on the energy efficiency of protocols, data aggregation directly aims at significantly reducing the amount of data to be transported, and it hence complements other approaches.

Although data aggregation techniques have been heavily investigated, there are still imperfections to be improved on. Existing WSN applications fall into two classes depending on the information that is needed at the sink. The first class corresponds to cases where only simple function values of the data are needed at the sink (e.g., MAX/AVG/MED); we call this class "functional". In that case, the aggregation functions only extract certain statistical quantities from the collected data [31] and the original data are thus not fully recoverable. The second class of applications refers to cases where the sink wants to obtain the full data set. We call this class "recoverable". The recoverability can be achieved by applying distributed source coding technique, such as Slepian-Wolf coding [34], [23], to perform non-collaborative data compression at the sources. However, it is not exactly practical due to the lack of prior knowledge on the spatial data correlation structure. Collaborative in-network compression makes it possible to discover the data correlation structure through information exchange [14], [19] but it either requires a simple correlation structure [14], or often results in high communication load [19] that may potentially offset the benefit of recoverable aggregation technique.

As a newly developed signal processing technique, *compressed sensing* (CS) promises to deliver a full recovery of signals with high probability from far fewer samples than their original dimension, as long as the signals are *sparse* or *compressible* in some domain [8]. Although this technique may suggest a way of reducing the volume of the data to transmit over WSNs without the need for adapting to the data correlation structure [22], the complexity of the interaction between data routing and CS-based aggregation has postponed the development on this front until very recently [27], [25]. Moreover, as these recent studies rely on conventional signal/image processing techniques (e.g., DCT, wavelets) to sparsify the collected data, they require regular WSN deployments (e.g., grids), which make them less practical.

In this paper, we propose a new data aggregation technique derived from CS, and we aim at **minimizing** the total energy consumption when collecting data from nodes of an **arbitrarily** deployed WSN for an application requiring full data recovery. We make use of *diffusion wavelets*; this facilitates a high fidelity recovery of data sensed by nodes in an arbitrary network by allowing both spatial and temporal correlations to be naturally taken into account. To the best of our knowledge, we are the first to address the problem of CS-based data aggregation and recovery in **arbitrary** networks. More specifically, we are making the following contributions:

- We propose a scheme that exploits compressed sensing for data aggregation.
- We employ diffusion wavelets to design the sparse basis for data recovery. This technique delivers high fidelity recovery for data collected in arbitrary WSNs by naturally taking into account spatial (and temporal) correlations.

Preliminary results were presented in the Proceedings of the 8th IEEE SECON, 2011 [38]. This work is supported in part by AcRF Tier 1 Grant RG 32/09, the Start-up Grant of NTU, and AcRF Tier 2 Grant ARC15/11.

L. Xiang and J. Luo are with the School of Computer Engineering, Nanyang Technological University, 639798 Singapore. E-mail: xiangliu@pmail.ntu.edu.sg; junluo@ntu.edu.sg.

C. Rosenberg is with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Email: cath@ece.uwaterloo.ca.

[1]We define *data aggregation* in a general sense. It refers to any transformation that summarizes or compresses the data acquired and received by a certain node and hence reduces the volume of the data to be sent out.

- We show that, by choosing a proper sparse basis, we can partition a WSN into subnetworks and apply compressed data aggregation to these subnetworks individually. Compared with non-partitioning case, this significantly improves energy efficiency without sacrificing the fidelity of data recovery.
- We study, for our CS-based aggregation scheme, the minimum energy data gathering problem. We first provide the characterization of the optimal solutions and prove the hardness of the problem. Then both optimal and heuristic approaches are presented to solve the problem.
- We demonstrate the superiority of our compressed data aggregation scheme through experiments on extensive datasets: compared with non-aggregation, our proposal achieves significant energy saving while delivering data to the end with high fidelity.

In the remainder of this paper, we first introduce the framework for our compressed data aggregation scheme in Sec. II. Then we explore the applicability of CS on arbitrary networks in Sec. III, by designing proper sparse basis with the help of diffusion wavelets. In Sec. IV, we formulate the minimum energy compressed data aggregation problem and address it by both optimization and heuristic approaches. We evaluate the recovery fidelity and energy efficiency in Sec. V, before concluding our work in Sec. VII.

II. Compressed Data Aggregation

We introduce our novel *compressed data aggregation* (CDA) scheme in this section. After preparing readers with necessary background on compressed sensing, we define the network model and then present the CDA framework.

A. Compressed Sensing

Given a set of data that can be sparsely represented in a proper basis, CS theory asserts that one can recover the data set from far fewer samples than its original dimension [8]. Suppose an n*-dimensional* signal $\mathbf{u} \in \mathbb{R}^n$ has an m*-sparse representation* under a proper basis Ψ, i.e., $\mathbf{u} = \Psi\mathbf{w}$, where $\|\mathbf{w}\|_{\ell_0} = m \ll n$ (i.e., the count of the non-zero elements in $\mathbf{w}$ is m). The CS coding process computes a compressed *k-dimensional* **coded** vector $\mathbf{v} = \Phi\mathbf{u}$, where Φ is a $k \times n$ "sensing" matrix[2] whose row vectors are largely incoherent with Ψ [8]. Then the CS decoding process reconstructs the signal as $\hat{\mathbf{u}} = \Psi\hat{\mathbf{w}}$, where $\hat{\mathbf{w}}$ is the optimal solution of the following convex optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^n} \|\mathbf{w}\|_{\ell_1} (= \sum_i |w_i|) \qquad \text{subject to} \quad \mathbf{v} = \Phi\Psi\mathbf{w}. \quad (1)$$

Note that, as the decoding probability grows with n in power law [7], CS works better for data sets of large size.

The practical performance of CS coding depends on the sparsity of the signal, as well as the decoding algorithm. The basis that enables the sparsest representation depends on the particular structure or features of the original signal $\mathbf{u}$. Conventional compression basis (e.g., Fourier, DCT) can only deal with data sampled in vector form (i.e., aligned in line) or in matrix form (i.e., aligned in lattices). However, thanks to diffusion wavelets, we are able to find the sparse representation for data sampled on arbitrary manifolds or graphs. We will discuss this issue in detail in Sec. III. On the reconstruction process front, algorithms have been proposed based on either standard interior-point methods (e.g., ℓ_1_MAGIC [6]) or greedy pursuit schema (e.g., CoSaMP [32]).

As opposed to the existing distributed source coding techniques (e.g., Slepian-Wolf coding [34]), compressed sensing is truly model-free: no prior knowledge about the underlying correlation structure is required for the encoding process. In fact, only lightweight computations (simple multiplications and summations) are incurred during the encoding; the actual computational load is shifted to the decoding end. As we will show in Sec. II-C, CS coding allows for a fully distributed way to compress the network data traffic. Due to these properties, CS becomes extremely suitable for data aggregation in resource scarce WSNs.

B. Network Model

We represent a WSN by a connected graph $G(V, E)$, where the vertex set V corresponds to the nodes in the network, and the edge set E corresponds to the wireless links between nodes (so we use "edge" and "link" interchangeably hereafter). Let n and ℓ be the cardinalities of V and E, respectively. There is a special node $s \in V$ known as the sink that collects data from the whole network. As we aim at monitoring **widespread events**, data from all sensor nodes are demanded. Assume each of the n nodes acquires a sample u_i, the ultimate goal for a data aggregation scheme in WSN is to gather sufficient information to recover the n-dimensional signal (i.e., sensor readings) $\mathbf{u} = [u_1, \cdots, u_n]^T$ at the sink while minimizing some other criteria (e.g., volume of traffic, energy). Note that, each sample is represented as a 32-bit **floating-point** number in TinyOS [5] and it is encapsulated in a packet.

Let $\mathbf{c} : E \to \mathbb{R}_0^+$ be a cost assignment on E, with $c_{ij} : (i,j) \in E$ being the energy expense of sending one unit of data across link (i,j). Assuming identical rate R and bandwidth W for all links, we can show that c_{ij} is proportional to the path loss on link (i,j),[3] hence is a function of the link length. Also let $\mathbf{x} : E \to \mathbb{R}_0^+$ be a load allocation on E, with $x_{ij} : (i,j) \in E$ being the data traffic load imposed by a certain routing/aggregation scheme on link (i,j). We assume that all nodes are loosely time synchronized and the data collection proceeds in rounds. At the beginning of each round, every sensor node produces one unit of data (one sample), and the sink acquires all information at the end of this round. We also assume that there are no packet losses in our paper.

C. Overview of CDA Framework

Without data aggregation, each node needs to send its sample to the sink and one process that can be optimized

[2]In theory, the sensing matrix Φ should satisfy the *restricted isometry principle* (RIP). Both Gaussian random matrices and Bernoulli random matrices are considered as good candidates for Φ.

[3]This follows from the Shannon's Theorem that suggests an identical signal-to-noise ratio at all receiving ends (given R and W), and from the common assumption that path loss is the only term in channel gain.

is the routing. In any case, nodes around the sink will carry heavy traffic as they are supposed to relay the data from the downstream nodes. Directly applying CS to data collection suggests a way to alleviate this bottleneck. Suppose k is predefined and known by the whole network, and each node i is aware of its own k-dimensional coding vector ϕ_i. To illustrate the so called *plain CS aggregation* [30] (initially introduced in [22], [27]), we rewrite the CS coding as

$$\mathbf{v} = \Phi\mathbf{u} = u_1\phi_1 + \cdots + u_n\phi_n.$$

Now the idea becomes clear: each node i first codes its sample into a k-dimensional vector using ϕ_i, then this coded vector $\mathbf{v}$ rather than the raw data is transmitted (i.e., k, rather than 1, packets are sent). The aggregation is done by summing the coded vectors whenever they meet. As floating-point notation allows calculations over a wide range of magnitudes, we consider 32 bits are enough to represent any coded element and hence the packet size will not change during aggregation. We can represent the traffic load (in terms of the number of packets) by the dimension of a data vector, as in [27]. Therefore, the traffic load on any link is always k. Eventually, the sink collects the aggregated k-dimensional vector rather than n-dimensional raw data, then the decoding algorithm is used to recover the n raw samples. In the later exposition, we term k, the dimension of the coded vector $\mathbf{v}$ (i.e., the number of aggregated packets) as the *aggregation factor*. It is worth noting that the encoding process is actually done in a distributed fashion on each individual node, by simply performing some multiplications and summations whose computational cost can be negligibly small. The actual computational load is shifted to the decoding end where the energy consumption is not a concern.

However, directly applying CS coding on every sensor node might not be the best choice. As shown in Fig. 1, suppose $n-1$ nodes are each sending one sample to the n-th node, the outgoing link of that node will carry n samples if no aggregation is performed. If we apply the plain CS aggregation, it will force every link to carry k samples, leading to unnecessary higher traffic at the early stages. Therefore, the

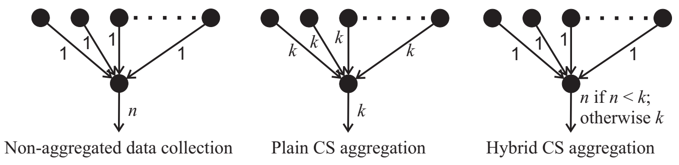


Fig. 1. Comparison of different data collection mechanisms. The link labels correspond to the carried traffic.

proper way of applying CS is to start CS coding only when the outgoing samples will become no less than k, otherwise, raw data transmission is used. We coined this scheme as *hybrid CS aggregation* in our previous work [30]. Obviously, hybrid CS aggregation marries the merits of non-aggregation and plain CS aggregation: it reduces the total traffic load while preserving the integrity of the original data. Therefore, we adopt hybrid CS aggregation as the basis of our CDA framework. Note that two types of traffic are imposed by the CDA framework, namely the *encoded traffic* and the *raw traffic*; they can be differentiated by a flag in a data packet.

In a practical implementation, given a fixed aggregation factor k, a basis ϕ and a routing, a node i waits to receive from all its downstream neighbors[4] all the data they have to send. Only upon receiving more than $k-1$ raw samples or any encoded samples, node i will start encoding and aggregating by creating a vector $u_j\phi_j$ for every uncoded sample u_j it receives (including u_i). These vectors, along with the already coded samples (if any), are combined into one vector through a summation. Finally, node i will send out exactly k encoded samples corresponding to the coded vector.

According to the description above, we need an implicit synchronization in generating Φ. As depicted in Fig. 1, the CS coding for some samples may be carried out at other nodes instead of the sources, so the i-th column of Φ has to be the same wherever it is generated. We achieve this goal by associating a specific pseudo-random number generator (a publicly known algorithm and its seed) with a node i: it indeed meets the i.i.d. criterion among matrix entries, while avoiding any explicit synchronization among nodes.

Two main problems under the CDA framework are

- P1 How to recover the data collected from WSNs with arbitrary network topologies using proper sparse basis?
- P2 What is the routing **scheme** that minimizes the total energy consumption?

In the following, we first handle P1 by introducing customized diffusion wavelets basis in Sec. III. Then, to address P2, we define and tackle the *minimum energy compressed data aggregation* (MECDA) problem in Sec. IV.

III. High Fidelity Data Recovery

As existing CS-based aggregation techniques make use of conventional signal/image compression basis (e.g., DCT basis), regular WSN deployments (e.g., grids) are required. However, nodes are usually randomly deployed rather than being aligned in lattices, in order to avoid high deployment cost. To cope with arbitrary network topologies, we employ diffusion wavelets as the sparse basis for the collected data. In particular, we customize the basis such that it allows us to i) partition a WSN into subnetworks for improving energy efficiency (Sec. III-C) and ii) jointly exploit spatial and temporal correlations among data (Sec. III-D).

A. Diffusion Wavelets

Generalizing the classical wavelets, *diffusion wavelets* [11] enables multiscale analysis on general structures such as manifolds or graphs. As opposed to dilating a "mother wavelet" by powers of two to generate a set of classic wavelet bases, the dyadic dilation generating diffusion wavelets relies on a *diffusion operator*. Here diffusion is used as a smoothing and scaling tool to enable coarse grained and multiscale analysis.

Let us take an arbitrary graph G as an example to illustrate the idea. Suppose the weighted adjacency matrix of G is $\Omega = [\omega_{ij}]$, where ω_{ij} is the weight of edge (i,j). Let $\Lambda = [\lambda_{ij}]$ be the *normalized Laplacian* of G, i.e., $\lambda_{ij} = 1$ if $i = j$;

[4]Node i's downstream neighbors are the nodes that have been specified by the routing scheme to forward their data to the sink through i.

otherwise $\lambda_{ij} = -\frac{\omega_{ij}}{\sqrt{\sum_p \omega_{ip} \sum_p \omega_{pj}}}$. It is well known that Λ characterizes the degree of correlations between function values taken at vertices of the graph G [10]. Roughly speaking, each eigenvalue (and the corresponding eigenvector) represents the correlation under a certain scale. In order to decompose the signal sampled on a graph in a multiscale manner, one may consider partitioning the range space of Λ. The idea behind diffusion wavelets is to construct a *diffusion operator* O from Λ, such that they share the same eigenvectors whereas all eigenvalues of O are smaller than 1. Consequently, recursively raising O to power 2 and applying a fixed threshold to remove the diminishing eigenvalues (hence the corresponding eigenvectors and the subspaces spanned by them) leads to a dilation of the null space but a shrinkage of the range space; this naturally produces space splitting.

More specifically, O^{2^j} is computed at the j-th scale, eigenvalue decomposition is derived for it, and the resulting eigenvectors form a basis that (qualitatively) represents the correlation over neighborhood of radius 2^j hops on the graph. Denote the original range space of O by $U_0 = \mathbb{R}^n$, it is split recursively: at the j-th level, U_{j-1} is split into two orthogonal subspaces: the scaling subspace U_j that is the range space of O^{2^j}, and the wavelet subspace V_j as the difference between U_j and U_{j-1}. Given a specified decomposition level γ, the diffusion wavelet basis Ψ is the concatenation of the orthonormal bases of $V_1, \ldots, V_\gamma$ (wavelet functions) and U_γ (scaling function), sorted in descending order according to their corresponding frequencies.

Interested readers are referred to [11] for detailed exposition. We want to point out that different diffusion operators lead to different wavelets bases, therefore the art of our later design lies in the proper choice of an operator.

B. Basis for Spatial Correlation

As $\mathbf{u}$ is sampled by a WSN with nodes arbitrarily deployed, the basis Ψ has to accommodate this irregularity. This makes diffusion wavelets basis an ideal choice. According to Sec. III-A, diffusion wavelets are generated by diffusion operator O, which stems from the weighted adjacency matrix $\Omega = [\omega_{ij}]$. As ω_{ij} represents the correlation between the data sampled at nodes i and j, we propose to make ω_{ij} a function of distance for representing the spatial correlation.

Let d_{ij} be the Euclidean distance between nodes i and j. Assuming these distances are known, we define

$$\omega_{ij} = \begin{cases} d_{ij}^{\alpha} & i \neq j, \\ \beta & \text{otherwise}, \end{cases} \tag{2}$$

where $\alpha < 0$ and β is a small positive number. As a result, the normalized Laplacian becomes

$$\lambda_{ij} = \begin{cases} 1 - \frac{\beta}{\sum_p d_{ip}^{\alpha}} & i = j, \\ -\frac{d_{ij}^{\alpha}}{\sqrt{\sum_p d_{ip}^{\alpha} \sum_p d_{pj}^{\alpha}}} & \text{otherwise}. \end{cases} \tag{3}$$

Here the constant β is used to tune the spectrum of the graph G, hence the structure of diffusion wavelets.

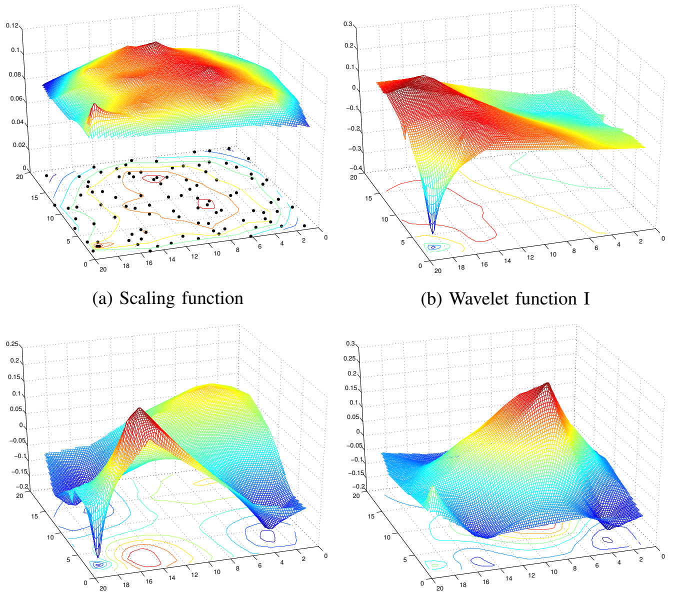
In a practical monitoring process, we initially collect data without aggregation. Using these data as the ground truth, we can estimate α and β in (2) through trial-and-error. Then both estimated parameters are used for later aggregation.

Proposition 1: The eigenvalues of Λ lie in $[0, 2]$, and the maximum eigenvalue $\sigma_{\max}(\Lambda)$ is a decreasing function in β. The proof is omitted; it follows from Lemma 1.7 in [10].

Based on this proposition, two straightforward choices of the diffusion operator O are (I is the identity matrix):

$$O = I - \Lambda \quad \text{or} \quad O = \Lambda/2;$$

both have their eigenvalues ranged from 0 to 1. Therefore, keeping raising a dyadic power to O will make all the eigenvalues diminish eventually. So we partition the range space of O and group the eigenvectors to form the basis, by thresholding on the diminishing eigenvalues of O^{2^j}. Based on the above construction procedure, we generate the diffusion wavelets for a WSN with 100 nodes and illustrate some of them in Fig. 2.



(a) Scaling function (b) Wavelet function I

(c) Wavelet function II (d) Wavelet function III

Fig. 2. Diffusion wavelets for a 100-node WSN, with $O = I - \Lambda$, $\alpha = -1$ and $\beta = 1$. (b)-(d) are three examples of the low frequency wavelets. The node distribution is plotted in (a). Although these wavelets are discrete, we present them as interpolated surfaces to facilitate visual illustration.

C. Joint Recovery with Network Partition

For large-scale networks, the aggregation factor k needs to be sufficiently large in order to allow for a high fidelity data recovery at the sink. This may substantially limit the energy efficiency brought by our CDA framework. Fortunately, we could expect the sparsity property to still hold for localized subsets of the network. To improve energy efficiency while still maintaining recovery fidelity, we seek to partition the WSN (excluding the sink) into a set $\mathcal{V}$ of **disjoint** localized subnetworks $\{V_i\}_i$, and the sink gathers information from these subnetworks individually. We have $\bigcup_{V_i \in \mathcal{V}} V_i = V \backslash \{s\}$ and $V_i \bigcap V_j = \emptyset, i \neq j$. For example, we can geometrically partition the region occupied by a WSN into multiple areas that are centered at the sink s. Then, V_i is the set of sensors that belong to area i. Denote by n_i and k_i the number of nodes and the aggregation factor for the subnetwork G_i. As the aggregation factor is an increasing function of the network size, we would have $k_i < k$ as the CS coding is performed on smaller datasets ($n_i < n$). Although the sink still collects $k = \sum_{G_i \in \mathcal{G}} k_i$ coded samples, the aggregated traffic in each subnetwork is reduced to only k_i, which significantly lowers the total energy consumption.

The network partition causes the sensing matrix Φ to have a block-diagonal shape:

$$\Phi = \begin{bmatrix} \Phi_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Phi_2 & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Phi_{|\mathcal{G}|} \end{bmatrix},$$

where Φ_i is a $k_i \times n_i$ matrix with entries as specified in Sec. II-A. Now the question is whether a CS coding with Φ may still deliver high fidelity recovery at the decoding end. Fortunately, we have the following result:

Proposition 2: Let $\mathcal{U}$ be the set of m-sparse vectors that share the same sparse index set $\mathcal{I}$ of basis vectors, i.e., $\mathbf{u}_i = \sum_{j\in\mathcal{I}} w_j(\mathbf{u}_i)\boldsymbol{\psi}_j$, $\forall \mathbf{u}_i \in \mathcal{U}$. If the coherence [8] between Φ and a subset of Ψ's columns indexed by $\mathcal{I}$ is low, the recovery of any $\mathbf{u} \in \mathcal{U}$ by solving Eqn. (1) is exact with high probability.

The proof is straightforward given [7]. We will show in Sec. V-A1 that data mostly share the same sparse index set of diffusion wavelets: the set containing the "low frequency" components of the basis. Empirically, these components often have their "energy" spread across all coordinates (see Fig. 2). As a result, if Ψ is incoherent with a full sensing matrix and $|\mathcal{G}|$ is not too large, so is it with a block-diagonal Φ. Finally, the conditions stated in *Proposition 2* does not constrain how a WSN should be partitioned, so we have the freedom to choose a partition that favors energy efficiency, for example, a balanced partition that minimizes $\max_i k_i$. Also thanks to this freedom, we can focus on one subnet in the remaining discussions of this paper.

D. Joint Spatial and Temporal Recovery

As the diffusion wavelets basis stems from a graph that represents the data correlation, we can extend the basis defined for spatial correlations to include temporal correlations as well. The idea is that, for a given time period indexed by $\mathcal{R}$, we replicate the graph G $|\mathcal{R}|$ times and index them by $r \in \mathcal{R}$. Within each graph G^r, the weighted adjacency matrix is still Ω in (2). Between two graphs G^{r_1} and G^{r_2}, the weight between node i in G^{r_1} and node j in G^{r_2} is given by $\omega_{ij}^{r_1 r_2} = \omega_{ij}/g(|r_1 - r_2|)$, where $g(\cdot)$ is an increasing function. This extended adjacency matrix is given by

$$\tilde{\Omega} = \begin{bmatrix} \Omega & \Omega I^{-1}_{g(|r_1-r_2|)} & \cdots & \Omega I^{-1}_{g(|r_1-r_{|\mathcal{R}|}|)} \\ \Omega I^{-1}_{g(|r_2-r_1|)} & \Omega & \ddots & \Omega I^{-1}_{g(|r_2-r_{|\mathcal{R}|}|)} \\ \vdots & \ddots & \ddots & \vdots \\ \Omega I^{-1}_{g(|r_{|\mathcal{R}|}-r_1|)} & \Omega I^{-1}_{g(|r_{|\mathcal{R}|}-r_2|)} & \cdots & \Omega \end{bmatrix},$$

where $I^{-1}_{g(|r_1-r_2|)}$ is a diagonal matrix with $g^{-1}(|r_1 - r_2|)$ on its diagonal. The temporal correlation represented by $\tilde{\Omega}$ can be fine-tuned by choosing $g(\cdot)$ appropriately: the larger the first-order derivative of $g(\cdot)$, the faster the temporal correlation attenuates. Based on this extension, we can derive the diffusion operator and hence diffusion wavelets, following exactly the same procedure as presented in Sec. III-B. Note that, $g(\cdot)$ is again estimated through trial-and-error as in Sec. III-B.

The benefits of involving spatial and temporal correlations together are twofold. Firstly, for large-scale WSNs with hundreds or thousands of nodes, the number of samples k to be acquired (by the sink) for each round could be reduced while still achieving the same level of recovery fidelity. Secondly, for small-scale WSNs with only tens of nodes, CS often fails to deliver satisfactory recovery fidelity for individual rounds, which stems from the asymptotic nature of the CS theory. Nevertheless, with a small set of samples in space but a large one in time, we could still expect promising results for CDA if we compensate the small size in space by the large size in time. This intuition will be confirmed in Sec. V-A.

IV. Minimum Energy Compressed Data Aggregation (MECDA)

In this section, we begin with the definition of the MECDA problem. Based on our findings on the structure of the optimal solutions, we propose a mixed integer programming formulation along with a greedy heuristic. Another two algorithms are also introduced so as to benchmark our greedy heuristic.

A. Problem Statement and Hardness Analysis

Recall that c_{ij} is the cost of transmitting a unit of data over link (i,j), and x_{ij} is the traffic load over (i,j) (Sec. II-B). Our problem, termed *minimum energy compressed data aggregation* (MECDA), is to minimize the total cost (or energy consumption) $\sum_{(i,j)\in E} c_{ij}x_{ij}$, when applying CDA to collect data at the sink. If a WSN is partitioned into subnetworks to improve energy efficiency (see Sec. III-C), the optimization is performed over individual subnets.

To solve this problem, we need to jointly allocate the raw/coded traffic and find the optimal routing that minimizes the energy consumption defined above. We will first show that we can restrict the search for an optimal routing to trees. This is due in part to the objective function (i.e., we are only interested in the energy consumption).

Proposition 3: There exists an optimal solution of MECDA involving a routing tree rooted at the sink.

Proof: It is obvious that, if CS is not applied, a shortest path tree rooted at the sink connecting all sources is always an optimal strategy for energy efficiency. Suppose the nodes performing CS coding are given by an oracle when applying CS aggregation. Then in order to minimize the energy cost, raw traffics originated from the rest nodes should be destined to the nearest coding nodes. Therefore, shortest path trees rooted at coding nodes compose (part of) an optimal solution. Moreover, if a coding node a splits its coded vector and sends the resulting vectors to different next-hop neighbors on the way towards the sink, the CS coding cannot be properly performed and hence additional flows are introduced into the routing path from a towards the sink, contradicting our objective of minimizing the energy consumption. Therefore, an optimal routing for MECDA can be achieved by restricting data traffic (raw and coded) on a tree rooted at the sink. ■

Consequently, without loss of generality, we restrict the data aggregation on a directed **tree** rooted at the sink. The nodes that route their data to the sink through node i are called the *descendants* of i. We now formally specify the abstract model for our CDA scheme.

Definition 1 (CDA): Given a certain routing tree T, the outgoing traffic from node i is

$$x_{ij:(i,j)\in E(T)} = \min\left\{ \sum_{j:(j,i)\in E(T)} x_{ji} + 1, \quad k \right\}.$$

This restricts that the out-going flow of a node is always upper bounded by k. We call a node that performs CS coding as an *aggregator* and otherwise a *forwarder* hereafter.

In fact, *Definition 1* exhibits a special aggregation function, which is nonlinear and heavily dependent on the routing strategy. Therefore, our MECDA problem aims at finding a tree and allocating the traffic load $\mathbf{x}$ properly in a given round, through **joint** routing and aggregator assignment, such that the total energy consumption is minimized. By investigating the interactions between CS aggregation and routing tree structure, we draw some interesting observations in the following.

1) Characterization of the Optimal Solution: Each optimal solution of MECDA provides an optimal *configuration*, in terms of routing paths and aggregator assignment. Here are several conditions that characterize an optimal configuration.

Proposition 4: In an optimal configuration, we have the following properties.

1) The network is partitioned into two complementary sets: *aggregator set* A ($s \in A$) and *forwarder set* F, i.e., $A \cap F = \emptyset$ and $A \cup F = V$.
2) The routing topology for nodes in A is a minimum spanning tree (MST) of the subgraph induced by A.
3) For every node $i \in F$, the routing path to A is via some node $\hat{j} \in A$ that minimizes the cost of the shortest path, i.e., $\hat{j} = \arg\min_{j\in A}(\mathrm{SP}_{ij})$.
4) Each member of F has less than $k-1$ descendants, whereas each leaf of the MST induced by A and rooted at the sink has no less than $k-1$ descendants in F.

The proof is postponed to Appendix A to maintain fluency; we just illustrate an optimal configuration by Fig. 3. In fact, the above conditions assert that the optimal routing tree consists of a "core" – an MST for A, as well as a "shell" – a set of shortest path trees (SPTs) that connect F to A. As the cost minimization within each set is trivial, the difficulty in solving the problem should lie in the partition of V into A and F. Note that, the plain CS aggregation is a special case of CDA, where we have $A = V$ and $F = \emptyset$ and the optimal solution is trivial: an MST of G. As a result, we focus on investigating the general CDA hereafter.

2) Hardness of the Problem: Based on our characterization in Sec. IV-A1, MECDA does not appear to admit a straightforward solution of polynomial time complexity, given its graph partitioning nature. We hereby analyze the problem complexity. Our results establish the NP-hardness of MECDA, through a nontrivial reduction from the *maximum leaf spanning tree* (MLST) problem [16]. As a byproduct, we also obtain the inapproximability of MECDA.

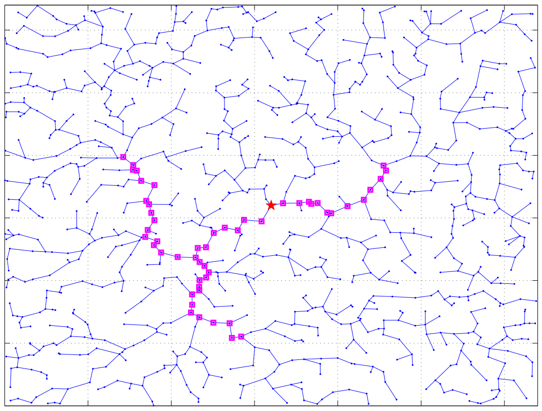


Fig. 3. An optimal CDA tree in a 1024-node WSN, with $k = 150$. The sink is represented by the pentagram and the aggregators are marked as squares; the rest are forwarders. The original graph G is a complete graph, with the edge cost being a function of the distance between its two ends. We only plot the tree edges to avoid confusion.

We first introduce the decision version of MECDA, termed CS Aggregation Tree Cost Problem (CSATCP).

INSTANCE: A graph $G = (V, E)$, a cost assignment $\mathbf{c}: E \to \mathbb{R}_0^+$, an aggregation factor (integer) k, and a positive number B.

QUESTION: Is there a CS aggregation tree such that the total cost is less than B?

We denote by CSATCP^k the problem instance with a specific value for the aggregation factor k. Since we need MLST in the later proof, we also cite it from [16].

INSTANCE: A graph $G = (V, E)$, a positive integer $K \le |V|$.

QUESTION: Is there a spanning tree for G in which K or more vertices have degree 1?

We first show that, for two specific values of k, CSATCP^k can be solved in polynomial time.

Proposition 5: CSATCP^1 and CSATCP^{n-1} are both P-problems.

Proof: For $k = 1$, every node belongs to A and sends out only one sample. Therefore, a (polynomial time) minimum spanning tree oracle would answer the question properly. For $k = n-1$ (or any larger values), every node apart from s can be put into to F and no CS coding is performed. Therefore, a (polynomial time) shortest path tree oracle would answer the question properly. ■

The proof also suggests that the traditional min-energy functional aggregation and non-aggregation are both special cases of MECDA. Unfortunately, other parameterized versions of CSATCP (hence MECDA) are intractable.

Proposition 6: CSATCP^k, with $2 \le k < n-1$, are all NP-complete problems.

We again postpone the proof to the Appendices (Appendix B), where we construct a reduction from MLST to CSATCP^k.

As a byproduct of this proof, we are also able to state the inapproximability of MECDA as an optimization problem.

Corollary 1: MECDA does not admit any *polynomial time approximation scheme* (PTAS).
The proof is sketched in Appendix B; it follows directly from the MAX SNP-completeness of the MLST (optimization) problem [15] and our proof to *Proposition 6*.

B. Solution Techniques

In this section, we first formulate MECDA as a mixed integer programming (MIP) problem, which gives the exact solution to MECDA. Concerning the NP-hardness of MECDA, we also present a greedy heuristic that efficiently finds a near-optimal solution. This is followed by two other algorithms with proven approximation ratios for benchmarking.

1) An Optimization Formulation: Essentially, we formulate the MECDA problem as a special minimum-cost flow problem. The main difference between CDA and non-aggregation is the flow conservation: CDA does not conserve flow at the aggregators. Therefore, we need to extend the flow conservation constraint for every node $i \in V \backslash \{s\}$:

$$\begin{aligned} \textstyle\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} + (n-k)y_i &\geq 1 \\ \textstyle\sum_{j:(i,j)\in E} x_{ij} - (k-1)y_i &\geq 1 \end{aligned}$$

where $y_i = 1$ if node i is an aggregator; otherwise $y_i = 0$. If i is a forwarder, the first constraint is just the conventional flow conservation, and the second constraint trivially holds. While if i is an aggregator, the second constraint states the outgoing traffic is always k. Then the first constraint becomes trivial since we have $n - \sum_{j:(j,i)\in E} x_{ji}$ as a lower bound on the LHS if we plug in the second constraint.

It is not sufficient to have only the extended flow conservation constraints, as the extension may allow loops. According to *Proposition 4*, we need to constrain that the set of links carrying positive flows form a spanning tree. Let $z_{ij} \in \{0,1\}$ be an indicator of whether a link is a tree edge and $\bar{\mathbf{x}} \geq 0$ be a virtual link flow assignment, we first propose an alternative formulation of a spanning tree.[5]

$$\begin{aligned} \sum_{j:(i,j)\in E} \bar{x}_{ij} - \sum_{j:(j,i)\in E} \bar{x}_{ji} - \frac{1}{n-1} &\geq 0 \quad \forall i \in V\backslash\{s\} \\ z_{ij} - \bar{x}_{ij} &\geq 0 \quad \forall (i,j) \in E \\ \textstyle\sum_{(i,j)\in E} z_{ij} - (n-1) &= 0 \end{aligned}$$

The proof for the correctness of this formulation is omitted; it follows directly from the fact that a connected subgraph whose vertex set is V and whose edge set has a cardinality $n-1$ is a spanning tree of G. Indeed, the first constraint asserts that the subgraph is connected: there is a positive (virtual) flow between every node i and s. The other two constraints confine the cardinality of the edge set involved in the subgraph. Note that the virtual flow vector $\bar{\mathbf{x}}$ is used only to specify the connectivity, it has nothing to do with the real flow vector $\mathbf{x}$.

[5]Conventional spanning tree formulation (e.g., [12]) involves an exponential number ($2^{|V|}$) of constraints $\sum_{(i,j):i\in S,j\in S} z_{ij} \leq |S| - 1,\ \forall S \subseteq V$. By introducing the virtual link flow assignment, our formulation exhibits a more compact form than the existing one, with $|V| + |E|$ constraints.

The final step is to make the connection between x_{ij} and z_{ij}, such that only edges carrying positive flows are indicated as tree edges. This is achieved by two inequalities applied to every edge $(i,j) \in E$: $x_{ij} - z_{ij} \geq 0$ and$z_{ij} - k^{-1}x_{ij} \geq 0$, which imply that $x_{ij} = 0 \Leftrightarrow z_{ij} = 0$ and $x_{ij} > 0 \Leftrightarrow z_{ij} = 1$. In summary, we have the following extended min-cost flow problem as the MIP formulation of MECDA.

$$\text{minimize} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{4}$$

$$\begin{aligned} \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} + (n-k)y_i &\geq 1 \quad \forall i \in V\backslash\{s\} && (5)\\ \sum_{j:(i,j)\in E} x_{ij} - (k-1)y_i &\geq 1 \quad \forall i \in V\backslash\{s\} && (6)\\ x_{ij} - z_{ij} &\geq 0 \quad \forall (i,j) \in E && (7)\\ z_{ij} - k^{-1}x_{ij} &\geq 0 \quad \forall (i,j) \in E && (8)\\ \sum_{j:(i,j)\in E} \bar{x}_{ij} - \sum_{j:(j,i)\in E} \bar{x}_{ji} - \frac{1}{n-1} &\geq 0 \quad \forall i \in V\backslash\{s\} && (9)\\ z_{ij} - \bar{x}_{ij} &\geq 0 \quad \forall (i,j) \in E && (10)\\ \sum_{(i,j)\in E} z_{ij} - (n-1) &= 0 && (11)\\ x_{ij}, \bar{x}_{ij} \geq 0, \quad z_{ij} &\in \{0,1\} \quad \forall (i,j) \in E && (12)\\ y_i &\in \{0,1\} \quad \forall i \in V\backslash\{s\} && (13) \end{aligned}$$

We will use the optimal solutions obtained by solving this MIP to benchmark the performance of our heuristics in small-scale WSNs in Sec. V-B1.

2) A Greedy Algorithm: As explained in Sec. IV-A1, the difficulty of MECDA lies in partitioning V into A (the "core") and F (the "shell"). One can expect that in general $|A| \ll |F|$ for a practical network topology and aggregation factor. Observing that, we now present a greedy heuristic that is based on the principle of "growing the core". The details are given in **Algorithm 1**.

Algorithm 1: MECDA_GREEDY

```
Input: $G(V,E)$, $s$, $k$
Output: $T$, $A$, cost
1  $A = \{s\}$;  $F = V\backslash\{s\}$;  cost $= \infty$
2  repeat
3      forall the $i \in \mathrm{B}(A)$ do
4          $A_{\text{test}} = A \cup \{i\}$;  $F_{\text{test}} = F\backslash\{i\}$
5          $\{\text{cost}_{\text{MST}}, L\} \leftarrow \mathrm{MST}(A_{\text{test}})$
6          $\{\text{cost}_{\text{SPF}}, \mathbf{t}\} \leftarrow \mathrm{SPF}(F_{\text{test}}, A_{\text{test}})$
7          if $k \cdot \text{cost}_{\text{MST}} + \text{cost}_{\text{SPF}} \leq \text{cost}$ AND
           $\min_{l\in L} t_l \geq k-1$ then
8              cost $= k \cdot \text{cost}_{\text{MST}} + \text{cost}_{\text{SPF}}$
9              $A_{\text{cand}} = A_{\text{test}}$;  $F_{\text{cand}} = F_{\text{test}}$
10         end
11     end
12     $A = A_{\text{cand}}$;  $F = F_{\text{cand}}$
13 until $A$ unchanged;
14 $T = \mathrm{MST}(A) \cup \mathrm{SPF}(F, A)$
15 return $T$, $A$, cost
```

The algorithm maintains two lists, A and F, recording respectively the aggregator set and forwarder set. We term by $\mathrm{MST}(A)$ the MST induced by A, and by $\mathrm{SPF}(F, A)$ the *shortest path forest* (SPF) that connects each $i \in F$ through the shortest path to its nearest aggregator $\hat{j} = \arg\min_{j\in A}(\mathrm{SP}_{ij})$. While the former is readily computed by Prim's algorithm [13], the latter can be obtained by simply performing linear searches on an all-pairs shortest paths table (obtained in advance by Floyd-Warshall algorithm [13]). The outcome of $\mathrm{MST}(A)$ and $\mathrm{SPF}(F, A)$ includes the edge costs in $\mathrm{MST}(A)$ ($\mathsf{cost}_{\mathrm{MST}} = \sum_{(i,j)\in \mathrm{MST}(A)} c_{ij}$), path costs in $\mathrm{SPF}(F, A)$ ($\mathsf{cost}_{\mathrm{SPF}} = \sum_{i\in F} \min_{j\in A} \mathrm{SP}_{ij}$), the leaf node set L in $\mathrm{MST}(A)$, and a vector $\mathbf{t} = \{t_l\}_{l\in A}$ counting the number of descendants for every node in A. The cost incurred by a certain partition A and F is $k{\cdot}\mathsf{cost}_{\mathrm{MST}} + \mathsf{cost}_{\mathrm{SPF}}$.

Starting with a trivial assignment that $A = \{s\}$ and $F = V\backslash\{s\}$, the algorithm proceeds in iterations. We denote by $\mathrm{B}(A) = \{i \in F | \exists j \in A : (i, j) \in E\}$ the neighboring nodes of A. For each round, the algorithm greedily moves one node from $\mathrm{B}(A)$ to A following two criterions: 1) the optimality characterization for A is satisfied, i.e., every leaf node in $\mathrm{MST}(A)$ has no less than $k-1$ descendants, and 2) the action leads to the greatest cost reduction. Consequently, the core keeps growing, leading to more links carrying lighter aggregated traffic instead of heavier raw traffic. So the total energy cost keeps decreasing. The algorithm terminates if no further core expansion is allowed, where the cost reaches a local minimal. Upon termination, the algorithm returns the aggregation tree $T = \mathrm{MST}(A) \cup \mathrm{SPF}(F, A)$, along with the aggregator set A. It is easy to verify that $\{T, A\}$ satisfies all the conditions, in particular 4), stated in *Proposition 4*, because otherwise A can be further expanded and the algorithm would not have terminated.

Proposition 7: **Algorithm 1** has a polynomial time complexity, which is $\mathcal{O}\left((n-k)^2n^2 + n^3\right)$.
We refer to Appendix C for the detailed proof.

Note that in a practical system, data aggregation starts from the leave nodes in T and proceeds towards the sink level by level, which demands a loose synchronization between a node and its children in T. Assuming a practical MAC and a reasonable degree in T, the delay for all descendants to successfully transmit data to their aggregator in T is bounded. Therefore, each aggregator waits for a bounded time period before performing CS coding; this guarantees the level by level processing of CDA.

Note that **Algorithm 1** is also amenable to a distributed implementation. Initially, an all-to-all flooding is deployed to obtain the network-wide information, where $\mathcal{O}(n^2)$ messages are needed. This phase accomplishes the topology discovery; all-pair shortest paths are also computed and maintained locally at each node. Then A expands in rounds: each $i \in \mathrm{B}(A)$ computes the new cost of adding itself into A and competes with other candidates in terms of the reduced cost; the winner leading to the most cost reduction will broadcast globally that it becomes a new aggregator. Each competition is upper bounded by n^2 messages, and at most $n - k$ nodes can be nominated as aggregators. This results in $\mathcal{O}\left(n^2(n-k)\right)$ messages. Once A is determined, the tree is formed naturally since the network topology and the core A are known to all. Therefore, the overall message complexity is $\mathcal{O}(n^2(n-k))$.

3) A Randomized Algorithm: In large-scale WSNs, computing the optimal configuration becomes extremely hard. Therefore, we seek to benchmark our greedy heuristic by a randomized algorithm [18] that admits a provable approximation ratio. Given a graph $G(V, E)$, a set $D \subseteq V$ of demands, and a parameter $M > 1$, the so called *connected facility location* (CFL) aims at finding a set $A \subseteq V$ of facilities to open, such that the connection cost within A and that between D and A is minimized. By setting $D = V$ and $M = k$, MECDA is reduced to CFL, hence an upper bound for CFL also holds for MECDA. The main body of this algorithm is given in **Algorithm 2**. Here ρ is the approximation ratio for the Steiner tree problem given by a specific approximation algorithm. Later we will use Kou's algorithm [24], which is a well-known 2-approximate algorithm for minimum Steiner tree problem.

Algorithm 2: CFL_RANDOM

```
1 Each node $i \in V\backslash\{s\}$ becomes an aggregator with a
  probability $1/k$, denote the aggregator set by $A$ and the
  forwarder set by $F$;
2 Construct a $\rho$-approximate Steiner tree on $A$ that forms
  the CS aggregation tree on the core, denote the cost as
  $\mathsf{cost}_{\mathrm{Steiner}}$;
3 Connect each forwarder $j \in F$ to its closest aggregator in
  $A$ using the shortest path, denote the cost as $\mathsf{cost}_{\mathrm{SPF}}$.
4 $\mathsf{cost} = k \cdot \mathsf{cost}_{\mathrm{Steiner}} + \mathsf{cost}_{\mathrm{SPF}}$.
```

Note that even though **Algorithm 2** is a $(2+\rho)$-approximation for CFL [18] and hence for MECDA, it hardly suggests a meaningful solution for MECDA as it barely improves the network energy efficiency compared with non-aggregation, which we will show in Sec. V-B3. Therefore, we use the randomized algorithm only to benchmark our greedy heuristic rather than solving MECDA.

4) An MST-based Heuristic: Besides the randomized algorithm, our greedy heuristic can be benchmarked by a deterministic one as well. The algorithm starts with an MST on the whole network and rooted at the sink, namely $\mathrm{MST}(V)$. Then a counting procedure is taken place to get the number of descendants ($\mathbf{q}$) for each node. If a node has no less than $k-1$ descendants, it is promoted to be an aggregator; otherwise, it is labelled as a forwarder. The pseudo-code is given in **Algorithm 3**, corresponding to line 2-14 in **Algorithm 1**.

Algorithm 3: MST_PRUNED

```
1 $\{T, \mathbf{q}\} \leftarrow \mathrm{MST}(V)$
2 if $q_i \geq k-1$ then
3     $A = A \cup \{i\}$;   $x_{ij:(i,j)\in E(T)} = k$
4 else
5     $F = F \cup \{i\}$;   $x_{ij:(i,j)\in E(T)} = q_i + 1$
6 end
7 $\mathsf{cost} = \sum_{(i,j)\in E(T)} c_{ij}x_{ij}$
```

Proposition 8: **Algorithm 3** gives a solution to MECDA problem with an approximation ratio of k.

Proof: Suppose the optimal configuration of MECDA induces a routing tree T^*. It is obvious that $\mathsf{cost}(T^*) \geq \mathsf{cost}(\mathrm{MST}(V))$. While running **Algorithm 3**, we find a solution with $\mathsf{cost}(T) \leq k \cdot \mathsf{cost}(\mathrm{MST}(V))$. This gives the approximation ratio $= \mathsf{cost}(T)/\mathsf{cost}(T^*) \leq k$. ■

Due to the same reason explained in Sec. IV-B3, **Algorithm 3** only serves as another benchmark for **Algorithm 1** instead of solving MECDA.

V. Performance Evaluation

In the following, we evaluate the performance of CDA from two aspects. We first validate the performance of using CS coding for data recovery, then we demonstrate the energy efficiency brought by CDA using MECDA.

A. Recovery Fidelity of CDA

We intend to demonstrate the recovery performance of CDA with the use of diffusion wavelets. Note that the decoding procedure takes place at the sink that is capable of solving the convex optimization problem, i.e., Eqn. (1). We use both synthetic dataset as well as real dataset for evaluation.

During our experiments, we have tested different values of α and β in the Laplacian (3) and tried both $O = I - \Lambda$ or $O = \Lambda/2$. According to our observations, $O = I - \Lambda$ performs much better than $O = \Lambda/2$ as the sparse basis, while $\alpha \in [-1, -1/3]$ and $\beta \in [0, 2]$ provide good sparse representation for the sampled data (e.g., Fig. 5). In the following, we fix $\alpha = -1$ and $\beta = 1$, and use $O = I - \Lambda$.

1) Compressibility of Data: In order to show that CS aggregation works well for network-partitioned WSNs, we need to demonstrate that the data $\mathbf{u}$ collected by a WSN only has low frequency components when projected onto the diffusion wavelets basis (see Sec. III-C). Here we use the data generated by the `peaks` function in Matlab, depicted in Fig. 4. In Fig. 5, we plot the diffusion wavelets coefficients for data obtained for WSNs of different sizes. Sorting the coefficients in descending order by their frequencies, it is evident that the data contain mostly low frequency components of the diffusion wavelets basis.

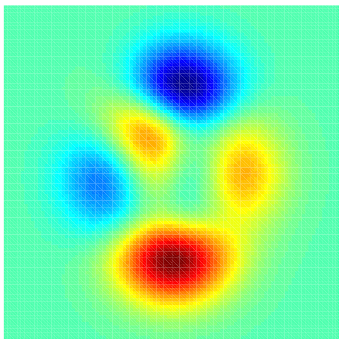


Fig. 4. Visualization of the `peaks`.

2) Spatial Recovery: Fig. 6(a) visualizes a set of relative humidity data obtained from [4]; it serves as the underlying signal under surveillance. Then we arbitrarily deploy WSNs on the field. The direct recovery in Fig. 6(c) represents the ideal case for data compression and recovery where both the data set and the sparse basis are known *à priori* (though it is not practical for WSNs). One can see the CS recovery leads to comparable performance with the direct recovery: the data pattern has been maintained in both cases. Therefore, CS recovery is superior to direct recovery in that it does not rely on the global information or the prior knowledge of the sparse basis to compress the volume of traffic.

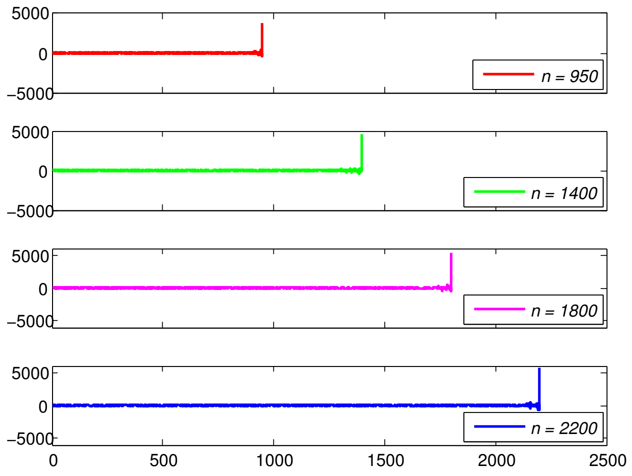


Fig. 5. Diffusion wavelets coefficients for WSNs with different sizes.

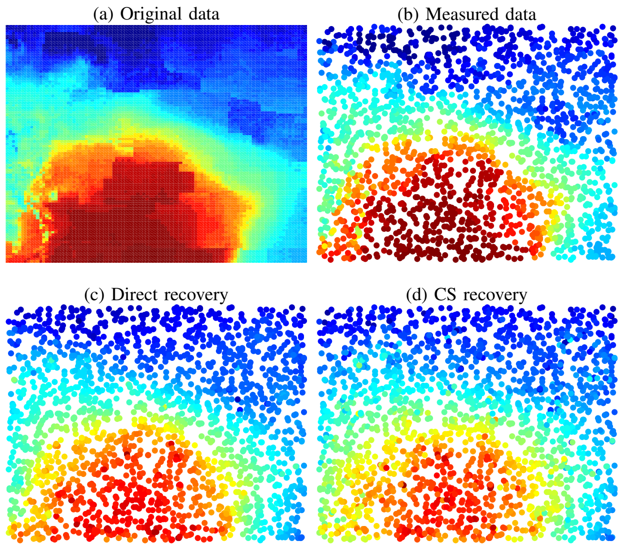


Fig. 6. Visualization of the physical phenomenon, the sampled data, as well as the reconstructed data. A 2048-node arbitrary network is deployed upon this area with its measured data depicted in (b). In (c), we take the 75 largest diffusion wavelet coefficients to directly recover the data. While in (d), the CS recovery is obtained by solving Eqn. (1) with $k = 300$.

To show the recovery fidelity of CDA under network partitions, we plot in Fig. 7 the recovery errors of CS decoding at the sink. We use the abbreviations ST (single tree) and MT (multiple trees) to indicate the cases with and without network partition, respectively. In this comparison, we partitioned the network into four disjoint parts with each one forming an aggregation tree rooted at the sink. The recovery error is defined as $\epsilon = \|\mathbf{u} - \hat{\mathbf{u}}\|_{\ell_2} \cdot \|\mathbf{u}\|_{\ell_2}^{-1}$. The aggregation factor for the single tree case is ranged from 100 to 300. Accordingly, we take the aggregation factor for each subtree in the tree partition cases to be $k_i = k/4$ or $k_i = k/3$. Obviously, as the aggregation factor keeps increasing, the recovery error gets smaller and smaller, and it is supposed to become negligible eventually. As we will discuss in Sec. V-B, however, the aggregation

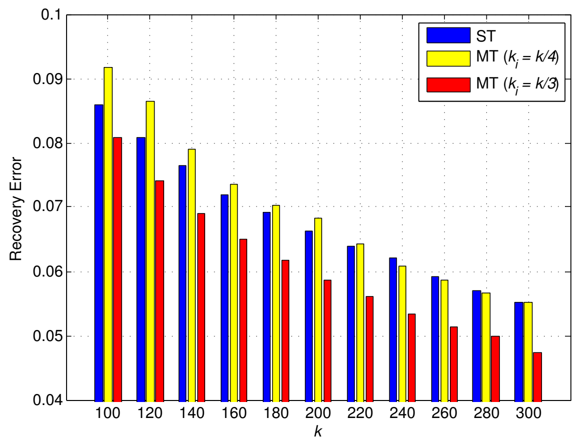


Fig. 7. CS recovery errors for the data sensed by 2048-node WSNs.

factor also affects the routing cost. Fortunately, Fig. 7 also shows that with tree partition and a proper aggregation factor, one can expect a promising recovery performance and much lower energy consumption (see Sec. V-B for details).

3) Joint Spatial and Temporal Recovery: Although CS becomes less effective for spatial recovery in small-scale WSNs, leveraging on diffusion wavelets, we can naturally add the temporal domain into consideration. In this section, our joint spatial and temporal recovery scheme is tested upon the data provided by the Met Office [3], which records the monthly climate information gathered from 33 weather stations in the UK ($n = 33$). We use a 6-month record (March 2009 to September 2009) for experiment, and we take $k = 5$ for each month. We may recover the weather data either independently for each month or jointly for all the 6 months. Concerning the joint spatial and temporal recovery, we set $g(\cdot) = \exp(|r_1 - r_2|)$. For the ease of illustration, we align the spatially distributed data in a sequence of the station indices and concatenate the monthly data into a row vector, as depicted in Fig. 8. As expected, for such a small-scale network, solely

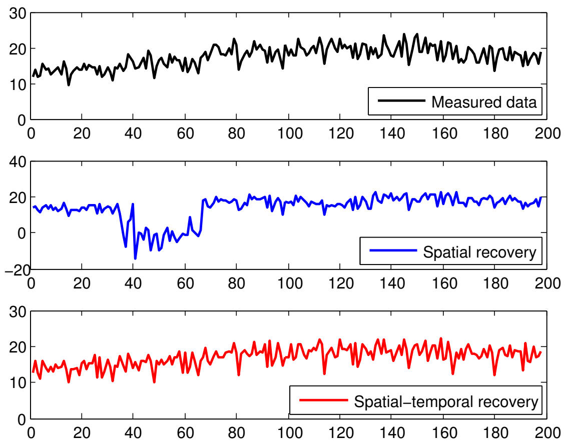


Fig. 8. Comparing independent spatial recovery and joint spatial and temporal recovery for UK temperature.

spatial recovery sometimes compromises the recovery fidelity, whereas the joint spatial and temporal recovery achieves a high fidelity recovery ($\epsilon = 8\%$) for all 6 months.

B. Energy Efficiency of CDA

To obtain the optimal solution for MECDA, we use CPLEX [2] to solve the MIP formulation given in Sec. IV-B1. The solvers for MECDA_GREEDY, CFL_RANDOM, and MST_PRUNED are developed in C++, based on Boost graph library [1]. We consider two types of network topologies: in *grid networks*, nodes are aligned in lattices with the sink being at a corner; and in *arbitrary networks*, nodes are randomly deployed with the sink being at the center. We also consider networks of different size but identical node density. The underlying communication graph is a complete graph; each link bears an energy cost proportional to the cube of the distance between its two ends, i.e., $c_{ij} = d_{ij}^3$. Note that such a communication graph is actually the worst case for a given vertex set V, as it results in the largest edge set. Less sophisticated communication graphs could be produced by a thresholding technique, i.e., removing edges whose weights go beyond a certain threshold, but that would just simplify the solution. The CFL_RANDOM solver runs ten times on each network deployment, so the mean value is taken for comparison. The MECDA_GREEDY and MST_PRUNED solvers suggest a deterministic solution for each specific instance. For arbitrary networks, we generate ten different deployments for each network size and we use the boxplot to summarize the results. On each box, the central mark is the median, the lower (upper) edge of the box are the 25th (75th) percentile, the whiskers extend to the extreme observations not considered outliers, and outliers are indicated individually as "+".

1) Efficacy of MECDA_GREEDY*:* We first compare the results obtained from the MIP and the greedy heuristic, aiming at demonstrating the near-optimality of MECDA_GREEDY. As MECDA is an NP-complete problem, the optimal solution to its MIP formulation can be obtained only for WSNs of small size. Therefore, we report the results for WSNs of 20 and 30 nodes, with $k \in \{4, 6, 8, 10\}$, which are summarized by boxplot in Fig. 9. It is evident that the

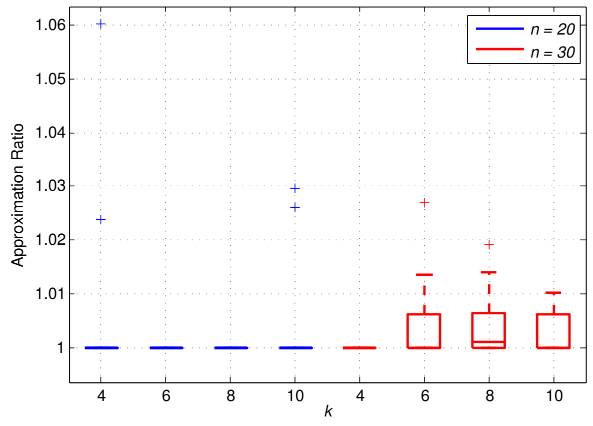


Fig. 9. Benchmarking MECDA_GREEDY in small-scale networks.

(empirical) approximation ratio is very close to 1, confirming the near-optimality of our heuristic. We will further validate the efficacy of MECDA_GREEDY using CFL_RANDOM and MST_PRUNED for large-scale WSNs, and we will also study the energy efficiency gained by CDA.

2) Bad Performance of Plain CS Aggregation: In Fig. 10, we compare CDA (results of MECDA_GREEDY) with plain

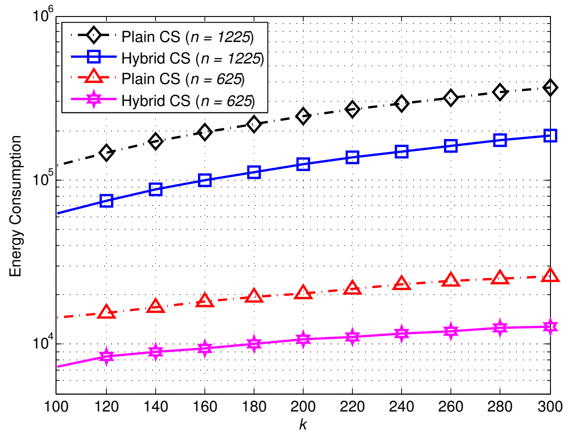


Fig. 10. Comparing CDA against plain CS aggregation.

CS aggregation, showing the bad performance of the latter one. As depicted, the results are obtained from grid networks consisting of 625 nodes and 1225 nodes, with aggregation factor k ranging from 100 to 300. Evidently, plain CS aggregation always consumes several times more energy than CDA. The reason can be explained as follows: plain CS aggregation overacts by forcing every node to be an aggregator, even though the aggregators are often in the minority for optimal configurations (see Fig. 3). In fact, plain CS aggregation is less energy efficient than non-aggregation unless k becomes unreasonably small ($\ll 100$).

3) Efficiency of CDA: To demonstrate the efficiency of CDA, we first consider a special case, i.e., a grid network with 1225 nodes in Fig. 11(a), then we proceed to more general cases, i.e., arbitrary networks with 2048 nodes in Fig. 11(b). We again set $k \in [100, 300]$. Four sets of results are compared, namely non-aggregation, and CDA solved by MECDA_GREEDY, CFL_RANDOM, as well as MST_PRUNED solvers. One immediate observation is that, compared with non-aggregation, CDA brings a remarkable cost reduction. In grid networks, 75% of the energy is saved in the best case. Even for the arbitrary networks, this saving can be up to 50%. The energy saving is gradually reduced as k increases, because the increase of k leads to the "shrinking" of the core, making the aggregation tree more and more like the SPT. Nevertheless, we have shown in Fig. 6 and Fig. 7 that $k = 10\%$ of n is sufficient to allow a satisfactory recovery ($\epsilon < 7\%$). Therefore, we can expect CDA to significantly outperform non-aggregation in general. Moreover, as the results obtained from MECDA_GREEDY are always bounded from above by those obtained from both CFL_RANDOM and MST_PRUNED (which have proven approximation ratios), the efficacy of MECDA_GREEDY in large-scale networks is also confirmed. This also explains why neither the randomized nor the pruned algorithm is suitable for solving MECDA: they may lead to solutions that are less energy efficient than non-aggregation (the energy consumptions of CFL_RANDOM and MST_PRUNED go beyond that of non-aggregation when

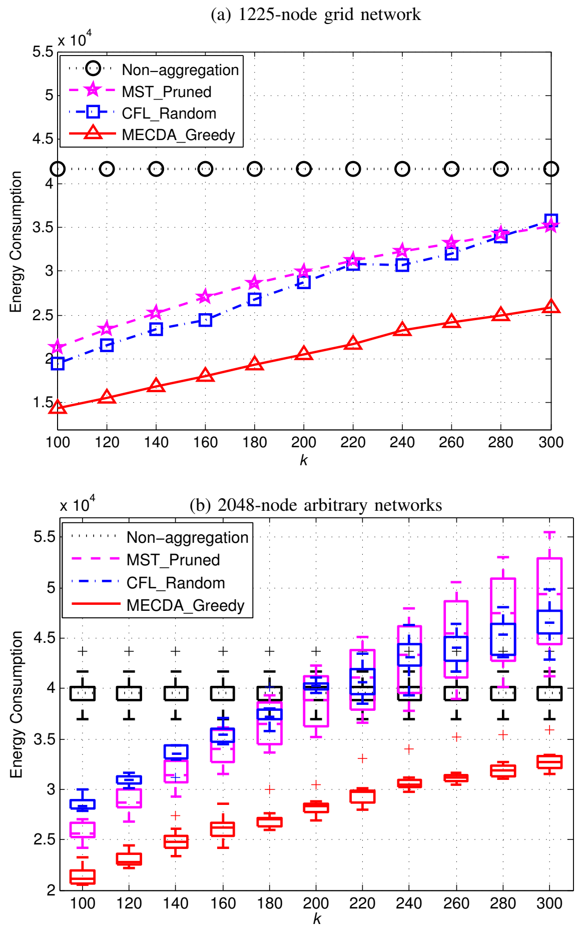


Fig. 11. Comparing CDA against non-aggregation and benchmarking MECDA_GREEDY in large-scale networks.

$k \geq 200$), whereas MECDA_GREEDY always finds a solution better than non-aggregation.

In Fig. 12, results for non-aggregation and CDA

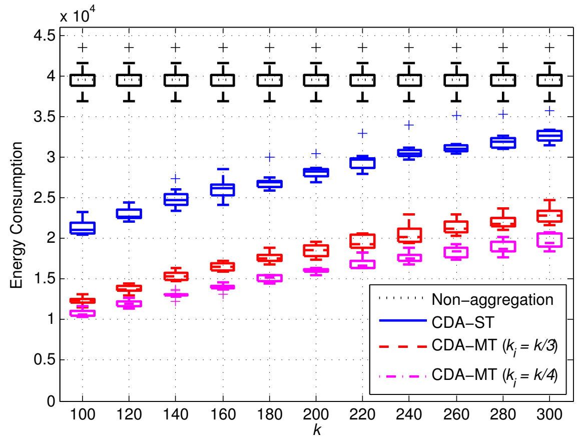


Fig. 12. Comparing the cost for CDA with and without network partition on arbitrary networks of 2048 nodes.

with/without network partition are compared to validate the better energy efficiency of CDA with network partition. MECDA_GREEDY is applied to individual partitions to construct aggregation trees. Along with the results in Sec. V-A, we can see that, if the network is partitioned into four subnetworks and the aggregation factor is accordingly set ($k_i = k/4$), up to 70%$\sim$50% energy can be saved with a little sacrifice on the recovery accuracy (compared with single tree CDA). However, if we boost the aggregation factor slightly, i.e., $k_i = k/3$, better recovery performance is achieved (see Fig. 7) while still maintaining significant energy saving. In practice, one could adjust the tree partition and aggregation factor according to application requirements.

VI. Related Work and Discussions

Data aggregation is one of the major research topics for WSNs, due to its promising effect in reducing data traffic. Given the page limitation, we only discuss, among the vast literature, a few contributions that are closely related to our proposal. Applying combinatorial optimizations to data aggregation was introduced in [17], assuming an aggregation function concave in the input. Whereas [17] aims at deriving an algorithm with a provable bound (albeit arbitrarily large) for all aggregation functions, we are considering a specific aggregation function that is inspired by our CDA, and we propose fast near-optimal solution techniques for practical use. Involving the correlation structure of data, other types of optimal data aggregation trees are derived in [36], [14]. However, as we explained in Sec. I, such approaches are too correlation structure dependent, hence not as flexible as our CS-based aggregation.

Compressed sensing is a recent development in signal processing field, following several important contributions from Candés, Donoho, and Tao (see [8] and the references therein). It has been applied to WSN for single hop data gathering [22], but only a few proposals apply CS to multihop networking. In [27], a throughput scaling law is derived for the plain CS aggregation. However, as we pointed out in Sec. V-B, plain CS aggregation is not an energy efficient solution. In addition, our results in [30] also demonstrate the disadvantage of plain CS aggregation in terms of improving throughput.

[33] investigates the routing cost for CS aggregation in multi-hop WSNs where the sensing matrix is defined according to the routing paths. [25] draws the observation that the sensing matrix has to take the characteristics of the sparse domain into account. The routing-dependent or domain-dependent design, unfortunately, contradicts the spirit of CS that sensing matrices can be random and easily generated. None of these proposals [27], [33], [25] can recover CS aggregated data from arbitrarily deployed WSNs, which is one of the key issues that we address in this paper and [37].

Though we require reliable transmissions to be handled at a lower layer, an alternative solution is over-sampled CS source coding [9]. Due to the inherent randomization of CS, packet loss is somewhat equivalent to reducing k at the coding end. Therefore, data recovery at the sink can still be performed (though bearing a larger error) in the face of packet loss.

VII. Conclusion

Leveraging on the recent development of compressed sensing, we have proposed a *compressed data aggregation* (CDA) scheme for WSN data collection in this paper. Our major contributions are twofold: 1) We have designed a proper sparse basis based on diffusion wavelets to achieve high fidelity recovery for data aggregated from arbitrarily deployed WSNs. We have developed this idea to allow for arbitrary network partitions and to integrate temporal correlations along with the spatial ones, which can significantly reduce energy consumption while maintaining the fidelity of data recovery. 2) We have investigated the minimum energy CDA problem by characterizing its optimal configurations, analyzing its complexity, as well as providing both an exact solution (for small networks) and approximate solutions (for large networks). For performance evaluation, we have carried out extensive experiments on both synthetic datasets and real datasets. The results, on one hand, demonstrate that high fidelity data recovery can be achieved by properly designing the sparse basis; and on the other hand, validate the significant energy efficiency in data collection.

Appendix A
Characterizing the Optimal Solution of MECDA

Condition 1) holds trivially. For every aggregator $i \in A$, the following statement holds: all nodes on the routing path from i to s are aggregators. This is so because if i is an aggregator, it has at least $k-1$ descendants in the spanning tree. Consequently, the parent of i has at least k descendants, justifying itself as an aggregator. Repeating this reasoning on the path from i to s confirms that the above statement is true. Now, as every aggregator sends out exactly k units of data, the minimum energy routing topology that spans A is indeed an MST for the subgraph induced by A, which gives us condition 2). Note that it is condition 1) that allows us to decompose the minimization problem into two independent problems: one for A and one for F.

For nodes in F, as they do not perform CS coding, the minimum energy routing topology should be determined by the shortest path principle. However, the destination of these shortest paths is not s, but the whole set A, as the energy expense inside A is independent of that of F. Therefore, for each node $i \in F$, it needs to route its data to a node $\hat{j} \in A$ that minimizes the path length; this is indeed condition 3). Condition 4) follows directly from the property of an aggregator: it has at least $k-1$ descendants in the spanning tree. To ensure optimal energy efficiency, each forwarder is restricted to have less than $k-1$ descendants. Q.E.D.

Appendix B
NP-Completeness of CSATCP^k

We first show that CSATCP^k is in NP. If a non-deterministic algorithm guesses a spanning tree, the partition of V into A and F can be accomplished in polynomial time ($\mathcal{O}(n\ell)$ for a rough estimation), simply by counting the number of descendants for each node. Then to test if the total cost is below B or not only costs another $n-1$ summations and multiplications to compute the total cost of the spanning tree. This confirms the polynomial time verifiability of CSATCP^k, hence its membership in NP.

Next, we prove the NP-completeness of CSATCP^k for $2 \leq k < n-1$ through a reduction from MLST. In fact, given an instance $G(V,E)$ of MLST, the goal is to partition V into two sets: leaf and non-leaf, which is similar to what needs to be done for CSATCP^k. We first extend $G(V,E)$ in three steps. First, we add an auxiliary node s and connect it to every vertex in V. Second, we attach to every vertex in V a path containing $k-2$ auxiliary vertices. Now, we have an extended graph $G'(V',E')$, as shown in Fig. 13. Finally, we assign a

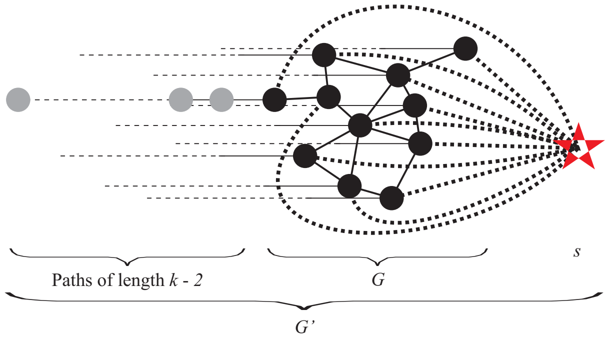


Fig. 13. Reduction from MLST to CSATCP^k. Given an instance $G(V,E)$ of MLST, we extend the graph by (1) adding an auxiliary node s and connect it to every vertex in V and (2) attach to every vertex in V a path containing $k-2$ auxiliary vertices.

cost 1 to every edge in E', except those between V and s whose costs are set to be $k+\varepsilon$ with ε being a small positive number. Given a certain parameter B, the answer to CSATCP^k can be obtained by finding the minimum cost spanning tree on G'. Note that the special cost assigned to the edges between s and V forces this spanning tree to use exactly one edge between s and V (incurring a constant cost of $k(k+\varepsilon)$), and to choose other edges in the rest of the graph G'.

Due to condition 4) of *Proposition 4*, a minimum cost configuration of CSATCP^k will put all the auxiliary nodes on the paths attached to V into F, and the total cost of these paths is a constant $\frac{1}{2}(k-1)(k-2)|V|$. The remaining step partitions V into A and F^{k-1}, with nodes in the second set each sending $k-1$ units of data, such that the total cost is minimized. This is equivalent to the following problem:

$$\begin{aligned} \text{minimize} \quad (k-1)|F^{k-1}| &+ k|A| && (14)\\ F^{k-1}\cup A &= V && (15)\\ F\cap A &= \emptyset && (16) \end{aligned}$$

with an additional constraint that A induces a connected subgraph of G. Since $k\left(|F^{k-1}|+|A|\right) = k|V|$ is a constant, the objective is actually to maximize the cardinality of $|F^{k-1}|$. Therefore, if we consider F^{k-1} as the leaf set of the spanning tree for V, this problem is exactly MLST.

In summary, what we have shown is the following: suppose we have an oracle that answers CSATCP^k correctly, then for every instance G of MLST, we simply extend it to G' following the aforementioned procedure. Given the above shown equivalence, the oracle will also answer MLST correctly. In particular, if the answer to CSATCP^k with $B = k(k+\varepsilon) + \left[\frac{1}{2}(k-1)(k-2)+k\right]|V| - K$ is true, then the answer to MLST is true. Now, given the NP membership of CSATCP^k and the NP-completeness of MLST [16], we have the NP-completeness of CSATCP^k. Q.E.D.

According to the proof, we also show that MECDA does not admit any PTAS. Denote by $\mathrm{APX}_{\mathcal{P}}$ and $\mathrm{OPT}_{\mathcal{P}}$ the approximated objective (by a PTAS) and the optimal value of a problem $\mathcal{P}$, then assuming the existence of a PTAS to MECDA implies that $\frac{\mathrm{APX}_{\mathrm{MECDA}}}{\mathrm{OPT}_{\mathrm{MECDA}}} \leq 1+\epsilon$. Given the aforementioned instance and let $f(k,|V|) = k(k+\varepsilon) + \left[\frac{1}{2}(k-1)(k-2)+k\right]|V|$, we have

$$\begin{aligned} \frac{\mathrm{APX}_{\mathrm{MECDA}}}{f(k,|V|)-\mathrm{OPT}_{\mathrm{MLST}}} &\leq 1+\epsilon \\ \Rightarrow \quad \frac{f(k,|V|)-|F^{k-1}|}{f(k,|V|)-\mathrm{OPT}_{\mathrm{MLST}}} &\leq 1+\epsilon \\ \Rightarrow \quad 1-\epsilon\left(\frac{f(k,|V|)}{\mathrm{OPT}_{\mathrm{MLST}}}-1\right) &\leq \frac{|F^{k-1}|}{\mathrm{OPT}_{\mathrm{MLST}}} \end{aligned}$$

As $f(k,|V|) > |V| > \mathrm{OPT}_{\mathrm{MLST}}, \forall k > 0$, we have $\left(\frac{f(k,|V|)}{\mathrm{OPT}_{\mathrm{MLST}}}-1\right) \geq 0$. Therefore, replacing $\epsilon\left(\frac{f(k,|V|)}{\mathrm{OPT}_{\mathrm{MLST}}}-1\right)$ by $\hat{\epsilon}$ suggests the existence of a PTAS for MLST. However, this contradicts the fact that MLST is MAX SNP-complete [15]. Consequently, the inapproximability of MECDA follows.

Appendix C
Computational Complexity of MECDA_Greedy

Recall in Algorithm 1, for each round of adding one aggregator, all $i \in \mathrm{B}(A)$ are tested. While within each testing phase, the routing tree is computed as the combination of an MST and SPTs. The iteration proceeds until no further expansion for the core. We analyze the computational complexity for Algorithm 1 in the following. First, the all-pairs shortest paths are computed in advance, which leads to a complexity of $\mathcal{O}(n^3)$ and contributes additively to the overall complexity. Considering the r-th outer iteration, we have $r+1$ elements in A and $n-r-1$ elements in F for each testing partition. While the complexity of $\mathrm{SPF}(F,A)$ is $\mathcal{O}\left((r+1)(n-r-1)\right)$ as only pairwise distances are compared from $j \in F$ to A, $\mathrm{MST}(A)$ incurs a complexity of $\mathcal{O}((r+1)^2)$ for the best implementation [13]. The cardinality of $\mathrm{B}(A)$ is bounded by $n-r$, so the whole testing phase for admitting the $r+1$-th aggregator costs $\mathcal{O}\left(\left[(r+1)^2+(r+1)(n-r-1)\right](n-r)\right)$. And the program proceeds at most $n-k$ iterations before ending. Therefore, the total complexity of all iterations is

$$\begin{aligned} &\mathcal{O}\left(\sum_{r=1}^{n-k}\left[(r+1)^2+(r+1)(n-r-1)\right](n-r)\right) \\ =\; &\mathcal{O}\left(n\sum_{r=1}^{n-k}(r+1)(n-r)\right) \\ =\; &\mathcal{O}\left(n^2\textstyle\sum_{r=1}^{n-k} r\right) = \mathcal{O}\left((n-k)^2n^2\right) \end{aligned}$$

Now, adding the initial complexity $\mathcal{O}(n^3)$ of computing the all-pairs shortest paths, the complexity of Algorithm 1 is $\mathcal{O}\left((n-k)^2n^2+n^3\right)$ Q.E.D.

References

[1] "Boost." [Online]. Available: http://www.boost.org/

[2] "IBM-ILOG CPLEX 11.0." [Online]. Available: http://www.cplex.com/

[3] "Met Office: UK's National Weather Service." [Online]. Available: http://www.metoffice.gov.uk/weather/uk/

[4] "National Digital Forecast Database." [Online]. Available: http://www.weather.gov/ndfd/

[5] "TinyOS." [Online]. Available: http://www.tinyos.net/

[6] E. Candès and J. Romberg, "ℓ_1-MAGIC." [Online]. Available: http://www.acm.caltech.edu/l1magic/

[7] E. Candès, J. Romberg, and T. Tao, "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information," *IEEE Trans. Info. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[8] E. Candès and M. Wakin, "An Introduction to Compressive Sampling," *IEEE Signal Processing Mag.*, vol. 25, no. 3, 2008.

[9] Z. Charbiwala, S. Chakraborty, S. Zahedi, Y. Kim, M. Srivastava, T. He, and C. Bisdikian, "Compressive Oversampling for Robust Data Transmission in Sensor Networks," in *Proc. of IEEE INFOCOM*, 2010.

[10] F. Chung, *Spectral Graph Theory*. Providence, R.I: AMS Press, 1997.

[11] R. R. Coifman and M. Maggioni, "Diffusion Wavelets," *Appl. Comp. Harm. Anal.*, vol. 21, no. 1, 2006.

[12] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. New York: John Wiley and Sons, 1998.

[13] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge: The MIT Press, 2001.

[14] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer, "Network Correlated Data Gathering with Explicit Communication: NP-completeness and Algorithms," *IEEE/ACM Trans. on Networking*, vol. 14, no. 1, 2006.

[15] G. Galbiati, F. Maffol, and A. Morzenti, "A Short Note on the Approximability of the Maximum Leaves Spanning Tree Problem," *Elsevier Information Processing Letters*, vol. 52, no. 1, 1994.

[16] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[17] A. Goel and D. Estrin, "Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk," in *Proc. of the 14th ACM-SIAM SODA*, 2003.

[18] A. Gupta, A. Kumar, and T. Roughgarden, "Simpler and Better Approximation Algorithms for Network Design," in *Proc. of the 35th ACM STOC*, 2003.

[19] H. Gupta, V. Navda, S. Das, and V. Chowdhary, "Efficient Gathering of Correlated Data in Sensor Networks," *ACM Trans. on Sensor Networks*, vol. 4, no. 1, 2008.

[20] K. Han, Y. Liu, and J. Luo, "Duty-cycle-aware minimum-energy multicasting in wireless sensor networks," *IEEE/ACM Trans. on Networking*, 2012 (accepted to appear).

[21] K. Han, L. Xiang, J. Luo, and Y. Liu, "Minimum-Energy Connected Coverage in Wireless Sensor Networks with Omni-Directional and Directional Features," in *Proc. ACM MobiHoc*, 2012, pp. 85–94.

[22] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, "Compressed Sensing for Networked Data," *IEEE Signal Processing Mag.*, vol. 25, no. 3, 2008.

[23] S. He, J. Chen, D. Yau, and Y. Sun, "Cross-layer Optimization of Correlated Data Gathering in Wireless Sensor Networks," in *Prof. of the 7th IEEE SECON*, 2010.

[24] L. T. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Info.*, vol. 15, pp. 141–145, 1981.

[25] S. Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega, "Spatially-Localized Compressed Sensing and Routing in Multi-hop Sensor Networks," in *Proc. of the 3rd GSN (LNCS 5659)*, 2009.

[26] X.-Y. Li, W.-Z. Song, and W. Wang, "A Unified Energy-Efficient Topology for Unicast and Broadcast," in *Prof. of ACM MobiCom*, 2005.

[27] C. Luo, F. Wu, J. Sun, and C.-W. Chen, "Compressive Data Gathering for Large-Scale Wireless Sensor Networks," in *Proc. of the 15th ACM MobiCom*, 2009.

[28] J. Luo and J.-P. Hubaux, "Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks," in *Proc. of the 24th IEEE INFOCOM*, 2005, pp. 1735–1746.

[29] ——, "Joint Sink Mobility and Routing to Increase the Lifetime of Wireless Sensor Networks: The Case of Constrained Mobility," *IEEE/ACM Trans. on Networking*, vol. 18, no. 3, pp. 871–884, 2010.

[30] J. Luo, L. Xiang, and C. Rosenberg, "Does Compressed Sensing Improve the Throughput of Wireless Sensor Networks?" in *Proc. of the IEEE ICC*, 2010.

[31] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation Service for Ad-hoc Sensor Networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, 2002.

[32] D. Needell and J. A. Tropp, "CoSaMP: Iterative Signal Recovery from Incomplete and Inaccurate Samples," *Commun. ACM*, vol. 53, no. 12, pp. 93–100, 2010.

[33] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorz, "On the Interplay between Routing and Signal Representation for Compressive Sensing in Wireless Sensor Networks," in *ITA*, 2009.

[34] D. Slepian and J. Wolf, "Noiseless Encoding of Correlated Information Sources," *IEEE Trans. on Information Theory*, vol. 19, no. 4, 1973.

[35] R. Subramanian and F. Fekri, "Sleep Scheduling and Lifetime Maximization in Sensor Networks: Fundamental Limits and Optimal Solutions," in *Proc. of 5th ACM IPSN*, 2006.

[36] P. von Richenbach and R. Wattenhofer, "Gathering Correlated Data in Sensor Networks," in *Proc. of the 2nd ACM DIALM-POMC*, 2004.

[37] L. Xiang, J. Luo, C. Deng, A. Vasilakos, and W. Lin, "DECA: Recovering Fields of Physical Quantities from Incomplete Sensory Data," in *Proc. of the 9th IEEE SECON*, 2012, pp. 107–115.

[38] L. Xiang, J. Luo, and A. V. Vasilakos, "Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks," in *Proc. of the 8th IEEE SECON*, 2011.

[39] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station," in *Prof. of the 9th ACM MobiHoc*, 2008.

Liu Xiang received her BS degree in Electronic Engineering from Tsinghua University, China. She is currently a PhD candidate in Computer Science working at the School of Computer Engineering, Nanyang Technological University in Singapore. Her research focuses on efficient data collections for wireless sensor networks. She is a student member of the IEEE. More information can be found at http://www.ntu.edu.sg/home2009/xi0001iu/.

Jun Luo received his BS and MS degrees in Electrical Engineering from Tsinghua University, China, and the PhD degree in Computer Science from EPFL (Swiss Federal Institute of Technology in Lausanne), Lausanne, Switzerland in 2006. From 2006 to 2008, he has worked as a post-doctoral research fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. In 2008, he joined the faculty of the School of Computer Engineering, Nanyang Technological University in Singapore, where he is currently an assistant professor. His research interests include wireless networking, distributed systems, multimedia protocols, network modeling and performance analysis, applied operations research, as well as network security. He is a Member of both IEEE and ACM. More information can be found at http://www3.ntu.edu.sg/home/junluo/.

Catherine Rosenberg was educated in France (Ecole Nationale Supérieure des Télécommunications de Bretagne, Diplôme d'Ingénieur in EE in 1983 and University of Paris, Orsay, Doctorat en Sciences in CS in 1986) and in the USA (UCLA, MS in CS in 1984), Dr. Rosenberg has worked in several countries including USA, UK, Canada, France and India. In particular, she worked for Nortel Networks in the UK, AT&T Bell Laboratories in the USA, Alcatel in France and taught at Purdue University (USA) and Ecole Polytechnique of Montreal (Canada). Since 2004, Dr. Rosenberg is a faculty member at the University of Waterloo where she now holds a Tier 1 Canada Research Chair in the Future Internet. Her research interests are broadly in networking with currently an emphasis in wireless networking and in traffic engineering (Quality of Service, Network Design, and Routing). She has authored over 100 papers and has been awarded eight patents in the USA. She is a fellow of the IEEE. More information can be found at http://ece.uwaterloo.ca/∼cath/.