Delay constrained relay node placement in two-tiered wireless sensor networks: A set-covering-based algorithm

Chaofan Ma\textsuperscript{a,b,c,d}, Wei Liang\textsuperscript{a,b,c,*}, Meng Zheng\textsuperscript{a,b,c,*}

\textsuperscript{a} Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China  
\textsuperscript{b} State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China  
\textsuperscript{c} Key Laboratory of Networked Control Systems, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China  
\textsuperscript{d} The University of Chinese Academy of Sciences, Beijing 100049, China

\section{ARTICLE INFO}

\textbf{Keywords:}  
Wireless Sensor Networks  
Set cover  
Delay  
Relay node placement  
Approximation algorithm

\section{ABSTRACT}

As Wireless Sensor Networks (WSNs) are widely used in time-critical applications, e.g., factory automation and smart grid, the importance of Delay Constrained Relay Node Placement (DCRNP) problem is becoming increasingly noticeable. Considering the benefits in terms of energy efficiency and scalability brought by the two-tiered topology, this paper studies the DCRNP problem in two-tiered WSNs. To address the NP-hardness, a Two-phase Set-Covering-based Algorithm (TSCA) is proposed to approximately solve this problem. To be specific, in the first phase, a Connectivity-aware Covering Algorithm (CCA) places Relay Nodes (RNs) to fully cover distributed sensor nodes with respect to delay constraints, and meanwhile CCA tries to reduce the number of connected components in the topology constructed in this phase so as to save the RNs deployed to build network connectivity. In the second phase, the network connectivity is built in obedience to delay constraints by a Set-Covering-based Algorithm (SCA) through an iterative manner, which formulates the deployment of RNs at each iteration as the set covering problem and solves this problem using a classic set covering algorithm. In addition, the elaborated analysis of time complexity and approximation ratio of the proposed algorithms is given out. Finally, extensive simulations demonstrate that TSCA can significantly save deployed RNs in comparison to existing algorithms.

\section{1. Introduction}

Wireless Sensor Networks (WSNs) attract considerable attentions in recent years for their immense potentials in the applications of environment monitoring, battlefield reconnaissance and biomedical observation (Yick et al., 2008). WSNs consist of battery-powered Sensor Nodes (SNs) sensing certain circumstance information, and one or several sinks collecting the information sensed by SNs. Typically, the communication range of an SN is very limited due to its constrained power supply (Bredin et al., 2010; Hou et al., 2005). Thus, Relay Nodes (RNs) equipped with larger communication radii and more energy are often placed in WSNs to build network connectivity under various constraints, such as fault tolerance and energy-efficiency (Kashyap et al., 2006; Wu et al., 2008; Olariu and Stojmenovic, 2006). Due to the high costs of RNs, a minimum number of RNs are expected to be placed for building the connectivity, which is termed as the Relay Node Placement (RNP) problem. In recent years, the RNP problem has been extensively studied (Lin and Xue, 1999; Ma et al., 2016a; Cheng et al., 2007; Lloyd and Xue, 2007; Misra et al., 2010; Srinivas et al., 2009; Yang et al., 2012; Han et al., 2010).

As WSNs are progressively employed in the time-critical applications, e.g., industrial automation and smart grid (Kumer et al., 2014; Gungor and Hancke, 2009; Lin et al., 2014), the importance of the Delay Constrained RNP (DCRNP) problem is highlighted recently. Taking the factory automation as an example, the data sensed by SNs is typically time-sensitive, such as alarm notification and information for feedback control, and thus the importance of receiving the data at the sink in a timely manner is noticeable (Liang et al., 2011; Zheng et al., 2015). However, the literature about the DCRNP problem are very limited (Bhattacharya and Kumar, 2010, 2014; Nigam and Agarwal, 2014; Ma et al., 2016b; Sitanyah et al., 2014; Ma et al., 2016c). Bhattacharya and Kumar, (2010, 2014) first prove that the DCRNP problem is NP-hard, and a Shortest Path Tree based Iterative Relay Pruning (SPTiRP) algorithm is proposed. SPTiRP preliminarily builds a shortest path tree to connect each SN and the sink, then it saves the deployed RNs by gradually removing the RNs on the shortest path tree. This leads to a limitation that the finally deployed RNs can only be those included in the originally built shortest path tree, and the worst
case of this algorithm happens when all the RNs of the optimal solution have been missed by the shortest path tree. 

Nigam and Agarwal (2014) formulate the DCRNP problem as a linear programming problem, and propose a branch-and-cut algorithm to optimally solve the DCRNP problem. However, the proposed algorithm can only solve a special case of the DCRNP problem (each of the source nodes cannot have a singleton node cut), and the time complexity of this algorithm grows exponentially, which prohibits its use in large scale problems. 

Sitanayah et al. (2014) study the fault-tolerant RNP problem with a subset of the DCRNP problem (each of the source nodes cannot have a singleton node cut), and the time complexity of this algorithm grows exponentially, which prohibits its use in large scale problems. 

In view of the benefits in terms of energy efficiency and scalability brought by two-tiered WSNs (Estrin et al., 1999; Pan et al., 2003), this paper studies the DCRNP problem in two-tiered WSNs and proposes a Two-phase Set-Covering-based Algorithm (TSCA). TSCA solves the DCRNP problem in two phases, i.e., the covering phase, in which the RNs for covering distributed SNs are deployed, and the connecting phase, in which the RNs for building network connectivity are deployed. Two algorithms, i.e., the Connectivity-aware Covering Algorithm (CCA) and the Set-Covering-based Algorithm (SCA), are proposed for the two phases, respectively. In summary, the contributions of this paper are listed as follows:

- First, an algorithm-CCA is presented for the covering phase. The CCA first deploys a set of RNs to fully cover distributed SNs with respect to delay constraints imposed on these SNs, and in the meanwhile CCA tries to reduce the number of connected components in the topology constructed in this phase so as to save the RNs deployed in the connecting phase. We prove that CCA is a polynomial-time algorithm with an approximation ratio of $\ln n + 1$, where $n$ is the number of SNs.
- Second, an algorithm-SCA is proposed for the connecting phase. The SCA builds network connectivity in obedience to delay constraints in an iterative manner, which formulates the deployment of RNs at each iteration as the set covering problem and employs the typical greedy-set-covering algorithm to solve this set covering problem. We prove that SCA is also a polynomial-time algorithm but with an approximation ratio of $O(\ln n)$.
- Third, an algorithm-TSCA is designed based on CCA and SCA to the DCRNP problem in two-tiered WSNs. Through rigorous analysis, we prove that TSCA is a polynomial-time algorithm with an approximation ratio of $O(\ln n)$.
- Fourth, extensive simulations are performed on the NS-3 simulator to demonstrate the effectiveness of the proposed algorithms. Simulation results show that TSCA saves more deployed RNs in comparison to existing works while meeting the delay constraints.

The rest of this paper is organized as follows: Problem formulation is given out in Section 2. CCA and SCA are proposed in Section 3 and Section 4, respectively. Then, TSCA is presented in Section 5 to the DCRNP problem in two-tiered WSNs. Simulation results are shown in Section 6. Finally, Section 7 concludes the whole paper.

2. Problem formulation

First of all, some reasonable assumptions are given out to facilitate the analysis of the DCRNP problem as published works (Bhattacharya and Kumar, 2010, 2014; Nigam and Agarwal, 2014):

- The end-to-end delay is measured in terms of hop count.
- This paper only considers the widely-used many-to-one communication pattern (Li and Mohapatra, 2007), in which SNs may transmit their sensed information to the sink via multi-hop paths.
- Due to the existence of obstacles and forbidden regions in real deployment environment, RNs cannot be deployed at will. Therefore, this paper assumes that RNs can only be deployed at some predetermined Candidate Deployment Locations (CDLs).

In this paper, the DCRNP problem is studied in the two-tiered WSNs, in which the data generated by SNs can only be sent to their 1-hop neighbor RNs, and SNs cannot relay any data. We consider a two-tiered WSN consisting of a set of SNs $S = \{s_1, s_2, \ldots, s_m\}$, a set of CDLs $C = \{c_1, c_2, \ldots, c_n\}$ and a sink $K$. The communication radii of the SNs and the RNs are $r$ and $R$, respectively. Typically, $R \geq r$. Without loss of generality, we assume that the communication radius of sink is larger than $R$. The WSN is formulated as an undirected graph $G = (V, E)$, where $V = S \bigcup \bigcup \bigcup \{K\}$ is the node set, and $E$ is the edge set. $\forall u, v \in V (u \neq v)$, if $u$ and $v$ are the two ends of an edge in $E$, $u$ and $v$ should fulfill the following conditions:

- If $u \in S$ or $v \in S$, then $u$ and $v$ should meet that $\|u - v\| \leq r$;
- If $u \notin S$ and $v \notin S$, then $u$ and $v$ should meet that $\|u - v\| \leq R$,

where $\|u - v\|$ denotes the Euclidean distance between $u$ and $v$. Besides, in this paper a path between $u$ and $v$ in graph $G$ is denoted by $p(u, v)$.

Definition 1 (DCRNP Problem in Two-Tiered WSNs). For an undirected graph $G = (V, E)$, the DCRNP problem searches for an induced subgraph $G' = (V', E')$ of $G$, where $V' = S \bigcup \{K\}$, and $C'$ is a subset of $C$ with the minimum cardinality such that the following condition is satisfied: there exists at least one path, which complies with the delay constraint, between each SN and the sink, and the internal nodes on this path are RNs.

CDLs on the induced subgraph $G'$ are selected to deploy RNs. Thus the terms RN and CDL are used interchangeably in this paper. Besides, a node on $G'$ can be an SN, a RN or the sink. Then, let the notation $C(p)$ represent the hop count of path $p$.

The DCRNP problem in two-tiered WSNs can be formulated as an optimization problem:

\[
\text{Minimize} \quad |C'| \\
\text{s. t.} \quad \forall s \in S, \exists p_{s, (s, K)} \in C(p_{s, (s, K)}) \leq \Delta(s) \quad \text{and} \\
N(p_{s, (s, K)}) \cap S = \{s\},
\]

where $\Delta(s)$ is the delay constraint imposed on SN $s$, $p_{s, (s, K)}$ denotes a path between $s$ and $K$ in graph $G$ and $N(\bullet)$ represents the set of nodes on path $\bullet$. In this paper, a path meeting the delay constraint is termed as a feasible path.

The DCRNP problem has been proved NP-hard (Bhattacharya and Kumar, 2010). Therefore, the objective of this paper is to devise an efficient polynomial time algorithm to approximately solve the DCRNP problem. In two-tiered WSNs, the network connectivity relies on the mesh network composed of RNs, and SNs only need to transmit their packets to the RNs covering them. This means that the deployed RNs can be classified into two categories, i.e., the RNs for covering distributed SNs and those for building the connectivity of mesh network. Correspondingly, this paper solves the DCRNP problem in two phases, i.e., the covering phase, in which RNs for covering SNs are placed, and the connecting phase, in which RNs for building connecting are placed.
3. Algorithm for covering phase

3.1. Preliminaries

In the covering phase, a minimum set of RNs are expected to be deployed to fully cover SNs with respect to delay constraints. Let \( \mathcal{H}(u, v) \) denote a shortest (i.e., least hop count) path between nodes \( u \) and \( v \) in graph \( G \). SN \( s \) is said to be effectively covered by CDL \( c \) if the following conditions are satisfied:

\[
\| c - s \| \leq r, \tag{2a}
\]

\[
C(\mathcal{H}(c, K)) \leq \Delta(s) - 1. \tag{2b}
\]

Conditions (2a)–(2b) show that there exists at least one feasible path passing through \( c \) between the sink \( K \) and SN \( s \). Therefore, if \( c \) is used to place an RN to cover \( s \), there is no need to worry about the violation of delay constraint imposed on \( s \). Let \( \mathcal{S} \) denote the set of all SNs that can be effectively covered by CDL \( c \). We call \( c \) a Possible Position (PP) if \( |\mathcal{S}| \geq 1 \).

Correspondingly, the problem in the covering phase is to select a minimum set \( Y \) of PPs to place RNs such that \( \mathcal{S} \cup \bigcup_{c \in Y} \mathcal{S} = \mathcal{S} \), which is usually termed as the set cover problem (CormenT et al., 2001). Next, the concept of Possible Area (PA) is introduced.

**Definition 2 (Possible Area).** A Possible Area (PA), \( A \), is defined as the set of all the PPs covering the same SNs, i.e.,

\[
\forall u, v \in A (u \neq v), \ X(u) = X(v), \tag{3}
\]

where \( X(A) \) denotes the SNs effectively covered by the PPs in \( A \).

Fig. 1 shows the reason for introducing the concept of PA, where the ovals and the points in these ovals represent PAs and their associated PPs, respectively. The edge between two PPs in different PAs implies that the RNs deployed at these PPs can communicate with each other directly. In this figure, PPs \( c_1, c_4, c_5, c_8, \) and \( c_9 \) are originally selected to deploy RNs, and the resulting topology is shown in the left subfigure of Fig. 1(b). Alternatively, according to the definition of PA, we can use another PP in the same PA to substitute a selected PP. For example, we can substitute \( c_3 \) by \( c_4 \) and substitute \( c_5 \) by \( c_6 \), which leads to a new topology shown in the right subfigure of Fig. 1(b). We can see that fewer RNs will be deployed for the new topology to build the connectivity, which indicates that we can adjust each selected PP to save the deployed RNs.

3.2. Algorithm description

In the covering phase, CCA first searches a set \( \mathcal{A} \) of all the PAs. Then, a classic set cover algorithm is employed to select a subset \( \mathcal{A} \) of PAs to fully cover SNs, i.e., \( \bigcup_{c \in \mathcal{A}} \mathcal{X}(c) = \mathcal{S} \). In this paper, the Greedy-Set-Covering (GSC) algorithm proposed in (CormenT et al., 2001) is employed by CCA. Next, one PP is selected from each PA in \( \mathcal{A} \) to deploy RN such that SNs are fully covered. We term the above operation as the PP selection procedure. The objective of the PP selection procedure is to further save the RNs deployed in the connecting phase. CCA is explicitly detailed in Algorithm 1. Besides, the RNs deployed in the covering phase are termed as Covering Phase RNs (CPRs), and the delay constraint of each CPR \( y \) is calculated as

\[
\Delta(y) = \min_{c \in \mathcal{X}(c)} \Delta(s) - 1. \tag{4}
\]

**Algorithm 1.** Connectivity-aware Covering Algorithm (CCA).

**Input:** The set \( \mathcal{S} \) of SNs, a set \( C \) of CDLs, the sink \( K \), the communication radii of the RNs and the SNs and the delay constraints.

**Output:** A set \( Y \) of RNs to fully cover SNs.

begin

1. **foreach** \( c \in C \) **do**
   - **if** \( c \) is a PP **then**
     - \( \mathcal{P} = \mathcal{P} \cup \{c\} \);
   2. Classify PPs in \( \mathcal{P} \) into different PAs, and denote the set of PAs by \( \mathcal{A} \);
   3. \( \mathcal{A} = \) a subset of \( \mathcal{A} \) searched by GSC to fully cover SNs;
   4. GSA is employed to select a PP from each PA in \( \mathcal{A} \), and a set \( Y \) of RNs are deployed at these selected PPs;

return \( Y \)

the set of all the PPs covering the same SNs, i.e.,

\[
\forall u, v \in A (u \neq v), \ X(u) = X(v), \tag{3}
\]

where \( X(A) \) denotes the SNs effectively covered by the PPs in \( A \).

Fig. 1 shows the reason for introducing the concept of PA, where the ovals and the points in these ovals represent PAs and their associated PPs, respectively. The edge between two PPs in different PAs implies that the RNs deployed at these PPs can communicate with each other directly. In this figure, PPs \( c_1, c_4, c_5, c_8, \) and \( c_9 \) are originally selected to deploy RNs, and the resulting topology is shown in the left subfigure of Fig. 1(b). Alternatively, according to the definition of PA, we can use another PP in the same PA to substitute a selected PP. For example, we can substitute \( c_3 \) by \( c_4 \) and substitute \( c_5 \) by \( c_6 \), which leads to a new topology shown in the right subfigure of Fig. 1(b). We can see that fewer RNs will be deployed for the new topology to build the connectivity, which indicates that we can adjust each selected PP to save the deployed RNs.

**Definition 3.** If there is at least one edge between PP \( c \) and the PPs in PA \( A \), we say that PP \( c \) is adjacent to PA \( A \).

In the PP selection procedure, we expect the connected components (which are termed as components for short in following paper) in the topology built in the covering phase as few as possible such that fewer RNs will be deployed to build the connectivity in the connecting phase. To this end, a Greedy-based Selection Algorithm (GSA) is designed for the PP selection procedure. In GSA, each PP is assigned a weight which indicates the number of PAs (in \( \mathcal{A} \)) adjacent to the PP. As illustrated in Fig. 1(a), \( c_6 \) is adjacent to \( A_4 \) and \( c_9 \) is adjacent to \( A_3 \) and \( A_5 \). Consequently, in each iteration of GSA, a PP \( c \) with the largest weight
and the PPs adjacent to \( c \) are selected to deploy RNs. GSA is described in Algorithm 2.

Fig. 2 gives an example to illustrate GSA, where the solid line segments denote the edges between selected PPs and the dashed line segments denote the edges between remaining unselected PPs. In the first iteration, since \( c_1 \) is adjacent to most PAs, i.e., \( A_4 \), \( A_6 \) and \( A_7 \), \( c_1 \) is selected from \( A_8 \) to deploy RN, and PPs adjacent to \( c_1 \) are also selected. However, from Fig. 2(a) we can observe that \( c_1 \) may be adjacent to multiple PPs in a PA, e.g., \( c_1 \) is adjacent to \( c_2 \) and \( c_3 \) in \( A_6 \). In this case, GSA will select the one with largest weight. Furthermore, if these PPs have an identical weight, GSA will select the one nearest to the sink (measured in terms of hop count). Therefore, \( c_2 \) and \( c_6 \) are selected from \( A_4 \) and \( A_6 \) to deploy RNs, respectively. In the first iteration, \( c_1 \), \( c_2 \), \( c_4 \) and \( c_6 \) are selected from the PAs containing them, and build a component. At the end of this iteration, the remaining PPs in \( A_4 \), \( A_6 \), \( A_7 \) and \( A_8 \) are deleted.

Then, in the second iteration, GSA first searches for a set of remaining PPs that are adjacent to the selected PPs. If this set is non-empty, PPs in this set are selected. Otherwise, GSA will select PPs as the previous iteration. In Fig. 2(b), \( c_7 \) is adjacent to the component built in the previous iteration, and thus, it is selected. In the end of this iteration, the components built previously is renewed, i.e., the component composed of \( c_1 \), \( c_2 \), \( c_4 \) and \( c_6 \) includes one more PP \( c_7 \) now, and \( A_3 \) and its PPs are deleted. As the repetition of GSA, a topology is built gradually. In the previous iteration, \( c_{11} \) is selected due to the fact that it is closer to the sink than the other PPs in \( A_5 \).
Algorithm 2. Greedy-based Selection Algorithm (GSA).

Input: A set $\mathcal{A}$ of PAs, the sink $K$ and the communication radii of the RNs and the SNs.
Output: A set $Y$ of RNs which fully cover SNs.

begin
\begin{align*}
K = \emptyset; & \text{ a set to store components in resulting topology.} \\
\text{while } \mathcal{A} \neq \emptyset & \text{ do} \\
\text{if } \mathcal{C}_k \neq \emptyset & \text{ then} \\
\text{sort } \mathcal{C}_k \text{ in a descending order according to the weight;} \\
\text{while } \mathcal{C}_k \neq \emptyset & \text{ do} \\
\text{c = the first PP in } \mathcal{C}_k; \\
\text{foreach } u \in \mathcal{C}_k & \text{ do} \\
\text{if } u \text{ and c belong to the same PA then} \\
\mathcal{C}_k = \text{ overline}C_k - \{u\}; \\
\mathcal{K} = \mathcal{K} \cup \{c\}, \text{ and delete the PA including c from } \mathcal{A}; \\
\text{else} \\
\mathcal{C}_{tmp} = \text{the largest weighted PP and the PPs adjacent to it;} \\
delete PAs including PPs in } C_{tmp} \text{ from } \mathcal{A}; \\
sort } C_{tmp} \text{ in a descending order according to the weight;} \\
\text{foreach } u \in C_{tmp} & \text{ do} \\
\text{foreach } v \in (C_{tmp} - \{q\}) & \text{ do} \\
\text{if } v \text{ and u belong to the same PA then} \\
C_{tmp} = C_{tmp} - \{v\}; \\
C_{tmp} = C_{tmp} \cup \{c\}, \mathcal{K} = \mathcal{K} \cup C_{tmp}; \\
\end{align*}

return $Y$

3.3. Algorithm analysis

Let $N = m + n + 1$, where $m$ and $n$ are the number of CDLs and SNs, respectively. We first analyze the time complexity of GSA. The two inner loops of GSA have the same time complexity $O(N^2)$. The running time for the sorting operation (i.e., lines 2 and 8) is also $O(N^2)$. As the main loop iterates for at most $N$ times, the time complexity of GSA is $O(N^3)$.

CCA consists of three steps. In the first step, PAs are searched, and the time complexity of this step is $O(N^2)$. In the second step, the GSC algorithm whose time complexity is $O(N^3)$ is employed to find a set cover, and thus the time complexity of this step is $O(N^3)$. In the third step, GSA is employed in the PP selection procedure, which indicates the time complexity of this step is $O(N^3)$. Thus, the time complexity of CCA is $O(N^2) + O(N^3) + O(N^3) = O(N^3)$.

The approximation ratio of the GSC algorithm is $\ln n + 1$ (Cormen et al., 2001). As the GSC algorithm is employed to find a set cover for SNs and GSA does not change the size of this set cover, the approximation ratio of CCA is the same as the approximation ratio of the GSC algorithm, i.e., $\ln n + 1$.

4. Algorithm for connecting phase

4.1. Preliminaries

Clearly, a feasible solution to the DCRNP problem should be at least connected. This indicates that a feasible solution should contain a spanning tree, and thus a tree rooted at the sink and connecting all the CPRs deployed in the covering phase is built by SCA in the connecting phase. Given a tree $T$ and two different nodes of $T$, $u$ and $v$, we denote $p_{\star}(u, v)$ as a path of $T$ between $u$ and $v$. $\forall y \in Y$, if $C(p_{\star}(K, y)) \leq \Delta(y)$, we call $T$ a feasible tree, where $Y$ is the set of CPRs. In order to facilitate the explanation of SCA, we introduce the concept of level in a tree. Given a feasible tree $T$, the level of a node $q$ in $T$ is the hop count of the
path between the sink $K$ and node $q$, i.e., $C(p_q(K, q))$, and the set of nodes at the $k$th level of $T$ is denoted by $L^k_T$. As illustrated in Fig. 3, $L^1_T = \{c_1, c_2\}$, $L^2_T = \{c_2, c_3, c_4\}$, $L^3_T = \{y_1, y_2, c_5\}$, and $L^4_T = \{y_3, y_4\}$.

Considering that the nodes at each level should have 1-hop neighbors at their adjacent levels to maintain the network connectivity, we deploy RNs from the sink to the CPRs by level such that the CPRs are connected to the sink with the help of the gradually deployed RNs. Suppose that we have deployed RNs at the $(k-1)$th level in $T$, and next attempt to deploy RNs for the $k$th level. Let $\mathcal{V}(L_{k-1}^T)$ denote the set of 1-hop downstream neighbors of the nodes in $L_{k-1}^T$. Obviously, the nodes at the $k$th level should be 1-hop neighbors of the nodes at the $(k-1)$th level, i.e., $L_k^T \subseteq \mathcal{V}(L_{k-1}^T)$.

As all the 1-hop neighbors of the nodes at the $(k-1)$th level are found, i.e., $\mathcal{V}(L_{k-1}^T)$ is known, we need to select the nodes at the $k$th level (i.e., $L_k^T$) from $\mathcal{V}(L_{k-1}^T)$. In addition, the nodes in $L_k^T$ should ensure that each unconnected CPRs can be connected to the sink via a feasible path passing through one of these nodes. As shown in Fig. 4, the node at the 1st level is determined, i.e., $L^1_T = \{c_1\}$, and the neighbors of $c_1$ is known, i.e., $\mathcal{V}(L^1_T) = \{c_2, c_3, c_4\}$, where the dashed line segments denote the feasible paths of the unconnected CPRs. Fig. 4 shows that $y_1$ can be connected to the sink via a feasible path passing through $c_2$ and $y_2$ can be connected to the sink via feasible paths passing through $c_3$ or $c_4$, and thus, the nodes at the 2nd level can only be selected from $\{c_2, c_3, c_4\}$. Therefore, for each node in $\mathcal{V}(L_{k-1}^T)$, we should first design an approach to find the unconnected CPRs, whose feasible paths pass through this node, and this approach is shown in the following part of this section.

\footnote{As the RNs are deployed in a top-down manner, we will omit the word “downstream” for concise presentation in the rest of this paper.}

Let $P$ be the set of all feasible paths between CPR $y$ and the $K$. Then, $\forall C(p(y, K)) \leq \Delta(y), p(y, K) \in P$. The set of nodes lying on the feasible paths of $y$ is $\bigcup_{p \in P} N(x)$.

**Theorem 1.** Let $q$ be an arbitrary CDL or CPR. The sufficient and necessary condition for $q \in \bigcup_{p \in P} N(x)$ is that

$$C(H(q, y)) + C(H(q, K)) \leq \Delta(y).$$

(5)

**Proof.** Sufficiency: We build two shortest paths, i.e., $H(q, y)$ and $H(q, K)$, which connect $q$ and $y$ and connect $q$ and $K$, respectively. Then, we can build a path $p$ between $y$ and $K$ by combining $H(q, y)$ and $H(q, K)$. According to the inequality (5), the hop count of $p$ is given by

$$C(p) = C(H(q, y)) + C(H(q, K)) \leq \Delta(y),$$

(6)

which confirms that $p$ is a feasible path of $y$. Therefore, we can conclude that $q \in \bigcup_{p \in P} N(x)$. So far, we have completed the first part of the proof.

Necessity: From $q \in \bigcup_{p \in P} N(x)$ we know that there is at least one path $p (p \in P)$ passing through $q$. Therefore, we have $C(p) \leq \Delta(y)$.

Let $\mathcal{P}(\bar{p})$ be a segment of path $p$ between $q$ and $y (K)$. Then, we have

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Parameter Setting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>total packets</td>
<td>1000</td>
</tr>
<tr>
<td>packet length</td>
<td>2000 bits</td>
</tr>
<tr>
<td>path loss exponent</td>
<td>4</td>
</tr>
<tr>
<td>data rate</td>
<td>250 kbps</td>
</tr>
<tr>
<td>noise floor</td>
<td>$-115$ dBm</td>
</tr>
<tr>
<td>reference distance</td>
<td>1 m</td>
</tr>
<tr>
<td>average path loss at reference distance</td>
<td>20 dBm</td>
</tr>
</tbody>
</table>
Since $p$ is a feasible path and consists of $p_1$ and $\tilde{p}$, the following formulation holds

$$C(p) + C(\tilde{p}) \leq \Delta(y).$$

Plugging inequalities (7) into (8), we can achieve

$$C(H(q, y)) + C(H(q, K)) \leq \Delta(y).$$

This completes the proof of Theorem 1.

Theorem 1 tells that if $q$ satisfies inequality (5), at least one feasible path of $y$ should pass through node $q$. Let $q \in \mathcal{V}(L_1^{k-1})$, and $\mathcal{F}_q$ be the set of unconnected CPRs which are located at the levels deeper than $k$. According to Theorem 1, for any CPR, say $y \in \mathcal{F}_q$, satisfying

$$C(H(y, q)) + C(p_1(q, K)) = C(H(y, q)) + k \leq \Delta(y),$$

at least one feasible path of $y$ should pass through $q$. The set of CPRs fulfilling inequality (10) in $\mathcal{F}_q$ is denoted by $Q(q)$ and we say that $q$ can connect each CPR in $Q(q)$ to the sink in obedience to delay constraint.

Given a set $\mathcal{V}(L_k^{k+1})$ of 1-hop neighbors of the nodes in $L_k^{k-1}$, a subset $L_q^k$ will be selected from $\mathcal{V}(L_k^{k-1})$ as the nodes at the kth level. First of all, the nodes in $L_q^k$ should connect all the CPRs in $\mathcal{F}_q$ to the sink in obedience to delay constrains, i.e., $\bigcup_{q \in Q} Q(q) = \mathcal{F}_q$. Then, to save the cost of RN deployment at this level, the CDLs in $L_q^k$ should be as few as possible, i.e., a minimum subset $L_q^k$ is expected to be selected from $\mathcal{V}(L_k^{k-1})$ since the CPRs in $L_q^k$ are already determined when $L_1^k$ is fixed (i.e., the 1-hop neighbor CPRs of the nodes in $L_1^k$ are automatically taken as the nodes in $L_q^k$). Mathematically, this is a set covering problem, and typical solution algorithms to the set covering problem are rich in literature.

However, the inequality (10) cannot ensure that the RNs deployed at current level are closer to the unconnected CPRs than the nodes at previous levels, which may lead to a large amount of redundantly deployed RNs. As shown in Fig. 5 (i.e., $\forall \in \{1, 2, 3\}$, $\Delta(y) = 7$), according to inequality (10) we can obtain that $Q(c_1) = \{y_1, y_2, y_3\}$, and $Q(c_2) = \{y_2, y_3\}$. Since $\mathcal{V}(L_q^k) = \{c_1, c_2\}$, to deploy as few RNs as possible at each level, we will select $c_1$ as the deployment location at the 1st level. This leads to a local optimal solution whose edges are represented by the blue dashed line segments in Fig. 5, while the optimal solution whose edges are represented by the black line segments are missed. In this example, totally three RNs are wasted.

To avoid the inefficient local optimum, we introduce a rule which guarantees that the RNs deployed at each level gradually move closer to the unconnected CPRs. Let $\mathcal{F}$ be a node at the $(k-1)$th level, and $q$ be a 1-hop neighbor of $\mathcal{F}$, i.e., $q \in \mathcal{V}(L_q^{k-1})$. Obviously, CPRs in $Q(q)$ can also be connected to the sink through $\mathcal{F}$. At this moment, we restrict each CPR, say $y$, in $Q(q)$ to satisfy that

$$C(H(y, q)) < C(H(\mathcal{F}, y)).$$

Finally, the CPRs in $Q(q)$ should meet both the inequalities (10) and (11).
4.2. Algorithm description

The proposed SCA is summarized in Algorithm 3. Specifically, SCA is composed of three steps. In the first step, SCA checks whether the problem to be solved in the connecting phase is solvable or can be solved without the help of RNs. If one of these two cases happens, SCA terminates. Otherwise, SCA carries out the second step.


**Input**: A set $Y$ of CPRs, a set $\hat{C} = C - Y$ of CDLs, a sink $K$.

**Output**: A tree $T$: if $T = \emptyset$, this implies that no feasible solutions exist for the connecting phase; otherwise, $T$ is a feasible tree.

```
begin
1   $t1 = \text{a shortest path tree rooted at } K \text{ and including all CPRs in } Y \text{ and a subset of CDLs in } \hat{C}$;
2   if $\forall y \in Y, p_{t1}(y, K) \leq \Delta(y)$ then
3       $t2 = \text{a shortest path tree rooted at } K \text{ and including all CPRs in } Y$
4       without any CDLs in $\hat{C}$;
5       if $\exists y \in Y, p_{t2}(y, K) > \Delta(y)$ then
6           input $Y$, $\hat{C}$ and $K$ into the second step of SCA;
7           $C' = \text{a subset of } \hat{C} \text{ returned by the second step of SCA}$;
8           input $Y$, $C'$ and $K$ into the third step of SCA;
9       else
10          $T = t2$;
11       end
12 else
13      $T = \emptyset$;
14 return $T$;
end
```

The second step of SCA is iteratively executed so as to allocate the nodes at each level. To be specific, in the $k$th iteration, SCA selects the nodes at the $k$th level. SCA first searches for 1-hop neighbors of the nodes on the $(k - 1)$th level, i.e., $\mathcal{V}(L_{T}^{k-1})$. Next, for each node $q$ in $\mathcal{V}(L_{T}^{k-1})$, SCA searches for the unconnected CPRs that can be connected to the sink through $q$, i.e., $Q(q)$. Then, a subset $L_{T}^{k}$ of $\mathcal{V}(L_{T}^{k-1})$ is selected such that $\bigcup_{q \in Q(q)} Q(q) = \mathcal{T}$. As the execution of the second step of SCA, a feasible tree $T$ that connects each CPR and the sink is gradually built.

The second step of SCA is illustrated in Fig. 6, in which the circles and triangles in the same color are the 1-hop neighbors of the nodes at the previous level, and the circles and triangles with red edges are the nodes selected at each level to build the feasible tree $T$. At the beginning of the second step, SCA first searches the 1-hop neighbors of the sink $K$, denoted by the black circles and triangles, and the set of unconnected CPRs is $\mathcal{T} = \{y_{1}, y_{2}, \ldots, y_{6}\}$. Then, the nodes in $\mathcal{V}(L_{T}^{0})$ are represented by the unconnected CPRs, and the Greedy-Set-Cover (GSC) algorithm (CormenT et al., 2001) is employed to find a set cover for the unconnected CPRs. As shown in Fig. 6, $c_{1}$ is selected at the first level. At the second level, SCA first finds the 1-hop neighbors of $c_{1}$, denoted by the blue circles and triangles, and the set of unconnected CPRs changes into $\mathcal{T} = \{y_{1}, y_{3}, \ldots, y_{6}\}$ due to the fact that $y_{1}$ is a 1-hop neighbor of the node $c_{1}$ at the previous level. Then, $y_{1}$ and $c_{3}$ are selected as the nodes at the 2nd level based on the set cover algorithm.

Algorithm 4. The second step of SCA.

**Remark 1.** Notice that, the situation that different neighbors (e.g., $q$ and $q'$) connect the same unconnected CPRs to the sink may happen, i.e., $Q(q) = Q(q')$. To deal with this situation, we introduce a weight for each neighbor and select the one with the least weight. This paper defines the weight of $q$ as follows

$$w(q) = \left| \bigcup_{x \in Q(q)} \mathcal{C}(H(q, x)) \right|,$$

where $\mathcal{C}(p)$ denotes the set of CDLs on path $p$.
The redundant RNs may be deployed by the second step of SCA, and thus, the third step of SCA is designed to save the redundantly deployed RNs and detailed in Algorithm 5. In the third step, SCA aims to remove the RNs in $C'$ deployed by the second step of SCA as many as possible. To this end, each RN is assigned a weight which equates the number of CPRs meeting the inequality (5). If the deletion of a RN results in the violation of delay constraint, this RN cannot be removed and is marked as checked. SCA keeps removing the unchecked RN with least weight from $C'$ until $C'$ is empty or the RNs in $C'$ are all marked as checked.

**Algorithm 5.** The third step of SCA.

Input: A set $Y$ of CPRs, a set $C'$ of CDLs, a sink $K$.
Output: A set of CDLs, $C'$.

begin
1. $k = 0, L_T^0 = K$;
2. $\mathcal{V}(L_T^k)$ = the 1-hop neighbors of $K$ in $Y$ and $C'$ CPRs in $\mathcal{V}(L_T^k)$ are automatically connected to the sink, i.e., $\forall y \in \mathcal{V}(L_T^k) \cap Y, y \in L_T^{k+1}$;
3. $\bar{Y}_k = Y, \bar{Y}_k = \bar{Y}_k - \mathcal{V}(L_T^k)$;
   while $\bar{Y}_k \neq \emptyset$ do
     foreach $y \in L_T^{k+1}$ do
       calculate $Q(y)$ and $\omega(y)$;
       $\text{tmp}\bar{Y}_k = \bar{Y}_k - \bigcup_{y \in L_T^{k+1}} Q(y)$;
     if $\text{tmp}\bar{Y}_k = \emptyset$ then
       $k = k + 1$;
       $\mathcal{V}(L_T^k)$ = the 1-hop neighbors, which are selected from $\bar{Y}_k$ and $\hat{C}$, of the nodes in $L_T^k$;
     each CPR $y$ in $\mathcal{V}(L_T^k)$ are automatically connected to its 1-hop neighbor in $L_T^k$, i.e., $\forall y \in \mathcal{V}(L_T^k) \cap \bar{Y}_k, y \in L_T^{k+1}$, then
       $\bar{Y}_k = \bar{Y}_k - \mathcal{V}(L_T^k)$;
     else
       foreach $c \in \mathcal{V}(L_T^k)$ do
         calculate $Q(c)$ and $\omega(c)$;
         $Q(c) = Q(c) - \bigcup_{u \in L_T^{k+1}} Q(u)$;
       $\text{tmp}\text{Re} = \text{a subset of } \mathcal{V}(L_T^k) \text{ found by GSC (Cormen, 2001) to fully cover } \text{tmp}\bar{Y}_k, L_T^{k+1} = L_T^{k+1} \cup \text{tmp}\text{Re}$;
       $\hat{C}' = \hat{C} - \mathcal{V}(L_T^k), k = k + 1, C' = C' \cup \text{tmp}\text{Re}$;
       $\mathcal{V}(L_T^k)$ = the 1-hop neighbors, which are selected from $\bar{Y}_k$ and $\hat{C}$, of the nodes in $L_T^k$;
     each CPR $y$ in $\mathcal{V}(L_T^k)$ are automatically connected to its 1-hop neighbor in $L_T^k$, i.e., $\forall y \in \mathcal{V}(L_T^k) \cap \bar{Y}_k, y \in L_T^{k+1}$, then
       $\bar{Y}_k = \bar{Y}_k - \mathcal{V}(L_T^k)$;
     end
   end
   return $C'$;
4.3. Algorithm analysis

4.3.1. Time complexity

First, the time complexity of the first step of SCA is determined by the shortest path tree algorithm whose complexity is given by $O(N \lg N)$ (Bhattacharya and Kumar, 2014). Second, we analyze the time complexity of Algorithm 4. In the inner loop (lines 4–5), the shortest path tree algorithm is executed and lasts at most $N$ loops. Therefore, the time complexity of the inner loop of Algorithm 4 is $O(N \lg N)$. Then, the Greedy-Set-Cover with a time complexity of $O(N \lg N)$ (Cormen et al., 2001) is employed to find a minimum set cover in each iteration of the main loop. The time complexity of searching 1-hop neighbors (line 9) is $O(N \lg N)$. Since the main loop will run at most $\Delta_{\text{max}}$ loops, the time complexity of Algorithm 4 is given by $O(N \lg N \cdot \Delta_{\text{max}})$. In the third step of SCA, the shortest path tree algorithm runs at most $N$ times, and the time complexity to sort the remaining CDLs is $O(N \lg N)$. In turn, the time complexity of Algorithm 3 is $O(N \lg N)$. In summary, the total time complexity $t_{SCA}$ of SCA can be calculated as follows

$$t_{SCA} = O(N \lg N) + O(N^2) + O(N^3) = O(N^3).$$

4.3.2. Approximation ratio

Let $T^*$ and $T$ denote the optimal feasible tree and the feasible tree returned by SCA, respectively. Thus, the ratio between the optimal solution and the solution returned by the SCA algorithm is given by

$$r_{SCA} = \frac{\sum_{i=1}^{c} |L_i^c - \bigcup_{j=1}^{l} L_j^b - Y|}{\sum_{i=1}^{c} |L_i^c - \bigcup_{j=1}^{l} L_j^b - Y|} = \frac{\sum_{i=1}^{c} |L_i^c| - |Y|}{\sum_{i=1}^{c} |L_i^c| - |Y|}.$$  \hspace{1cm} (15)

Furthermore, we have

$$r_{SCA} = \frac{\sum_{i=1}^{c} |L_i^c| - |Y|}{\sum_{i=1}^{c} |L_i^c| - |Y|} < \frac{\sum_{i=1}^{c} |L_i^c|}{|OPT_k| \sum_{i=1}^{c} |L_i^c|}.$$

Let $OPT_k$ be a minimum set cover for the $k$th level of $T$. As GSA is employed to solve the set covering problem at each level, the approximation ratio of GSA (Cormen et al., 2001) is given by

$$\forall k \in \{1, 2, \ldots, l\}, \frac{|L_i^c|}{|OPT_k|} \leq \ln |T_k| + 1 \leq |Y| + 1.$$  \hspace{1cm} (16)

From Algorithm 1, we know that the second step of SCA implies the nonexistence of feasible solutions to the DCRNP problem without the help of RNs. Therefore, the optimal solution to DCRNP contains at least one RN, i.e.,

$$\left| \bigcup_{i=1}^{c} L_i^c - Y \right| \geq 1.$$  \hspace{1cm} (18)

which further implies that

$$\sum_{i=1}^{c} |L_i^c| \geq |Y| + 1.$$  \hspace{1cm} (19)

Besides, due to the fact that $|OPT_k| \leq |T_k| \leq |Y|$, combining inequalities (17)–(19), we can conclude that
\[ r_{\text{SCA}} < \sum_{i=1}^{l} \left( \frac{|\text{APT}_i|}{|\text{OPT}_i|} \right) \leq \frac{\ln|Y| + 1}{|Y| + 1} |Y| + 1 \]
\[ \leq \Delta_{\max}(1 - \frac{1}{|Y| + 1})|Y| + 1 = e(\ln|Y| + 1), \]
where \( e = (1 - \frac{1}{|Y| + 1}) \Delta_{\max} (0 < e < \Delta_{\max}). \)
As \( n \geq |Y| \), we can conclude that SCA is an \( O(\ln n) \)-approximation algorithm.

5. Algorithm to the DCRNP problem in two-tiered WSNs

5.1. Algorithm description

TSCA is composed of two phases, i.e., the covering phase and the connecting phase, in which CCA and SCA are employed, respectively. To be specific, in the covering phase, a set of CPRs are deployed by CCA to fully cover SNs in obedience to delay constraints. Then, in the connecting phase, the connectivity of the high-tier mesh network consisting of RNs is built by SCA. TSCA is explicitly described in Algorithm 6.

Algorithm 6. Two-tiered SCA (TSCA).

**Input:** The set \( S \) of SNs, a set \( C \) of CDLs and a sink \( K \).
**Output:** A set \( Y_f \) of deployed RNs.

\[
\text{begin}
1 \quad Y = \text{CPRs deployed by Algorithm 1;}
2 \quad T = \text{the feasible tree built by SCA to connect each CPR to the sink without violating the delay constraint;}
3 \quad Y_f = \text{the set of RNs on } T;
\text{return } Y_f;
\text{end}
\]

5.2. Algorithm analysis

5.2.1. Time complexity

The time complexity of TSCA is straightforward. Let \( t_{\text{CCA}} \) denote the time complexity of CPA. Then, the time complexity of TSCA is given by

\[ t_{\text{TSCA}} = t_{\text{CCA}} + t_{\text{SCA}} = O(N^3) + O(N^3) = O(N^3). \]  

(21)

5.2.2. Approximation ratio

Let \( \text{OPT} \) and \( \text{APT} \) be the optimal solution and the solution returned by TSCA, respectively. Let \( \text{OPT}_{ij} \) and \( \text{APT}_{ij} \) be the optimal solution and the solution returned by CCA in the covering phase, respectively. Then, we have that

\[ |\text{OPT}_{ij}| \leq |\text{APT}_{ij}| \leq (ln n + 1)|\text{OPT}_{ij}| \leq (ln n + 1)|\text{OPT}|, \]

(22)

where \((ln n + 1)\) is the approximation ratio of CCA.

Let \( T \) and \( \text{APT}_{ij} \) be the feasible tree returned by SCA and the set of RNs deployed in the connecting phase on \( T \), respectively. Thus, the ratio between \( |\text{APT}_{ij}| \) and \( |\text{OPT}| \) is given by

\[ \frac{|\text{APT}_{ij}|}{|\text{OPT}|} = \frac{|\sum_{i=1}^{l} |A_{ij}| - |\text{APT}_{ij}|}{|\text{OPT}|}. \]

(23)

where \( l \) is the number of levels of \( T \).

Similar to formulations (15)–(16), we have

\[ \frac{|\text{APT}_{ij}|}{|\text{OPT}|} < \Delta_{\max}(\ln n + 1). \]

(27)

5.3. Inequality (27) is trans-

Finally, considering inequalities (22) and (27), we have

\[ r_{\text{TSCA}} = \frac{|\text{APT}|}{|\text{OPT}|} \leq \frac{|\text{APT}_{ij}| + |\text{APT}_{ij}|}{|\text{OPT}|} \leq (\ln n + 1)|\text{OPT}| + \Delta_{\max}(\ln n + 1)|\text{OPT}| \]

(28)

which implies that the approximation ratio of TSCA is \( O(\ln n) \).

6. Simulation results

In simulations, SNs are randomly placed in a square field with the side length of 100 m, and the number \( n \) of SNs varies from 10 to 100. To ensure that the DCRNP problem has a feasible solution, in each simulation run, we set 400 randomly distributed CDLs in the deployment field. Without loss of generality, all the SNs have the same delay constraint \( \Delta \). The simulations are carried out under two scenarios, the homogeneous scenario, i.e., \( r = R = 10 \) or 15, and the heterogeneous scenario, i.e., \( r=10 \) and \( R=15 \). Numerical results are generated based on the method of batch means with 50 simulation runs for the confidence level of 95%.

These simulations are performed on an NS-3 simulator over a computer with Intel Core i5-2430M CPU and 2 GB RAM. The PHY and
MAC layers specified in IEEE 802.15.4 standard are used in simulations. As this paper only focuses on the impact of hop counts on network delay, to avoid collisions, each simulation run is divided into equal time slots, and in each time slot, only one SN is allowed to transmit packets. Totally 1000 packets with the same length will be transmitted by each SN. The parameters used in simulations are set as Table 1. The transmit power of a node is determined through statistical observation on the NS-3 simulator under the restriction that the PRR of a link between two neighboring nodes is larger than 95%. Then, the transmit power of a node is set as $-47.4$ dBm when its radius is 10 m, or $-40.4$ dBm when its radius is 15 m.

### 6.1. Number of deployed RNs

We first validate that CCA can further save the RNs deployed in the connecting phase. CCA achieves this by employing GSA in the PP selection procedure. Thus, to perform a fair comparison, we devise a revised CCA (rCCA) as baseline, which randomly selects a PP from each PA returned by the third step (i.e., line 3 in Algorithm 1) in the PP selection procedure instead of GSA. Therefore, given a set of SNs, rCCA and CCA deploy the same number of RNs in the covering phase. Then, SCA is employed in the connecting phase to build the network connectivity for the RNs deployed by rCCA and CCA, respectively, and we compare the number of RNs deployed in the connecting phase. The results shown in Fig. 7 exhibit that CCA deploys much fewer RNs than rCCA by introducing the GSA in the PP selection procedure. This implies that the covering phase and the connecting phase are tightly related, and should be studied jointly.

Then, we evaluate the performance of TSCA. The latest algorithm, denoted by PSH (Ma et al., 2016b), to the DCRNP problem in two-tiered WSNs is taken as a baseline. The simulation results are shown in Fig. 8. Obviously, TSCA deploys fewer RNs than PSH under different conditions, which verifies the effectiveness of TSCA. Fig. 8 also shows that both TSCA and PSH deploy more RNs as the delay constraints decrease, which implies that there exists a trade-off between the number of deployed RNs and the end-to-end delay.

### 6.2. Average delay

The simulation results on average delay are shown in Fig. 9. In addition to PSH, we also compare TSCA with another RNP algorithm—ARNPc (Misra et al., 2010), which deploys RNs without considering delay constraints, to show the impact of delay constraints on average delay. We can see from Fig. 9 that the average delays of networks built by ARNPc are much larger than those built by the other two algorithms and increase significantly with the number of SNs, which shows that delay constraints should be carefully concerned when time-
sensitive networks are designed. Furthermore, we can learn from Figs. 8 and 9 that increasing communication radii is another way to reduce the number of deployed RNs and decrease the end-to-end delay, but this comes at a price of increasing the energy consumption.

6.3. Transmission reliability

We also perform simulation on end-to-end transmission reliability, which is measured in terms of Packet Reception Rate (PRR) in this paper. The PRR is defined as the ratio of the total received packets to the total transmitted packets. The simulation results are shown in Fig. 10, which exhibits that TSCA and PSH can build a network ensuring a relatively constant PRR, whereas the PRRs in the networks built by ARNPc decease dramatically with the number of SNs since ARNPc constructs network topologies without considering delay constraints. In addition, Fig. 10 also shows that networks suffer a performance degradation in transmission reliability when the delay constraint grows larger. This implies that we can alter $\Delta$ to control the end-to-end reliability.

6.4. Running time

The comparison of the running time between PSH and TSCA is shown in Fig. 11, which indicates that these algorithms all require an increasing running time as the decrease of the communication radii or the delay constraint. Besides, although TSCA has a larger running time than PSH, considering the noticeable amount of the RNs saved by TSCA, the extra running time is worthwhile.

7. Conclusion

In this paper, we have studied the DCRNP problem in two-tiered WSNs and proposed an algorithm-TSCA to address this problem. Specifically, TSCA solves this problem in two phases, i.e., the covering phase and the connecting phase, in which CCA and SCA are employed, respectively. CCA deploys RNs to fully cover distributed SNs with respect to delay constraints, and meanwhile tries its best to save the RNs deployed in the connecting phase. Then, SCA builds network connectivity in obedience to delay constraints, which formulates the deployment of RNs as the set covering problem and employs GSC to search such a set cover. Through rigorous analysis, we have proved that TSCA is a polynomial-time algorithm with an approximation ratio of $O(\ln n)$. Finally, the effectiveness of TSCA has been verified through extensive simulations on an NS-3 simulator.

Existing works employ the ideal geometric disk wireless channel model, which cannot accurately capture the characteristics of unreliable wireless channel. Moreover, these works are only verified through simulations. Therefore, we expect to devise algorithms that is capable of guiding real-world RN deployments and verify their efficiency via experiments in the future.
Fig. 10. The comparison of PRR between TSCA, PSH and ARNPc under different radii and delay constraints. (a) $r = R = 10$. (b) $r = 10, R = 15$. (c) $r = R = 15$.

Fig. 11. The comparison of running time between PSH and TSCA under different radii and delay constraints. (a) PSH. (b) TSCA.
Acknowledgment

This work was supported by the Natural Natural Science Foundation of China under grant 61673371 and 71661147005, Youth Innovation Promotion Association, CAS (2015157) and the Program Project of Beijing Municipal Education Commission (KM 201511417008), Open Project of Beijing Key Laboratory of Information Service Engineering (ZK20201402).

References


Chaofan Ma received the B.S degree in Electrical Engineering and Automation from Henan Polytechnic University, China, in 2008. He received the M.S degree in Circuits and Systems from Dalian University of Technology, China, in 2013. He is currently working toward the Ph.D. degree in Detection Technology and Automatic Equipment at Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests are in wireless sensor networking, and embedded systems.

Wei Liang received the Ph.D. degree in Mechatronic Engineering from Shenyang Institute of Automation, Chinese Academy of Sciences, in 2002. She is currently serving as a professor of Shenyang Institute of Automation, Chinese Academy of Sciences. Her research interests are in the areas of wireless sensor network, dynamic scheduling theory and system simulation.

Meng Zheng received the B.S. degree in Information and Computing Science, and the M.S. degree in Operational Research and Cybernetics from Northeastern University, Shenyang, P.R. China, and the Ph.D degree in Mechatronic Engineering from University of Chinese Academy of Sciences, Beijing, P.R. China, in 2005, 2008 and 2012, respectively. He had been a visiting student from 2010 to 2012 in the Fraunhofer Institute for Telecommunication, Heinrich-Hertz-Institute, Berlin, Germany. Currently, he is serving as an associate professor at the Key Laboratory of Networked Control Systems, Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests are in the areas of wireless ad hoc and sensor networks, cognitive radio network and security in smart grid.