Does Compressed Sensing Improve the Throughput of Wireless Sensor Networks?

Jun Luo Liu Xiang
School of Computer Engineering
Nanyang Technological University (NTU), Singapore
Emails {junluo,xi0001iu}@ntu.edu.sg

Catherine Rosenberg
Department of Electrical and Computer Engineering
University of Waterloo, Canada
Emails: cath@engmail.uwaterloo.ca

***Abstract*—Although *compressed sensing* (CS) has been envisioned as a useful technique to improve the performance of *wireless sensor networks* (WSNs), it is still not very clear how exactly it will be applied and how big the improvements will be. In this paper, we propose two different ways (*plain-CS* and *hybrid-CS*) of applying CS to WSNs at the networking layer, in the form of a particular data aggregation mechanism. We formulate three flow-based optimization problems to compute the throughput of the *non-CS*, *plain-CS*, and *hybrid-CS* schemes. We provide the exact solution to the first problem corresponding to the non-CS case and lower bounds for the cases with CS. Our preliminary numerical results are only for a low-power regime. They illustrate two crucial insights: first, applying CS naively may not bring any improvement, and secondly, our hybrid-CS can achieve significant improvement in throughput.**

***Index Terms*—Wireless sensor networks, compressed sensing, data aggregation, routing, scheduling.**

I. Introduction

Improving the performance (in terms of throughput, lifetime, delay, etc.) of *wireless sensor networks* (WSNs) is a recurring issue of the wireless networking community. It is well known that proper data aggregation techniques[1] may significantly reduce the amount of data transmission load carried by a WSN and may hence improve its performance in every aspect. However, conventional aggregation techniques have many drawbacks. First, if only statistical quantities such as mean and max are extracted from the sensory data [1], [2], other features of these data are lost and hence this aggregation technique only applies to particular applications that require limited information from a WSN. Secondly, distributed source coding technique, such as Slepian-Wolf coding [3], may be applied to allow non-collaborative data compression at the sources, but the lack of prior knowledge of the data correlation structure could render it impossible to perform the coding operations. Finally, whereas collaborative in-network compression makes it possible to discover the data correlation structure through information exchange [4], [5], the resulting high computation and communication load may potentially offset the benefit of this aggregation technique.

In this paper, we consider the application of a new decentralized compression technology known as *compressed sensing* (CS) [6], [7] to in-network data aggregation, and we only focus on the network throughput as the objective. We first show a naive application of CS where the encoding is performed at every source. We then propose to apply CS only to relay nodes that are overloaded. These are what we call respectively the *plain-CS* and *hybrid-CS* schemes in the following. We formulate three flow-based optimization problems to compute the throughput of the *non-CS*, plain-CS, and hybrid-CS schemes, and we provide the **exact solution** to the problem corresponding to the non-CS case, as well as **lower bounds** for the two cases with CS. In formulating these problems, we assume that the transmission through wireless links can be scheduled in a conflict-free manner, and we make use of an SINR-based interference model. The resulting joint routing, compression, and scheduling problem is notoriously hard, but the tools that we have developed recently allow us to deal with link scheduling problems of very large scale [8], [9] and can be adapted to numerically solve the aforementioned three problems (or certain simplified versions of them).

The contribution of our work, apart from formulating the joint compression and scheduling problems, is the crucial insights gained from the numerical solutions of these problems. In particular, we show that applying CS naively may not bring any improvement, while our hybrid-CS scheme can achieve significant throughput improvement in the low-power regime. Previous proposals to apply CS to WSNs are concerned with either single-hop data aggregation [10] or efficient data dissemination [11]. To the best of our knowledge, there has been no prior work that has quantified the improvement in throughput by applying CS as an in-network data aggregation mechanism. Due to their high complexity, the optimization models and tools that we propose can only be used for offline studies. These offline studies are important because of the engineering insights they deliver, which may provide guidelines for practical designs.

The remaining of our paper is organized as follows. In Section II, we briefly review the theory of CS and describe how we model CS from a networking standpoint. We then formulate the three problems for non-CS, plain-CS and hybrid-CS in Section III. In Section IV, we show the numerical results obtained from solving the problems for WSNs with a grid topology. Finally, we conclude the paper in Section V.

[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 reduce the volume of the data to be sent out.

II. Overview of Compressed Sensing

In this section, we first briefly introduce the basic theory of CS, and then we explain in detail how we believe CS could be applied as an in-network data aggregation mechanism for WSNs.

A. Compressed Sensing Basics

As a novel sensing/sampling paradigm, CS theory asserts that one can recover certain signals from far fewer samples than what have been acquired from the sensors, if those signals can be sparsely represented in a proper basis [6]. Let us illustrate the idea in a WSN scenario. Assume a WSN of n nodes, each node acquiring a sample (e.g., humidity) x_i. The ultimate goal of the WSN is to collect the vector $\mathbf{x} = [x_1, \cdots, x_n]^T$ at the sink. We say $\mathbf{x}$ has an *m-sparse representation* if there exists a proper basis $\Psi = [\psi_1, \ldots, \psi_n]^T$, s.t. $\mathbf{x} = \sum_{i=1}^{m} z_i \psi_i$ and $m \ll n$. Now the CS theory suggests that, under certain conditions, instead of collecting $\mathbf{x}$, we may collect $\mathbf{y} = \Phi\mathbf{x}$, where $\Phi = \{\phi_{j,i}\}$ is a $k \times n$ "sensing" matrix whose entries are i.i.d. zero-mean random variables with variance $\frac{1}{k}$. Consequently, we can recover $\mathbf{x}$ from $\mathbf{y}$ by solving the convex optimization problem ($\|\mathbf{z}\|_{\ell_1} = \sum_i |z_i|$)

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

and letting $\mathbf{x} = \Psi\hat{\mathbf{z}}$, with $\hat{\mathbf{z}}$ being the optimal solution of (1). The condition that guarantees the correctness of this recovery is given by

$$k \geq C \cdot m \cdot \log n \tag{2}$$

where C is some small constant. In particular, as suggested by the "four-to-one" practical rule introduced in [6], $k = 4m$ is generally sufficient. Now, the meaning of "compressed" is pretty clear: the sink needs to collect only $k \ll n$ samples (as $m \ll n$ by assumption and $k \approx m$) to reconstruct the sensory data represented by the n samples.

According to the description above, Φ and Ψ are two keys to applying CS in WSNs. Using pseudo-random number generators to produce the entries of Φ, we can meet the i.i.d. criterion while avoiding actually transmitting Φ by seeding the generators using publicly known numbers. For example, if we associate a specific generator (a publicly known algorithm and its seed) with a node i, the i-th column of Φ, $\boldsymbol{\phi}_i$, can be generated anywhere with consistent output. In particular, the sink needs to store all the n seeds, such that it can generate Φ in order to process the compressed data. Although the Ψ that yields the sparsest representation of $\mathbf{x}$ may not be known, *wavelets* are in general considered as a good candidate for Ψ, as explained in [7]. For more advanced topics on CS theory, we refer readers to [6], [7] and the references therein.

B. Compressed Sensing as Data Aggregation

In the remaining of our paper, we adopt a high level model of CS. In particular, we call $\rho = \frac{n}{k}$ the *compression ratio* of CS. We assume ρ is constant (and known) as far as $n \geq n_{\min}$. Given the de facto four-to-one rule discussed above, our underlying assumption is that, as far as $n \geq n_{\min}$, the sparsity index m is proportional to the dimension n of a data vector. Also, the knowledge of the constant ρ (for the given n-dimensional data vector) can be obtained from past statistics on the sensory data. Of course, these intuitive simplification assumptions deserve further validation in the future.

We use Fig. 1 to illustrate the idea of CS-based data aggregation as compared to conventional data collection (called *non-CS* in the following). For the non-CS shown in Fig. 1(a), a node receiving $s-1$ packets (each packet corresponding to a data sample from a node, and the value of s depending on routing) will send out s packets (the $s-1$ received packets plus its own data sample); the sink, in particular, will need to receive all the n samples. Hence in the non-CS scenario, the load of a node is typically higher the closer it is from the sink.

CS-based network operation differs a lot from the non-CS case. Using CS, the sink needs only to receive k packets instead of n. Obviously, in order to use CS, each node i needs to know the value of n, i.e., how many nodes participate in the aggregation (could be the whole WSN or a subnet if a partition has been performed, as we will explain later) and the value of ρ. From these two values, it computes $k = \frac{n}{\rho}$ and generates locally k values $\phi_{j,i}$ ($1 \leq j \leq k$). It then creates a vector $x_i[\phi_{1,i}, \cdots, \phi_{k,i}]^T$, where x_i is its own sensory data. Typically, node i will wait to receive from all its downstream neighbors (i.e., i's neighbors that have been specified by routing to forward their data to the sink through i) all the data they have to send before starting transmitting. Each received packet is an element of a column vector of size k similar to $x_i[\phi_{1,i}, \cdots, \phi_{k,i}]^T$ and it carries its index from 1 to k so that it can be **added** to the data already waiting in i with the same index (either locally produced or received from a downstream neighbor). Then node i will send exactly k packets corresponding to the aggregated column vector. As long as routing is done to avoid duplications of packets, the sink will receive exactly k aggregated packets that it will transform back into a column vector $\mathbf{x}$.

Now the difference between CS and non-CS operations becomes clear: CS operation requires each node in the WSN to send exactly k packets irrespective of what it has received, which means, compared with non-CS, more work/load for the nodes far away from the sink and less work/load for the nodes close to the sink; the latter were, along with the sink, the main bottleneck of the non-CS data collection. As our studies are concerning offline network dimensioning, we assume a reliable network, i.e., a network without packet losses. We leave the issue of coping with unreliable links for future studies.

Note that it is not wise to perform the above CS operation on the whole network since $k = \frac{n}{\rho}$ may still be quite large. This suggests that we should **partition** the WSN into subnets and perform the CS operation independently in each subnet. In addition, the easiest way to avoid duplication is to perform routing on a spanning tree with the sink as the root. Now there are clearly many ways to select such a spanning tree. For example, given a WSN whose sink has a degree of δ, a spanning tree may consist of δ subtrees, each rooted at one of

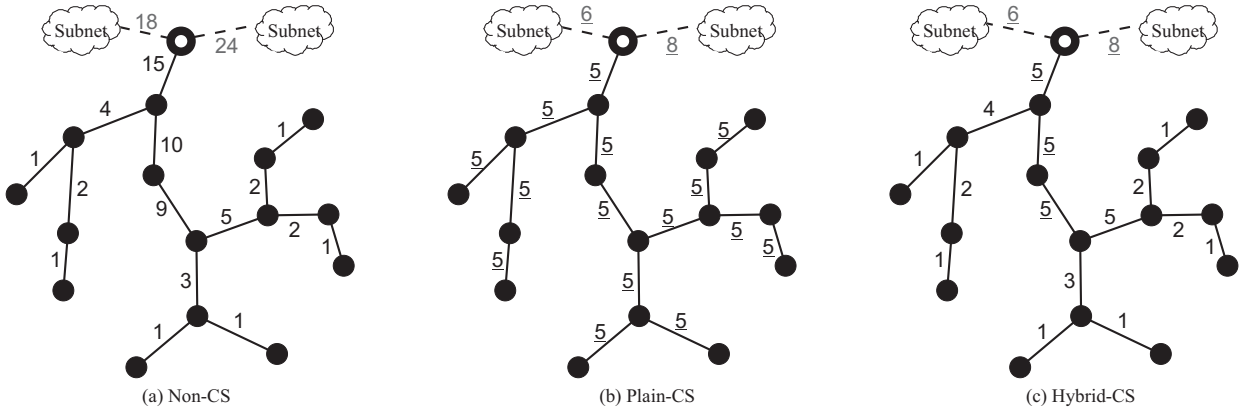


Fig. 1. Comparison of different data collection mechanisms. The link labels correspond to the carried traffic. The underlined labels indicate CS encoded traffic, assuming $\rho = 3$. The sink is represented by the hollow circle.

the δ children and having the same number of nodes $\tilde{n} = \frac{n}{\delta}$ (of course building such a partition may not always be possible). To apply CS on each subtree we need $\tilde{n} \geq n_{\min}$. In that case, each subtree is responsible for sending $\tilde{k} = \frac{\tilde{n}}{\rho} < \frac{n}{\rho}$ packets and the sink will then receive in total k packets as before, but intermediate nodes in the network will take a much lower load ($\tilde{k} = \frac{k}{\delta}$ instead of k). We are not claiming that the best solution is necessarily to create a balanced partition of δ subnets even though we suspect that probably it is often true.

In the problem formulation described later, we will partition the network into disjoint subnets and let individual subnets aggregate data samples independently of the other subnets. Such a partition is valid as far as the size of each subnet is not smaller than $n_{\min}$. We illustrate an unbalanced partition (in particular on one of the subtrees) in Fig. 1(b). Note that whereas the CS operation (along with routing) is done independently on each subnet, the link scheduling should still be performed globally, as the interference generated by a link (no matter which subtree it belongs to) has a global impact on the rest of the network. This makes our optimization problems hard to solve even if we assume that the routing is determined by predefined subtrees.

III. System Model and Problem Formulation

In this section, we first define the models for various components of a WSN, then we present the problem formulations.

A. Models and Assumptions

We model the WSN as a set $\mathcal{N}$ of nodes, with $|\mathcal{N}| = n$, and a sink Θ. Each node $i \in \mathcal{N}$ is associated with a geographical location. We assume that (i) all nodes send sensory data (through multihop routing if necessary) to the sink with the same rate λ, (ii) time is slotted and all the nodes are synchronized, and (iii) the network is operated in a conflict-free and scheduled manner.

We assume that all the nodes have the same *transmit power* P_{tx} and the same *data-rate* c. This is only for ease of exposition; our approach does accommodate multiple powers and rates. We assume that the channel gain from a node i to another node j is quasi-static, since we consider fixed wireless networks. For simplicity, we model the channel gain as isotropic path-loss given by $(\frac{d_{ij}}{d_0})^{-\eta}$ where d_{ij} denotes the *distance* from node i to node j, d_0 is the *near-field crossover distance* and η is the *path-loss exponent*. The feasibility of a wireless link is based on whether a bit-error-rate (BER) less than a tolerable maximum can be achieved on the link. We assume that this BER requirement translates into a minimum SINR (signal-to-interference-and-noise ratio) requirement corresponding to an SINR threshold β. We define $\mathcal{L}$ as the set of all feasible links. Specifically, a link $l = (i, j)$ is feasible (or $l \in \mathcal{L}$) if $\frac{P_{\text{tx}}}{N_0}(\frac{d_{ij}}{d_0})^{-\eta} \geq \beta$ where N_0 is the *thermal noise power* in the frequency band of operation. Let $|\mathcal{L}| = L$, and let l_O and l_D denote the origin and destination of link l, respectively.

We use the following SINR-based interference model. Let $\boldsymbol{\zeta} \subset \mathcal{L}$ denote a set of links. When all the links in $\boldsymbol{\zeta}$ are simultaneously active, the SINR perceived by link $l \in \boldsymbol{\zeta}$ is given by

$$\gamma_l(\boldsymbol{\zeta}) = \frac{P_{\text{tx}}(\frac{d_{l_O l_D}}{d_0})^{-\eta}}{N_0 + \sum_{k \in \boldsymbol{\zeta} \setminus \{l\}} P_{\text{tx}}(\frac{d_{k_O l_D}}{d_0})^{-\eta}} \tag{3}$$

We say a set of links $\boldsymbol{\zeta}$ is an *independent set* (ISet) if no two links share the same node and, for every link $l \in \boldsymbol{\zeta}$, we have $\gamma_l(\boldsymbol{\zeta}) \geq \beta$. It is clear that all the links belonging to an ISet can be scheduled at the same time in a conflict-free fashion. We define $\mathcal{I}$ to be the collection of all ISets

$$\mathcal{I} = \{\boldsymbol{\zeta} | \gamma_l(\boldsymbol{\zeta}) \geq \beta, \ \forall \ l \in \boldsymbol{\zeta}\} \tag{4}$$

Let $\mathcal{I}_l$ denote the set of ISets that contain link l. We use the SINR-based interference model rather than other more frequently used ones (e.g., protocol model) simply because it is more realistic [12].

Let $\mathcal{S}$ denote the power set of $\mathcal{L}$. A *transmission schedule* is an $|\mathcal{S}|$-dimensional vector $\hat{\boldsymbol{\alpha}} = [\alpha_{\boldsymbol{\zeta}}]_{\boldsymbol{\zeta} \in \mathcal{S}}$, and we can interpret $\alpha_{\boldsymbol{\zeta}}$ as the fraction of time allocated to a link set $\boldsymbol{\zeta}$. To make a schedule conflict-free, we need $\alpha_{\boldsymbol{\zeta}} > 0$ only if the set $\boldsymbol{\zeta}$ is an ISet (otherwise $\alpha_{\boldsymbol{\zeta}} = 0$) and $\sum_{\boldsymbol{\zeta} \in \mathcal{I}} \alpha_{\boldsymbol{\zeta}} \leq 1$. Therefore,

a *conflict-free transmission schedule* is an $|\mathcal{I}|$-dimensional vector $\boldsymbol{\alpha} = [\alpha_{\boldsymbol{\zeta}}]_{\boldsymbol{\zeta}\in\mathcal{I}}$.

B. Throughput Maximization for The Case without CS

In the non-CS data collection, we formulate our joint routing and scheduling problem as a regular flow optimization problem where we can assume that there is only one **many-to-one** flow ending at the sink. We define the link-set incidence matrix $Q = \{q_{l,\zeta}\}_{l\in\mathcal{L},\zeta\in\mathcal{I}}$

$$q_{l,\zeta} = \begin{cases} 1 & \text{if } l \in \zeta \in \mathcal{I} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Note that each column $\mathbf{q}_\zeta$ of Q is a vector that represents an ISet ζ and that the number of columns is $|\mathcal{I}|$ which is generally very large. We also define the standard node-arc incidence matrix $A = \{a_{i,l}\}_{i\in\mathcal{N},l\in\mathcal{L}}$

$$a_{i,l} = \begin{cases} +1 \text{ or } -1 & \text{if } i = o(l) \text{ or } i = d(l) \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Let r_l be the amount of flow going over a link l, and denote by $\mathbf{r} = [r_1, \cdots, r_L]^T$ the associated link flow vector. Given the network model and the definitions, we want to maximize the throughput of the many-to-one flow, which is formulated in the following:

$$\max_{\mathbf{r},\boldsymbol{\alpha}\geq\mathbf{0}} \quad \lambda \tag{7}$$

$$A\mathbf{r} \quad \geq \quad \lambda\mathbf{1} \tag{8}$$

$$c\cdot Q\boldsymbol{\alpha} \quad \geq \quad \mathbf{r} \tag{9}$$

$$\boldsymbol{\alpha}^T\mathbf{1} \quad \leq \quad 1 \tag{10}$$

where $\mathbf{1} = [1, 1, \cdots, 1]^T$. The problem explicitly maximizes flow rate λ over all possible link flows $\mathbf{r}$ (or equivalently routing) and link schedules $\boldsymbol{\alpha}$. Note that involving both the flow conservation constraint (8) and the scheduling constraints (9,10) yields a very general formulation that jointly takes routing and scheduling into consideration. We will solve this problem **exactly** in the following. Due to the space constraint, we omit the discussions on how to cope with the huge dimension of the matrix Q. Interested readers are referred to [8], [9] for details. Note that the solution of this problem can be viewed as an upper bound of the throughput achievable by a WSN under non-CS case using a random access MAC and/or a fixed routing.

As shown in Fig. 1(a), the non-CS data collection puts an increasing load on nodes closer to the sink, as they have to relay more data than those that are further. Consequently, the bottlenecks in terms of throughput are usually located at those "last-hop" nodes. Note that we use this scenario **only** to illustrate the differences between the non-CS data collection and CS-based data aggregations; we are not claiming that the solutions for any problem formulated in this paper yield necessarily the kind of partitions shown in the figure.

C. Throughput Maximization with Plain CS

As already described in Sec. II-B, one straightforward (but naive) way of applying CS is the following: upon acquiring a sensory data x_i, node i generates a random vector $\boldsymbol{\phi}_i$ and sends $x_i\boldsymbol{\phi}_i + \sum_{j\in\Pi_i} x_j\boldsymbol{\phi}_j$, where Π_i is the set of downstream neighbors of i. In the best possible case, the transmissions are sequenced so that a node h hops away from the sink has received data from all its downstream (i.e., $h+1$ hops) neighbors before encoding and sending the data. Translated into a flow model, this yields a totally egalitarian load allocation, as every link carries the same flow rate $k\lambda$, where k, as explained in Sec. II, is the dimension of $\boldsymbol{\phi}_i$. We illustrate this idea by Fig. 1(b), where $k = 5$ (as $n = 15$ for the represented subnet and $\rho = 3$ by assumption).

We will assume single-path routing in the following even if it is not strictly a requirement. This effectively means that the routing optimization should be done by choosing the best partition and then the optimal spanning tree for each subnet of a WSN. Let $\mathcal{T}$ be a *node disjoint tree cover* of $\mathcal{N}$. Let $T_i \in \mathcal{T}$ be a tree rooted at a node i that is one-hop from the sink Θ. Let $\hat{\mathcal{T}}$ be the *extended tree cover* such that, for $\hat{T}_i \in \hat{\mathcal{T}}$, $V(\hat{T}_i) = V(T_i) \cup \{\Theta\}$ where $V(T)$ is the set of vertices of tree T and $E(\hat{T}_i) = E(T_i) \cup \{(i, \Theta)\}$ where $E(T)$ is the set of edges of tree T. We apply CS to each $\hat{T}_i$ with an identical ρ, which can only be done if $n_i \geq n_{\min}$ where $n_i = |V(T_i)|$. In other words, for $l \in E(\hat{T}_i)$, the traffic load is $\frac{n_i}{\rho}\lambda$. Now the throughput maximization becomes

$$\max_{\boldsymbol{\alpha}\geq\mathbf{0};\ \mathcal{T}:n_i\geq n_{\min},\ \forall T_i\in\mathcal{T}} \quad \lambda \tag{11}$$

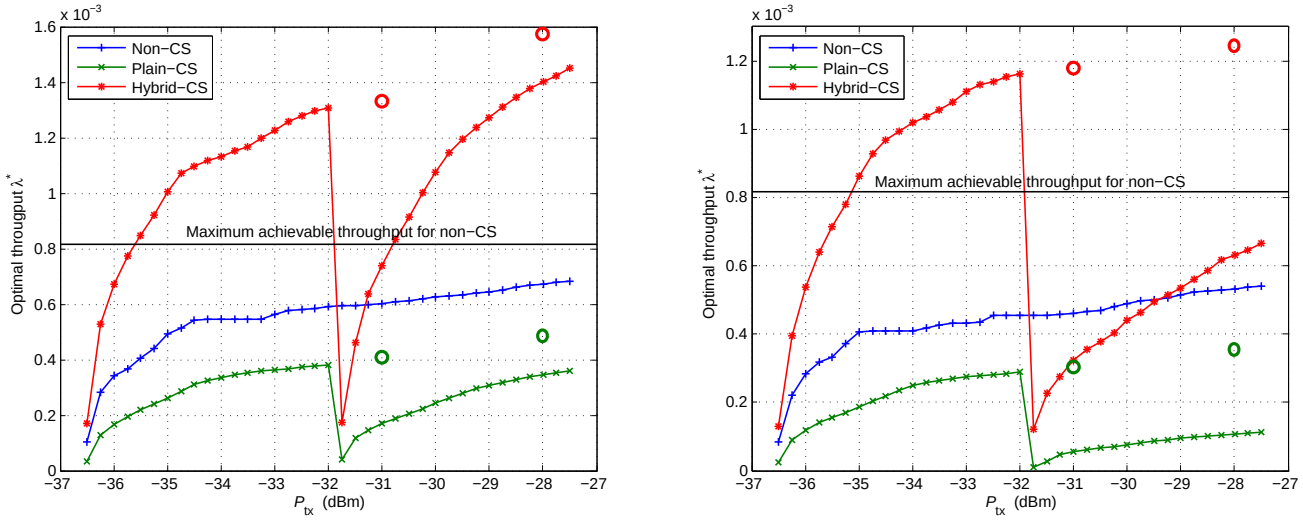
$$c\sum_{\zeta\in\mathcal{I}} q_{l,\zeta}\alpha_\zeta \quad \geq \quad \frac{n_i}{\rho}\lambda \qquad \begin{cases} \forall\ l \in E(\hat{T}_i) \\ \forall\ \hat{T}_i \in \hat{\mathcal{T}} \end{cases} \tag{12}$$

$$\boldsymbol{\alpha}^T\mathbf{1} \quad \leq \quad 1 \tag{13}$$

Here we put $\mathcal{T}$ as the routing optimization variable, which replaces the flow conservation (8) used for the non-CS case. The link load $k_i\lambda$ on the RHS of (12) depends on the size n_i of the sub-tree T_i (whose extension $\hat{T}_i$ includes the link l under consideration), as $k_i = \frac{n_i}{\rho}$. By this we mean to jointly search for the optimal tree cover and the optimal scheduling upon it. Unfortunately, the tree cover problem is in general very hard [13]. Therefore, we later solve the problem by fixing a "good" tree cover and obtain the optimal schedule upon it. This gives us a feasible solution that can be seen as a **lower bound** on the optimal solution for problem (11-13).

D. Throughput Maximization with Hybrid CS

It can be observed that directly applying CS in the way suggested in Sec. III-C creates too much load in the earlier stages of the data collection. More precisely, starting coding right from the leaf nodes of a tree might be counter-productive as they have to transmit k packets for each data sample as opposed to one packet in the non-CS case. Hence we propose a *hybrid-CS* policy to improve the performance: the non-CS scheme is applied in the earlier stages of the data collection (starting from the leaf nodes), and the CS-based



(a) Grid network with a sink at the center.

(b) Grid network with a sink at one corner.

Fig. 2. Throughput comparison among non-CS, plain-CS, and hybrid CS.

compression is only applied at a node whose incoming traffic intensity becomes larger or equal to $k\lambda$. This idea is explained by Fig. 1(c). Therefore, an intermediate node might have to encode multiple data samples coming from its child nodes. This will be done by generating locally the column vector ϕ's for each sample.

We continue using the terminology defined in Sec. III-C. Let $\Pi_{i,j}$ be the set of *child nodes* of node j (including j itself) in T_i. It is clear that, according to our hybrid-CS policy, the outgoing links of node j will carry a load of $\lambda \min\left(|\Pi_{i,j}|, \frac{n_i}{\rho}\right)$, where the first term refers to the flow conservation in the non-CS data collection, and the second term indicates the use of CS to encode the data. Now the throughput maximization is slightly different from (11–13):

$$\max_{\boldsymbol{\alpha}\geq\mathbf{0};\ \mathcal{T}:n_i\geq n_{\min},\ \forall T_i\in\mathcal{T}} \quad \lambda \tag{14}$$

$$c\sum_{\zeta\in\mathcal{I}} q_{l,\zeta}\alpha_\zeta \quad \geq \quad \min\left(|\Pi_{i,l_O}|, \frac{n_i}{\rho}\right)\lambda \quad \begin{cases}\forall\ l\in E(\hat{T}_i)\\ \forall\ \hat{T}_i\in\hat{\mathcal{T}}\end{cases} \tag{15}$$

$$\boldsymbol{\alpha}^T\mathbf{1} \quad \leq \quad 1 \tag{16}$$

The optimal solution can be obtained by jointly searching for the optimal tree cover and the optimal scheduling upon it, which is at least as hard as solving problem (11–13). Therefore, we, again, solve the problem later by decoupling the tree cover from the scheduling upon it. This gives us a feasible solution that can be seen as a **lower bound** on the optimal solution for problem (14-16). According to Fig. 1(c), it is clear that hybrid-CS combines the advantage of both non-CS and plain-CS: it gets rid of the excessive load at leaf nodes by applying non-CS at an earlier stage of the data collection, and it uses CS to reduce the load carried by the last-hop bottleneck.

IV. Numerical Results

We assume a grid network of 1225 (35×35) nodes (including the sink), and we consider two cases differing in terms of the position of the sink: one at the center and another at one corner. We assume that the distance between two closest nodes is 8m. For radio propagation, we assume $N_0 = -100$dBm, $d_0 = 0.1$m, and $\eta = 3$. Also, we fix the rate $c = 1$, take $\beta = 6.4$dB, and we investigate the optimal throughput as a function of P_{tx}. The computations for all the cases start at the transmit power $P_{\min}$ that just allows the network to be connected. Note that, although each value of $P_{\text{tx}} \geq P_{\min}$ may not yield a new set of links (hence a new topology), it might produce new ISets. Let the degree of the sink be $\delta(P_{\text{tx}})$ for a given P_{tx}.

For the cases with CS, we need to create a partition so that each subnet has at least $n_{\min}$ nodes. This means that when P_{tx} increases, even if $\delta(P_{\text{tx}})$ increases, the number of subnets in the partition will not grow beyond $\frac{n}{n_{\min}}$. For our current computations, we use Dijkstra's algorithm to generate the tree cover $\mathcal{T}$ (which **decouples** routing from link scheduling), and we set $\rho = 5$. It is trivial to see that, as P_{tx} keeps increasing, the sink will have more and more one-hop neighbors and hence there will be more trees in a tree cover. We do not allow a partition in which one of the subtrees becomes too small, i.e, $n_i < n_{\min} = 150$. We have limited our computations to low values of P_{tx} since our computational tool requires a memory space larger than what a 32-bit program can handle (recall that the joint scheduling problem is on the whole network that has 1225 nodes and thus there is a huge set of potential ISets when P_{tx} is large). We have started to upgrade our tool to 64-bit to allow us to tackle larger transmit powers.

The results for both cases are shown in Fig. 2. The black lines in both figures show the maximum achievable throughput of non-CS data collection if we keep increasing P_{tx} until

every node has a direct link to the sink. It is clear from the comparisons that plain-CS might not yield higher throughput than non-CS, but hybrid-CS will definitely bring a significant improvement (apart from certain points that we will discuss later). One may argue that it is not fair to compare a lower bound on the throughput of plain-CS to the optimal throughput of non-CS. Although this is true, what our preliminary results illustrate is the importance of performing CS carefully if **substantial** throughput gains need to be achieved. Of course, increasing ρ will improves the performance of both plain-CS and hybrid-CS. However, assuming $\rho = 5$ implies that $m \leq \frac{k}{4} = \frac{n}{20}$, which is already quite sparse for realistic sensory data set. Therefore, we are not expecting a significant increase in ρ.

The figures also show one major problem with using Dijkstra's algorithm to generate $\mathcal{T}$: it always seeks the min-hop routing greedily. Whenever new links are created due to a transmit power increase, the algorithm will produce a tree cover using these new links. Unfortunately, as these links have just been created, their SNRs are too low to allow spatial reuse with other links. This explains the big "falls" in the curves at $P_{\text{tx}} = -31.75$dBm: they correspond to the transmit power where the first "diagonal" links are created. Note that, at the maximum power shown in the figures (i.e., -27.5dBm), only links between closest neighbors and across the shortest diagonal are feasible. In order to show that the "falls" were due to the way we constructed the tree cover and not to an inherent problem with either the CS schemes or the particular network topology, we computed the performance of the plain-CS and hybrid-CS schemes at powers greater than -32dBm by fixing $\mathcal{T}$ to be the one computed for $P_{\text{tx}} = -32$dBm (i.e., at a power level that does not make the shortest diagonal links feasible). Clearly while we have a fixed $\mathcal{T}$, we are still allowing the scheduling to take full advantage of a higher transmit power that may create new ISets. As expected, the performances with a fixed $\mathcal{T}$ (represented by the circles in the figures) do not exhibit any decrease. For an arbitrary network topology, we expect the same phenomenon to persist as far as we use Dijkstra's algorithm to generate $\mathcal{T}$; using a grid topology simply amplifies it and hence allows us (and the readers) to observe it easily.

V. Conclusion

In this paper, we investigate the benefit of applying compressed sensing (CS) to data collection in wireless sensor networks (WSNs). We first describe a naive way of applying CS called plain-CS, then we propose a hybrid-CS scheme that combines conventional data collection (non-CS) with plain-CS. We formulate and solve three flow-based optimization problems that characterize the throughput under the three schemes. The most important insights we acquired from this study are: (i) applying CS naively may not bring any improvement, and (ii) our hybrid-CS can achieve significant improvement in throughput as compared with non-CS. Our preliminary numerical results are only for a low-power regime.

It will be part of our future work to develop better *partitioning strategies* for our hybrid-CS scheme. Also, we need to validate some of the assumptions made in the paper using realistic sensory data. Moreover, we will be looking at the combination of the physical layer CS (e.g., [11]) and our networking layer CS to further improve the throughput of WSNs. Finally, we will also study how CS could benefit other performance metrics of WSNs, such as lifetime and delay.

References

[1] 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.

[2] J. Gao, L. Guibas, N. Milosavljevic, and J. Hershberger, "Sparse Data Aggregation in Sensor Networks," in *Proc. of the 4th ACM IPSN*, 2007.

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

[4] 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.

[5] 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.

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

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

[8] J. Luo, A. Girard, and C. Rosenberg, "Efficient Algorithms to Solve a Class of Resource Allocation Problems in Large Wireless Networks," in *Proc. of the ICST WiOpt*, 2009.

[9] J. Luo, C. Rosenberg, and A. Girard, "Engineering Wireless Mesh Networks: Joint Scheduling, Routing, Power Control and Rate Adaptation," *IEEE/ACM Trans. on Networking (to appear)*, 2010, http://www3.ntu.edu.sg/home/junluo/documents/TMPAlgo.pdf.

[10] M. Rabbat, J. Haupt, A. Singh, and R. Nowak, "Decentralized Compression and Predistribution via Randomized Gossiping," in *Proc. of the 3th ACM IPSN*, 2006.

[11] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive Wireless Sensing," in *Proc. of the 3th ACM IPSN*, 2006.

[12] A. Iyer, C. Rosenberg, and A. Karnik, "What is the Right Model for Wireless Channel Interference?" *IEEE Trans. on Wireless Communications*, vol. 8, no. 5, 2009.

[13] G. E. amd N. Garg, J. Könemann, R. Ravi, and A. Sinha, "Min-Max Tree Covers of Graphs," *Elsevier Operations Research Letters*, vol. 32, no. 4, 2004.