Research Article

Constructing Maximum-Lifetime Data-Gathering Tree in WSNs Based on Compressed Sensing

Zhengyu Chen,¹,² Geng Yang,² Lei Chen,² and Jian Xu²

¹School of Electronic and Information Engineering, Jinling Institute of Technology, Nanjing 211169, China
²Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing, Jiangsu 210003, China

Correspondence should be addressed to Zhengyu Chen; zych@jit.edu.cn

Received 3 February 2016; Accepted 26 April 2016

Academic Editor: Jun Luo

Copyright © 2016 Zhengyu Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Data gathering is one of the most important operations in many wireless sensor networks (WSNs) applications. In order to implement data gathering, a tree structure rooted at the sink is usually defined. In most wireless sensor networks, nodes are powered by batteries with limited energy. Prolonging network lifetime is a critical issue for WSNs. As a technique for signal processing, compressed sensing (CS) is being increasingly applied to wireless sensor networks for saving energy. Compressive sensing can reduce the number of data transmissions and balance the traffic load throughout networks. In this paper, we investigate data gathering in wireless sensor networks using CS and aim at constructing a maximum-lifetime data-gathering tree. The lifetime of the network is defined as the number of data-gathering rounds until the first node depletes its energy. Based on the hybrid-CS data-gathering model, we first construct an arbitrary data-gathering tree and then use the random switching decision and optimal parent node selecting strategy to adjust the load of the bottleneck node and prolong the network lifetime. Simulation results show that the proposed algorithm outperforms several existing approaches in terms of network lifetime.

1. Introduction

Wireless sensor networks (WSNs) consist of a great number of nodes that sense the environment and collaboratively work to process and route the sensory data. They have a great application value in the fields of habitat monitoring, medical care, battlefield monitoring, and so on. Data gathering is a basic operation in WSNs, where sensors are responsible for collecting all sensory data and delivering them to the sink node [1]. Typically, sensors have strictly limited communication abilities and energy resources; therefore, how to reduce energy consumption in data gathering so as to prolong network lifetime is an important issue [2].

The problem of energy efficient data gathering in WSNs has been extensively investigated. Data-gathering methods based on cluster and tree are proposed in many literatures [3, 4]. The goal of such methods is to construct a network topology in order to use energy resources of sensor node effectively. However, these methods cannot overcome the “hot pot” problem; that is, nodes close to the sink would suffer from heavy loads and their energy would be exhausted soon, which would shorten the network lifetime. Data aggregation [5] and source coding technique [6] are two efficient methods to overcome the “hot pot” problem. However, data aggregation adopts a simple aggregation function that only extracts certain statistical information from the sensory data. The sink node cannot recover all sensory data. Hence, this technique can only be applied to particular applications that require partial information from WSNs. The distributed source coding technique performs noncollaborative data compression at the source nodes; it is not exactly practical in WSNs either due to the lack of prior knowledge of the data correlation or because of resulting in high communication load.

Compressed sensing is a signal processing technique for efficiently acquiring and reconstructing a signal, by finding solutions to underdetermined linear systems [7]. This technique promises to successfully recover the original signal
from far fewer measurements than its original dimension, as long as the signal is sparse or compressible in some domain. Recently, the growing interest in CS has inspired many works in WSNs [8–11]. Many researches show that the CS technique suits well data gathering in WSNs because the sensory data is generally quite sparse in nature [8]. The authors in [12] made the first effort to speculate the potential of using CS theory for data gathering in multihop WSNs. However, they have not proposed a real scheme based on this initial idea. The authors in [8] firstly presented a complete framework for data gathering based on CS in large-scale WSNs where a large number of sensor nodes are densely deployed and sensor readings are spatially correlated. In Compressive Data Gathering (CDG) proposed in [8], the sensory data are transmitted as weighted sums on a routing path so as to reduce global communication cost without introducing intensive computation or complicated transmission control. However, CDG can result in many unnecessary data transmissions during data-gathering procedure.

In [13], the authors introduced two data-gathering models, that is, plain-CS and hybrid-CS. The plain-CS model which is the same as the CDG scheme uses CS encoding for all nodes in the network. The hybrid-CS model applies CS only to relay nodes that are overloaded. The hybrid-CS model marries the merits of nonaggregation data gathering and plain-CS scheme. Recently, Xiang et al. in [14] introduced a Minimum Energy Compressed Data Aggregation (MECDA) algorithm to implement the data gathering in WSNs based on hybrid-CS technique. The aim of MECDA is to allocate the traffic load properly in a given data-gathering round, through joint routing and aggregator assignment, such that the total energy consumption is minimized. Unfortunately, MECDA only takes into account the overall energy consumption of sensor network and does not consider balancing the energy consumption of each node; therefore, the energy of some sensor nodes may be exhausted quickly, which will affect the network lifetime eventually. Therefore, it is significant for the researchers to design a maximum-lifetime data-gathering method which balances the energy consumption of sensor nodes by using CS theory in designing of underlying routing.

In this paper, we propose a method of constructing maximum-lifetime data-gathering tree (MLDGT) in WSNs based on the hybrid-CS data-gathering model. The contributions of this paper are as follows:

(1) We first define the problem of maximum-lifetime data-gathering tree in WSNs based on compressed sensing, prove that it is NP-complete, and then analyze the characterizations of the optimal solutions.

(2) We propose a novel algorithm, which used random switching and optimal parent node selecting strategy to keep transferring descendants of bottleneck node to realize load balancing and extending the lifetime of WSNs.

(3) Extensive experiments have been conducted to verify that our algorithm achieves higher lifetime than existing approaches.

The rest of the paper is organized as follows. In Section 2, we give a brief overview of applying compressed sensing in data gathering of WSNs. Section 3 formally defines the problem of maximum-lifetime data-gathering tree and analyzes the complexity of the problem. The proposed algorithms are detailed and analyzed in Section 4. Simulation results are presented in Section 5 and conclusions are drawn in Section 6.

2. Preliminaries

2.1. Compressed Sensing Basics. In CS framework, the real-valued signal \( X \in \mathbb{R}^N \) can be reconstructed from a far fewer measurements than its original dimension \( N \) as long as the signal \( X \) is sparse or compressible in a certain domain. Let \( X = [x_1, x_2, \ldots, x_N]^T \) be \( N \)-dimensional original data that can be represented in some orthonormal basis \( \Psi = [\psi_i]_{i=1}^N \) (let \( \psi_i \) be \( N \)-dimensional column vector) as \( X = \sum_{i=1}^N \theta_i \psi_i = \mathbf{\Psi} \theta \). If the coefficient vector \( \theta = [\theta_1, \theta_2, \ldots, \theta_N]^T \) is \( k \)-sparse, that is, there are only \( k \) (\( k < N \)) nonzero entries in vector \( \theta \), the signal is \( k \)-sparse in basis \( \Psi \). Let \( \Phi = [\phi_1, \phi_2, \ldots, \phi_N] \) be the \( M \times N \)-dimensional measurement matrix whose row vectors are largely incoherent with \( \Psi \). Suppose \( Y \) is the measurements of the \( k \)-sparse \( X \), that is, \( Y = \Phi X = \Phi \Psi \theta \).

According to CS theory, we can recover \( X \) from \( Y \) by solving the following optimization problem:

\[
\min_{\theta \in \mathbb{R}^N} ||\theta||_0 \quad \text{s.t.} \quad Y = \Phi \Psi \theta.
\]

Suppose \( \hat{\theta} \) is the optimal solution of the optimization problem (1). The recovery signal of \( X \) can be obtained by

\[
\hat{X} = \Psi \hat{\theta}.
\]

The difference between the original signal \( X \) and recovery signal \( \hat{X} \) represents the practical performance of CS coding, which depends on the scarcity of the original signal, as well as on the reconstruction algorithm. Because the sensory data of WSNs is generally quite sparse in nature, CS suits well data gathering in WSNs [15, 16]. In this paper, we define the ratio of the dimension of measurement vector \( Y \) to the dimension of original sensory data vector \( X \) as sampling rate \( \lambda \); that is, \( \lambda = M/N \). The sampling rate is an important parameter that affects the energy consumption of nodes in a data-gathering tree that is constructed based on CS.

2.2. Overview of Data Gathering Based on Compressed Sensing in WSNs. For the traditional data-gathering method of WSNs, nodes around the sink need to transmit more raw data and consume more energy than the downstream nodes of the sensor network. The unbalanced energy consumption has a major impact on network lifetime. In [8], Luo et al. proposed CDG for data gathering in large-scale WSNs which is an efficient way to alleviate the bottleneck.
To illustrate the idea of data gathering based on CS, we rewrite the CS coding as

$$ Y = \Phi X = x_1\phi_1 + x_2\phi_2 + \cdots + x_r\phi_r + \cdots + x_n\phi_n, $$

where $x_i$ is sensory data of node $v_i$ and $\phi_i$ is $i$th column vector of measurement matrix $\Phi$. In CDG, instead of individual readings, the weighted sums of sensor readings are delivered to the sink. In the process of data gathering, node $v_i$ first multiplies its reading $x_i$ with $\phi_i$ and then adds the product to the $M$-dimensional column vector $\sum_{j=1}^{M} x_j\phi_j$ which is received from children nodes. The sum $\sum_{j=1}^{M} x_j\phi_j = \sum_{j=1}^{M} x_j\phi_j + x_i\phi_i$ will be sent to the parent node of $v_i$. If each node performs this operation, the compressed measurement will be completed during the process of data gathering. Eventually, the sink collects the $M$-dimension vector $Y$, and then the decoding algorithm is used to recover the $N$ original sensory data. The method mentioned above is referred to as plain-CS model. In plain-CS data gathering, each node needs to send exactly an $N$-dimension vector irrespective of what it has received; there is no “hot pot” phenomenon.

However, the plain-CS model may lead to unnecessary higher traffic at the earlier stage of data gathering. Therefore, Luo et al. in [13] proposed the hybrid-CS data-gathering model. The main idea of hybrid-CS model is that the non-CS scheme is applied in the earlier stages of data gathering, and the CS-based compression is only applied at node whose incoming traffic intensity becomes larger than $M$ or equal to $M$. The data-gathering process based on hybrid-CS model is depicted in Figure 1 through a simple chain-type topology.

In this paper, we focus on maximizing the lifetime of data-gathering trees based on hybrid-CS model. As can be seen from the above analysis, the energy consumption characteristic of sensor node brings about a new challenge for constructing a maximum-lifetime data-gathering tree based on hybrid-CS model.

3. System Model and Problem Formulation

3.1. System Model. We consider a static wireless sensor network with $N$ sensor nodes $\{v_1, v_2, \ldots, v_N\}$ and one sink node $S$. All sensor nodes are randomly deployed over an $A \times A$ Euclidean plane to continuously monitor the environment and periodically report data to the sink. We use an undirected graph $G(V, E)$ to represent the WSNs, where $V$ is the set of all nodes and $E$ is the set of edges. The underlying communication graph is a unit disk graph. All nodes have the same transmission range $r$. There is an edge $(v_i, v_j) \in E$ whenever the Euclidean distance between nodes $v_i$ and $v_j$ satisfies $\|v_i - v_j\| \leq r$. Each node has unique initial energy. Assume that $E(v_i)$ denotes the initial energy of node $v_i$. The sink is AC-powered which has infinite power supply and powerful computation ability. Since data communication is the dominant energy consumption in sensor network, we can only take into account the energy consumption of communication. At each data-gathering round, each node generates a reading which is referred to as one-unit data. We denote by $E_{Rx}$ and $E_{Tx}$ the energy consumption of one-unit data transmission and reception, respectively.

The architecture for hybrid-CS data gathering that we refer to in [13, 14, 17] is shown in Figure 2. A spanning tree rooted at sink is adopted to gather the readings of the whole network. In Figure 2, the arrowed line represents the transmission link, and the data on the left side of each line is ready for transmitting. For leaf nodes, such as nodes $v_1 \sim v_L$ and $v_{L+3} \sim v_{L+6}$, the original sensory data will be transmitted to their parents. For other nodes, the outgoing traffic has two possibilities:

1. When the number of data received from node $v_i$'s children is less than $M$, where $M$ is the dimension of column vector $\phi_i$, node $v_i$ forwards the received and its own data directly. The node $v$ is referred to as forwarding node. In Figure 2, the outgoing traffic of node $v_{L+7}$ is data set $\{x_{i+3}, \ldots, x_{i+6}, x_{i+7}\}$, in which $x_{i+7}$ transmit the raw sensory data $x_1, x_2, \ldots, x_M$ directly, and the amount of transmitting data of each node is no more than $M$. But nodes $n_{M+1}, n_{M+2}, \ldots, n_N$ transmit data based on CS; that is, node $v_L$ ($M < L < N$) transmits data $\sum_{i=1}^{L} x_i\phi_i$, and the transmission load of each node in set $\{n_{M+1}, n_{M+2}, \ldots, n_N\}$ is equal to $M$.

In this paper, we focus on maximizing the lifetime of data-gathering trees based on hybrid-CS model. As can be seen from the above analysis, the energy consumption...
is its own sensory data; the others are received from its children.

(2) When the number of data received from node $v$’s children nodes is more than $M$, node $v$ performs the CS coding according to (3) and transmits exactly $M$ encoded data corresponding to the aggregated column vector. We call node $v$ the aggregating node. In Figure 2, node $v_{i+1}$ is an aggregating node and performs the CS coding $\sum_{j=1}^{i+1} x_j \phi_j$. The outgoing traffic of $v_{i+1}$ is always $M$.

Based on the above analysis, in a given data-gathering tree $T$, the outgoing traffic of node $v$ is

$$R_{Tx} (T, v) = \begin{cases} \sum_{u \in \text{Ch}(v)} R_{Tx} (T, u) + 1, & \text{if } \sum_{u \in \text{Ch}(v)} R_{Tx} (T, u) < M; \\ M, & \text{otherwise}, \end{cases} \quad (4)$$

where Ch($v$) is the set of $v$’s children nodes. Thus, the energy consumption of node $v$ in each data-gathering round is

$$C (T, v) = \begin{cases} E_{Tx} R_{Tx} (T, v) + E_{Rx} \sum_{u \in \text{Ch}(v)} R_{Tx} (T, u), & \text{if node } v \text{ is not a leaf node} \\ E_{Tx}, & \text{otherwise}. \end{cases} \quad (5)$$

3.2. Problem Formulation. In this paper, we define a data-gathering round as the process of gathering all data from nodes to the sink, regardless of how much time it takes [18].

The lifetime of a node $v$ in tree $T$ is defined as the number of data-gathering rounds since it began to work until its death:

$$L_{\text{node}} (T, v) = \frac{E (v)}{C (T, v)}. \quad (6)$$

The lifetime of a tree $T$ is defined as the lifetime of the first dead node in the tree:

$$L_{\text{tree}} (T) = \min_{v \in V} \{L_{\text{node}} (T, v)\}. \quad (8)$$
For any network $G$, we refer to $S(G)$ as the set of all possible trees for $G$. The problem of finding a maximum-lifetime data-gathering tree (MLDGT) based on hybrid-CS model can be defined as

$$
\max_{T \in S(G)} L_{\text{tree}}(T) = \min_{T \in S(G), v \in V} \{L_{\text{node}}(T, v)\}. \quad (9)
$$

This problem that is similar to the maximum-lifetime problem proposed in [18–20] is an NP-complete problem. The unique property of the problem is the energy consumption model of the node in data-gathering tree based on hybrid-CS model, which brings about new challenges for constructing data-gathering tree. Since the problem is NP-complete, we try to find an approximate solution to solve the problem. Details of the solution techniques are shown in the next section.

4. Solution Techniques

4.1. Basic Ideal of Solution Techniques. To facilitate the analysis, we first provide some relevant assumptions and definitions in this section. We denote by $P(T, v_i)$ the path from node $v_i$ to sink $S$ in spanning tree $T$. The expression $v_i \in P(T, v_j)$ represents the notion that node $v_j$ is a relaying node on the path $P(T, v_i)$. In a spanning tree $T$, the nodes on the path $P(T, v_j)$ may have different loads. We define path load, $l_{\text{path}}(T, v_i)$, of node $v_i$ as the maximum load of all nodes along the path from node $v_i$ to sink $S$; that is,

$$
l_{\text{path}}(T, v_i) = \max \{l_{\text{node}}(T, v_j) \mid v_j \in P(T, v_i)\}. \quad (10)
$$

And the node with the largest load on the path $P(T, v_j)$ is called bottleneck node on the path $P(T, v_j)$ of node $v_j$.

Formula (9) can be transformed into an equivalent form:

$$
\max_{T \in S(G)} L_{\text{tree}}(T) \iff \\
\min_{T \in S(G), v \in V} \left\{ \frac{1}{L_{\text{node}}(T, v)} \right\} = \min_{T \in S(G), v \in V} \{l_{\text{node}}(T, v)\}. \quad (11)
$$

Let $Q(T)$ denote the set of all leaf nodes in tree $T$. Assume that $q = |Q(T)|$ is the number of nodes in set $Q(T)$, and the elements of set $Q(T)$ are $u_1, u_2, \ldots, u_q$. Thus, the sensor node set $V$ can be divided into $q$ paths of $q$ leaf nodes; that is,

$$
V = P(T, u_1) \cup P(T, u_2) \cup \cdots \cup P(T, u_p) = \bigcup_{u_i \in Q(T)} P(T, u_i). \quad (12)
$$

Based on the division of node set $V$, formula (11) is equal to

$$
\max_{T \in S(G)} L_{\text{tree}}(T) \iff \\
\min_{T \in S(G), v \in V} \{l_{\text{node}}(T, v)\} = \min_{T \in S(G), v \in V} \{l_{\text{node}}(T, v) \mid v \in P(T, u_i)\} = \min_{T \in S(G)} \{\max_{v \in P(T, u_i)} \{l_{\text{node}}(T, v) \mid v \in P(T, u_i)\}\},
$$

$$
\max \{l_{\text{node}}(T, v) \mid v \in P(T, u_2)\}, \ldots, \\
\max \{l_{\text{node}}(T, v) \mid v \in P(T, u_r)\}
$$

$$
\min_{T \in S(G)} \left\{ l_{\text{path}}(T, u_1), l_{\text{path}}(T, u_2), \ldots, l_{\text{path}}(T, u_p) \right\} = \min_{T \in S(G)} \left\{ \min_{u_i \in Q(T)} \max_{v \in P(T, u_i)} \{l_{\text{node}}(T, v) \mid v \in P(T, u_i)\} \right\}.
$$

Therefore, the MLDGT problem is transformed to construct a spanning tree, in which the maximum path load of leaf node is minimal. Inspired by [19], in order to solve the problem of formula (13), we need to find a spanning tree in which the difference between the maximum path load and the minimum path load of leaf node is minimal. Therefore, the MLDGT problem can be described as follows:

$$
\max_{T \in S(G)} L_{\text{tree}}(T) \iff \\
\min_{T \in S(G), v \in V} \{l_{\text{path}}(T, u_i) \mid u_i \in Q(T)\} \iff \\
\min_{T \in S(G)} \left\{ \max_{u_i \in Q(T)} l_{\text{path}}(T, u_i) \mid u_i \in Q(T)\right\} - \min_{T \in S(G)} l_{\text{path}}(T, v) \mid v \in Q(T)\right\}.
$$

(14)

For a given connectivity group $G$, we assume that tree $T_{\text{opt}}$ is an optimal spanning tree which provides the maximum lifetime. Let $\sigma_{\text{opt}}$ be the difference between the maximum path load and the minimum path load of leaf node in tree $T_{\text{opt}}$; that is,

$$
\sigma_{\text{opt}} = \max \left\{ l_{\text{path}}(T_{\text{opt}}, v_i) \mid v_i \in Q(T_{\text{opt}})\right\} - \min \left\{ l_{\text{path}}(T_{\text{opt}}, v_i) \mid v_i \in Q(T_{\text{opt}})\right\}. \quad (15)
$$

Each connectivity group $G$ has a unique $\sigma_{\text{opt}}$ which depends on the node’s initial energy, network deployment, and network connectivity of WSNs. It is an NP-complete problem for finding $\sigma_{\text{opt}}$ of any given connectivity group.

We can enumerate all spanning trees of graph $G$ and find a spanning tree which minimizes the maximum path load of leaf nodes. However, the number of the spanning trees of graph $G$ can be an exponential function of the number of nodes $N$. Therefore, we try to find an approximate solution to minimize the maximum path load of leaf node in the data-gathering tree. Details of the algorithm are described in the next section.

4.2. Algorithm Description. In this section, we design an approximate algorithm to deal with the problem raised by formula (13). Our algorithm uses randomized tree transformation and switching method and optimal parent node selecting strategy that make full use of the property of energy consumption in data gathering based on CS to improve convergence speed of the algorithm.
Our algorithm starts from an arbitrary tree $T$ and then keeps transferring descendants of bottleneck node to nodes with smaller path load iteratively. In order to make the algorithm converge, we define a terminate parameter $\sigma$ as the maximum allowable difference in the path loads among all leaf nodes. If the current spanning tree $T'$ satisfies the condition

$$\max \left\{ l_{\text{path}} \left( T', v_i \right) \mid v_i \in Q \left( T' \right) \right\} - \min \left\{ l_{\text{path}} \left( T', v_i \right) \mid v_i \in Q \left( T' \right) \right\} \leq \sigma, \quad (16)$$

the iterative operation will be terminated, and $T'$ is the final result. It is obvious that the terminate parameter $\sigma$ is more close to $\sigma_{\text{opt}}$, and the algorithm result is more close to the optimal solution. Therefore, our algorithm may be superior well under proper choice of terminate parameter $\sigma$.

However, if the current spanning tree $T'$ does not satisfy condition (16), we need to continue to adjust the load of bottleneck node. We use Figure 3 to illustrate the details of our algorithm. Figure 3 shows a part of the spanning tree $T'$. In Figure 3, the solid lines with arrow indicate the directions of data transmission in the tree $T'$ while the dotted lines without arrow represent the edges in graph $G$. The omitted parts of the tree $T'$ are represented by the dotted lines with arrow. $T_v$ denotes the subtree rooted at node $v_i$.

In Figure 3, suppose that node $v_i$ has the highest load in current tree $T'$. We need to switch some descendant nodes of $v_i$ to new parent node. Node $v_i$ has three children nodes, $v_{i+1}$, $v_{i+5}$, and $v_{i+6}$. Those nodes can be selected to switch to an alternate path. If node $v_{i+4}$ is selected, it may have two potential parent nodes, $v_{i+1}$ and $v_{i+2}$. Assume that node $v_{i+4}$ is switched to node $v_{i+1}$. After switching, the load of node $v_{i+1}$ increases and may become greater than the load of node $v_i$ by more than the terminate parameter $\sigma$. Consequently, the path load of leaf nodes in $T'_{v_{i+1}}$ may exceed the path load of leaf nodes in $T'_{v_i}$ by more than $\sigma$. In the next iteration, node $v_{i+4}$ may be switched to node $v_i$ again. In such case, we say that an oscillation has occurred. This phenomenon influences convergence and convergence speed of the algorithm. In order to reduce the influence of the oscillation, we adopt some mechanisms such as random switching decision and optimal parent node selecting strategy.

In the random switching decision, whether a node is selected for switching depends on its switching probability. For a considered node, we generated a random number, 0 or 1, according to its switching probability. If the random number is 1, the corresponding node is selected to switch; otherwise, the node is not selected. It is more likely to generate random number 1 when the switching probability is higher. The initial switching probability for each node is 1/2. Then, the switching probability $p(v_i)$ of node $v_i$ is decided by the number of switching instances where node $v_i$ is selected. The switching probability can be expressed as

$$p \left( v_i \right) = 2^{-[n(v_i)+1]}, \quad (17)$$

where $n(v_i)$ denotes the switching numbers of node $v_i$. When a node has been selected for switching many times, there may be an oscillation that happened on such node. We use formula (17) to give such node with a smaller switching probability, reduce the chance of the node being selected, and give other nodes more opportunities. This method is beneficial to reduce the oscillation.

In Figure 3, if node $v_{i+4}$ is switched many times and the algorithm cannot converge, the switching probability of node $v_{i+4}$ will decrease, and the opportunities of other $v_i$’s children node, such as $v_{i+5}$ or $v_{i+6}$, will increase. This feature is advantageous to improve the convergence of the algorithm.

We have used a random switching decision to select a switching node. We will design an optimal parent node selecting strategy for assigning an optimal parent node to the selected switching node. Intuitively, in graph $G$, each neighbor of a selected switching node is likely to be the parent node of the switching node. However, in fact, only a part of the neighbor nodes are qualified to be the parent eventually. Because the path loads of some neighbor nodes are high originally, the node switching may increase the path load of such nodes, and the oscillation may happen. Therefore, we should choose some neighbor nodes as the potential parent nodes. Suppose that $v_j$ is a switching node and its original path load is $l_{\text{path}}(T, v_j)$ in tree $T$. If $v_j$ is switched to a neighbor node $v_j$, the new tree $T'$ is constructed, and the new path load of $v_j$ is $l_{\text{path}}'(T', v_j)$. If the new path load of $v_j$ is bigger than the original path load, the neighbor node $v_j$ cannot become a potential parent node. The set of $v_j$’s potential parent nodes can be determined as follows:

$$P_{v_j} = \left\{ v_j \mid l_{\text{path}}'(T', v_j) < l_{\text{path}}(T, v_j), \ v_j \in \text{Nei}_{v_j} \right\}, \quad (18)$$

where $\text{Nei}_{v_j}$ is the set of $v_j$’s neighbor nodes.

In neighbor nodes set $\text{Nei}_{v_j}$, there may be two kinds of node: the first is the forwarding node and the other is the aggregating node. As described in formula (4), when neighbor node $v_j$ is an aggregating node, the outgoing traffic of aggregating node $v_j$ is a constant value, $M$. If $v_j$ switches to $v_j$, the outgoing traffic of $v_j$ is also a constant value $M$; that is,
Input: graph $G$, arbitrary spanning tree $T$, $l_{\text{path}}(T, v_i)$ and $l_{\text{node}}(T, v_i)$ for each $v_i \in V$, $\beta_{\text{max}}$

Output: Data-gathering tree

1. Set $\beta_v \leftarrow 0$ and $p(v_i) = 1/2$ for each $v_i \in V$;
2. Let $v_u$ be the node with the highest load in tree $T$;
3. While $\beta_v \leq \beta_{\text{max}}$ do
4. if $\max(l_{\text{path}}(T, v_i) \mid v_i \in Q(T)) - \min(l_{\text{path}}(T, v_i) \mid v_i \in Q(T) \leq \sigma)$ then
5. Return $T$;
6. else
7. Set $\alpha \leftarrow \text{Ch}(v_u)$$\beta_v = \beta_v + 1$;
8. While $\alpha \neq \emptyset$ do
9. Remove node $v_v$ from $\alpha$ in FIFO order;
10. $\alpha = \alpha \cup \text{Ch}(v_u)$;
11. if $P_v = \emptyset$ then
12. $\alpha = \alpha \cup \text{Ch}(v_u)$;
13. else
14. if SwitchingDecision($p(v_u) = 1$) then
15. $\text{Par}_{v_u} = \arg \min_{v_b \in P_v} l_{\text{path}}(T', v_b) / l_{\text{path}}(T, v_b)$; $T'$ is new parent of $v_b$;
16. $\text{Par}_{v_u}$ is new parent of $v_v$;
17. $p(v_v) = p(v_b) / 2$;
18. break;
19. else
20. $\alpha = \alpha \cup \text{Ch}(v_b)$;
21. end if
22. end if
23. end while
24. end if
25. Select $v_v$ as the node with the highest load;
26. end while
27. Return $T$

Algorithm 1: Construction of data-gathering tree.

4.3 Details of the Algorithm. The core of our algorithm is shown in Algorithm 1. In our algorithm, $\beta_v$ counts how many times node $v_v$ is selected as the node with the highest load in tree $T$, and $p(v_v)$ denotes switching probability of node $v_v$. The initial values of $\beta_v$ and $p(v_v)$ for each $v_v \in V$ are set to 0 and 1/2, respectively. In line (2), $v_v$ denotes the node with the highest load in tree $T$, whose children nodes will be selected for switching. $\beta_{\text{max}}$ denotes the maximum allowed number of times that a node can be selected for switching. It is an important parameter that affects the convergence of the algorithm. If the node $v_v$ with the highest load has been selected for $\beta_{\text{max}}$ times, the switching procedure will be terminated (line 3) and the data-gathering tree $T$ is returned (line 27). Within the while loop, if the current spanning tree $T$ satisfies inequality (16), the iterative operations of the algorithm will be terminated, and the data-gathering tree $T$ is returned (lines 4)-(5)). Otherwise, the children of $v_v$ are inserted in queue $\alpha$, and each node in queue $\alpha$ is qualified to be selected to switch. If queue $\alpha$ is not empty, we remove a node $v_b$ from queue $\alpha$ in FIFO order and then determine $v_b$’s potential parents set $P_v$, using formula (18) (lines 9)-(10). If node $v_b$ has no potential parent, the children nodes $\text{Ch}(v_b)$ of $v_b$ are added to the queue and considered for switching in subsequent rounds (line 12). If $v_b$ has some potential
parents, a random decision is made based on its current switching probability \( p(v_b) \) through the switching decision function. If the outcome of the decision is to switch, a node \( \text{Par}_u \) is chosen as the parent of node \( v_b \) through formula (20), and then the tree \( T \) is updated by switching the subtree \( T_{v_b} \) rooted at node \( v_b \) to \( \text{Par}_u \). The new loads \( l_{\text{path}}(T, v_i) \) and \( l_{\text{node}}(T, v_j) \) of each \( v_i \in V \) are updated (lines (14)-(16)). Because node \( v_b \) is switched, the subsequent switching probability becomes a half of its original; that is, \( p(v_b) = p(v_b)/2 \) (line (17)). Since node \( v_b \) is switched and the tree \( T \) is updated, we should terminate the while loop (line (18)) and then reselect a new node \( v_b \) with the highest load (line (25)) for subsequent iterations. If the outcome of the decision is not to switch, node \( v_b \) remains with its current parent and the descendants of \( v_b \) are added to the queue for subsequent consideration (lines (19)-(20)).

4.4. Analysis of Algorithm. In this section, we present the time complexity of our algorithm. The calculation of \( l_{\text{path}}(T, v_i) \) and \( l_{\text{node}}(T, v_j) \) for each \( v_i \in V \) that is done during the initialization can be performed in \( O(N) \) time. Lines (1)-(2) of our algorithm can be performed in \( O(N) \) time. On line (4), since there can be at most \( O(N) \) leaf nodes in current spanning tree, the calculation of the maximum and minimum path load of leaf node takes at most \( O(N) \) time. The initialization of \( \alpha \) on line (7) takes \( O(C_{\text{max}}) \) time, where \( C_{\text{max}} \) is the maximum number of children of any node. On line (10), we determine the potential parents of the selected switching node \( v_b \). Because each neighbor of the switching node \( v_b \) has the potential to become the parent, we need to calculate \( v_b \)'s new path load \( l_{\text{path}}(T', v_b) \) when \( v_b \) switch to each neighbor. The calculations take \( O(Q_{\text{max}}) \) times, where \( Q_{\text{max}} \) is the maximum number of neighbors of a node in graph \( G \). In order to determine \( l_{\text{path}}(T', v_b) \), we need to calculate the load of each node on the path from the selected switching node to the sink. It takes at most \( O(D) \) time where \( D \) is the diameter of the network. Therefore, determining the potential parents on line (10) takes \( O(DQ_{\text{max}}) \) time. Updating \( \alpha \) on lines (12) and (20) takes \( O(C_{\text{max}}) \) time. Selecting the ultimate parent on line (15) takes at most \( O(Q_{\text{max}}) \) since the new path load has been calculated on line (10). On line (16), updating \( l_{\text{path}}(T, v) \) and \( l_{\text{node}}(T, v) \) takes \( O(N) \) time. Since \( C_{\text{max}} \leq N \) and \( Q_{\text{max}} \leq N \), the running time for operations from line (9) to line (22) is dominated by \( O(DQ_{\text{max}} + N) \). Because the subtree \( T_{v_b} \) rooted at node \( v_b \) can have at most \( N \) descendants, the second loop on line (8) can run for \( O(N) \) times. Therefore, the running time for the second loop takes \( O(N(DQ_{\text{max}} + N)) \) time. If all nodes periodically participate in switching, the loop on line (3) runs at most \( Np_{\text{max}} \) times. Hence, the total time complexity of our algorithm is \( O(N^2(DQ_{\text{max}} + N)) \).

5. Performance Evaluation

In this section, we evaluate the performance of our algorithm through simulation experiments.

5.1. Simulation Setup. We evaluate the performance of our algorithm using Matlab simulations. In our simulation, we randomly deploy sensor nodes in a square region of size 100 m × 100 m. The number of sensors is varied from 100 to 300. We consider three scenarios associated with the different location of the sink in the network. In Scenario 1, the sink was placed at the corner of the deployment area, that is, at the (0 m, 0 m) coordinate. In Scenario 2, the sink was placed at one side of the deployment area, specifically at the (50 m, 0 m) coordinate. In Scenario 3, the sink was placed in the middle of the deployment area, namely, at the (50 m, 50 m) coordinate. Each sensor is randomly assigned an initial energy between 0.5 and 1 joule (J). We assumed that all nodes have the same energy consumption model. The radio transmission range of each node is set to 25 m. Similar to [18], the energy required to receive data is 50 nJ/bit and the energy required to transmit data is 100 nJ/bit. The size of one-unit data is 16 bytes. The energy consumption for one-unit data transmission and reception is \( E_{\text{Rx}} = 6400 \) nJ and \( E_{\text{Tx}} = 12800 \) nJ. We generate 100 random networks and present the average results for comparisons. We evaluate the lifetime performance measured as the number of data-gathering rounds until the first node runs out of energy.

The choice of the terminate parameter \( \sigma \) governs convergence speed of the algorithm and lifetime of the tree produced at converged point. However, it is not possible to know the optimal value of \( \sigma \) for a random connectivity graph. Intuitively, it may appear that choosing \( \sigma \) as low as possible results in a higher lifetime tree. When \( \sigma \) is too low, the algorithm will perform more switching to find a spanning tree to meet the terminate parameter. However, such spanning tree may not topologically exist in a given connectivity graph representing the sensor network. The oscillations may occur and diminish the lifetime of the final tree. Therefore, we need to learn the appropriate ranges of \( \sigma \) for different network configurations. To solve this problem, we generate 100 random deployed networks for each scale sensor network as training inputs to our algorithm. Then, we measure their lifetimes with varying \( \sigma \) values and select the optimal \( \sigma \) value for simulation experiments. The maximum allowed switching time \( \beta_{\text{max}} \) is set to 1000.

5.2. Simulation Results. In this section, we first compare our algorithm with the MITT [18], RaSMaLai [19], and MECDA [14] data-gathering schemes for Scenario 2. The lifetime performance is compared as a function of the number of nodes. The sampling rate is set to 10% and 20%.

It can be seen from Figure 4 that our algorithm outperforms the compared algorithms. Furthermore, the lower the sampling rate, the better the lifetime performance of our algorithm. When the sampling rate is low, more nodes in the network can become the aggregating nodes, so as to save energy. The MITT algorithm is based on a min-max-weight spanning tree. The algorithm improves the lifetime of a given tree by switching the parent of the node under consideration. The MECDA algorithm uses hybrid-CS technology, and the objective of the algorithm is to minimize the total energy consumption in a data-gathering tree. It does not consider the balance of energy consumption and energy consumption of...
each node. The nodes near the sink may have higher energy consumption, which affects the lifetime of the network. The RaSMaLai algorithm uses randomized switching scheme to maximize the lifetime of data collection trees based on the concept of the bounded balanced trees. However, because the algorithm collects the raw data directly, the nodes close to the sink need to relay data of all the downstream nodes on the path, so that the energy consumption is higher, and the network lifetime is shorter. Our algorithm uses both the hybrid-CS strategy and the random switching method to adjust the load of bottleneck node, thus obtaining the best lifetime performance.

Figure 5 shows the lifetime of different algorithms when the sampling rate is varied from 10% to 30%, and the number of nodes is 200 and 300, respectively. As shown in Figure 5, with the increase of the number of nodes, the nodes near the sink need to forward more data, the energy consumption increases, and the lifetime is reduced. With the increase of sampling rate, the lifetime for our algorithm and MECDA decreases. This is because when the sampling rate increases, the maximum data length \( M \) is increased, and the intermediate nodes need to gather more data to become aggregating nodes, the energy consumption of such nodes increases, and the network lifetime is reduced. However, the lifetime of RaSMaLai does not change with the varying of sampling rate, because it does not use the CS strategy. When the sampling rate is over 30%, the performance of our algorithm is more and more close to that of the RaSMaLai algorithm. The reason is that when the sampling rate is large, few nodes can become the aggregating node, and the characteristic of energy consumption is consistent with the RaSMaLai algorithm.

As shown in Figure 6, we compare our algorithm with MECDA in different scenarios. No matter which scenario is used, our algorithm outperforms the MECDA algorithm. In Scenario 1, the performance of our algorithm is more superior, because when the sink is placed at \((0, 0)\) coordinate, the nodes around the sink need to forward more data of downstream nodes and have heavy loads, and thus our algorithm can make the load of such nodes become more balanced by switching. However, in Scenario 3, the performance improvement of our algorithm is relatively small; this happens since the sink located at the middle naturally gives better balancing for each algorithm.
6. Conclusion

In this paper, we have studied the problem of constructing a maximum-lifetime data-gathering tree. We used hybrid-CS theory to data gathering in wireless sensor networks. We first constructed an arbitrary data-gathering tree, used a random switching decision to select a bottleneck node's child as switching node, and then designed an optimal parent node selecting strategy for assigning an optimal parent to the selected switching node. We kept transferring descendants of the bottleneck node to realize load balancing until the terminate parameter or maximum allowed switching time is satisfied. Simulation results show that the proposed algorithms can significantly increase the lifetime of WSNs and outperform several existing approaches in terms of network lifetime.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research is supported by the Natural Science Foundation of Jiangsu Province under Grants nos. BK20130096 and BK20171754, the National Natural Science Foundation of China under Grants nos. 61273051, 61272084, 61202004, 61202353, 61302084, and 61302158, the Postdoctoral Science Foundation of China under Grant no. 2015M581794, the Project of Natural Science Research of Jiangsu University under Grants nos. 15KJB520027 and 14KJB520031, the Postdoctoral Science Foundation of Jiangsu Province under Grant no. 1501023C, the Key Project of Natural Science Research of Jiangsu University under Grant no. 11KJA520002, and the Natural Science Foundation of Anhui Province (no. I608085MF127).

References

[1] S. L. Ji and Z. P. Cai, “Distributed data collection in large-scale asynchronous wireless sensor networks under the generalized physical interference model,” IEEE/ACM Transactions on Networking, vol. 21, no. 4, pp. 1270–1283, 2013.
[2] T. Duong and T. Nguyen, “Data collection in sensor networks via the novel fast markov decision process framework,” IEEE Transactions on Wireless Communications, vol. 14, no. 8, pp. 4522–4533, 2015.
[3] Y. Wu, Z. Mao, S. Fahmy, and N. B. Shroff, “Constructing maximum-lifetime data-gathering forests in sensor networks,” IEEE/ACM Transactions on Networking, vol. 18, no. 5, pp. 1571–1584, 2010.
[4] H. Lin and H. Uster, “Exact and heuristic algorithms for data-gathering cluster-based wireless sensor network design problem,” IEEE/ACM Transactions on Networking, vol. 22, no. 3, pp. 903–916, 2014.
[5] X. Xu, X.-Y. Li, P.-J. Wan, and S. Tang, “Efficient scheduling for periodic aggregation queries in multihop sensor networks,” IEEE/ACM Transactions on Networking, vol. 20, no. 3, pp. 690–698, 2012.
[6] S. He, J. M. Chen, D. K. Y. Yau, and Y. Sun, “Cross-layer optimization of correlated data gathering in wireless sensor networks,” IEEE Transactions on Mobile Computing, vol. 11, no. 11, pp. 1678–1691, 2012.
[7] E. J. Candès and M. B. Wakin, “An introduction to compressive sampling,” IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 21–30, 2008.
[8] C. Luo, F. Wu, J. Sun, and C. W. Chen, “Compressive data gathering for large-scale wireless sensor networks,” in Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom ’09), pp. 145–156, Beijing, China, September 2009.
[9] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, “On the interplay between routing and signal representation for Compressive Sensing in wireless sensor networks,” in Proceedings of the Information Theory and Applications Workshop (ITA ’09), pp. 206–215, San Diego, Calif, USA, February 2009.
[10] J. Wang, S. Tang, B. Yin, and X.-Y. Li, “Data gathering in wireless sensor networks through intelligent compressive sensing,” in Proceedings of the IEEE INFOCOM (INFOCOM ’12), pp. 603–611, IEEE, Orlando, FL, USA, March 2012.
[11] Y. Tang, B. Zhang, T. Jing, D. Wu, and X. Cheng, “Robust compressive data gathering in wireless sensor networks,” IEEE Transactions on Wireless Communications, vol. 12, no. 6, pp. 2754–2761, 2013.
[12] J. Haupt, W. U. Bajwa, M. Babbat, and R. Nowak, “Compressed sensing for networked data: a different approach to decentralized compression,” IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 92–101, 2008.
[13] J. Luo, L. Xiang, and C. Rosenberg, “Does compressed sensing improve the throughput of wireless sensor networks?” in Proceedings of the IEEE International Conference on Communications (ICC ’10), pp. 1–6, Cape Town, South Africa, May 2010.
[14] L. Xiang, J. Luo, and A. Vasilakos, “Compressed data aggregation for energy efficient wireless sensor networks,” in Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON ’11), pp. 46–54, Salt Lake City, UT, USA, June 2011.
[15] Y. Li, J. Zhou, and H. Xiong, “Global correlated data gathering in wireless sensor networks with compressive sensing and randomized gossiping,” in Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM ’11), pp. 1–5, IEEE, Houston, Tex, USA, December 2011.
[16] B. W. Zhang, X. Z. Cheng, N. Zhang, Y. Cui, Y. Li, and Q. Liang, “Sparse target counting and localization in sensor networks based on compressive sensing,” in Proceedings of the 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM ’11), pp. 2255–2263, IEEE, Shanghai, China, April 2011.
[17] L. Xiang, J. Luo, and C. Rosenberg, “Compressed data aggregation: energy-efficient and high-fidelity data collection,” IEEE/ACM Transactions on Networking, vol. 21, no. 6, pp. 1722–1735, 2013.
[18] J. B. Liang, J. X. Wang, J. N. Cao, J. N. Chen, and M. M. Lu, “An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks,” in Proceedings of the 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2010), pp. 1–5, San Diego, Calif, USA, March 2010.
[19] S. K. A. Imon, A. Khan, M. Di Francesco, and S. K. Das, “RaSMaLai: a randomized switching algorithm for maximizing lifetime in tree-based wireless sensor networks,” in *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM ’13)*, pp. 2913–2921, Turin, Italy, April 2013.

[20] S. K. A. Imon, A. Khan, M. Di Francesco, and S. K. Das, “Energy-efficient randomized switching for maximizing lifetime in tree-based wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, pp. 1401–1415, 2014.
