# Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks*

Liu Xiang Jun Luo
School of Computer Engineering
Nanyang Technological University, Singapore
Email: {xi0001iu, junluo}@ntu.edu.sg

Athanasios Vasilakos
Department of Electrical and Computer Engineering
National Technical University of Athens, Greece
Email: vasilako@ath.forthnet.gr

***Abstract*—As a burgeoning technique for signal processing, *compressed sensing* (CS) is being increasingly applied to wireless communications. However, little work is done to apply CS to multihop networking scenarios. In this paper, we investigate the application of CS to data collection in wireless sensor networks, and we aim at minimizing the network energy consumption through joint routing and compressed aggregation. We first characterize the optimal solution to this optimization problem, then we prove its NP-completeness. We further propose a mixed-integer programming formulation along with a greedy heuristic, from which both the optimal (for small scale problems) and the near-optimal (for large scale problems) aggregation trees are obtained. Our results validate the efficacy of the greedy heuristics, as well as the great improvement in energy efficiency through our joint routing and aggregation scheme.**

## 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, which include, among others, energy conserving sleep scheduling (e.g., [1]), topology control (e.g., [2]), mobile data collectors (e.g., [3]), and data aggregation[1] (e.g., [4]). Whereas the first three approaches (and many others) focus on the efficiency of networking techniques that transport the sensory data, data aggregation directly aims at significantly reducing the amount of data to be transported, and it hence complements other approaches and is deemed as the most crucial mechanism to achieve energy efficient data collection for WSNs.

Although data aggregation techniques have been heavily investigated, there are still imperfections to be improved on. First, lossy aggregation that adopts simple aggregation functions (e.g., MIN/MAX/SUM) only extracts certain statistical quantities from the sensory data [5], [6], other information is thus lost and hence this aggregation technique only applies to particular applications that require limited information from a WSN. Secondly, though one may, in theory, apply distributed source coding technique, such as Slepian-Wolf coding [7], [4], to perform non-collaborative data compression at the sources, it is not exactly practical due to the lack of prior knowledge of the data correlation structure. Finally, whereas collaborative in-network compression makes it possible to discover the data correlation structure through information exchange [8], [9], it either requires a simple correlation structure [8], or often results in high communication load [9] that may potentially offset the benefit of this aggregation technique.

As a newly developed signal processing technique, *compressed sensing* (CS) promises to deliver a full recovery of signals w.h.p. from far fewer measurements than their original dimension, as long as the signals are *sparse* or *compressible* in some domain [10]. Although this technique appears to suggest a way of reducing the sensory data traffic for WSNs without the need for adapting to the data correlation structure [11], the complication involved in the interaction between data routing and CS-based aggregation has postponed the development on this front until very recently [12], [13]. In this paper, we promote a new data aggregation technique derived from CS, and we aim at minimizing the total energy consumption of a WSN in collecting sensory data from the whole network. To the best of our knowledge, we are the first to investigate the minimum energy CS-based data aggregation problem under a **combinatorial** framework. More specifically, we are making the following contributions in this paper:

- We define the problem of *minimum energy compressed data aggregation* (MECDA), and we provide the characterizations of the optimal solutions.
- We analyze the complexity of the (parameterized) decision versions of MECDA, and we prove the NP-completeness of MECDA in general (through a tricky reduction from the *maximum leaf spanning tree* problem [14]), and also relate MECDA with the existing non-aggregation and lossy aggregation approaches.
- We present a nontrivial mixed-integer programming (MIP) formulation of MECDA, through which we may obtain the optimal solutions for small scale WSNs, and we also propose a greedy heuristic that delivers near-optimal solutions for large scale WSNs.
- We report a large set of numerical results, which, on one hand, validate the efficacy of our heuristic, and on the other hand, demonstrate the energy efficiency of our CS-based data aggregation.

The remaining of our paper is organized as follows. We first give a brief overview of applying compressed sensing in

*This work is supported in part by the Start-up Grant of NTU and AcRF Tier 1 Grant RG 32/09.

[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.

978-1-4577-0093-4/11/$26.00 ©2011 IEEE

networking in Sec. II. Then we define our minimum energy data aggregation problem and present the characterizations of its optimal solution in Sec. III. After showing the NP-completeness of our problem in Sec. IV, we present two solution techniques in Sec. V. We further report numerical results to demonstrate energy efficiency of our hybrid CS data aggregation approach in Sec. VI. Finally, we discuss the related work and the limitation of our current approach in Sec. VII, before concluding our paper in Sec. VIII.

## II. Compressed Data Aggregation: An Overview

In this section, we first briefly introduce the basic theory of CS, and then we explain in detail how CS can be applied to data collection in WSNs.

### A. Compressed Sensing Basics

Suppose a signal $\mathbf{u} = [u_1, \cdots, u_n]^T$ has an *m-sparse representation* under a proper basis $\Psi = [\psi_1, \ldots, \psi_n]$, s.t. $\mathbf{u} = \sum_{i=1}^{m} w_i \psi_i$ and $m \ll n$. The theory of CS states that, under certain conditions, instead of directly collecting $\mathbf{u}$, we only need to collect $k = \mathcal{O}(m \log n)$ measurements $\mathbf{v} = \Phi\mathbf{u}$, where $\Phi = [\phi_1, \ldots, \phi_n]$ is a $k \times n$ "sensing" matrix[2] whose row vectors are largely incoherent with $\Psi$. Consequently, we can perfectly recover $\mathbf{u}$ from $\mathbf{v}$ w.h.p. by solving the convex optimization problem ($\|\mathbf{w}\|_{\ell_1} = \sum_i |w_i|$)

$$\min_{\mathbf{w} \in \mathbb{R}^n} \|\mathbf{w}\|_{\ell_1} \qquad \text{subject to} \quad \mathbf{v} = \Phi\Psi\mathbf{w}, \tag{1}$$

and by letting $\mathbf{u} = \Psi\hat{\mathbf{w}}$, with $\hat{\mathbf{w}}$ being the optimal solution.

The practical performance of CS coding depends on the sparsity of the signal, as well as the reconstruction algorithm. As networked data is generally quite sparse in nature, CS suits well for data collection in WSNs.

### B. CS Coding in Networking Context

Assume a WSN of $n$ nodes with each one acquiring a sample $u_i$, the ultimate goal of the WSN is to collect all data $\mathbf{u} = [u_1, \cdots, u_n]^T$ at a particular node called *sink*. Without data aggregation, each node needs to send its sample to the sink following a routing path, hence nodes around the sink will carry heavy traffic as they are supposed to relay the data from the downstream nodes. Fortunately, applying CS to data collection suggests a way to alleviate the bottleneck. To illustrate the idea, we rewrite the CS coding as

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

Now the idea of CS-based aggregation becomes clear [11]: each node $i$ first expands its sample to a $k$-dimension vector through a column vector $\phi_i$, then this encoded vector rather than the raw data is transmitted. The aggregation is done by summing the coded vectors whenever they meet, therefore, the traffic load on the aggregation path is always $k$. Eventually, the sink collects the aggregated $k$-dimension vector rather than $n$ raw samples, then the decoding algorithm is used to recover the $n$ raw samples. 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.

### C. Hybrid CS Aggregation

Interestingly, 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; or will carry 1 sample if lossy aggregation is performed. If we apply the CS principle directly, the so called *plain CS aggregation* will force every link to carry $k$ samples, leading to unnecessary higher traffic at the early stage transmissions. Therefore, the proper way of

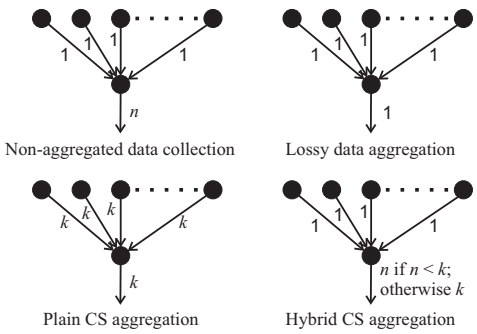


Fig. 1. Comparison of different data collection mechanisms. The link labels correspond to the carried traffic.

applying CS is to start CS coding only when the outgoing samples will become no less than $k$, otherwise, raw data collection (*non-aggregation* hereafter) is used. We coined this scheme as *hybrid CS aggregation* in our previous work [15]. Obviously, hybrid CS aggregation marries the merits of non-aggregation and plain CS aggregation: it reduces the traffic load while preserving the integrity of the original data set. Note that two types of traffic are imposed by the hybrid CS aggregation, namely the *encoded traffic* and the *raw traffic*, which can be differentiated by a flag carried by a data packet.

In a practical implementation, all nodes are initialized with non-aggregation mode by default. Given the publicly known threshold $k$, a node $i$ waits to receive from all its downstream neighbors[3] all the data they have to send. Only upon receiving more than $k-1$ raw samples or any encoded samples, node $i$ will switch to the CS aggregation mode 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 aggregated column vector.

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

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

According to the description above, we need an implicit synchronization in generating $\Phi$, i.e., for a given round of data collection, the $i$-th column of $\Phi$ has to be the same wherever it is generated (as CS coding is not necessarily performed at a source node). 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.

## III. Problem Statement and Solution Characterization

### A. Model and Problem

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). There is a special node $s \in V$ known as the sink that collects data from the whole network. We denote by $n$ and $\ell$ the cardinalities of $V$ and $E$, respectively. 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 $(i,j)$. 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 $(i,j)$. The objective of our problem is to minimize the total cost (or energy consumption)

$$\sum_{(i,j)\in E} c_{ij} x_{ij}$$

We assume that all nodes are roughly time synchronized and the data collection proceeds in rounds. At the beginning of each round, every node produces one unit of data (one sample), and the sink collects all information at the end. We assume no packet loss is incurred; this can be achieved by properly scheduling the network with effective MAC mechanisms (e.g., [16]). Based on the system orientation in Sec. II-C, we claim that no encoded traffic can coexist with any raw traffic on a link. This is so because once CS aggregation is initiated, allowing additional raw data transmission would only hurt the energy efficiency. Besides, encoded traffic cannot be split once it is introduced, otherwise it would be impossible to append new data at a later stage. So the encoded traffic is constrained on a spanning tree. While for the raw traffic, there is always an optimal single-path routing strategy. Consequently, without loss of generality, we restrict the data aggregation on a **tree** rooted at the sink. The nodes that route their data to the sink through node $i$ are called the *descendants* of $i$. We hereby formally specify the abstract model for the hybrid CS aggregation scheme.

*Definition 1 (Hybrid CS Aggregation):* Given a certain routing tree $T$, the outgoing traffic from node $i$ is

$$x_{ij:(i,j)\in T} = \begin{cases} \sum_{j:(j,i)\in T} x_{ji} + 1, & \text{if } \sum_{j:(j,i)\in T} x_{ji} < k-1; \\ k, & \text{otherwise.} \end{cases}$$

While the first case corresponds to the non-aggregation where the conventional flow conservation holds, the second case represents the CS coding for which the encoded traffic is always $k$. 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 *minimum energy compressed data aggregation* (MECDA) problem aims at 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 and present them in the next section.

### B. Characterizing the Optimal Solution

Each optimal solution of MECDA suggests an optimal *configuration*, in terms of routing paths and aggregator assignment. We present several conditions that characterize an optimal configuration in this section.

*Proposition 1:* In an optimal configuration, we have

1) The network is partitioned into two disjoint sets: *aggregator set* $A$ ($s \in A$) and *forwarder set* $F$, i.e., $A \subseteq V$, $F \subseteq V$, $A \cup F = V$, and $A \cap F = \emptyset$.
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 from it to some node $\hat{j} \in A$ is a shortest path, and $\hat{j}$ is the one that minimizes the length of this path, i.e., $p_{i\hat{j}} = \min_{j\in A}(\mathrm{SP}_{ij})$, where $\mathrm{SP}_{ij}$ refers to the shortest path from $i$ to $j$.
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, here we just illustrate an optimal configuration by Fig. 2. In

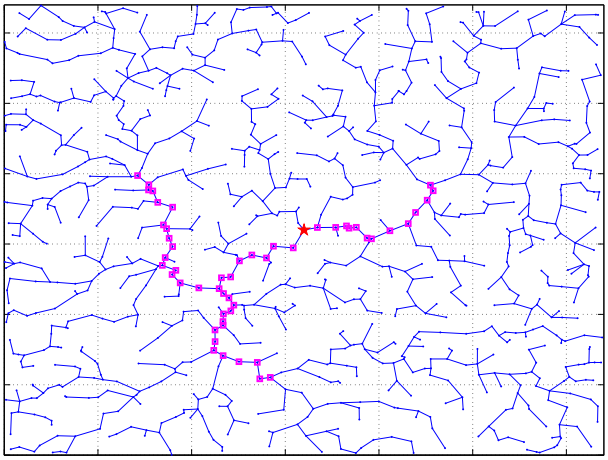

Fig. 2. An optimal CS aggregation tree in a 1024-node WSN, with $k = 150$. The sink is represented by the pentagram and the aggregators are marked as squares; the rest are forwarders. The original graph $G$ is a complete graph, with the edge cost being a function of the distance between its two ends. We only plot the tree edges to avoid confusion.

fact, the above conditions assert that an optimal configuration is a *spanning tree* rooted at $s$. It 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, if the plain CS aggregation is applied, we have $A = V$ and $F = \emptyset$. Therefore, the optimal solution is trivial: an MST of $G$. As a result, we focus on investigating the hybrid CS aggregation hereafter.

## IV. Complexity Analysis

Based on our characterization in Sec. III-B, MECDA does not appear to admit a straightforward solution of polynomial time complexity, given its graph partitioning nature. Therefore, we hereby analyze the problem complexity. Our results establish the NP-completeness of MECDA, through a nontrivial reduction from the *maximum leaf spanning tree* (MLST) problem [14]. As a byproduct, we also obtain the inapproximability of MECDA.

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 [14].

> INSTANCE: A graph $G = (V, E)$, a positive integer $K \leq |V|$.
>
> 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 2:* 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 lossy aggregation and non-aggregation are both special cases of MECDA. Unfortunately, other parameterized versions of CSATCP (hence MECDA) are intractable.

*Proposition 3:* CSATCP$^k$, with $2 \leq k < n - 1$, 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 omitted due to the space limitation; it follows directly from the MAX SNP-completeness of the MLST (optimization) problem [17] and our proof to *Proposition 3*.

## V. Min-Energy Compressed Data Aggregation

Given the hardness and inapproximability results for MECDA, we take two strategies to tackle MECDA in this section. We first come up with a nontrivial mixed-integer programming (MIP) formulation, which allows us to obtain optimal solutions to MECDA in small scale WSNs. Then we propose a heuristic that solves the problem efficiently. A randomized algorithm is borrowed from [18] to benchmark our heuristic in large scale WSNs.

### A. A Mixed Integer Program Formulation

Essentially, we formulate the MECDA problem as a special minimum-cost flow problem. The main difference between hybrid CS aggregation and non-aggregation is the flow conservation: hybrid CS aggregation 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 $y_i = 0$. The first constraint is just the conventional flow conservation if $i$ is not an aggregator (remember we assume that every node generates one unit of data in each round of data collection). If $i$ is an aggregator ($y_i = 1$), the first constraint trivially holds, as 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. The second constraint states the constant outgoing flow if $i$ is an aggregator; otherwise $\sum_{j:(i,j)\in E} x_{ij} \geq 1$ trivially holds.

It is not sufficient to have only the extended flow conservation constraints, as the extension may allow loops. According to *Proposition 1*, 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.[4]

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

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 flow between every node $i$ and $s$. Another 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 $\mathbf{x}$.

The final step is to make the connection between $x_{ij}$ and $z_{ij}$, such that only edges carrying positive flows are indicated

[4] Conventional spanning tree formulation (e.g., [19]) involves an exponential number of constraints $\sum_{(i,j):i\in S, j\in S} z_{ij} \leq |S| - 1, \ \forall S \subseteq V$.

as tree edges. This is achieved by two inequalities applied to every edge $(i,j) \in E$,

$$x_{ij} - z_{ij} \geq 0 \quad \text{and} \quad z_{ij} - k^{-1} x_{ij} \geq 0,$$

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.

$$\text{minimize} \quad \sum_{(i,j)\in E} c_{ij} x_{ij} \tag{2}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} + (n-k) y_i \;\geq\; 1 \quad \forall i \in V\backslash\{s\} \tag{3}$$

$$\sum_{j:(i,j)\in E} x_{ij} - (k-1) y_i \;\geq\; 1 \quad \forall i \in V\backslash\{s\} \tag{4}$$

$$x_{ij} - z_{ij} \;\geq\; 0 \quad \forall (i,j) \in E \tag{5}$$

$$z_{ij} - k^{-1} x_{ij} \;\geq\; 0 \quad \forall (i,j) \in E \tag{6}$$

$$\sum_{j:(i,j)\in E} \bar{x}_{ij} - \sum_{j:(j,i)\in E} \bar{x}_{ji} - \frac{1}{n-1} \;\geq\; 0 \quad \forall i \in V\backslash\{s\} \tag{7}$$

$$z_{ij} - \bar{x}_{ij} \;\geq\; 0 \quad \forall (i,j) \in E \tag{8}$$

$$\sum_{(i,j)\in E} z_{ij} - (n-1) \;=\; 0 \tag{9}$$

$$x_{ij}, \bar{x}_{ij} \geq 0, \quad z_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{10}$$

$$y_i \in \{0,1\} \qquad \forall i \in V\backslash\{s\} \tag{11}$$

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

### *B. A Greedy Heuristic*

As explained in Sec. III-B, 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 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**.

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 [20], the latter can be obtained by simply performing linear searches on an all-pairs shortest paths table (obtained in advance by Floyd-Warshall algorithm [20]). 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 $\mathbf{t}$ that counts the number of descendants for every node in $A$. The cost incurred by a certain partition $A$ and $F$ is the sum of $\mathsf{cost}_{\mathrm{MST}}$ and $\mathsf{cost}_{\mathrm{SPF}}$.

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, and the algorithm terminates if no further core expansion is allowed. 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 $\{T, A\}$ satisfy all the conditions, in particular 4), stated in *Proposition 1*, because otherwise $A$ can be further expanded and the algorithm would not have terminated.

**Algorithm 1:** MECDA_Greedy

**Input**: $G(V,E)$, $s$, $k$
**Output**: $T$, $A$

$A = \{s\}$; $F = V\backslash\{s\}$; $\mathsf{cost} = \infty$
**repeat**
  **forall the** $i \in \mathrm{B}(A)$ **do**
    $A_{\mathsf{test}} = A \cup \{i\}$; $F_{\mathsf{test}} = F\backslash\{i\}$
    $\{\mathsf{cost}_{\mathrm{MST}}, L\} \leftarrow \mathrm{MST}(A_{\mathsf{test}})$
    $\{\mathsf{cost}_{\mathrm{SPF}}, \mathbf{t}\} \leftarrow \mathrm{SPF}(F_{\mathsf{test}}, A_{\mathsf{test}})$
    **if** $\mathsf{cost}_{\mathrm{MST}} + \mathsf{cost}_{\mathrm{SPF}} \leq \mathsf{cost}$ *AND* $\min_{l\in L} t_l \geq k-1$ **then**
      $\mathsf{cost} = \mathsf{cost}_{\mathrm{MST}} + \mathsf{cost}_{\mathrm{SPF}}$
      $A_{\mathsf{cand}} = A_{\mathsf{test}}$; $F_{\mathsf{cand}} = F_{\mathsf{test}}$
    **end**
  **end**
  $A = A_{\mathsf{cand}}$; $F = F_{\mathsf{cand}}$
**until** $A$ *unchanged*;
$T = \mathrm{MST}(A) \cup \mathrm{SPF}(F, A)$
**return** $T$, $A$

*Proposition 4:* **Algorithm 1** has a polynomial time complexity, which is $\mathcal{O}\left((n-k)^2 n^2 + n^3\right)$.
We refer to Appendix C for the detailed proof.

### *C. A Randomized Algorithm*

In large scale WSNs, computing the optimal configuration becomes extremely hard. To this end, we seek to benchmark our greedy heuristic by a randomized algorithm [18] whose approximation ratio is provably good. 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 the upper bounds for CFL also hold for MECDA. The main body of this algorithm is given in **Algorithm 2**. Here $\rho$ is the approximation ratio for the Steiner tree problem given by a specific algorithm. Later we will approximate the Steiner tree by constructing an MST on the extended graph where each edge weight is redefined as the shortest distance between the two ends on the communication graph. So we have $\rho = 2$ [21] in our implementation of the randomized algorithm.

**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;
3 Connect each forwarder $j \in F$ to its closest aggregator in $A$ using the shortest path.

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. VI-C. Therefore, we use the randomized algorithm only to benchmark our greedy heuristic rather than solving MECDA.

### D. A Practical Implementation

We hereby present a practical implementation adapted from **Algorithm 1** in Sec. V-B. Due to the iterative computations for the core and shell, it is too costly (in message passing) to rely on a fully distributed implementation. Instead, we shift the computation load to the sink: it first acquires the network topology information, then computes $\{T, A\}$ using **Algorithm 1**, and finally disseminates the information to the whole WSN. The network topology can be acquired in a distributed fashion. Firstly, nodes exchange information locally within a WSN, such that each node obtains the knowledge of its one-hop neighbors. The neighborhood information is then collected by the sink, which is used to recover the underlying network topology and to compute the aggregation tree. After this tree initialization phase, nodes may still join (new deployments) or leave (failures) the WSN. For gradual node joining or leaving, the tree can be adapted by locally growing or pruning the core $A$. However, the adapted tree might not satisfy conditions 2) and 3) stated in *Proposition 1*. Therefore, after a massive node joining or leaving (supposed to be a rare event), the aggregation tree needs to be re-initialized.

## VI. Numerical Simulations

In this section, we first introduce our experiment setting, and then present the results obtained from the solution techniques described in Sec. V.

### A. Experiment Setting

To obtain the optimal solution, we use CPLEX [22] to solve the MIP formulation given in Sec. V-A. The greedy and randomized solvers are developed in C++, based on Boost graph library [23]. We consider two types of network topologies: in *grid networks*, nodes are aligned in lattices with sink being at a corner; and in *arbitrary networks*, nodes are randomly deployed with sink being at the center. We also consider networks of different size where the node density is identical. The underlying communication graph is a complete graph; each link has a weight proportional to the cube of the distance between its two ends. 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 randomized solver runs ten times on each network deployment, so the mean value is taken for comparison. For arbitrary networks, we generate ten different deployments for each network size and we use the boxplot to summarize the results, which shows five quantities: lower quartile (25%), median, upper quartile (75%), and the two extreme observations.

### B. Efficacy of Greedy Algorithm

In this section, we compare the results obtained from the MIP and the greedy heuristic, aiming at demonstrating the near-optimality of the greedy algorithm. 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 with 20 and 30 nodes, with $k \in \{4, 6, 8, 10\}$, in Fig. 3. It is evident that the (empirical) approximation ratio is very close to 1, confirming the near-optimality of our heuristic.

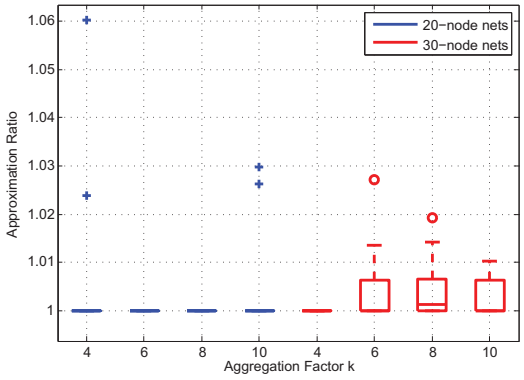


Fig. 3. Benchmarking the greedy heuristics in small networks.

### C. Efficiency of Hybrid CS Aggregation

Once the efficacy of the greedy algorithm is validated in Sec. VI-B, we are now ready to study the energy efficiency gained by properly applying the CS-based aggregation in large scale WSNs. Fig. 4 shows the comparisons of energy consumption between hybrid CS aggregation and plain CS aggregation, as well as that between hybrid CS aggregation and non-aggregation.

We first compare hybrid CS aggregation (results of greedy solver) with plain CS aggregation, showing the incompetence of the latter one. As depicted in Fig. 4(a), 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 hybrid CS aggregation. The reason can be explained as follows: plain CS aggregation overacts by forcing every node to be an aggregator, even though the

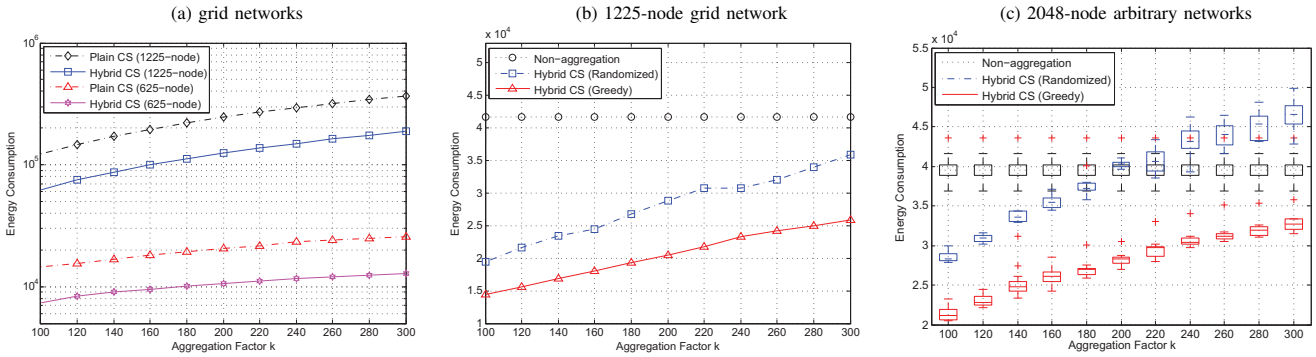


Fig. 4. Comparing the total energy consumption for plain CS aggregation, hybrid CS aggregation, and non-aggregation schemes.

aggregators are often in the minority for optimal configurations (see Fig. 2). In fact, plain CS aggregation is less energy efficient than non-aggregation unless $k$ becomes unreasonably small ($\ll 100$). Therefore, we shall later only compare hybrid CS aggregation with non-aggregation.

To demonstrate the efficiency of hybrid CS aggregation, we first consider a special case, a grid network with 1225 nodes in Fig. 4(b), then we proceed to more general cases, arbitrary networks with 2048 nodes in Fig. 4(c). We again set $k \in [100, 300]$. Three sets of results are compared, namely non-aggregation, and hybrid CS aggregation from both the greedy and randomized solvers. One immediate observation is that, compared with non-aggregation, hybrid CS aggregation brings a remarkable cost reduction. In grid networks, almost half the energy is saved in the worst case; even for the arbitrary networks, hybrid CS aggregation leads to over 20% energy saving in the worst case. The difference gap is slightly narrowed down 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 demonstrated in our companion work that $k = 10\%n$ is sufficient to allow a satisfactory recovery. Therefore, we can expect the hybrid CS aggregation to significantly outperform non-aggregation in general. Moreover, as the results obtained from our greedy solver are always bounded from above by those from the randomized solver (which has a proven approximation ratio), the efficacy of our greedy algorithm in large scale networks is also confirmed. This also explains why the randomized algorithm is not suitable for solving MECDA: it may lead to solutions that are less energy efficient than non-aggregation, whereas our greedy solver always finds a solution better than non-aggregation.

## VII. Related Work and Discussions

Data aggregation is one of the major research topics for WSNs, exactly due to its promising effect in reducing data traffic. Due to 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 [24], assuming an aggregation function concave in the input. Whereas [24] 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 the hybrid CS aggregation, and we propose fast near-optimal solution techniques for practical use. Involving the correlation structure of sensory data, other types of optimal data aggregation trees are derived in [25], [8]. 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 celebrated contributions from Candés, Donoho, and Tao (see [10] and the references therein). It has been applied to WSN for single hop data gathering [11], but only a few proposals apply CS to multihop networking. In [12], a throughput scaling law is derived for the plain CS aggregation. However, as we pointed out in Sec. VI-C, plain CS aggregation is not an energy efficient solution. In addition, our results in [15] also demonstrate the disadvantage of plain CS aggregation in terms of improving throughput. In [13], the focus is rather on creating sparse CS projection through clustering than finding optimal routing topology. Though we assume reliable transmissions throughout this paper, unreliable links can be handled by applying oversampled CS source coding [26]. Thanks to its inherent randomization, compressive oversampling can neutralize the stochastic nature of wireless link disturbances and hence compensate channel erasures, which eventually makes the data recovery at the sink largely immune to packet losses.

For specific surveillance scenarios where the physical phenomena are identically distributed in the network area and the sensor nodes are uniformly distributed, we may assume that the sparse dimension $m$ of the sensory data is proportional to the network size $n$, which directly translates to the proportionality between $k$ and $n$. As we mentioned in Sec. VI-C, the core grows larger as $k$ decreases, leading to an increasing energy efficiency. If we can partition the network into several sub-networks, and carry out the hybrid CS aggregation independently within each part, we are able to further cut down the energy consumption. For instance, every

tree branch rooted at the sink may serve as a sub-network [15]. Also, if a proper sparse projection [13] can be found, the resulting $k$ for each cluster is also smaller than that for the whole network. We are on the way of exploring this direction to further improve the energy efficiency of CS aggregation.

## VIII. Conclusion

In this paper, we have investigated the energy efficiency aspect of applying compressed sensing (CS) to data collection in wireless sensor networks (WSNs). We have first defined the problem of minimizing energy consumption through joint routing and compressed aggregation. Then we have characterized the optimal solution to this optimization problem, and also proven its NP-completeness. We have further proposed two solution techniques to obtain both the optimal (for small scale problems) and the near-optimal (for large scale problems) aggregation trees. We have shown the prominent improvement in energy efficiency of our CS-based data aggregation through numerical simulations.

We plan to extend our current work in mainly two directions. On one hand, we are interested in further reducing the energy consumption by involving network partition, as discussed in Sec.VII. On the other hand, we are also considering the cases where not all nodes are sources, which may require different heuristics to tackle.

## IX. Acknowledgements

We are grateful to the anonymous reviewers for their constructive feedback.

## References

[1] 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.

[2] X.-Y. Li, W.-Z. Song, and W. Wang, "A Unified Energy-Efficient Topology for Unicast and Broadcast," in *Prof. of the 11th ACM MobiCom*, 2005.

[3] 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.

[4] 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.

[5] 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.

[6] S. Cheng, J. Li, Q. Ren, and L. Yu, "Bernoulli Sampling Based $(\epsilon, \delta)$-Approximate Aggregation in Large-Scale Sensor Networks," in *Proc. of the 29th IEEE INFOCOM*, 2010.

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

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

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

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

[16] "IEEE 802.15.4-2006." [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

[17] 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.

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

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

[21] V. Vazirani, *Approximation Algorithms*. New York: Springer-Verlag, Berlin, 2001.

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

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

[24] 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.

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

[26] 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 the 29th IEEE INFOCOM*, 2010.

## Appendix A
## Characterizing the Optimal Solution of MECDA

Condition 1) holds trivially, as the partition is determined by performing CS coding or not, while each node is bounded to fall into either side. 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 the fact $i$ is an aggregator implies that $i$ 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. One may verify that, if the condition did not hold, either the node should be an aggregator (for a member of $F$) or it should not be an aggregator (for a leaf of $A$). 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. 5. Finally, we assign a cost

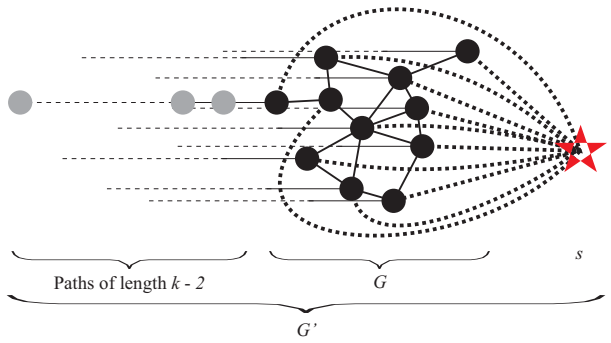


Fig. 5. Reduction from MLST to CSATCP$^k$. Given an instance $G(V,E)$ of MLST, we extend the graph by (1) adding an auxiliary node $s$ and connect it to every vertex in $V$ and (2) attach to every vertex in $V$ a path containing $k-2$ auxiliary vertices.

1 to every edge in $E'$, except those between $V$ and $s$ whose costs are set to be $k+\varepsilon$ with $\varepsilon$ 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 the condition 4) of *Proposition 1*, 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 configuration 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:

$$\begin{aligned} \text{minimize} \quad & (k-1)|F^{k-1}| + k|A| && (12)\\ & F^{k-1} \cup A = V && (13)\\ & F \cap A = \emptyset && (14) \end{aligned}$$

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 [14], we have the NP-completeness of CSATCP$^k$. Q.E.D.

According to the proof, we may also show that MECDA does not admit any PTAS. This idea is to reduce an approximation of MLST to that of MECDA. Consequently, the inapproximability of MECDA follows from the MAX SNP-completeness of MLST [17].

## 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, within each testing phase an MST and an SPF are computed. 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 [20]. 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.

In fact, the $n-k$ outer iterations only happen for linear networks. Given an arbitrary network, the algorithm may require far less than $n-k$ outer rounds to terminate, which suggests that the actual complexity could be much lower.