Pervasive and Mobile Computing 22 (2015) 113–125

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

# Adaptive compressive sensing based sample scheduling mechanism for wireless sensor networks

Jie Hao [a], Baoxian Zhang [a,*], Zhenzhen Jiao [a], Shiwen Mao [b]

[a] Research Center of Ubiquitous Sensor Networks, University of Chinese Academy of Sciences, Beijing 100049, China

[b] Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201, USA

ARTICLE INFO

*Article history:*
Available online 14 February 2015

*Keywords:*
Compressive sensing
Sample scheduling
Energy efficiency
Wireless sensor network

ABSTRACT

Sample scheduling is a crucial issue in wireless sensor networks (WSNs). The design objectives of efficient sample scheduling are in general two-folds: to achieve a low sample rate and also high sensing quality. Recently, compressive sensing (CS) has been regarded as an effective paradigm for achieving high sensing quality at a low sample rate. However, most existing work in the area of CS for WSNs use fixed sample rates, which may make sensor nodes in a WSN unable to capture significant changes of target phenomenon, unless the sample rate is sufficiently high, and thus degrades the sensing quality. In this paper, to pursue high sensing quality at low sample rate, we propose an adaptive CS based sample scheduling mechanism (ACS) for WSNs. ACS estimates the minimum required sample rate subject to given sensing quality on a per-sampling-window basis and accordingly adjusts sensors' sample rates. ACS can be useful in many applications such as environment monitoring, and spectrum sensing in cognitive sensor networks. Extensive trace-driven experiments are conducted and the numerical results show that ACS can obtain high sensing quality at low sample rate.



## 1. Introduction

Wireless sensor networks (WSN) can be deployed to monitor physical phenomena, e.g., primary user signals in a cognitive sensor network. In such networks, massive sensor nodes work to sense their surrounding environments and report their sensed data to a fusion center. Sensor nodes are usually powered by batteries that are energy limited. Since all the sensor nodes are targeted to monitor the same physical phenomena and the sensor nodes are usually deployed in a sufficient high density to guarantee high sensing quality, spatial correlation among the sensing measurements from neighboring sensor nodes is highly expected. In addition, for each individual sensor node, temporal correlation usually exists among its measurements since its monitored physical phenomenon usually changes continuously. Such spatial and temporal correlations have been exploited in various WSN technologies (e.g., spectrum sensing in a cognitive sensor networks, data aggregation and compression, route selection, clustering, and etc.) while meeting certain sensing quality requirements.

To this end, compressive sensing (CS) allows a sparse analog signal to be represented by much fewer samples than that required by the Nyquist sampling theorem [1]. In other words, for a sparse signal consisting of $N$ samples, only a small number $M \ll N$ of encoded samples, generated by using the original $N$ samples and properly chosen transform coefficients, are needed to be reported and used to recover the original signal at the sink side. The mapping matrix from the original

* Corresponding author.
*E-mail address:* bxzhang@ucas.ac.cn (B. Zhang).

http://dx.doi.org/10.1016/j.pmcj.2015.02.002
1574-1192/

$N$-sample signal to the $M$-sample signal is called measurement matrix or projection matrix. The ratio between $M$ and $N$ is called the sample rate and the degree of consistency between the recovered signal and the original signal is called recovery quality or sensing quality. The objective of CS design is to use the minimum sample rate to recover the original signal subject to given recovery quality. Three key factors are involved in the CS design: a representation basis $\Psi$ to represent the interested signal in a sparse form, a measurement matrix $\Phi$ to transform the original $N$-sample signal into an $M$-sample signal with efficient transform coefficients and a recovery algorithm to recover the original signal. The recovery quality depends on all these three factors and the properties of the signal per se. It has been proven that the sparser the signal as represented by the representation basis, the better the recovery quality; the higher the incoherence between the representation basis and measurement matrix, the better the recovery quality. In general, almost all existing CS work (see [2–24]) focuses on the sophisticated design of the representation basis, measurement matrix, and recovery algorithm to pursue high recovery quality.

For phenomena monitoring applications using WSNs, traditionally, sensor nodes are often required to take samples according to a predetermined rate (e.g., once per minute). According to CS, each sensor node is allowed to sample the environment at a much lower sample rate than the predetermined rate without sacrificing the conformance between the actual phenomenon and the sensing reading collected. To maintain conformance, the sample rate should be tuned adaptively as the phenomenon changes. However, most existing work in the field of CS based sample scheduling for WSNs assume fixed sample rate, which may degrade the compression performance under dynamic phenomenon. To pursue stably high sensing quality with a sample rate as low as possible, it is critical to investigate adaptive CS schemes to trade-off the two performance goals.

An important application of CS is spectrum sensing in cognitive radio sensor networks. This is because channel availability is generally spatially and temporally correlated in such networks. Traditional spectrum sensing mechanisms requires each spectrum sensor to sense the licensed channels at the Nyquist sample rate for accurate spectrum sensing, which may lead to excess delay and waste of energy. With CS, each sensor node can perform spectrum sensing at a much lower sampling rate while still keeping high quality in spectrum sensing. Alternatively, spatial correlation in the spectrum sensor data can be exploited to turn off some spectrum sensors to preserve energy and extend the life time of the cognitive radio sensor network [19]. Since CS and also our adaptive CS mechanism proposed in this paper can work well for both phenomena monitoring and spectrum sensing, in the rest of this paper, we shall often take the application of phenomena monitoring as example when discussing mechanism implementation details although our design can also work well for spectrum sensing. In this sense, we treat "spectrum availability" as a phenomenon to be monitored.

In this paper, we propose an adaptive CS based sample scheduling mechanism (ACS) in order to achieve a low sample rate provided that a given sensing quality requirement is expected to be met. In ACS, each sensor node adjusts its sample rate on a per-checking-window basis. The implementation of ACS can be divided into two phases: Training phase and online phase. In the training phase, ACS needs to pre-collect certain amount of original sensing data (at high sample rate) for the phenomenon of interest, based on which it can build a hash table that empirically reflects the relationship between sample rate required for meeting a given desired sensing quality requirement and a sparsity degree (or change intensity). In the online phase, each sensor can decide its sample rate on per checking window basis. More specifically, it can decide its sample rate in the next checking window based on the signal sparsity degree or change intensity in the current checking window and also the hash table built during the training phase.

We accordingly propose the detailed design of ACS. The major contributions in this paper are as follows. First, we propose an adaptive sample scheduling mechanism to overcome the drawback that fixed sample rate compressive sensing mechanisms can fail to quickly react to significant phenomena change unless the sample rate is excessively high. We further present the detailed design of ACS based on signal sparsity and change intensity, respectively. Third, we conduct extensive numerical experiments using real data trace to validate the performance of ACS. The results demonstrate that ACS can achieve high performance as compared with existing work. That is, ACS can achieve desired sensing quality by much lower sample rate than existing mechanisms. Furthermore, experiment results show that each node independently adjusts its sample rate can achieve comparatively high performance as compared with collaborated sample control at different scales such as cluster-based or network-based. Largely reduced sample rate can largely reduce the energy consumed for environmental sampling and also for wireless communications, which makes ACS attractive for energy-constrained WSNs.

The reminder of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 introduces necessary preliminaries for this work and formulates the problem to be addressed. Section 4 proposes the ACS mechanism. Section 5 presents the numerical experiments to demonstrate the high performance of ACS. Section 6 concludes this paper.

## 2. Related work

Recently, CS has been applied to WSNs due to its high recovery quality. Fig. 1 provides a typical example illustrating how CS can be utilized to design transmission schedule in a WSN. In Fig. 1, all nodes form a chain topology where the packets $p_1, p_2, \ldots, p_N$ as generated by nodes $s_1, s_2, \ldots, s_N$, respectively, need to be transmitted to the sink node in the network. Using traditional transmission schedule, each node needs to separately transmit its own packet and all the packets of its downstream nodes (i.e., those nodes away from the sink node). As a result, the sink receives $N$ uncoded packets, $N(N+1)/2$ transmissions in total need to be carried out in the network and the closer a node to the sink, the more energy it will consume. By introducing CS as shown in Fig. 1, the $N$ raw packets can be represented by $M$ ($M \ll N$) encoded packets

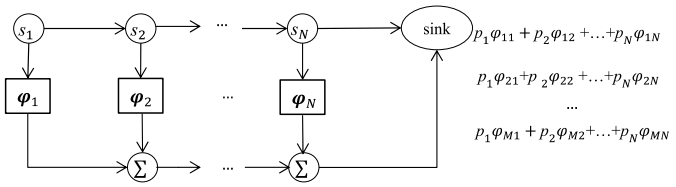


**Fig. 1.** A transmission schedule using CS.

and then recovered at the sink node with high probability. Each packet $p_i$ is weighted by $\boldsymbol{\varphi}_i = \{\varphi_{1i}, \varphi_{2i}, \ldots, \varphi_{Mi}\}$, where $\boldsymbol{\varphi}_i$ is the $i$th entry of measurement matrix $\Phi_{M\times N}$, $1 \leq i \leq N$. Accordingly, the sink node will receive $M$ encoded packets and thus only $MN$ transmissions in total are needed, and more important, all network nodes consume the same amount of energy for transmissions. In addition, CS is tolerant to transmission losses/failures since only a sufficient number of packets are needed for data recovery. Therefore, the CS-based transmission schedule has significant advantages in terms of energy consumption balance, energy efficiency, and transmission robustness. Owning to the above advantages, CS has been used in various sensor networks, such as data gathering sensor networks [2–5], cognitive radio sensor networks [19–22], MIMO based sensor networks [23,24], etc. We next introduce related work in these areas.

One major application of CS in WSNs is data gathering. For data gathering applications, most existing work introduces CS to obtain efficient transmission schedule in order to improve network capacity and throughput. CDG (Compressive Data Gathering) [2] focuses on snapshot data gathering. Instead of transmitting the original data directly, each node along a pre-built delivery structure multiplies its own data with a random coefficient in a way such that the sink can obtain weighted sums of all the original packets and then recover them. In CDG, the measurement matrix is randomly generated and the signal is sparsified by using DCT (Discrete Cosine Transform) or wavelet transform. CDG is not suitable for small-scale sensor networks where signal sparsity may not be prominent enough and hence the potential capacity gain will be small. Ji et al. [3] studied snapshot and continuous data gathering problem under Physical Interference Model. In order to pursue improved network capacity, snapshot data gathering uses network partition to schedule the CS process of each sub-network and hence order-optimal network capacity can be achieved; continuous data collection uses a pipeline scheduling algorithm to speed up the CS process in the network. Luo et al. [4] investigated the benefit of applying CS to data gathering tree in terms of network throughput. This hybrid CS avoids the excessive traffic load at leaf nodes and also takes advantage of CS to reduce the traffic load at those nodes near the sink so that hybrid CS can achieve significant improvement in terms of throughput. The CS based data gathering approach presented in [15] investigated the impact of sparse projection matrix generated by routing topology on the accuracy of the approximation. Compared with CS based on tree topology, CS based on clustered topology is demonstrated to have higher recovery quality and compression ratio [5], where Lee et al. analyzed the impact of clustering on the recovery quality and compression ratio and accordingly designed cluster-based CS mechanism. Most of the above work can utilize only spatial correlation, and they assume fixed CS schedule, which means the compression ratio is also fixed.

CS is also useful for spectrum sensing (referred to as "compressed spectrum sensing" in some work) in cognitive radio sensor networks. This is because the operations of cognitive ratio networks need continuous spectrum sensing. Based on the observation that wireless signals in open-spectrum networks are usually sparse in the frequency domain, Refs. [18–22] introduce CS to reduce the spectrum sensing frequency. Ref. [18] employs random sub-Nyquist-rate and also incorporates the wavelet based edge detector to recover the location of frequency bands. Ref. [19] seeks to balance the trade-off between energy saving and recovery accuracy in terms of the number of active sensor nodes and the corresponding estimation error. Ref. [20] first transforms the received analog signal to a digital signal using an analog-to-information converter instead of analog-to-digital converter and uses the autocorrelation of this compressed signal to reconstruct the signal spectrum. This approach can achieve accurate power spectrum density estimation and also the probability of detecting signal occupancy. Refs. [21,22] exploit the joint sparsity between multiple cognitive radios to further improve the sensing performance. Multiple cognitive radios enforce consensus among local spectral estimates of cognitive ratios and perform collaborative spectrum sensing. It is demonstrated that distributed compressive spectrum sensing leverages spatial diversity to alleviate wireless fading and thus has superior performance.

CS has also been used in channel coding, analog transmission, sample schedule design, and related areas. Here, we introduce some typical work in these areas. Compressive wireless sensing (CWS) [6] is designed for single hop networks and it combines CS design and amplitude modulated analog transmission to deliver linear projections of sensor readings. CWS exploits the advantage of CS in reducing the latency of data gathering in a single hop network. However, CWS is impractical for multi-hop sensor networks due to the difficulty in node synchronization at the multi-hop scale. Oversampled CS source coding [7] uses CS as an application layer erasure coding strategy for recovering missing data over erasure wireless channels. Wu et al. [8] focused on the design of CS-based sample schedule for soil moisture monitoring applications. In [8], the authors exploited CS based on temporal correlation at an individual sensor node and they aimed to obtain a sample schedule with a low sample rate while achieving comparable sensing quality by using a sufficiently high sample rate without CS. It is shown that a difference matrix and uniform measurement schedule can attain a good tradeoff between signal sparsity degree and incoherence. This CS design achieves high sensing quality. However, it uses a fixed sample rate and thus may lead to failure

in capturing significant change of its target phenomenon. Ref. [14] utilized principal component analysis (PCA) to capture spatial/temporal correlation of signals. The combination of PCA and CS is able to effectively self-adapt to unpredictable changes in the signal statistics by a feedback control loop that estimates the signal recovery error. Refs. [16,17] were targeted at improving energy cost balance. Ref. [17] exploited both compressibility and heterogeneity and designed a probabilistic sampling method to provide an accurate temporal–spatial profile for a given energy budget. Ref. [16] adjusted sensor node sampling workload according to solar energy availability.

From the above introduction, we have the following observations. Only a few existing CS mechanisms study the design of adaptive sample schedule from the perspective of energy consumption balance while most work mainly use a fixed sample rate which may lead to unsatisfied sensing quality (if choosing too low sample rate) or excessive resource consumption (if choosing too high sample rate). Also, different from existing work which compresses original data pre-collected, we focus on the scenario that sampling the target phenomenon with adaptive sample rate in real time without pre-collecting high resolution data. In this paper, we study how to achieve adaptive CS-based sample scheduling for phenomena monitoring or spectrum sensing using a WSN. The objective is to achieve a low sample rate while meeting give recovery quality as the monitored target change with time.

## 3. Preliminaries

In this section, we present some preliminaries of compressive sensing. Existing work in CS can generally be categorized into two types: SMV-based (single measurement vector) and MMV-based (multiple measurement vectors). SMV-based CS compresses one-dimensional signals and it is motivated by an observation that for an $N$-sample signal that is $K$-sparse (i.e., the signal can be formulated as a sum of $K$ incoherent basis functions from some known basis), only $K+1$ projections of the signal onto the incoherent basis are required to reconstruct the signal with high probability. However, obtaining the exact $K+1$ projections needs a combinatorial search which results in unacceptably high complexity. Fortunately, $M = cK$ (typically, $c = 3$ or 4) projections are sufficient for signal reconstruction with high probability, which has been proved to provide the same result to that by using combinatorial search [1]. MMV is an extension of SMV and is more frequently used in practice such as image compression and movie compression. One straightforward solution to MMV is to divide an MMV into multiple SMVs and then solve each SMV individually. However, in [9], Baron et al. proved that utilizing the correlation among MMVs in distributed compressive sensing can further reduce $M$ while keeping comparably high sensing quality.

First, we introduce CS based on SMV. Consider a discrete $K$-sparse signal denoted by vector $x_{N\times 1}$, we can reconstruct $x$ from vector $y_{M\times 1} = \Phi_{M\times N} x_{N\times 1}$ ($K < M < N$) if $x$ is sparse enough, i.e., $K = \|x\|_o \ll N$. $K/N$ is referred to as sparsity degree, $M/K$ is referred to as the oversampling rate, $M/N$ is referred to as the sample rate, $\Phi$ is referred to as the measurement matrix as well as the sample schedule matrix. However, in practice, a signal may not be sparse or sufficiently sparse. In this case, representation in an alternative domain is necessary to represent $x$, i.e., $x_{N\times 1} = \Psi_{N\times N} s_{N\times 1}$, where $\Psi$ is the representation basis and $s$ is sparse with $\|s\|_o \ll N$. The compressive sensing formula can be rewritten as

$$x_{N\times 1} = \Psi_{N\times N} s_{N\times 1}, \qquad y_{M\times 1} = \Phi_{M\times N}\Psi_{N\times N} s_{N\times 1}. \tag{1}$$

To recover the original signal with high probability is equivalent to finding the solution to Eq. (2).

$$\min_{\tilde{s}\in R} N\|\tilde{s}\|_o \quad \text{s.t. } y_{M\times 1} = \Phi_{M\times N}\Psi_{N\times N}\tilde{s}_{N\times 1} \tag{2}$$

where $\tilde{s}$ is the recovered signal in the representation basis $\Psi$ and $R^N$ represents $N$-dimensional real space. Let $\tilde{x}$ represent the corresponding recovered signal. There is also an alternative kind of approaches seeking to solve the $l_1$ norm minimization problem instead of the $l_0$ norm based problem in Eq. (2). To recover the original $x$ with high accuracy, elaborately chosen $\Phi_{M\times N}$, $\Psi_{N\times N}$, and the recovery algorithm need to be jointly evaluated. Many choices are feasible for CS. DTC (Discrete cosine transform) and wavelet are widely used for representation basis, Gaussian or pseudo random measurement matrices are widely used, and Basis Pursuit [10] and SL0 (Smoothed $l_0$) [11] are good choices for the recovery algorithm.

Specifically, regarding sample scheduling, vector $x$ is a series of expected sample measurements by a sensor node at a high sample rate, vector $y$ is the compressed measurements actually sensed and reported by the sensor node after using CS. Then, we have $y = \Phi_{M\times N} \times x$, where $\Phi_{M\times N}$ is the measurement matrix. To facilitate the understanding, we let the sample interval at $x$ be a basic unit of time. Please note that in compressed spectrum sensing, the basic unit of time is usually set to be the reciprocal of the Nyquist sampling rate. We say that $x$ samples at times $1, 2, \ldots, N$. The measurement matrix $\Phi_{M\times N}$ is a binary matrix. The element of $\Phi_{M\times N}$ at position $i \times j$ is either '1' or '0', where '1' means the sensor node takes its $i$th sample at time $j$ and '0' means not. There is only one '1' in each row and at most one '1' in each column. Let $S$ represent the resulting sample schedule, which is a row vector where each item is the sum of a corresponding column of $\Phi_{M\times N}$. The element of $S$ at position $j \times 1$ ($1 < j < N$) is either '1' or '0' where '1' means the sensor node samples once at time $j$ and '0' means not. With CS, a sensor node can take $M$ samples instead of $N$ samples since the $N$ samples can be recovered from the $M$ samples with high probability.

Please note that in the context of spectrum sensing, CS can be performed after the analog-to-digital converter or during analog sampling. The former still needs Nyquist-rate uniform sampling while the latter can reduce the analog-to-digital sampling rate directly. Our mechanism proposed in this paper can be applied to both cases, which makes our mechanism flexible.

In [8], Wu et al. compared several design choices for representation basis, measurement matrix, and recovery algorithm and claimed that the combination of representative basis $\Psi_D = M_D^{-1}$, measurement matrix $\Phi_U$, and recovery algorithm SL0 (Smoothed $l_0$) proposed in [11] is a good choice as compared with other combinations. To be more specific, $M_D$ is called difference matrix and is formulated as

$$M_D = \begin{bmatrix} -2 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}. \tag{3}$$

The design motivation of the representative basis is that the target phenomena usually stay stable in short time. $\Phi_U$ corresponds to uniform sample scheduling, i.e., with $\Phi_U$ measurements are taken once per $\lfloor N/M \rfloor$ units of time. The recover algorithm SL0 approximates the $l_0$ norm of a signal $s$ by a smooth function $F_\sigma(s)$, where $\sigma$ is a parameter determining the approximation quality. SL0 can quickly find a solution in $l_0$ norm by minimizing $F_\sigma(s)$ for a very small value of $\sigma$.

However, this design choice above has a drawback such that it may fail in capturing significant change of target monitoring phenomenon unless the sample rate is excessively high. For example, in environment monitoring application, humidity and moisture will dramatically change (increase) after rainfall, surface temperature may drop dramatically after sunset and etc. In cognitive radio sensor networks, the channel occupation may increase significantly in rush time but stay low during idle time. To address this problem, in this paper, we focus on maintaining desired stable recovery accuracy and aim to propose an adaptive CS based sample scheduling mechanism (ACS) which can quickly adjust sensors' sample rates according to the dynamics of monitored phenomenon.

## 4. Proposed ACS mechanism

In this section, we propose our ACS mechanism. ACS aims at only sampling the phenomena of interest at a very low sample rate while meeting expected sensing quality. First, we present the basic idea behind ACS. Second, we present the detailed design of ACS including its representation basis, measurement matrix design, and recovery algorithm. Third, we present the sample rate adjustment strategy in ACS.

### 4.1. ACS overview

We consider a network scenario wherein massive sensor nodes are deployed to monitor a target phenomenon in an interested area. The sensor nodes sample the phenomenon according to their own sample schedules and report their measurements to a sink node. In ACS, the sample schedules of sensor nodes are decided by using compressive sensing, where the measurements of all sensor nodes in the network form an MMV which can have high spatial and temporal correlations. In this section, we propose an efficient sample scheduling mechanism called ACS, which performs adaptive compressive sensing in order to achieve a low sample rate while meeting desired stably high sensing quality.

In ACS, each sensor node monitors its surrounding phenomenon at a sample rate which changes with the monitored phenomenon subject to given sensing quality. That is, each node estimates the change intensity (or sparsity degree) of its monitored phenomenon during a certain period of time called checking window whose size is denoted by $N$, decides on how to adjust its sample rate in the following checking window based on certain knowledge of target phenomenon obtained during a training phase. By adaptively adjusting the sample rates of sensor nodes, ACS achieves a low sample rate while meeting desired stably high sensing quality.

When using ACS for supporting compressive spectrum sensing, joint design of the analog-to-digital module and ACS can achieve optimized performance. However, in this paper, we focus ourselves on the sample rate control mechanism. The joint design issue is out of the scope of this paper. Next we will present the ACS in detail.

### 4.2. ACS design

For a network consisting of a number of sensor nodes, spatial and temporal correlation can be jointly used to support efficient compressive sensing. In this subsection, we describe how our designed ACS mechanism works. For a sensor network constituent of $J$ sensor nodes, the readings (without using CS) of the $J$ sensor nodes form multiple measurements vectors (MMV) $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_J]$, where $\boldsymbol{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jN}]^T$, $1 \le j \le J$, is the measurements taken by sensor $j$ and $x_{jk}$ is the reading of sensor $j$ at time $k$, $1 \le k \le N$.

To ease the presentation, in this paper, ACS permutes the multiple measurement vectors $\boldsymbol{X}$ into a single measurement vector $\boldsymbol{X}_C$, and then solves the CS problem based on $\boldsymbol{X}_C$ since an MMV could be considered as multiple correlated SMVs.

Next, we illustrate how ACS works when $J = 2$ as an example. The cases with $J > 2$ can be similarly deduced. For the case $J = 2$, we permute $\boldsymbol{X}$ and generate a new single measurement vector $\boldsymbol{X}_C$ of size $(J \times N) \times 1$ where $\boldsymbol{X}_C = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}$. Now the problem to be solved is transformed to the design of representation basis, measurement matrix, and recovery algorithm.

For the representation basis: Consecutive measurements taken by a sensor node is considered to be temporally correlated (suppose the sample interval is small), i.e., $|x_{j(n+1)} - x_{jn}| < \varepsilon,\ 1 < j \leq J,\ 1 \leq n < N$ and $\varepsilon$ is a small positive number. It is intuitive to design a representation basis $\Psi = D_C^{-1}$ to sparsify $\mathbf{X}_C$ so that $D_C\mathbf{X}_C = [x_{11}, x_{12} - x_{11}, \ldots, x_{1N} - x_{1(N-1)}, x_{21}, x_{22} - x_{21}, \ldots, x_{2N} - x_{2(N-1)}]^T$ is a sparse signal. We deduce

$$D_C = \begin{bmatrix} D_1 & 0 \\ 0 & D_1 \end{bmatrix}, \quad \text{where } D_1 = \begin{bmatrix} 2 & 0 & \cdots & \cdots \\ -1 & 1 & \cdots & \cdots \\ 0 & -1 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}. \tag{4}$$

For the measurement matrix: We choose a measurement matrix such that ACS performs uniform sampling (in each checking window). Note that different checking windows can have different sample rates. The reason to choose uniform sample schedule is that intense change of the target signal could happen at any time and uniform sample schedule is a good choice to achieve high data accuracy when no apriori knowledge regarding the monitored signal is known in advance [12]. We accordingly design a sample schedule $S$ according to the measurement matrix $\Phi$ as follows. For a given sensor node and given $N$ and $M$, where $N$ represents window size and $M$ represents the number of compressed samples to be taken in the window, we have the following.

If $M/N \leq 0.5 : S_j = \begin{cases} 1 & \text{if } j = i \times \lfloor N/M \rfloor \text{ where } i = 1, 2, \ldots, M \\ 0 & \text{otherwise.} \end{cases}$

If $M/N > 0.5$: $S$ is determined as follows. First, we assign $S_j = 1$ if $j$ is an even number and $1 \leq j \leq N$; Second, we randomly select the remaining $M - \lfloor N/2 \rfloor$ items among the remaining odd items and set their values to 1.

We take a simple example to illustrate the sample scheduling procedure. Given $N = 10$ and $M = 2$, since $M/N = 0.2 \leq 0.5$, ACS uniformly samples from the original signal: $S = [0, 0, 0, 0, 1, 0, 0, 0, 0, 1]$. Given $N = 10$ and $M = 7$, we first set $S_2, S_4, \ldots, S_{10}$ as 1, then randomly select 2 items among $S_1, S_3, \ldots, S_9$ to be 1. So one possible result would be [0, 1, 1, 1, 0, 1, 0, 1, 1, 1].

Once the sample schedule in a checking window is determined, a sensor node will only sample the environment and then report its sampling result when $S_j = 1,\ 1 \leq j \leq N$.

With the above sample schedule $S$, $\Phi$ is decided as follows: $\Phi_{ij} = \begin{cases} 1, & \text{if } i = \sum_{k=1}^{j} S_k \text{ and } S_j = 1 \\ 0, & \text{otherwise.} \end{cases}$

With the measurement matrix for each sensor node, each having its own sample rate, the measurement matrix for the entire network is as follows: $\Phi_U = \begin{bmatrix} \Phi_1 & 0 \\ 0 & \Phi_2 \end{bmatrix}$ for $J = 2$.

For the recovery algorithm: Smoothed L0 (SL0) [11] which aims to minimize the $l_0$ norm is adopted in ACS. Different from most existing algorithms that pursue $l_1$ norm minimization instead of $l_0$ norm, SL0 tries to minimize the $l_0$ norm directly. Specifically, it approximates the $l_0$ norm of a signal $s$ by a smooth function $F_\sigma(s)$, where $\sigma$ is a parameter determining the approximation quality. The bigger $\sigma$ is, the smoother $F_\sigma(s)$ but also the worse approximation to the $l_0$ norm will be; and the smaller $\sigma$ is, the better approximation to the $l_0$ norm but the harsher $F_\sigma(s)$ is. SL0 can quickly find a solution in $l_0$ norm by minimizing $F_\sigma(s)$ for a very small $\sigma$. The complexity of SL0 is low as $O(M^2)$.

### 4.3. Sample rate control mechanism

In CS theory, sample rate is defined to equal to $M/N$. Intuitively, in ideal condition, the sample rate should be tuned in real time based on the change intensity of the target signal. The more intense the target signal changes in a short period of time, the less sparse the sensed signal is, the higher the sample rate within that period of time should be; otherwise, the more stable the signal is, the more sparse the signal is, thus the lower the sample rate should be.

In this subsection, we propose mechanisms for adaptive sample rate control, which inspects the measurements in each checking window, whose size is $N$, and adjusts the sample rate in the next checking window accordingly. To pursue high sensing quality, we expect the sensing quality of each checking window exceeds a desired threshold, denoted by $\text{SNR}^*$. There are two key metrics useful for the sample rate adjustment. One is the sparsity degree of the signal and the other is the change intensity of the signal. The change intensity indicates how intense a signal varies with time. The change intensity of a window is measured by the SNR (Signal to Noise Ratio), where the signal is the one of the current window and noise is defined as the signal of current window minus that of its preceding window. Fig. 2 shows how sparsity degree and change intensity of a signal (measurements of relative humidity) vary over time as observed on per checking window basis. From this figure, it is seen that the sparsity degree and change intensity associated with different checking windows change over time. The significant change part of the original signal (see the top subfigure) accords to the peak points of sparsity degree (see the middle subfigure) and the steep drop of change intensity (see the bottom subfigure). Motivated by the observation in Fig. 2, it is possible to adjust the sample rate for the next checking window by leveraging the sparsity degree or change intensity detected in the preceding checking window since sparsity degrees (or change intensities) in neighboring checking windows are very close. Later on, we will further show that our sample rate adjustment mechanism in ACS is highly resilient to certain error in sparsity degree or change intensity estimation. Next, we will present adaptive sample rate control mechanism based on sparsity degree and change intensity, respectively.

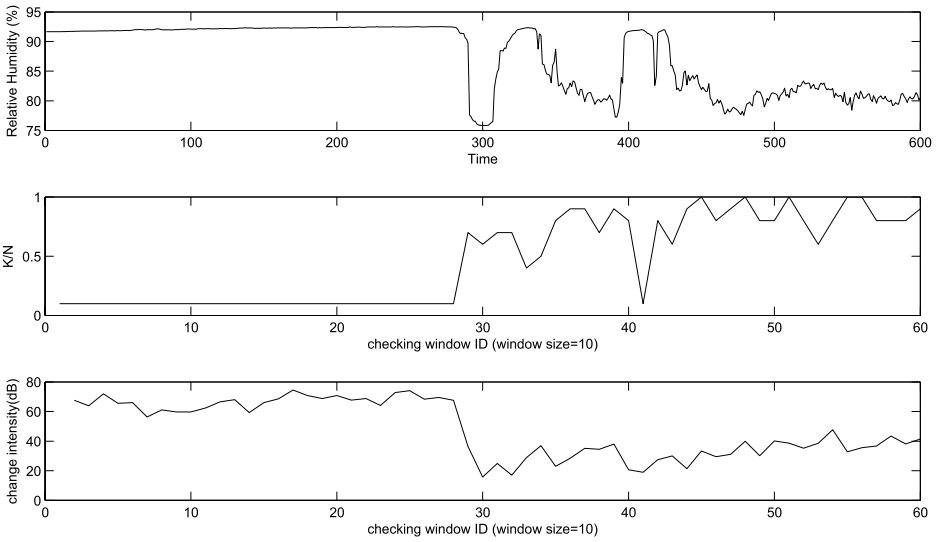


**Fig. 2.** The change trend of relative humidity per checking window.

*4.3.1. Sparsity based sample rate control mechanism*

In CS, it has been proved that to recover the original $K$-sparse signal $s$ with given $(\Phi, \Psi)$ pair, $M$ should satisfy the following conditions to achieve an overwhelming probability for exact recovery:

$$M \geq c \times \mu^2(\Phi, \Psi) \times K \times \log N, \tag{5}$$

where $c$ is a signal dependent positive constant and $\mu$ is the coherence between $\Phi$ and $\Psi$. In general, $M = 3K \sim 4K$ is sufficient to satisfy condition (5). Accordingly, condition (5) could be rewritten as:

$$sr = \frac{M}{N} \geq c \times \mu^2(\Phi, \Psi) \times \left(\frac{K}{N}\right) \times \log N, \tag{6}$$

where $sr$ represents the sample rate. Eq. (6) is for achieving *exact* recovery with a sufficiently high probability. Setting sample rate according to (6) can lead to excessively high sample rate. Also, (6) is suitable for the orthonormal matrices, which are not suitable for our case.

To address the above two considerations, ACS empirically identifies the relation between sample rate and sparsity degree. Such relation is built during a training phase. Then, in the online phase, ACS can adjust the sample rate of each sensor in the next checking window based on the signal sparsity degree (or change intensity) detected in the current checking window by looking up the hash table pre-built during the training phase. More detailed design of ACS is presented as follows.

In the training phase, ACS pre-collects a large amount of original measurements from all the sensor nodes in the network at a high sample rate as training data. Then, for each individual checking widow, the sample rate is increased gradually from a very small value until the network-wide recovery quality meets given threshold for the first time. Here, the network-wide recovery quality is calculated based on the measurements from all sensor nodes in the same checking window and further all the sensor nodes are assumed to have the same sample rate. Recovery quality is measured in terms of recovery SNR $= 20\log(\frac{\|x\|_2}{\|\tilde{x}-x\|_2})$, where $\tilde{x}$ is the recovered signal for the original signal $x$. This version of ACS mechanism is called ideal ACS hereafter, which can achieve the minimum sample rate while keeping the recovery SNR per checking window always above the desired given threshold. Although impractical to be actually implemented in practice, ideal ACS is useful for analyzing the properties of target signal and can also be used as a baseline for performance comparison.

Here, we use an example to illustrate the relation between sample rate and sparsity degree. Fig. 3 plots sample rate $sr$ versus sparsity degree $K/N$ when performing ideal ACS to monitor relative humidity. The data trace is from project SensorScope [13] and more detailed description of this project will be given in Section 5. In our training process, the checking window size is $N = 10$, accordingly $NJ = 230$, and $sr$ is chosen from $\{0.1, 0.2, \ldots, 1\}$. Recall that $J$ represents the network size. Furthermore, here, the sparsity degree was calculated on a per-network basis. That is, at each time, sparsity degree was evaluated based on the total $NJ = 230$ samples in a checking window and collected from all the network nodes. The given SNR threshold in this figure was selected as 35 dB.

Fig. 3 shows how the sample rate $sr$ changes with sparsity degree $K/N$. Based on the measurements in Fig. 3, we can build a hash table for mapping from $K/N$ to $sr$. Because the least sample rate required for a desired sensing quality is strongly signal dependent, there may exist some overlapping between different $K/N$ ranges associated with different $sr$. In ACS, we tend to choose larger sample rate for ensuring recovery quality when one $K/N$ corresponds to multiple sample rates (more exactly, two sample rates in Fig. 3). Specifically, we determine the hash table as a significance test problem. Starting from $sr = 1.0$

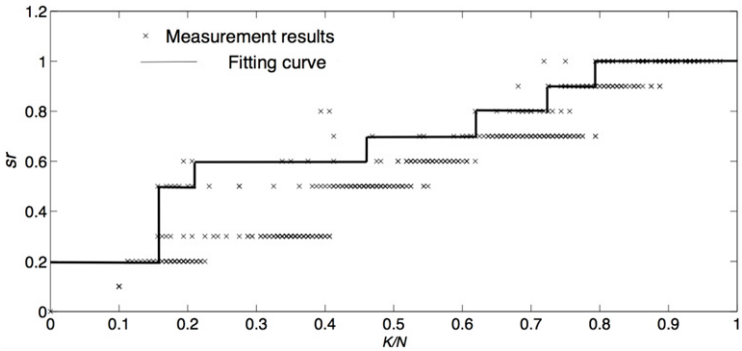


**Fig. 3.** Sample rate $sr$ versus sparsity degree $K/N$ for relative humidity monitoring when ideal ACS is performed.

to $sr = 0.1$, regarding each $sr$ value we choose its corresponding interval of $K/N$ with significance level $\alpha = 0.01$. That is, for each $sr$ value, at least $(1 - \alpha) \times 100 = 99\%$ of measurements fall into its corresponding $K/N$ interval and are said to pass the significance test. Fig. 3 plots the fitting curve for the hash table, below which those measurements are said to pass the significance test.

The online phase in ACS works as follows. With the pre-established hash table from sparsity degree to sample rate, at the end of each checking window, ACS recovers the signal of the window by using SL0, calculates the sparsity degree $K/N$ of the recovered signal, and then chooses a corresponding $sr$ for the next checking window by looking up the pre-built hash table. As an example, in Fig. 3, the interval of $K/N$ for $sr = 0.7$ is $[0.47, 0.63]$. That is, for a signal with sparsity degree falling into this interval, its sample rate (for the next window) will be set to 0.7. A salient feature of the hash table (see also the fitting curve in Fig. 3) is as follows: Overestimation of sparsity degree will not cause sacrifice in sensing quality but can cause certain oversampling if the estimated value drifts into the right neighbor stage (interval) in the fitting curve; Further, underestimation will not cause sacrifice in sensing quality either unless the estimated value drifts into the left neighbor stage. Experimental results show that ACS can achieve very high success rate ($>98\%$) for individual checking window to meet desired sensing quality. In this sense, we say the sample rate adjustment mechanism in ACS is highly resilient to estimation error of sparsity degree. Similar conclusion can also be drawn for change intensity based ACS (see later). Since the hash table is pre-established based on training data collected from all sensors in the network, in this sense, we say ACS has already considered both temporal/spatial correlations in its implementation. However, it should be noted that during the online implementation, ACS does not need to collect the original measurements at high sample rate.

The computation at each sensor node consists of signal recovery and sparsity degree computation. As mentioned in Section 4.2, ACS leverages algorithm SL0 whose computation complexity is $O(M^2)$ to recover the signal. Sparsity degree computation includes difference computation, and thus it has complexity $O(N)$. The overall computation complexity is $O(M^2 + N)$, which is usually acceptable as compared with the sampling overhead and transmission overhead it saved in particular consider the fact that nowadays many sensor nodes in the market are having increasing computational capabilities. In case computational overhead is indeed a big concern, such computation can be carried out at cluster head nodes or sink nodes in a WSN, which usually have high computational capabilities.

#### *4.3.2. Intensity based sample rate control mechanism*

The change intensity of target signal is also a good indicator to adjust the sample rate of sensors. ACS can check the temporal difference between the last two adjacent checking windows, if a relatively significant difference happens, we say that the phenomenon has changed significantly so that the sample rate (for the next checking window) should be augmented; otherwise, the sample rate stays at a low level. Please recall the change intensity, i.e., the signal intensity difference between two adjacent checking windows is measured by the SNR of the recovered signals associated with the two windows. Fig. 4 plots the fitting curve for the relation between change intensity and sample rate during a training phase by using certain measurements from SensorScope [13]. In Fig. 4, those measurement results below the fitting curve (on the left side) are said to pass the significance test. Similar to the sparsity degree based sample rate mechanism in Section 4.3.1, we can establish a hash table by using the fitting results in Fig. 4.

The online phase works as follows. At the end of a checking window, each sensor node recovers the signal of the window, calculates the SNR (representing change intensity) between the current window and its preceding window, and accordingly set the sample rate for the next checking window.

#### *4.3.3. Discussions*

After building the hash table, adaptive sample scheduling can be realized at different scales for tuning sensor nodes' sample rates. One is let the sink node be responsible for adjusting the sample rate and recovering the signal based on the data that it collects from all the sensor nodes in the network. The second method is that, for cluster-based networks, the nodes in the same cluster jointly determine their sample rate. The third method is that each sensor node adjusts its sample

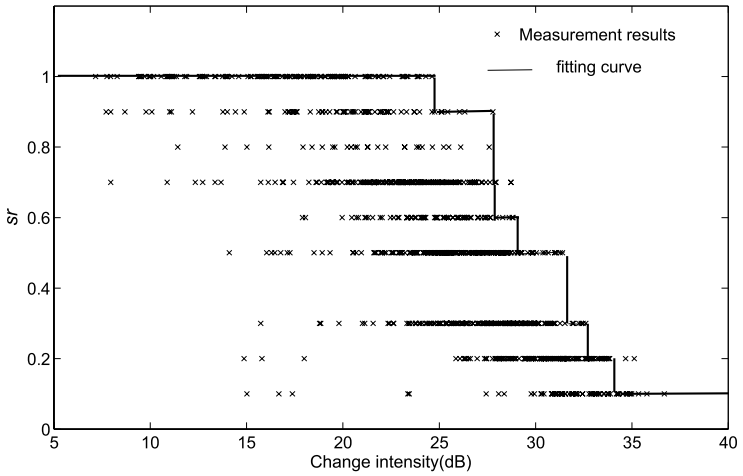


**Fig. 4.** Sample rate *sr* versus change intensity of signals for relative humidity monitoring.

rate independently. The first implementation method is referred to as "*network scale ACS*", the second is referred to as "*cluster scale ACS*" and the third is "*node scale ACS*" hereafter. Network scale ACS takes advantage of all the data of the network so that the sample rate control is done from the viewpoint of the whole network. However, it brings very high communication overhead and also high delay for sensor nodes to learn their new sample rate. New sample rate can only be calculated upon receipt of all data from all the sensor nodes, and then be disseminated to all the sensor nodes via reverse network-wide flooding operations. This may degrade the network performance and reduce its ability to quickly react to significant change of target signal which may be unacceptable for large-scale networks. Cluster scale ACS restricts the sample rate control to be within a cluster so that the protocol overhead and reaction delay for sample rate control is relatively low. Node scale ACS, has the lowest communication overhead.

Please note that the node scale version of ACS has also considered both temporal correlation at individual sensor nodes and the spatial correlation among different sensor nodes. This is because, during the training phase, the hash table from sparsity degree/change intensity to the sample rate *sr* is obtained by analyzing the training data from all sensor nodes in the network so that the relation between required sample rate and sparsity degree (or change intensity) was established based on both temporal and spatial correlation of signals in the whole network. The performance of the above three versions of ACS will be compared in Section 5.2.

The checking window size $N$ has big impact on ACS's compression performance. Too large $N$ may lead to excessively high sample rate but too small $N$ may be insufficient to perform CS. The reason is as follows. Large $N$ means there is a high probability that significant change of the target signal may happen in an individual checking window. To capture such (significant) change of target signal, we have to choose a high sample rate for the whole window although the signal change may just last for short time (as compared to the window size). In this case, oversampling is observed. In contrast, too small $N$ unnecessarily restricts the possible compression ratio for CS. $N = 1$ is an extreme case because a signal with $N = 1$ cannot be compressed at all for individual sensor nodes and ACS with $N = 1$ can only consider spatial correlation for compression. The impact of $N$ and how to properly set its value will be demonstrated via extensive simulations in Section 5.

A special case in the implementation of sparsity degree based ACS is as follows. We should not set sample rate to a value leading to $M = 1$ sample in a checking window. This is because, in such a case, no matter how intensely the target signal changes during the window, the sparsity $K$ will be surely 1, which results in inaccurate estimation of sparsity degree of the signal. Owning to this concern, we will not discuss the case $M = 1$ for sparsity degree based ACS hereafter.

## 5. Numerical results

In this section, we evaluate the efficiency of ACS by comparing it with related work through extensive experiments. For this purpose, we use environmental data to evaluate the performance of ACS. We first introduce the data trace used in our experiments and analyze the properties of different types of data to obtain certain apriori knowledge for implementing ACS. We then compare the performance of network/cluster/node scale ACS. Finally, we compare ACS with other CS schemes.

### *5.1. Experiment preparation*

The data trace used in our experiments is the environmental data collected by project SensorScope, deployed at Grand-St-Bernard (called Grand-St-Bernard Deployment hereafter) near the borders of Switzerland and Italy [13]. SensorScope is a distributed measurement system based on wireless sensor network with built-in capacity to produce high temporal and spatial density measures. Grand-St-Bernard is one of SensorScope's experiment fields and its sensor deployment layout is

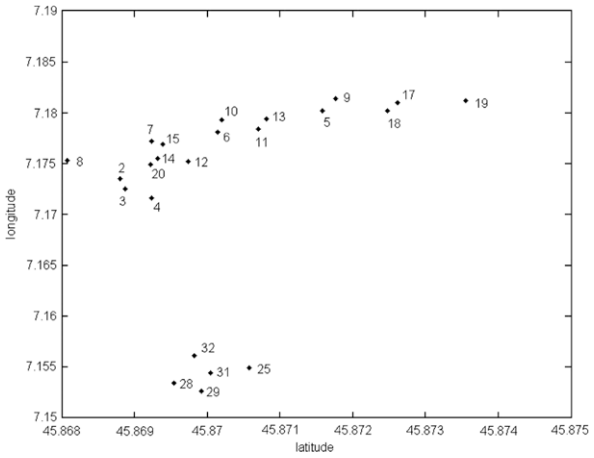


**Fig. 5.** Network layout of the Grand-St-Bernard Deployment [13].

shown in [13]. In the SensorScope project, the deployed sensor nodes measure key environmental data once per two minutes and the monitored phenomena include air temperature, humidity, surface temperature, incoming solar radiation, and etc. The total number of samples used in our experiments is 10,000, which are equivalent to 20,000 min continuous monitoring (i.e., about two weeks).

Before the performance evaluation, we first analyze the impact of different parameters on compression performance.

First, we evaluate the impact of checking window size $N$ on the performance of ACS. The metrics used in performance evaluation are average sample rate and the overall recovery quality. For this purpose, we varied $N$ from 8 to 100 when implementing ideal ACS to sample rain meter and relative humidity and see how the required sample rate for ideal ACS changes (subject to given desired recovery quality $\text{SNR}^* = 35$ dB in this subsection). Fig. 6(a) plots the average sample rates with varying $N$ for different phenomena. In the figure, the sample rate and SNR of relative humidity are more stable than those of rain meter. This is because their data characteristics: That is, relative humidity is relatively stable while rain meter may change significantly in short time. The performance regarding relative humidity is insensitive to the window size but that of rain meter changes dynamically with increasing window size. It is shown that $N = 10$ leads to the least sample rate. Fig. 6(b) plots the overall recovery SNR with varying $N$. Here, Overall SNR is a network parameter and it represents the network-wide recovery SNR based on all the samples collected from all sensor nodes throughout an experiment. We can see that $N = 8$ and $N = 10$ result in higher recovery SNR than that by other settings of $N$ and the gap between the recovery SNR by $N = 8$ and that by $N = 10$ is insignificant. Based on the above experiment results, we shall set $N = 10$ in the following tests since $N = 10$ could result in low sample rate and high overall recovery SNR.

Second, we fixed checking window size $N = 10$ to observe the sparsity degrees of different measurements. The experiment results show that, for ideal ACS, the (minimal required) average sample rate (subject to $\text{SNR}^* = 35$ dB) is about 0.81 for ambient temperature, 0.66 for surface temperature, 0.94 for solar radiation, 0.58 for relative humidity, 0.17 for soil moisture, 0.89 for watermark, 0.16 for rain meter, and 1 for wind direction. In other words, to monitor ambient temperature, relative humidity, soil moisture, watermark, rain meter, CS can bring improvements while it brings little gain for wind direction, which is non-sparse. To evaluate the gain by using ACS, we chose rain meter (a high sparse signal) and relative humidity (a moderate sparse signal) as the data sources in the following experiments.

### 5.2. Implementation of ACS at different scales

There are several ways to implement ACS as discussed in Section 4.3, i.e. network scale ACS, cluster scale ACS, and node scale ACS. In this subsection, we compare their performance. Regarding the cluster scale ACS, we simply divide the network into four clusters according to the positions of the sensor nodes. For example, in Fig. 5, the four clusters are accordingly formed as follows: {25, 28, 29, 31, 32}, {2, 3, 4, 8, 20, 14, 15, 7, 12}, {6, 10, 11, 13, 5, 9}, and {17, 18, 19}. Here, to compare the best possible performance of the three implementations of ACS, we assume the packet delivery delay is very small and thus ignored. We accordingly evaluate the performance of different implementations of ACS in terms of success ratio, average sample rate (avg. *sr* for short), min SNR, and overall recovery SNR. Success ratio is the ratio between the number of checking windows whose recovery SNR is above the required sensing quality (we chose 35 dB here) and the total number of checking windows. The average sample rate is the ratio between total number of compressed samples and the total number of original uncompressed samples. Min SNR is the minimum SNR among all checking windows.

Table 1 compares the performance of different implementations of ACS. To ease the presentation, we call different implementations of ACS by using two tuples. For example, ACS (sparsity, node) means node scale sparsity based ACS. From Table 1, we can see that no matter which version of ACS is carried out, intensity based ACS obtains similar success ratio, overall recovery SNR, and min SNR with sparsity degree based ACS but has a slightly lower sample rate. Based on the results

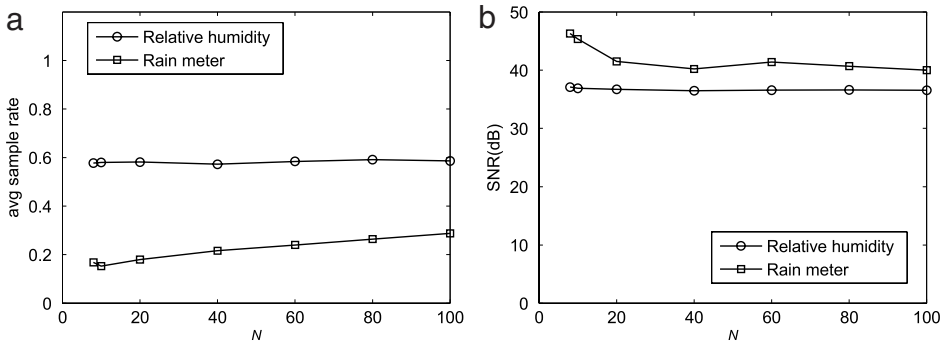


**Fig. 6.** Compression performance of ideal ACS versus $N$ when $SNR^* = 35$ dB for different phenomena monitoring.

**Table 1**
Performance of ACS for rain meter.

| | Avg. sample rate | Success ratio (%) | Min SNR(dB) | Overall SNR(dB) |
|---|---|---|---|---|
| ACS (sparsity, network) | 0.36 | 98.0 | 15.3 | 37.9 |
| ACS (sparsity, cluster) | 0.31 | 98.5 | 14.2 | 35.8 |
| ACS (sparsity, node) | 0.23 | 98.3 | 15.0 | 35.3 |
| ACS (intensity, network) | 0.28 | 98.8 | 9.8 | 36.4 |
| ACS (intensity, cluster) | 0.27 | 98.6 | 16.9 | 40.4 |
| ACS (intensity, node) | 0.23 | 98.3 | 12.4 | 37.7 |

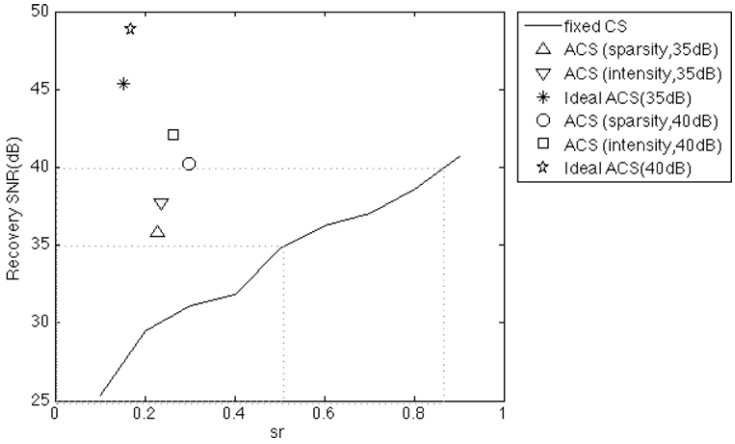


**Fig. 7.** Performance comparison of different mechanisms regarding rain meter.

in Table 1, we have the following observations: Node scale ACS does not lead to a higher sample rate for achieving comparable recovery quality compared with network/cluster scale ACS. The experiment based on relative humility has similar result. The above result is encouraging: To achieve high sensing quality and low sample rate, every node can independently adjust its own sample rate according to the hash table established during the training phase and needs no communication overhead for local coordination (suppose each individual sensor node in the network has sufficient computation capability for performing the processing task in ACS). In the next subsection, we will focus on exploring the performance of the node scale ACS.

### 5.3. Performance comparison

In this subsection, we compare the performance of the following three mechanisms: ACS, CS with fixed sample rate (called "fixed CS" for short hereafter), and ideal ACS, subject to $SNR^* = 35$ dB and 40 dB, for monitoring rain meter and relative humidity, respectively.

The superiority of ACS is significant for rain meter monitoring as shown in Fig. 7. In Fig. 7, fixed CS needs sample rates about 0.50 and 0.88 for $SNR^* = 35$ dB and 40 dB, respectively. In contrast, ACS can achieve higher success ratio and overall recovery SNR by using lower sample rate. Given $SNR^* = 35$ dB, ACS(intensity) only needs a sample rate of 0.23 to achieve overall recovery SNR of 37.7 dB; ACS(sparsity) utilizes sample rate of 0.23 to obtain overall recovery SNR 35.8 dB. Given $SNR^* = 40$ dB, ACS(intensity) leverages sample rate of 0.26 to achieve overall recovery SNR 42.2 dB while ACS(intensity)

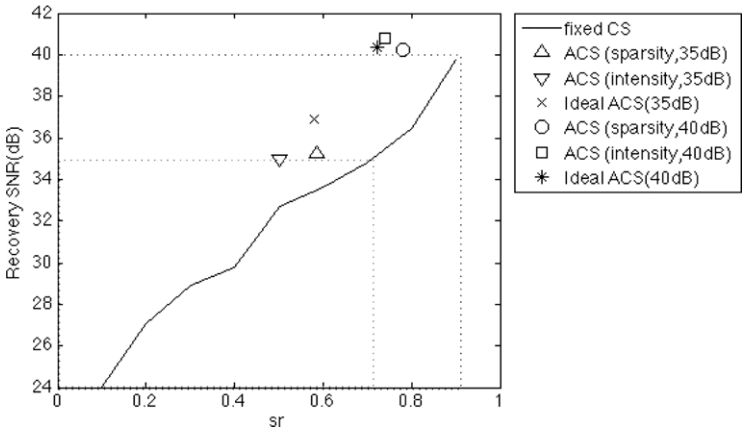


**Fig. 8.** Performance comparison of different mechanisms regarding relative humidity.

leverages sample rate of 0.30 to achieve overall recovery SNR 40.2 dB. Obviously, we can see that ACS largely outperforms fixed CS regarding rain meter monitoring. The reason is that ACS has the ability to capture significant change of target signal.

In addition to the experiments based on rain meter measurements, to better evaluate the performance of ACS, we conduct another experiment using the data trace of relative humidity. Different from the high sparsity of rain meter in the previous experiments, the average sample rate to achieve sensing quality 35 dB for relative humidity by using ideal ACS is more than 0.5. We chose such a signal as a moderately sparse signal. The experiment results are shown in Fig. 8. For $SNR^* = 35$ dB, fixed CS needs sample rate above 0.70. In contrast, ACS(sparsity) only needs sample rate of 0.58 for achieving overall recovery SNR 35.2 dB while ACS(intensity) only needs sample rate of 0.52 for achieving overall recovery SNR 35.0 dB. For $SNR^* = 40$ dB, fixed CS needs sample rate above 0.90. However, ACS(sparsity) only needs sample rate of 0.78 for overall recovery SNR 40.3 dB while ACS(intensity) only needs sample rate of 0.74 for overall recovery SNR 40.8 dB. From Fig. 8, we can draw the same conclusion on the benefit of ACS as that in the last experiment. The experimental results demonstrate again that ACS can achieve desired sensing quality with a much lower sample rate as compared with fixed CS.

In summary, from Figs. 7 and 8, we can draw a conclusion that ACS (either intensity based or sparsity based) achieves desired sensing quality by using much fewer samples compared with fixed CS.

## 6. Conclusion

In this paper, we studied the adaptive compressive sensing problem for wireless sensor networks and accordingly proposed an adaptive CS-based sample scheduling (ACS) mechanism to overcome the drawback that fixed sample rate in existing compressive sensing mechanisms is unable to quickly react to significant phenomena change. ACS can work with either of the following metrics, signal sparsity or change intensity, for window-by-window sample rate adjustment by using hash table built during training phase. Extensive experiments show that ACS can maintain high stable sensing quality with low sample rate.

## Acknowledgments

This work was supported in part by NSF of China under Grant Nos. 61173158, 61471339, and 61101133. Shiwen Mao's research was supported in part by the US National Science Foundation (NSF) under Grants CNS-0953513, and through the NSF Broadband Wireless Access and Applications (BWAC) Center site at Auburn University.

## References

[1] E.J. Candes, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.
[2] C. Luo, F. Wu, J. Sun, C.W. Chen, Compressive data gathering for large scale wireless sensor networks, in: Proc. ACM MobiCom'09, Beijing, China, 2009, pp. 145–156.
[3] S.L. Ji, R. Beyah, Y.S. Li, Continuous data collection capacity of wireless sensor networks under physical interference model, in: Proc. IEEE MASS'11, Valencia, Spain, 2011, pp. 222–231.
[4] J. Luo, L. Xiang, C. Rosenberg, Dose compressed sensing improve the throughput of wireless sensor network? in: Proc. IEEE ICC'10, Cape Town, South Africa, 2010, pp. 1–6.
[5] S. Lee, S. Pattem, M. Sathiamoorthy, Spatially localized compressed sensing and routing in multi-hop sensor network, in: Proc. GSN, Oxford, UK, 2009, pp. 11–20.
[6] J. Haupt, W.U. Bajwa, M. Rabbat, R. Nowak, Compressed sensing for networked data, IEEE Signal Process. Mag. 25 (2) (2008) 92–101.
[7] Z. Charbiwala, S. Chakraborty, S. Zahedi, K. Younghun, M.B. Srivastava, H. Ting, C. Bisdikian, Compressive oversampling for robust data transmission in sensor network, in: Proc. IEEE INFOCOM'10, San Diego, CA, USA, 2010, pp. 1–9.
[8] X.P. Wu, M.Y. Liu, In-situ soil moisture sensing: measurement scheduling and estimation using compressive sensing, in: Proc. IEEE IPSN'12, Beijing, China, 2012, pp. 1–12.

[9] D. Baron, M.B. Wakin, M.F. Duarte, S. Sarvotham, R.G. Baraniuk, Distributed compressed sensing, Technical Report ECE-0612, Electrical and Computer Engineering Department, Rice University, 2006.
[10] S. Chen, D. Donoho, Basis pursuit, in: Conference Record of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, Vol. 1, 1994, pp. 41–44.
[11] H. Mohimani, M. Babaie-Zadeh, C. Jutten, A fast approach for overcomplete sparse decomposition based on smoothed l0 norm, IEEE Trans. Signal Process. 57 (1) (2009) 289–301.
[12] H. Deng, B. Zhang, J. Zheng, Rate-constrained uniform data collection in wireless sensor networks, IET Commun. 5 (10) (2011) 1343–1350.
[13] http://sensorscope.epfl.ch/index.php/Environmental_Data.
[14] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, M. Zorzi, Sensing, compression, and recovery for WSNs: sparse signal modeling and monitoring framework, IEEE Trans. Wireless Commun. 11 (10) (2012) 3447–3461.
[15] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, M. Zorzi, On the interplay between routing and signal representation for compressive sensing in wireless sensor networks, in: ITA Workshop, 2009.
[16] R. Rana, W. Hu, C.T. Chou, Energy-aware sparse approximation technique (EAST) for rechargeable wireless sensor networks, in: Proc. EWSN, Coimbra, Portugal, 2010, pp. 306–321.
[17] Y. Shen, W. Hu, R. Rana, C.T. Chou, Nonuniform compressive sensing for heterogeneous wireless sensor networks, IEEE Sens. J. 13 (6) (2013) 2120–2128.
[18] Z. Fanzi, G.B. Giannakis, Compressed sensing for wideband cognitive radios, in: Acoustics, Speech and Signal Processing, 2007. Proc. ICASSP, Hawaii, USA, Vol. 4, 2007, pp. IV-1357–IV-1360.
[19] D. Hu, S. Mao, N. Billor, P. Agrawal, On the trade-off between energy efficiency and estimation error in compressive sensing, Ad Hoc Networks 11 (6) (2013) 1848–1857.
[20] Y.L. Polo, Y. Wang, A. Pandharipande, G. Leus, Compressive wide-band spectrum sensing, in: Proc. ICASSP, Taipei, Taiwan, 2009, pp. 2337–2340.
[21] Z. Fanzi, C. Li, Z. Tian, Distributed compressive spectrum sensing in cooperative multihop cognitive networks, IEEE J. Sel. Top. Signal Process. 5 (1) (2011) 37–48.
[22] Y. Wang, A. Pandharipande, Y.L. Polo, G. Leus, Distributed compressive wide-band spectrum sensing, in: IEEE Information Theory and Applications Workshop, 2009, pp. 178–183.
[23] L.Y. Song, Y. Zhang, R. Yu, W.Q. Yao, QoS-aware packet forwarding in MIMO sensor networks: a cross-layer approach, Wirel. Commun. Mob. Comput. 10 (6) (2010) 748–757.
[24] R. Yu, Y. Zhang, L.Y. Song, W.Q. Yao, Joint optimization of power, packet forwarding and reliability in MIMO wireless sensor networks, ACM/Springer Mob. Netw. Appl. (MONET) 16 (6) (2010) 760–770.