The following publication P. Guo, J. Cao and X. Liu, "Lossless In-Network Processing in WSNs for Domain-Specific Monitoring Applications," in IEEE Transactions on Industrial Informatics, vol. 13, no. 5, pp. 2130-2139, Oct. 2017 is available at https://doi.org/10.1109/TII.2017.2691586.

# Lossless In-network Processing in WSNs for Domain-specific Monitoring Applications

Peng Guo, Jiannong Cao and Xuefeng Liu

**Abstract**—Internet of things (IOT) is emerging as sensing paradigms in many domain-specific monitoring applications in smart cities, such as structural health monitoring (SHM) and smart grid monitoring. Due to the large size of the monitoring objects (e.g., civil structure or the power grid), plenty of sensors need to be deployed and organized to be a large scale of multi-hop wireless sensor networks (WSNs), which tends to have quite high transmission cost. In-network processing is an efficient way to reduce the transmission cost in WSNs. However, implementing in-network processing for above domain-specific monitoring usually requires to losslessly distribute a dedicate domain-specific algorithm into WSNs, which is much different from most existing in-network processing works. This paper conducts a case study of a classic centralized SHM algorithm ERA, and shows how to losslessly and optimally in-network process ERA, especially the typical feature extraction method SVD therein, in a WSN. Based on whether the intermediate data can be processed together or not by sensor nodes, we respectively implement tree-based in-network processing of SVD and chain-based in-network processing of SVD in WSNs. We prove that using an appropriate shallow light tree (SLT) as routes for tree-based in-network processing of SVD, can achieve the approximation ratio $1+\sqrt{2}$ (in terms of transmission cost), while for the chain-based in-network processing of SVD, we design two efficient heuristic algorithms for searching the optimal routes. Extensive simulation results validate the efficiency of these proposed schemes that are customized for SVD-based IOT applications.

**Index Terms**—Wireless Sensor Network (WSN), In-network processing, Matrix computation, Routing scheme.

## 1 Introduction

Many domain-specific monitoring systems in smart cities, such as structural health monitoring (SHM) [1] of skyscrapers, state estimation in smart grid [2], and fault diagnosis of machines [3], do not simply employ wireless sensing technology, but more and more simultaneously require some domain-specific informatics analysis methods. When the systems employ wireless sensor networks (WSNs), lots of sensor nodes usually need to be deployed and cooperatively monitor one common object (e.g., structure, smart grid or big machine). After the sink node gathers the raw data sampled at different regions of the object, a specialized centralized algorithm is then executed to extract some global features or estimate model parameters of the object.

However, implementing a WSN-based domain-specific monitoring system in practice is usually much challenging, which is much different from the conventional environmental monitoring applications of WSNs. For instance, in order to accurately capture the vibration response of civil infrastructure in SHM, sensors are required to sample the structure's responses at high sampling rate ($>$ hundreds of Hz) for a "long enough" period of time [4]. Consequently, the number of data collected at each sensor can reach the level of tens of thousands. Transmitting such a large number of data to the sink becomes a challenge to resource-limited sensor nodes, especially in multi-hop WSNs.

To deal with this challenge, in-network processing is an effective way. With in-network processing, only the important information or intermediate computation data rather than the raw data is transmitted. Hence, the transmission load in WSNs can be significantly reduced. However, implementing appropriate in-network processing in WSN-based domain-specific monitoring system has typical requirements. Different from conventional WSN applications where the in-network processing is simple (e.g., Max, Sum or top-k) and typically limited to spatial or temporal aggregation [5], or directly assumes a general aggregation function (e.g., constant size of output), the in-network processing in WSN-based domain-specific monitoring systems should not simply assume a common aggregation function in advance. Instead, they usually designate a specialized centralized algorithm, and the WSN should exactly distribute the algorithm into sensor nodes for in-network processing, so as to achieve the same quality as that of wired centralized monitoring systems (we call it as lossless in-network processing in this paper).

Unfortunately, it is not easy to exactly partition many domain-specific algorithms into distributed in-network processing functions in WSNs, as the algorithms are sophisticated and require all raw data from the deployed nodes for feature extraction, generally through various of matrix computations such as eigen decomposition [6], singular value decomposition (SVD) [7]. Distributing these matrix computations into WSNs is not a trivia.

Besides the hardness of partitioning the domain-specific algorithms, appropriate partition is also a key issue, in consideration of assigning the partitioned computations to

*Peng Guo (e-mail: guopeng@hust.edu.cn) and Xuefeng Liu (e-mail: csxfliu@gmail.com) are with the school of Electronic Information and Communications, Huazhong University of Science and Technology, China.*
*Jiannong Cao (e-mail: csjcao@comp.polyu.edu.hk) is with the Department of Computing, Hong Kong Polytechnic University, Hongkong.*

a WSN with arbitrary topology. Inappropriate partition of a centralized algorithm may be inefficient (or even hard) for the computation assignment, as the raw data in the network may not spread exactly as expected by the partitioned computations.

In this paper, we take example of a classic SHM algorithm: the eigensystem realization algorithm (ERA) [6] (with typical matrix computation SVD therein), and show how to losslessly in-network process ERA within an arbitrary WSN. Then, we study the optimization of the in-network processing, i.e, design the optimal routing scheme for the in-network processing to minimize the transmission cost. Contributions of the paper are summarized as follows.

- We implement two kinds (i.e., tree-based and chain-based) of lossless in-network processing for SVD in WSNs with arbitrary topology.
- We propose to employ a special *shallow light tree* (SLT) with appropriate parameter as the routing scheme for tree-based in-network processing of SVD, and prove that the transmission cost of the SLT can achieve approximation ratio $1+\sqrt{2}$.
- For chain-based in-network processing of SVD, we propose an efficient greedy algorithm and a simulated annealing algorithm to search the optimal routes.

Note that, as the key work in the paper is actually the optimal lossless in-network processing of the widely used SVD computation, the work can also be applied in many other industrial monitoring applications which rely on SVD factorization techniques to extract some features or patterns from the sampled data. Examples are non-destructive testing of machines [3] and state estimation of smart grid [8]. In addition, the proposed work can also help for distributed computing of some big data mining applications (e.g., urban traffic data [9] and smart meter data [10]) where the data spreads in a large network and SVD computation is widely employed.

The remainder of the paper is organized as follows. In Section 2, we review the related works. Section 3 presents the lossless in-network processing of ERA. The optimization of the in-network processing is discussed in Section 4. Simulations are described in Section 5, followed by the conclusions in Section 6.

## 2 Related work

Though lots of works on in-network processing have been done for WSNs [11]–[13]. The works are rather to be called as in-network aggregation, as usually a common and simple aggregation function, such as averaging the raw data, finding the maximum or top-K of the raw data, or just a general common compression formation, is pre-assumed in these works. Based on the aggregation functions, existing works usually focus on designing efficient (in terms of communication cost [11], [12] or delivery delay [13]) routing schemes for the in-network aggregation. However, unlike other monitoring applications, SHM usually targets at detecting possible structure damage, which is not straightforward but requires much domain knowledge. Aggregation functions for SHM should be specially designed according to a specialized SHM algorithm for the damage detection. In other words, we need to in-network process (or distribute) a dedicated SHM algorithm in WSNs.

Unfortunately, most SHM algorithms are centralized and require all raw data from the deployed nodes. Distributing these algorithms into a WSN is not easy. In [14], a multi-level damage localization strategy is proposed. In [15] and [16], some SHM algorithms such as the ERA [6] and the FDD [17] are made distributed through the idea of "divide and conquer". Other existing works in this area can be found in [18] and [19]. However, one problem that generally exists in these WSN-tailored distributed SHM algorithms is that the damage detection capability cannot be guaranteed to be comparable with the centralized SHM algorithm. This is because each cluster divided in WSNs only uses its local information, which can result in the ill-conditioned problem [6], and this inaccuracy cannot be rectified via the "stitching" process afterwards.

In recent works [1], the authors exactly distribute the ERA algorithm into WSNs. Meanwhile, the ERA is equivalently partitioned into pieces of computation tasks which require only local data, and the intermediate data that can be computed iteratively. This work takes one step to in-network process ERA in WSNs. However, the paper does not study how to efficiently in-network process ERA with optimal routing scheme to achieve the minimum transmission cost.

## 3 Distribution of ERA

In this section, we first briefly introduce the classic SHM algorithm ERA. Then, we show how to losslessly in-network process ERA by exactly distributing the computation therein.

### 3.1 Principle of SHM

According to the vibration theory, every structure has tendency to oscillate with much larger amplitude at some frequencies than others. These frequencies are called natural frequencies. For a structure with $n$ degrees of freedom, it has $n$ natural frequencies. When a structure is vibrating under one of its natural frequencies, it exhibits a corresponding vibration pattern for this natural frequency. For example, the natural frequencies and the corresponding mode shapes of a $n$-degrees-of-freedom structure are denoted respectively as:

$$\Omega = \{\omega_1, \omega_2, ..., \omega_n\}, \mathbf{\Psi} = [\mathbf{\Psi_1}, \mathbf{\Psi_2}, \cdots, \mathbf{\Psi_n}] \quad (1)$$

where $\omega_i$ and $\mathbf{\Psi_i} = [\phi_i^1, \phi_i^2, \cdots, \phi_i^n]^T$ $(i = 1, ..., n)$ are the $i^{th}$ natural frequency and the corresponding vibration pattern, respectively. $\phi_i^k (k = 1, 2, ..., n)$ is the value of $\mathbf{\Psi_i}$ at the $k^{th}$ degree of freedom.

$\omega_i$ and $\mathbf{\Psi_i}$ are internal vibration characteristic of structure and are quite important in SHM. By comparing the newly identified modal parameters with those obtained when the structure is healthy, possible damage can be detected and further located.

### 3.2 The ERA algorithm

ERA algorithm is a classic SHM algorithm that can estimate $\omega_i$ and $\mathbf{\Psi_i}$ of a structure with vibration data sampled by sensor nodes. Assume that a total of $M$ sensors are deployed on a structure and the collected raw data are denoted as $\mathbf{x}^i = [x^i(1), x^i(2), \cdot, x^i(N_{ori})]$, where $x^i(k)$ is the raw data sampled by the $i^{th}$ sensor at $k^{th}$ time step and $N_{ori}$ is the total number of data points collected in each node. With gathering all these raw data, ERA first calculates the cross-correlation function (CCF) between each nodes' raw data. Denote:

$$\mathbf{CCF} = [CCF_{\mathbf{x}^1\mathbf{x}^{ref}}, \cdots, CCF_{\mathbf{x}^M\mathbf{x}^{ref}}]^T \tag{2}$$

where $\mathbf{x}^{ref}$ is any element in set $\{\mathbf{x}^i\}$, and $CCF_{\mathbf{x}^i\mathbf{x}^{ref}}$ is the CCF between $\mathbf{x}^i$ and $\mathbf{x}^{ref}$. $CCF_{\mathbf{x}^i\mathbf{x}^{ref}}$ can be calculated with Welch's method [22] which first calculates the cross spectral density (CSD) between $\mathbf{x}^i$ and $\mathbf{x}^{ref}$ and then implement inverse Fourier transform (IFFT) on the CSD to obtain the CCF. Meanwhile, when calculating CSD, sufficient data is required. For example, the raw data measured in each round by a node may need to contain about $n_d = 20$ overlapped segments of data, and each segment of data may need to perform 2048-point IFFT. That is to say, $N_{ori}$ can be more than twenty thousands. Therefore, it's really difficult to gather these large amount of raw data from the resource-limited nodes to sink.

After obtaining CCF, ERA next establishes a Hankel matrix as follows. Denote $\mathbf{Y}(j) = [CCF_{\mathbf{x}^1\mathbf{x}^{ref}}(j), \cdots, CCF_{\mathbf{x}^M\mathbf{x}^{ref}}(j)]^T$, $j = 1, \cdots, N/2$. $CCF_{\mathbf{x}^1\mathbf{x}^{ref}}(j)$ is the $j^{th}$ component of $CCF_{\mathbf{x}^1\mathbf{x}^{ref}}$ when performing IFFT on the CSD, and $N$ is number of points of IFFT. Hence, the Hankel matrix can be denoted as:

$$\mathbf{H}(k-1) = \begin{bmatrix} \mathbf{Y}(k) & \mathbf{Y}(k+\alpha) & \cdot\cdot & \mathbf{Y}(k+\beta\alpha-\alpha) \\ \mathbf{Y}(k+1) & \mathbf{Y}(k+\alpha+1) & \cdot\cdot & \cdot\cdot \\ \vdots & & & \\ \mathbf{Y}(k+\alpha-1) & \mathbf{Y}(k+2\alpha-1) & \cdot\cdot & \mathbf{Y}(k+\beta\alpha-1) \end{bmatrix} \tag{3}$$

To identify the natural frequency and vibration pattern of the structure, we only need to calculate two Hankel matrices $\mathbf{H}(0)$ and $\mathbf{H}(1)$ ($\mathbf{H}(0), \mathbf{H}(1) \in \mathbb{R}^{\alpha M \times \beta}$). $\alpha$ and $\beta$ in Eq. 3 correspond to the number of block rows and columns in the Hankel matrix. Typical values of $\alpha$ and $\beta$ are: $\alpha = 20$ and $\beta = 100$, which are large enough to accurately identify the modal parameters [6].

The ERA then performs SVD on $\mathbf{H}(0)$:

$$\mathbf{H}(0) \overset{svd}{=} \mathbf{USV}^T \tag{4}$$

Both $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices. In addition, according to the vibration theory [20], for a structure whose vibration is dominated by the first $n$ modes, the rank of $\mathbf{H}(0)$ is only $2n$ and the singular values in $\mathbf{S}$ has $2n$ non-zero values: $\mathbf{S} = \begin{bmatrix} \mathbf{S}_{2n} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, where $\mathbf{S}_{2n} = diag(d_1, \cdots, d_{2n})$ are singular values of $\mathbf{H}(0)$. Hence, $\mathbf{H}(0)$ can be re-expressed as:

$$\mathbf{H}(0) \overset{svd}{=} \mathbf{U}_{2n}\mathbf{S}_{2n}\mathbf{V}_{2n}^T \tag{5}$$

where $\mathbf{U}_{2n} \in \mathbb{R}^{\alpha M \times 2n}$ and $\mathbf{V}_{2n} \in \mathbb{R}^{\beta \times 2n}$ are the first $2n$ columns of the matrices $\mathbf{U}$ and $\mathbf{V}$, respectively.

Finally, ERA uses Eq. 5 and the Hankel matrix $\mathbf{H}(1)$ constructed from Eq. 3 to find two matrices $\mathbf{A}$ and $\mathbf{\Gamma}$:

$$\mathbf{A} = \mathbf{S}_{2n}^{-1/2}\mathbf{U}_{2n}^T\mathbf{H}(1)\mathbf{V}_{2n}\mathbf{S}_{2n}^{-1/2}, \mathbf{\Gamma} = [\mathbf{I}_{2n}, \mathbf{0}]\mathbf{U}_{2n}\mathbf{S}_{2n}^{-1/2} \tag{6}$$

With performing the eigen decomposition on $\mathbf{A}$: $\mathbf{A} = \mathbf{\Phi}\Lambda\mathbf{\Phi}^{-1}$, the natural frequencies can be obtained according to the eigen values in $\Lambda$, and the vibration pattern can be calculated as $\mathbf{\Gamma\Phi}$.

### 3.3 Distribution of ERA

Generally, an in-network processing operation or function can be denoted as $F(\{data\})$, where $data$ can be raw data sampled by sensor nodes or intermediate data which is the computation result of another in-network processing function. According to ERA, the raw data sampled by sensor nodes follows such a process: *raw data*→*CCF*→*H(0)*→$\mathbf{A}$&$\mathbf{\Gamma}$ [1]. Since there are $M$ raw data, $M$ $CCF_{\mathbf{x}^i\mathbf{x}^{ref}}$ but unique *H(0)* and unique $\mathbf{A}$&$\mathbf{\Gamma}$, we partition the computation of ERA with a framework shown in Fig. 1. From Fig. 1, each node pre-processes its raw data to be a *CCF* data (i.e., $CCF_{\mathbf{x}^i\mathbf{x}^{ref}}$) with the Welch's method mentioned above. To facilitate each node to get the reference raw data $\mathbf{x}^{ref}$ for executing Welch's method, we take one sensor node's raw data as $\mathbf{x}^{ref}$ and let the node broadcast the data to all other nodes. Though the broadcast brings extra communication cost, it will facilitate sensor nodes to deliver the small-size *CCF* (equal to $N/2$) instead of the large-size raw data (about $N * n_d/2$ when $50\%$ overlap of the segments).

After pre-processing the raw data into *CCF* data, the network then begins to in-network process the SVD computation (i.e., Equation 3 and 5) on Hankel matrix $H(0)$ which is constructed with the input CCF data. To realize the in-network processing, we partition the computation of Equation 3 and 5 into three functions: $F_1(\{CCF\})$, $F_2(CCF, USV)$, and $F_3(\{USV\})$. In this context, the *CCF* data is regarded as the "raw data" while the *USV* data (i.e., the result of SVD computation) is regarded as the "intermediate data". The three functions are defined with *Algorithm* 1, 2, and 3, respectively. It can be seen, with the three in-network processing functions, each intermediate sensor node transmits one *USV* data instead of a group of *CCF* data.

**Algorithm 1** $F_1(\{CCF\})$

1: **Input:** *CCF* data
2: Establish Hankel matrix with the *CCF* data using Eq. 3
3: Perform SVD on the matrix using Eq. 5
4: **Return:** *USV* data

Note that, with the increment of the matrix's size, the computation cost of $F_3(\{USV\})$ at the intermediate sensor node may be much high, as the node needs to compute two times of reversing SVD and then perform SVD on the merged matrix. If the computation capability of nodes in WSNs is limited, $F_3(\{USV\})$ may not be allowed to run on the sensor nodes. This restriction will highly affect the

1. *H(1)* can be obtained with *H(0)* and $\mathbf{Y}(1+\beta\alpha)$

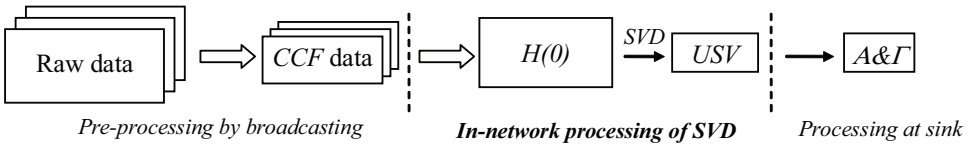



Fig. 1 Distribution of ERA.

**Algorithm 2** $F_2(CCF, USV)$

1: **Input:** *CCF* data and *USV* data
2: **if** The corresponding Hankel matrix of the *USV* data contains no more than 4 nodes' *CCF* data **then**
3: Calculate Hankel matrix with the *USV* data using Eq. 5
4: Update the Hankel matrix with *CCF* data using Eq. 3
5: Perform SVD on the Hankel matrix using Eq. 5
6: **else**
7: Directly update the *USV* data with the *CCF* data using methods in [1], [7]
8: **end if**
9: **Return:** updated *USV* data

**Algorithm 3** $F_3(\{USV\})$

1: **Input:** *USV* data
2: Calculate the corresponding Hankel matrixes with the *USV* data using Eq. 5
3: Merge the Hankel matrixes into one using Eq. 3
4: Perform SVD on the new Hankel matrix using Eq. 5
5: **Return:** updated *USV* data

routing design, which motivates us to design a novel routing scheme in Section 4.2.

### 3.4 Analysis of the typical computation mode

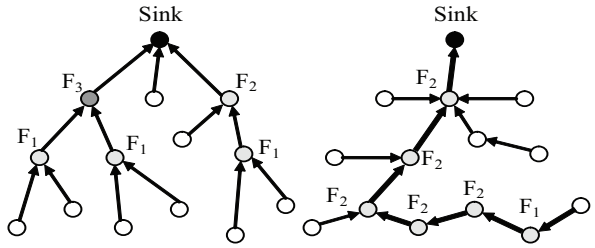
As we are concerned with the transmission cost in the in-network processing, here we analyze the sizes of the three in-network processing functions' results. Denote $|*|$ as the data size of $*$. The size of *CCF* at each node is $|CCF| = N/2$, and the size of *USV* of Hankel matrix for $i$ nodes is $|USV_i| = \alpha i * 2n + (2n)^2 + 2n * \beta$, where the three terms correspond to the data size of $\mathbf{U}_{2n}$, $\mathbf{S}_{2n}$ and $\mathbf{V}_{2n}$, respectively. Considering $N = 4096$, $n = 5$, $\alpha = 20$ and $\beta = 100$, $|USV_i| \approx iN/20 + P$, where $P = (2n)^2 + 2n * \beta = 1100$ is a constant value. It can be seen, transmitting *USV* instead of a group of *CCF* data is much more efficient. Moreover, it can be seen, for each in-network processing of SVD with one node's *CCF* data, the size of the *USV* always increases with a constant $r = \frac{N}{20}$. Based on the analysis, we have:

$$|F_1(CCF, CCF)| = |USV_2| = P + 2 * r \quad (7)$$

$$|F_2(CCF, USV_i)| = |USV_{i+1}| = P + (i+1) * r \quad (8)$$

$$|F_3(USV_i, USV_j)| = |USV_{i+j}| = P + (i+j) * r \quad (9)$$

From the equations above, the size of the in-network processing functions is always **a constant value $P$ plus a variant, and the variant is always proportional to the number of *CCF* data involved**. We name this typical computation mode as ***partial processing***.




(a) Tree-based processing (b) Chain-based processing

Fig. 2 Two types of in-network processing.

## 4 Optimal In-network Processing of SVD

In this section, we discuss how to optimally in-network process the SVD of ERA in WSN so as to minimize the overall transmission cost in the network. Since we've established the in-network processing functions for the SVD, our target is actually to find the optimal in-network processing routing for SVD.

We consider two cases for the routing design. If $F_3(\{USV\})$ is acceptable in WSNs, sensor nodes can process two intermediate data, which means each node can serve as parent node for two non-leave neighboring nodes. Hence, a tree-based routing can be designed for the in-network processing, as shown in Fig. 2 (a). However, in some applications of WSNs, the sensor nodes' computation resources may be too limited. Hence, $F_3(\{USV\})$ may not be acceptable in these WSNs, due to the high computation cost of $F_3(\{USV\})$. Under this circumstance, the intermediate data can only be processed with raw data. Thus, the in-network processing has to follow a single path to the sink in WSNs, as shown in Fig. 2 (b). Only the nodes on the path can execute in-network processing with $F_2(CCF, USV)$, while nodes outside of the path have to deliver their raw data to the nodes on the path for processing. We call this kind of routing as chain-based routing.

In the following, we discuss the optimization of these two typical routing patterns for in-network processing, respectively.

### 4.1 Optimal tree-based in-network processing for SVD

For tree-based in-network processing, our target is to design the optimal tree for the in-network processing. We notice that $F_1(\{CCF\})$ can actually be executed with only one input *CCF* data. Since $|F_1(CCF)| < |CCF|$, leaf nodes of an in-network processing tree should process their local *CCF* first, and then transmit the processing results (i.e., the *USV*) to their parents. The parent nodes do not need process their local *CCF* data in advance. Only after they receive the *USV* data from their children nodes, they begin to process the *USV* data and their local *CCF* data with $F_2(CCF, USV)$.

As $|F_2(CCF, USV_i)| = |F_3(USV_1, USV_i)|$, nodes assigned with $F_2(CCF, USV)$ is actually equivalent to be assigned with $|F_3(USV_1, USV_i)|$, in view of the transmission cost. Hence, considering the fact that all the data transmitted are *USV* data, we regard that all sensor nodes have gotten their own *USV* data with $F_1(\{CCF\})$ in advance (though the parent nodes do not need to do this). Hence, our task is actually to answer such a general question: how to optimally *"aggregate"* the *USV* data with a unique *"aggregation function"* $F_3(\{USV\})$?

Based on the analysis above, we formulate the problem of finding the optimal in-network processing tree (denoted as *OIT problem*) for SVD (or more generally for the typical *partial processing* computation) as follows.

- ***Given**: a network $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges between neighboring nodes. Initially, each node has data with equal size $P + r$.*
- ***Assume**: the data can always be aggregated by a node with aggregation function $F_3(\{data\})$ whose result's size is $P + (m + 1)r$, where $m$ is the number of descendant nodes sending data to the aggregation node.*
- ***Objective**: construct an optimal aggregation tree rooted by a given sink node, so as to minimize the overall transmission cost in $G$.*

It can be seen, the OIT problem is a special case of the traditional aggregation problems which have been proved to be NP-hard [21]. In the following, we present a sub-optimal tree for OIT, with which the transmission cost is less than $1 + \sqrt{2}$ times that of the optimal tree.

Denote $T_{opt} = (V, E')$ as the optimal in-network processing tree for *OIT problem*, where $E'$ is the set of edges in $T_{opt}$. Let $w(e_i)$ as the weight of edge $e_i$, and $P + r_i$ is the size of data transmitted on $e_i$, where $r_i = k_i * r$ ($k_i$ is an integer). The overall transmission cost of the in-network processing on $T_{opt}$ can be denoted as:

$$C_{opt} = \sum_{e_i \in E'} (P + r_i) w(e_i) \tag{10}$$

We first give a lower bound on $C_{opt}$ as follows.

LEMMA 1. *The cost of using an optimal in-network processing tree for OIT problem is bounded from below by $C_{opt} \geq P \cdot c_{MST} + r \cdot c_{SSP}$, where $c_{MST}$ is the cost of the minimum spanning tree (MST) of all nodes in $V$ (i.e., the sum of all edges' weights in MST), and $c_{SSP}$ is the sum of the costs of all the shortest paths to the sink node.*

*Proof.* According to Equation 10, we have:

$$\begin{aligned} C_{opt} &= \sum_{e_i \in E'} (P + r_i) w(e_i) \\ &= P \cdot \sum_{e_i \in E'} w(e_i) + \sum_{e_i \in E'} r_i \cdot w(e_i) \end{aligned} \tag{11}$$

It can be seen, there are two parts of cost in $C_{opt}$. One is $P \cdot \sum_{e_i \in E'} w(e_i)$, and another is $\sum_{e_i \in E'} r_i \cdot w(e_i)$. Meanwhile, $\sum_{e_i \in E'} w(e_i)$ is the cost of $T_{opt}$. Since $T_{opt}$ contains all nodes in $V$, its cost must be no less than $c_{MST}$, i.e., $\sum_{e_i \in E'} w(e_i) \geq c_{MST}$.

In addition, according to characteristic of *partial processing* computation, the variant part $r_i$ of the data size is cumulated directly during the computation. Since each node has raw data with variant part of size $r$, this part of size will be losslessly cumulated during the in-network processing. Therefore, $\sum_{e_i \in E'} r_i \cdot w(e_i)$ is equivalent to the cost for that all sensor nodes transmit their data with size $r$ to the sink along $T_{opt}$ without any aggregation. It is well known that, gathering all nodes' data to the sink without aggregation will have the minimum cost when the transmissions follow a shortest path tree. Hence, $\sum_{e_i \in E'} r_i \cdot w(e_i)$ must be no less than $r \cdot c_{SSP}$.

Therefore, the lemma follows immediately. □

Since finding $T_{opt}$ is NP-hard, we propose a sub-optimal tree for in-network processing of *partial processing* computation mode. In particular, we establish a *shallow light tree* (SLT) [23] with certain parameter as the sub-optimal tree. SLT is a spanning tree that balances the performance of the SPT ("shallow") and the MST ("light"). The basic idea of constructing SLT is simple: conducting deep first searching (DFS) within MST, if a node is becoming not *"shallow"* to the root, the path from the node to the sink in MST will be replaced by the path from the node to the sink in SPT. Metric for *"shallow"* is defined as the ratio $\alpha$ of the distance from the node to root in MST to that in SPT.

To optimally in-network process SVD with SLT, we need to calculate the optimal parameter $\alpha$ of SLT in a WSN with arbitrary topology. According to [23], given a graph $G = (V, E)$ and a number $\alpha > 1$, a SLT has the following two properties:

- The distance between any node and the root in the SLT is no more than $\alpha$ times the length of the shortest path from that node to the root in $G$.
- The total cost of a SLT $c_{SLT}$ is no more than $\frac{\alpha+1}{\alpha-1}$ times that of the MST of the graph $G$;

**Theorem 1.** *The cost of using a SLT with $\alpha = 1 + \sqrt{2}$ (denoted by $SLT_{1+\sqrt{2}}$) for* OIT problem *is no more than $1 + \sqrt{2}$ times that of the optimal in-network processing tree.*

*Proof.* Since the variant part $r_i$ of the data size is cumulated during the computation in $SLT_{1+\sqrt{2}}$ while the constant part $P$ of the data size is maintained on each edge in $SLT_{1+\sqrt{2}}$, the total cost of using $SLT_{1+\sqrt{2}}$ for *OIT problem* can be expressed as

$$\begin{aligned} C_{SLT(1+\sqrt{2})} &= P \cdot c_{SLT_{1+\sqrt{2}}} + \sum_{u_i \in V} r \cdot path(u_i, t)|_{SLT_{1+\sqrt{2}}} \\ &\leq P \cdot \frac{1 + \sqrt{2} + 1}{1 + \sqrt{2} - 1} c_{MST} + r \cdot (1 + \sqrt{2}) c_{SSP} \\ &= P \cdot (1 + \sqrt{2}) c_{MST} + r \cdot (1 + \sqrt{2}) c_{SSP} \\ &\leq (1 + \sqrt{2}) C_{opt} \end{aligned} \tag{12}$$

Therefore, the theorem follows immediately.

□

### 4.2 Optimal chain-based in-network processing for SVD

For the optimal chain-based in-network processing of SVD, our target is to find an optimal path for the

nodes executing the chain-based in-network processing (i.e., $F_2(CCF, USV)$) as well as the optimal routes for the remaining nodes without in-network processing in WSN, so as to minimize the overall transmission cost in the network. For convenience, we call the path for nodes executing the chain-based in-network processing as *in-network processing path*.

Finding the optimal *in-network processing path* is still a NP-complete problem, as it's easy to reduce the travelling salesman problem (TSP) to the problem (by assuming $r \to 0$). Hence, we design the heuristic algorithms for the problem. Meanwhile, we design a greedy algorithm and a simulated annealing algorithm.

#### 4.2.1 The proposed greedy algorithm

Our motivation on designing the greedy algorithm is based on the observation that: though detouring the *in-network processing path* to pass more nodes for processing their local *CCF* data will save transmission cost at the beginning, the size of the processing result increases gradually and will become much larger than that of raw data, making it finally not *worthy* to detour the path across even one more node for in-network processing. In other word, it should eventually follow the shortest path to the sink. Hence, to design the greedy algorithm, we assume a shortest path as the initial *in-network processing path* at first, and then gradually modify the path by detouring it step by step as long as the overall transmission cost in WSN can be reduced.

Hence, a key issue to implement the greedy algorithm is the calculation of the metric, i.e., the overall transmission cost in WSNs when given any *in-network processing path*. To calculate it, it is needed to ascertain the corresponding optimal routes for other nodes outside of the path. To this end, we notice that: the transmission load on the *in-network processing path*, i.e., $P + m * r$, is determined only by the number of nodes (i.e., $m$) delivering data to the path. In view of the size of the transmission load, we can regard that: once a node's *CCF* data is delivered to the *in-network processing path*, it will be processed to be an independent virtual data with size $r$, and the virtual data will be directly sent to sink along the path without being further processed.

Thus, to calculate the overall transmission cost in WSNs for a given *in-network processing path*, we separately calculate the transmission cost for the delivery of each *CCF* data from its source node to the sink, including the *CCF* data transmission stage and the virtual data transmission stage. To minimize the cost of *CCF* data transmission stage of a node (say $n_i$), we need to select the optimal node (say $n_j$) on the *in-network processing path* to process $n_i$'s *CCF*, and the delivery of $n_i$'s *CCF* data must follow the shortest path from $n_i$ to $n_j$. In this way, the optimal routing for $n_i$'s *CCF* transmission stage is obtained, and the total cost for the delivery of $n_i$'s *CCF* to the sink via $n_j$ can be calculated. We present the process on calculating the overall transmission cost in WSNs for a given *in-network processing path* with *Algorithm* 4.

Another issue is how to select the initial *in-network processing path* (or the start node $n_1$). To this end, the algorithm searches each node's shortest path to the sink in advance to calculate the corresponding overall cost. The shortest path with the minimum overall cost is selected as the initial *in-network processing path*.

**Algorithm 4** Calculation of the overall transmission cost in WSNs for a given *in-network processing path*

**Input:** the *in-network processing path* $\mathcal{P} = \{n_1, n_2, \ldots, n_k = sink\}$
Initialize overall transmission cost in WSN $C_{all}(\mathcal{P}) = 0$
**for** each node $n_i \in V - \{n_1, n_2, \ldots, n_k = sink\}$ **do**
  **for** each node $n_j \in \mathcal{P}$ **do**
    Calculate the length $L_{ij}$ of the shortest path between $n_i$ and $n_j$ in $G$.
    Calculate the length $l_{jk}$ of the path between $n_j$ and $n_k$ in $\mathcal{P}$.
    Calculate the cost for delivery $n_i$'s *CCF* to sink via $n_j$ as $C^j_{n_i} = (P + r) \cdot L_{ij} + r \cdot l_{jk}$.
  **end for**
  Calculate the minimum cost for delivery $n_i$'s *CCF* to sink as $C_{n_i} = min\{C^j_{n_i}\}$, $j \in [1, k]$.
  Record the corresponding path as the optimal routing for $n_i$'s *CCF* delivery.
  $C_{all}(\mathcal{P}) = C_{all}(\mathcal{P}) + C_{n_i}$
**end for**
**Return:** $C_{all}(\mathcal{P}) = C_{all}(\mathcal{P}) + P \cdot l_{1k} + r \cdot \sum_{m=1}^{k-1} l_{mk}$

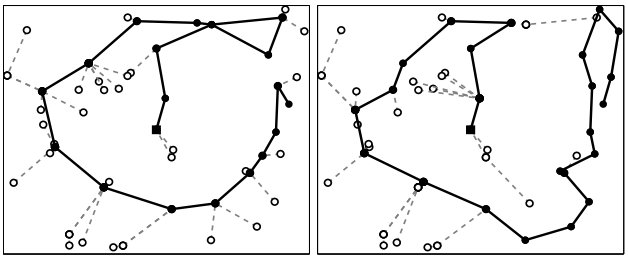
Hence, with the basic ideas introduced above, we formally present the proposed greedy algorithm with *Algorithm* 5.

**Algorithm 5** The proposed greedy algorithm

**for** each node $n_i \in G$ **do**
  Calculate the shortest path between $n_i$ and sink, and record the path as $\mathcal{P}_i$.
  Execute *Algorithm* 4 with $\mathcal{P}_i$, and get $C_{all}(\mathcal{P}_i)$.
**end for**
$C_{all} = min\{C_{all}(\mathcal{P}_i)\}$
Record the node whose corresponding overall cost is $C_{all}$ as $n^1$, and record the corresponding path as $\mathcal{P}^1$.
**for** $j : 1 \to |V|$ **do**
  **for** each neighbor node $n^j_t$ of $n^j$ that is not on $\mathcal{P}^j$ **do**
    Find the shortest path $\mathcal{P}_t$ between $n^j_t$ and the sink.
    Set combined path $\mathcal{P}'_t = \{n^1, n^2, \ldots, n^j, n^j_t\} \leftrightarrow \mathcal{P}_t$, where $* \leftrightarrow \star$ means to connect $*$ to $\star$.
    Execute *Algorithm* 4 with $\mathcal{P}'_t$, and get $C_{all}(\mathcal{P}'_t)$.
  **end for**
  **if** $min\{C_{all}(\mathcal{P}'_t)\} < C_{all}$ **then**
    Record the $n^j$'s neighbor node whose corresponding cost is $min\{C_{all}(\mathcal{P}'_t)\}$ as $n^{j+1}$, and record the corresponding combined path as $\mathcal{P}^{j+1}$.
    $C_{all} = min\{C_{all}(\mathcal{P}'_t)\}$
  **else**
    **Break**, and **return:** $\mathcal{P}^j$.
  **end if**
**end for**

#### 4.2.2 The simulated annealing algorithm

To avoid the local optimum in the greedy algorithm, we propose a two-tier simulated annealing algorithm as follows. In the inside tier of the simulated annealing algorithm, a given initial *in-network processing path* randomly detours to a neighbor node with probability related to the corresponding overall transmission cost. When the inside tier of the annealing process terminates, it returns the overall transmission cost for the final *in-network processing path*. In the outside



(a) The proposed greedy algorithm with cost $2.35 * 10^6$units and time 1.9s

(b) The simulated annealing algorithm with cost $2.29*10^6$units and time 156s

Fig. 3 The *in-network processing path* with two heuristic algorithms.

tier, we randomly select an initial shortest path as the input of the inside tier with probability related to the return of the inside tier. Detailed description of the two-tier simulated annealing algorithm is omitted for briefness.

Taking an instance of a random deployment with $100$ sensor nodes, Fig. 3 shows the results of the proposed greedy algorithm and the simulated annealing algorithm. It can be seen, though the *in-network processing path* with the proposed greedy algorithm leads to a bit higher overall transmission cost than that with the simulated annealing algorithm, the running time of the greedy algorithm is far smaller than that of the latter.

## 5 Simulations

To evaluate the performance of the proposed schemes, we use random-generated networks by randomly deploying $M$ nodes in area $Z$ with size $L * L$. The transmission range of each node is denoted by $R$. The weight of each edge between neighboring nodes is regarded to be proportional to the edge's length.

We leverage two metrics to evaluate the performance of the proposed schemes. One is the total transmission cost in WSNs. In particular, we define one transmission unit as the transmission cost for a node transmitting one bit information. The sizes of raw data and intermediate data are calculated according to section 3.4. Another metric is the running time of the proposed schemes. We execute the schemes on a computer with CPU type Intel $i3$-$2330M$ $2.2GHz$ and $2Gbyte$ of RAM, and measure the running time with Matlab7.0 tool.

### 5.1 Simulations for tree-based in-network processing

We first compare the transmission cost of the proposed in-network processing scheme with the traditional centralized ERA. For convenience, we name the proposed scheme as decentralized ERA in the simulations. As the decentralized ERA contains two phases, i.e., pre-processing and in-network processing, the reference raw data is assumed to be delivered along MST in pre-processing phase, while the *USV* data is aggregated along the proposed SLT in in-network processing phase. As for the traditional centralized ERA, we assume that all the raw data is gathered along SPT in the WSN.

Two scenarios are created to evaluate their performance in different network densities and network scales. We first fix the number of nodes to be $100$ but gradually increase the network density by increasing $R$ from $20$ to $40$. The area in this scenario is fixed to be $120 * 120$. Then we maintain the network density at a certain level but gradually increase the network scale by increasing the number of nodes from $100$ to $500$. Note that, to maintain the network density, when increasing the number of nodes, the size of the deployment area also needs to be increased. In both scenarios, $20$ simulations are performed for each parameter set, and the average value of each scheme is calculated.

Fig. 4(a)(b) shows the transmission cost of the two schemes at different transmission ranges and network scales, respectively. From Fig. 4(a)(b), the transmission cost of the proposed decentralized ERA can be one order of magnitude smaller than that of the centralized ERA. Therefore, with lossless in-network processing, the proposed decentralized ERA does significantly reduce the transmission cost in WSNs without any quality loss for the structural health monitoring. Furthermore, we can see from Fig. 4(b) that, the gap between the two schemes' costs increases sharply with the network scale, showing that the advantage of the decentralized ERA becomes more remarkable in large scale of WSNs.

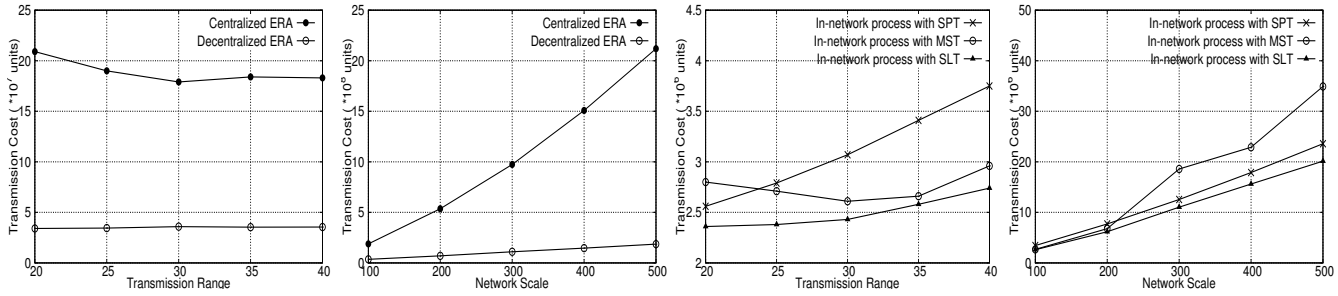
We further specially evaluate the transmission cost of the in-network processing phase in the proposed decentralized ERA. In particular, we compare the transmission cost of the proposed SLT-based in-network processing with those of the SPT-based in-network processing and the MST-based in-network processing. The simulation results are shown in Fig. 4(c)(d). It can be seen, the transmission cost of SLT-based in-network processing is always smaller than those of the SPT-based in-network processing and the MST-based in-network processing. Meanwhile, the cost of SLT-based in-network processing and that of SPT-based in-network processing increase steadily with the network density and network scale, while the cost of MST-based in-network processing various irregularly and sometime can be lower than that of SPT-based in-network processing.

### 5.2 Simulations for chain-based in-network processing

For chain-based in-network processing, we only compare the performance of the proposed greedy algorithm with that of the proposed simulated annealing algorithm in the in-network processing phase. To get the ground truth for reference, we also implement the brute force algorithm for optimal chain-based in-network processing in small scale in the simulation. We measure the transmission cost and the running time of the three algorithms.

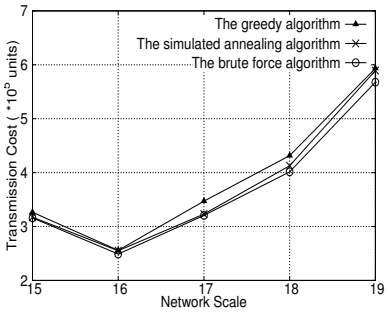
Fig. 5 shows the results of the three algorithms with slightly increasing the network scale from 15 to 19. It can be seen, the transmission costs of the proposed greedy algorithm and the simulated annealing algorithm are both very close to the optimal cost. However, the running time of the greedy algorithm is far smaller than that of annealing algorithm. And, when the network scale increases to $19$, the running time of the brute force algorithm is more than one hour, while that of the simulated annealing algorithm is $4.3s$ and that of the greedy algorithm is just $0.34s$.

We further evaluate the performance of the greedy algorithm and the simulated annealing algorithm in different

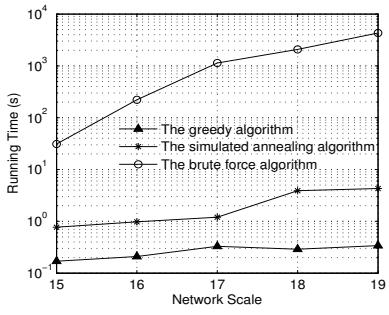



(a) (b) (c) (d)

Fig. 4 Simulation results for tree-based in-network processing.

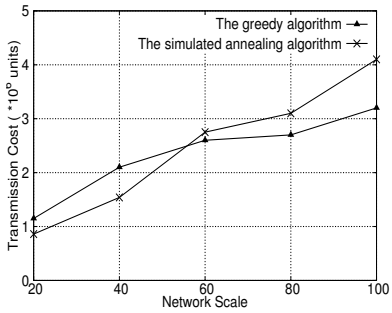



(a) Transmission Cost vs. Scale

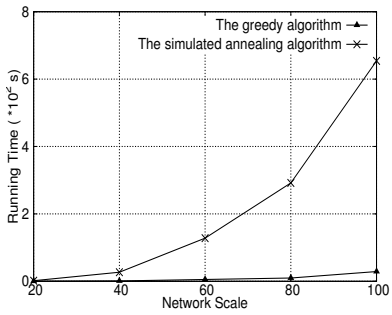



(b) Running Time vs. Scale

Fig. 5 Chain-based in-network processing with small scale

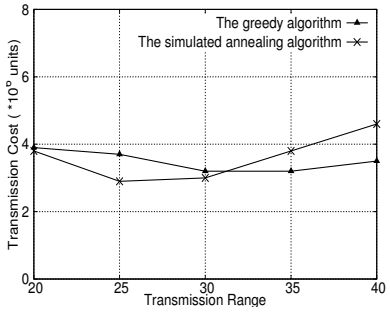



(a) Transmission Cost vs. Scale

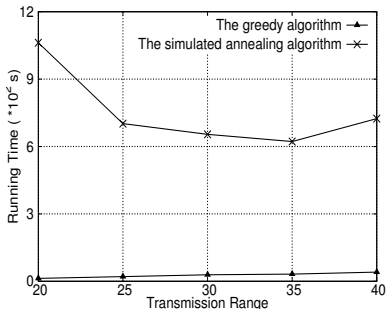



(b) Running Time vs. Scale

Fig. 6 Chain-based in-network processing with different scales




(a) Transmission Cost vs. Density




(b) Running Time vs. Density

Fig. 7 Chain-based in-network processing with different densities

network densities and network scales. Parameters are set in a similar way to that for tree-based in-network processing above. Fig. 6 and Fig. 7 show the simulation results. From Fig. 6, with the increment of network scale, both the transmission costs and running times of the two algorithms increase as expected. Meanwhile, the transmission cost with the greedy algorithm is even lower than that with the simulated annealing algorithm when the network scale is over 60. This is because larger network scale tends to result in unsteady performance of the simulated annealing algorithm. As for the performance of running time, the greedy algorithm far outperforms the simulated annealing algorithm, as shown in Fig. 6(b).

From Fig. 7, with the increment of transmission range, the running time with the greedy algorithm increases a little, while that with the simulated annealing algorithm decreases. This is because larger transmission range facilitates the greedy algorithm to have more candidate nodes to detour the in-network processing path, which is more prone to get the optimal path but requires more computation time. However, for the simulated annealing algorithm, larger transmission range facilitates the in-network processing path of the algorithm to arrive earlier at the sink, at the cost of irrational detour of the path. Comparing these two algorithms, the greedy algorithm still have close transmission cost to that with the simulated annealing algorithm while much lower running time than that of the latter.

## 6 Conclusions

In this paper, we discuss the lossless in-network processing of a given complex centralized algorithm in WSNs with arbitrary topology. We take example of a classic SHM algorithm ERA which contains typical feature extraction method SVD, and succeed to distribute ERA into WSNs. Meanwhile, we implement lossless tree-based in-network processing and chain-based in-network processing of SVD, respective for the case of unlimited computation capability and the case of limited computation capability of sensor nodes. Furthermore, we propose to utilize a certain SLT with appropriate parameter as the optimal routing scheme

for the tree-based in-network processing, and prove that the transmission cost with such a SLT can achieve approximation ratio $1 + \sqrt{2}$. For the optimization of the chain-based in-network processing which is a novel problem, we propose two efficient heuristic algorithms. Simulation results validate the efficiency of the proposed algorithms.

## ACKNOWLEDGEMENT

The work presented in this paper was supported in part by the NSF of China with Grant 61572217 and 61572218, and the NSFC/RGC Joint Research Scheme with RGC No: N PolyU51912 and NSFC Key Grant with project No: 61332004. Xuefeng Liu is the corresponding author.

## REFERENCES

[1] Xuefeng Liu, Jiannong Cao, Wen-zhan Song, Peng Guo, Zongjian He, *Distributed Sensing for High-Quality Structural Health Monitoring using WSNs*, IEEE Transactions on Parallel and Distributed System, vol. 26, no. 3, pp. 738-747, 2015.

[2] Y.-F. Huang, S. Werner, and et al., *State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid*, IEEE Signal Processing Magazine, vol. 29, no. 5, pp. 33-43, 2012.

[3] H. T. Pham and B.-S. Yang, *Estimation and forecasting of machine health condition using arma/garch model*, Mechanical Systems and Signal Processing, vol. 24, no. 2, pp. 546-558, 2010.

[4] S. Doebling, *Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review*, Los Alamos National Lab., Tech. Rep., 1996.

[5] B. Yu, J. Li, and et al., *Distributed data aggregation scheduling in wireless sensor networks*, in INFOCOM, 2009, pp. 2159-2167.

[6] J. Juang and R. Pappa, *Eigensystem realization algorithm for modal parameter identification and model reduction*, Journal of Guidance, Control, and Dynamics, vol. 8, no. 5, pp. 620-627, 1985.

[7] H. Zha and H. Simon, *On updating problems in latent semantic indexing*, SIAM Journal on Scientific Computing, vol. 21, no. 2, pp. 782-791, 1999.

[8] Julio Cesar Stacchini de Souza, Tatiana Mariano Lessa Assis, Bikash Chandra Pal, *Data Compression in Smart Distribution Systems via Singular Value Decomposition*, IEEE Transactions on Smart Grid, Vol., No.1, 2017, pp. 275-284.

[9] Muhammad Tayyab Asif ; Srinivasan Kannan ; Justin Dauwels ; Patrick Jaillet, *Data Compression Techniques for Urban Traffic Data*, IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2013.

[10] Yi Wang; Qixin Chen; Chongqing Kang; Qing Xia; Yuekai Tan; Zhijian Zeng; Min Luo, *Residential Smart Meter Data Compression and Pattern Extraction via Non-negative K-SVD*, IEEE Power and Energy Society General Meeting (PESGM), 2016.

[11] Yan Wu ; Dept. of Comput. Sci., Purdue Univ., West Lafayette, IN ; Fahmy, S. ; Shroff, N.B., *On the Construction of a Maximum-Lifetime Data Gathering Tree in Sensor Networks: NP-Completeness and Approximation Algorithm*, in INFOCOM, Phoenix, AZ, Apr. 2008.

[12] Weifa Liang, and Yuzhen Liu, *Online Data Gathering for Maximizing Network Lifetime in Sensor Networks*, IEEE Transactions on Mobile Computing, vol. 6, no. 1, 2007, pp. 2-11.

[13] XiaoHua Xu, Mo Li, XuFei Mao, Shaojie Tang, *A Delay-Efficient Algorithm for Data Aggregation in Multihop Wireless Sensor Networks*, IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 1, 2011, pp. 163-175.

[14] G. Yan, W. Guo, S. Dyke, G. Hackmann, and C. Lu, *Experimental validation of a multi-level damage localization technique with distributed computation*, Smart Structures and Systems, vol. 6, no. 5-6, pp. 561-578, 2010.

[15] A. Zimmerman and M. Shiraishi, *Automated modal parameter estimation by parallel processing within wireless monitoring systems*, Journal of Infrastructure Systems, vol. 14, no. 1, pp. 102-113, 2008.

[16] X.Liu, J. Cao, and et al., *Energy efficient clustering for wsn-based structural health monitoring*, in IEEE INFOCOM, 2011, pp. 1028-1037.

[17] R. Brincker, L. Zhang, and P. Andersen, *Modal identification from ambient responses using frequency domain decomposition*, in Proceedings of the 18th international modal analysis conference, 2000, pp. 625-630.

[18] T. Fu, A. Ghosh, E. Johnson, and B. Krishnamachari, *Energy-efficient deployment strategies in structural health monitoring using wireless sensor networks*, Structural Control and Health Monitoring, 2011.

[19] Y. Gao, B. Spencer Jr, and M. Ruiz-Sandoval, *Distributed computing strategy for structural health monitoring*, Structural control and health monitoring, vol. 13, no. 1, pp. 488-507, 2006.

[20] D. Inman, *Engineering vibrations*, Prentice Hall, 2006.

[21] A. Goel and D. Estrin, *Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk*, Proc. ACM/SIAM Symp. Discrete Algorithms, pp. 499-505, 2003.

[22] F. Harris, *On the use of windows for harmonic analysis with the discrete fourier transform*, Proc. of the IEEE, vol. 66, no. 1, pp. 51-83, 1978.

[23] S. Khuller, B. Raghavachari, N. Young, *Balancing minimum spanning trees and shortest-path trees*, Algorithmica, Vol. 14, no. 4, pp 305-321, 1995.

**Peng Guo** received his M.S. and Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2008, respectively. He is currently an Associate Professor at the school of Electronic Information and Communications in Huazhong University of Science and Technology. His research interests include wireless sensor networks, distributed computing and in-network processing. He has served as a reviewer for several international journals/conference proceedings.

**Jiannong Cao** received the MSc and PhD degrees in computer science from Washington State University, Pullman, Washington, in 1986 and 1990, respectively. He is currently the head and chair professor in the Department of Computing at Hong Kong Polytechnic University, Hong Kong. His research interests include parallel and distributed computing, mobile computing and big data analytics. He is a IEEE Fellow and a senior member of the China Computer Federation. He has served as a member of editorial boards of several international journals, a reviewer for international journals/conference proceedings, and also as an organizing/program committee member for many international conferences.

**Xuefeng Liu** received his M.S. and Ph.D. degree from Beijing Institute of Technology, China, and University of Bristol, UK, in 2003 and 2008, respectively. He is currently an Associate Professor at the school of Electronic Information and Communications in Huazhong University of Science and Technology. His research interests include wireless sensor networks and in-network processing. He has served as a reviewer for several international journals/conference proceedings.