Efficient Algorithms for Sensor Deployment and Routing in Sensor Networks for Network-structured Environment Monitoring

Shuguang Xiong†, Lei Yu‡, Haiying Shen‡, Chen Wang†, Wei Lu†
†IBM China Research Lab, Beijing, China
‡Department of ECE, Clemson University, Clemson, SC, USA
xiongshg@cn.ibm.com, {leiy, shenh}@clemson.edu, {wangcwc, luw}@cn.ibm.com

***Abstract*—When monitoring environments with wireless sensor networks, optimal sensor deployment is a fundamental issue and an effective means to achieve desired performance. Selecting best sensor deployment has a dependence on the deployment environments. Existing works address sensor deployment within three types of environments including one dimensional line, 2-D field and 3-D space. However, in many applications the deployment environments usually have network structures, which cannot be simply classified as the three types. The deployed locations and communications of sensor nodes are limited onto the network edges, which make the deployment problem distinct from that in other types of environments. In this paper, we study sensor deployment in network-structured environments and aim to achieve k-coverage while minimizing the number of sensor nodes. Furthermore, we jointly consider the optimization of sink deployment and routing strategies with the goal to minimize the network communication cost of data collection. To the best of our knowledge, this paper is the first one to tackle sensor/sink deployment under the deployment constraints imposed by the network structure. The hardness of the problems is shown. Polynomial-time algorithms are proposed to determine optimal sensor/sink deployment and routing strategies in tree-topology network structure. Efficient approximation algorithms are proposed for the general graph network structure and their performances are analyzed. Theoretical results and extensive simulations show the efficiency of the proposed algorithms.**

I. Introduction

Wireless sensor networks (WSNs) are increasingly deployed for many monitoring tasks, such as traffic monitoring [1], water quality monitoring [2], pipeline monitoring [3], [4]. These tasks often put forward requirements to WSNs on sensing quality, network lifetime, coverage, etc. Since the positioning of sensors has essential impact on these performances, sensor deployment is an important research issue, and extensively studied in the literature [5].

The strategies of sensor deployment are usually proposed with the goals to maximize area coverage [6]–[8] or sensing quality [9], [10], or to minimize the number of sensors [11] or communication cost [12], [13], while under the constraints of some other performance metrics. These works address sensor deployment problems in deployment environments of one dimensional line [4], [14], [15], 2-D plane field [7]–[11], [13], or 3-D space [6].

However, in many applications, we note that the underlying environments where sensor networks are deployed have network structures, such as river network, oil pipeline network and road network. The sensor nodes are usually manually deployed along the edges, i.e., riverways, pipelines and roads, to monitor river quality, oil temperature and traffic. That is, the deployment locations of sensors are limited into the underlying network structure. As we can see, few works address the sensor deployment problem with such deployment limitation imposed by the network structure. To fill the gap, this paper addresses sensor deployment in network-structured environments.

In WSNs, sensing coverage is a basic functionality and an important QoS (Quality of Service) measure of the network. To tolerate sensor failure and precisely localize the monitoring object, many applications may impose the requirement of k-coverage of the deployment area with $k > 1$, which means that every location in the area can be monitored by at least k sensors, where k is a given parameter. Due to its great importance, k-coverage has been widely studied in 2-D [16], [17] field and 3-D space [6]. On the other hand, to save the deployment cost of WSNs, users may want to deploy a minimum number of sensors while desired performances can be achieved. Therefore, in this paper, we consider the problem of deploying sensors to ensure that the network structure is k-covered, while minimizing the number of sensors. As opposed to previous works, the deployment locations of sensors are restricted by network structure in this problem.

Besides, data collection is an essential task for the monitoring applications, in which all sensors transmit their readings to the sinks. Since the amount of data may be large and sensor nodes are usually battery-powered, it is critical to design energy-efficient strategies for data collection to prolong network lifetime. Given a sensor deployment, the locations of sinks and the routing strategy of sensor data would have profound effects on the communication cost incurred by data collection. Therefore, in this paper we jointly consider the optimization problems of sink deployment and routing with the goal to minimize the communication cost. In particular, the deployed locations of sinks are limited to the intersections of network structures and the sensors transmit their readings along the edges to the sinks, which follows our observations from real applications like oil pipeline/road networks. In road networks, the traffic management center collects the information from sinks, which are usually deployed around the intersections to collect sensor readings from incident roads [1]. In pipeline networks, the heat stations are built along

the pipelines. Sinks are deployed with the stations, whose positions can be treated as intersections, to collect temperature and pressure readings and send them to the stations [4].

In summary, our contribution is as the following.

(1) We point out the current lack of works on optimal sensor deployment in network-structured environments and study the the problem of k-coverage sensor deployment on underlying networks with the goal to minimize the number of sensors. We prove that the problem is NP-Complete, and propose a polynomial-time optimal algorithm for tree-topology network structures and an approximation algorithm for general topology with performance analysis.

(2) Given a sensor deployment on a network structure, we study joint optimization of sink deployment on the intersections and routing strategies along the edges to minimize the total communication cost of data collection. We provide a polynomial-time optimal algorithm for tree-topology network structures and a heuristic algorithm for general topology.

(3) We provide theoretical analysis and also conduct extensive simulations to evaluate the performance of proposed algorithms. The results show the efficiency of our algorithms.

The rest of this paper is organized as follows. Section II summarizes related works. Section III introduces the network model and gives the problem description. Section IV formulates the sensor deployment problem and investigates its hardness and Section V proposes algorithms for network structure of tree and general graph topology. Section VI formulates the sink deployment and routing problem and Section VII proposes corresponding algorithms. Section VIII presents the simulation results. Section IX concludes this paper.

II. Related Work

There are extensive works that focus on node deployment in WSNs. In terms of the role of nodes, the existing works address the deployment of sensors, relays and data collectors(namely, sinks and base stations). They consider node deployment with various optimization goals including minimizing the number of sensor ndoes [11] and communication cost [12], [13], and maximizing coverage [6]–[8] and sensing quality [9], [10]. A comprehensive overview of node deployment is provided by Younis et al. [5].

Existing solutions for sensor deployment depend on the types of the deployment environment including one dimensional line [4], [14], [15], 2-D plane field [7]–[11], 3-D space [6]. Liu et al. [14] investigate the problem of how to optimally deploy the data back-haul nodes in a linear topology to maximize the lifetime of WSNs and propose a greedy deployment scheme to achieve near-optimal performance. Chen et al. [15] study the optimal sensor deployment in a linear network with two objectives that are to maximize the network lifetime and to minimize the application-specific cost, given the number of sensors and certain coverage requirement. Guo et al. [4] also study the linear sensor placement problem in monitoring oil pipelines with the goal of maximizing the network lifetime. Besides, most existing works address sensor deployment on 2-D plane field. S.Dhillon et al. [11] propose two algorithms to determine the minimum number of sensors and their locations, under the constraints of imprecise detections and possible obstacles in the terrain. Wang et al. [7] study the sensor deployment problem to minimize the number of sensors while achieving k-coverage of the area. Zou et al. [8] propose a virtual force algorithm as a sensor deployment strategy to maximize the sensor field coverage. Krause et al. [9] address sensor deployment with the goals to minimize the communication cost under the requirement of specified sensing quality and to maximize sensing quality subject to budget on the communication cost. Zhang et al. [10] study best sensor deployment to minimize estimation distortion at the fusion center. Besides, Andersen et al. [6] consider the problem of deploying wireless sensors in a three dimensional space to achieve a desired degree of coverage, while minimizing the number of sensors.

Existing works consider the optimal sink placement problem on 2-D plane. Youssef et al. [18] address the gateway placement with the goal to minimize the number of hops between a sensor and one of the gateways to reduce latency. Pan et al. [19] aim to locate the base station optimally such that the lifetime of battery-powered video nodes can be maximized. Shi et al. [20] consider the base station placement with the optimization objectives of network lifetime and capacity. As we can see, the deployment problems of sensors and sinks have not been addressed with location constraints imposed by network-structured environments. This paper fills the gap.

Many literatures have studied routing problems for reducing the communication cost of data collection and prolonging the network lifetime. For example, Park et al. [21] develop an online heuristic for the problem of routing message to maximize the network lifetime. Chang et al. [22] formulate the routing problem as a linear program, with the objective to maximize the network lifetime defined by the time until the network partition due to battery outage. Given the locations of sensors and the base station, Kalpakis et al. [23] propose algorithms to determine the routings for data collection from all sensors to the base station with the goal to maximize the network lifetime. Xiong et al. [24] formulate the maximum lifetime data collection problem as an integer program to get close to optimality. As opposed to these works, we jointly search optimal solutions for routing strategies and sink deployment to minimize the communication cost of data collection.

III. System Model and Problem Description

A. System Model

We assume a WSN is deployed in a network-structured environment, such as oil pipeline/road network. The environment can be modeled as an undirected graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$, in which a vertex $A \in \mathbb{V}$ refers to an intersection in the network, and an edge $AB \in \mathbb{E}$ stands for the connection between the two intersections. For convenience, an edge is regarded as a line segment, rather than its physical shape, i.e., a curve in real world. The edge can be treated as the straightened result of the real physical shape. The length of edge AB means the distance between intersection A and B, denoted as $|AB|$. The location of a point P on AB, denoted as $Loc(P)$, is represented by $(AB : x)$ or $(BA : |AB| - x)$, where x is the distance between P and A on edge AB. For two arbitrary points P and Q on the

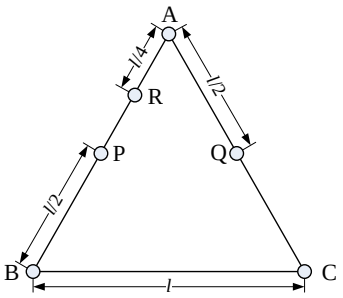

Fig. 1. A sample network-structured environment with three intersections A, B, C, and line segments AB, AC, BC of equal length l. P, R, and Q are three points on AB and AC, resp. Note that although the lengths of the line segments reflect the geographic distances, it is not always be the case.

edges in $\mathbb{E}$, a path from P to Q, denoted as $PA_1A_2 \dots A_mQ$, is a sequence of line segments PA_1, A_1A_2, ..., A_mQ, where PA_1 and A_mQ are parts of the edges in $\mathbb{E}$ where P and Q locate, resp., and $A_iA_{i+1} \in \mathbb{E}$ for $1 \leq i < m$. The length of the path is the sum of the lengths of the line segments. The *network distance* between P and Q is the minimum length of all the possible paths from P to Q, denoted as $\langle PQ \rangle$. For instance, in Figure 1, (PA, AQ) and (PB, BC, CQ) are two paths from P to Q, $\langle PR \rangle = l/4$ and $\langle PQ \rangle = l$.

Most existing works on sensor deployment in 2-D field assume the disc sensing model, i.e., the sensing range of a sensor is a disc with a certain radius centered at the position of the sensor. However, in network-structured environments, the sensing range of a sensor is restricted to the edges of the network. Therefore we use a new sensing model for the network-structured environment: the sensing range of a sensor A consists of every point B on the network with $\langle AB \rangle \leq d$, where d is a threshold provided by the user. Such assumption is reasonable and practical. For example, in water quality monitoring of a river network, the measurement of a sensor indicates the situation of waters within a certain distance to that sensor along the tributaries.

For environment monitoring, the sensors are deployed on the edges, i.e., roads or pipelines, to measure the traffic or oil temperature, while the sinks are deployed on the vertices to collect data from sensors on incident edges through wireless radios. In real environments, wireless links usually cannot be established across edges due to possible obstructions and long distances among the edges. Thus, we assume that there is no data transmission across edges, a sensor only forwards data to other sensors or sinks located on the same edge, and each sensor has one unit data to be delivered in a data collection task. Let the distance between two sensors P and Q be d, the communication cost for P to send one unit data to Q, denoted as $c(PQ)$, can be modeled as $c(PQ) = \alpha d^{\beta} + \gamma$, where α, β, γ are system-dependent parameters and $2 \leq \beta \leq 4$ [25].

B. Problem Description

In this paper, we focus on the following two problems:

(1) Given a network structure, how to deploy sensors to achieve k-coverage on the network structure while the number of sensors is minimized?

(2) Given a network structure and a sensor deployment, how to deploy sinks on the intersections of the network structure and how to determine a routing path from each sensor to a sink along each edge, such that the total communication cost incurred by a data collection task is minimized?

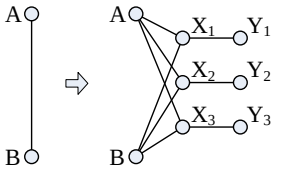

Fig. 2. The construction from an instance of the Vertex Cover problem to that of the Sensor Deployment problem. The edge AB is removed, while 6 vertices and 9 edges with a equal length of $\frac{d}{2}$ are added in the new graph.

IV. Hardness of Sensor Deployment Problem

Problem 1 (Sensor Deployment)**.** *Given a network structure $\mathbb{G}(\mathbb{V}, \mathbb{E})$, a distance threshold $d > 0$, and a coverage degree $k > 0$, the goal is to find out a set of points on the edges in $\mathbb{E}$ with the minimum size, where sensors are deployed, denoted as $\mathbb{S}$, such that for arbitrary point P on any edges in $\mathbb{E}$, $\exists \mathbb{Q} \subseteq \mathbb{S}$, where $|\mathbb{Q}| \geq k$ and $\langle PQ \rangle \leq d$, $\forall Q \in \mathbb{Q}$.*

Theorem 1. *The Sensor Deployment problem in decision version is NP-Complete, even if the coverage degree $k = 1$.*

Proof: We consider the decision version of the Sensor Deployment (SD) problem with $k = 1$, which asks whether such an $\mathbb{S}$ exists fulfilling the constraints with $|\mathbb{S}| \leq t$ for a given parameter t. First, we prove that the SD problem is in NP. Next, we conduct a polynomial-time reduction from the Vertex Cover (VC) problem to the SD problem.

A polynomial-time algorithm can be used to check the validity of an answer to the problem. For each $P \in \mathbb{S}$, all the paths starting from P with length d are computed, and the line segments on the paths are marked as "active". The validity of the answer can be verified by checking whether all the edges in $\mathbb{E}$ are fully covered by the active line segments. Since there are $O(t \cdot |\mathbb{E}|)$ active line segments, it requires time polynomial in the input size, and the problem is in NP.

Given a graph $\mathbb{G}^*(\mathbb{V}^*, \mathbb{E}^*)$, the VC problem asks for a set $\mathbb{C}^* \subseteq \mathbb{V}^*$ with the minimum size, such that each edge in $\mathbb{E}^*$ is incident to at least one vertex in $\mathbb{C}^*$. For an arbitrary instance $\mathbb{G}^*(\mathbb{V}^*, \mathbb{E}^*)$ of the VC problem, the instance of the SD problem can be constructed by (1) assigning $k = 1$ and d, and (2) building a graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$. Initially, $\mathbb{V} = \mathbb{V}^*$ and $\mathbb{E} = \emptyset$. Then for each edge $AB \in \mathbb{E}^*$, two groups of vertices are added into $\mathbb{V}$, which are $\mathbb{X} = \{X_1, X_2, X_3\}$ and $\mathbb{Y} = \{Y_1, Y_2, Y_3\}$, and nine edges are added into $\mathbb{E}$, as shown in Figure 2. The length of each edge is set to $d/2$. It is easy to see that the construction takes time polynomial in the size of $\mathbb{G}^*(\mathbb{V}^*, \mathbb{E}^*)$.

Now we show that the VC problem has an answer $\mathbb{C}^*$ of size t or less, *iff* the SD problem has an answer $\mathbb{S}$ of size t or less. On one hand, suppose $\exists \mathbb{C}^* \subseteq \mathbb{V}^*$ such that $\mathbb{C}^*$ is a vertex cover and $|\mathbb{C}^*| \leq t$, let $\mathbb{S} = \mathbb{C}^*$, then at least one vertex in $\{A, B\}$ is in $\mathbb{S}$ for any $AB \in \mathbb{E}^*$. Therefore, all the points on the edges in $\mathbb{E}$ are within a distance no more than d to A or B, and $\mathbb{S}$ is an answer to the SD problem. On the other hand, suppose the SD problem has an answer $\mathbb{S} \subseteq \mathbb{V}$ such that $|\mathbb{S}| \leq t$. If $\exists AB \in \mathbb{E}^*$ such that $A \notin \mathbb{S}$ and $B \notin \mathbb{S}$, $\mathbb{S}$ must have three points $\{P_i | 1 \leq i \leq 3\}$ where each P_i is on one of edges

Algorithm 1 The SDT Algorithm

INPUT: A tree rooted at R_r with weighted edges, k and d.
OUTPUT: $\mathbb{S} = \{S_i | 0 \leq i < t\}$ with $Loc(S_i)$ for $0 \leq i < t$.

$t = 0$ and $\mathbb{S} = \emptyset$;
Sensor_on_Tree(R_r);
return $\mathbb{S}$;
procedure Sensor_on_Tree(P)
$ret = 0$ and $y = 0$;
for each child C of vertex P **do**
  $l = |PC|$ – Sensor_on_Tree(C);
  while $l \geq 2d$ **do**
    $\mathbb{S} = \mathbb{S} \cup \{S_i | Loc(S_i) = (PC : l - d), t \leq i < t + k\}$;
    $t = t + k$ and $l = l - 2d$;
  if $l > d$ **then**
    $\mathbb{S} = \mathbb{S} \cup \{S_i | Loc(S_i) = (PC : l - d), t \leq i < t + k\}$;
    $t = t + k$;
    $ret = \max\{ret, 2d - l\}$;
  else if $l \geq 0$ **then**
    $y = \max\{y, l\}$;
  else
    $ret = \max\{ret, -l\}$;
if $y > ret$ **then**
  $ret = -y$;
  if $P = R_r$ **then**
    $\mathbb{S} = \mathbb{S} \cup \{S_i | Loc(S_i) = Loc(P), t \leq i < t + k\}$;
    $t = t + k$;
return ret;

$\{AX_i, BX_i, X_iY_i\}$ excluding A and B, such that the points $\{Y_i | 1 \leq i \leq 3\}$ can be covered. Clearly, all the points covered by the sensors deployed at $\{P_i | 1 \leq i \leq 3\}$ are within the coverage of the sensors deployed at A and B. Hence, a valid answer to the SD problem, denoted as $\mathbb{S}'$, can be obtained by replacing $\{P_i | 1 \leq i \leq 3\}$ with A and B for all $AB \in \mathbb{E}^*$ where $A \notin \mathbb{S}$ and $B \notin \mathbb{S}$. Since $|\mathbb{S}'| \leq t$ and $\forall AB \in \mathbb{E}^*$, at least one vertex in $\{A, B\}$ is in $\mathbb{S}'$, let $\mathbb{C}^* = \mathbb{S}'$, and $\mathbb{C}^*$ is an answer to the VC problem. Because the VC problem is NP-Complete [26], the SD problem is NP-Complete. ■

V. Algorithms for Sensor Deployment Problem

In this section, we present an algorithm called SDT (Sensor Deployment on Trees) to compute an optimal sensor deployment when the network structure is a tree, and an approximation algorithm named SDG (Sensor Deployment on Graphs) for general network structure. The optimality and approximation ratio of the algorithms are proved.

A. An Optimal Algorithm for Tree Topologies

Denote the tree rooted at P as $Tree(P)$, and the SDT algorithm computes an optimal or "partial-optimal" answer for $Tree(P)$, $\forall P \in \mathbb{V}$, with the corresponding *extra coverage* maximized. The answer is denoted by $\mathbb{S}(P)$, and we say that $\mathbb{S}(P)$ is partial-optimal, if (1) $Tree(P)$ cannot be fully k-covered by $\mathbb{S}(P)$ with $|\mathbb{S}(P)| = |\mathbb{S}_{opt}| - k$ where $\mathbb{S}_{opt}$ is an optimal answer for $Tree(P)$, and (2) $\mathbb{S}(P)$ becomes an optimal answer when additional k sensors deployed at P are added into $\mathbb{S}(P)$. The extra coverage of the sensors on $Tree(P)$ refers to their ability to cover the network structure. Specifically, if $Tree(P)$ is fully k-covered, the extra coverage is the maximum distance between P and any k-covered point $Q \notin Tree(P)$ by the sensors in $\mathbb{S}(P)$. Otherwise, it refers to the opposite number of the maximum distance between P and any point $Q \in Tree(P)$ not k-covered by $\mathbb{S}(P)$.

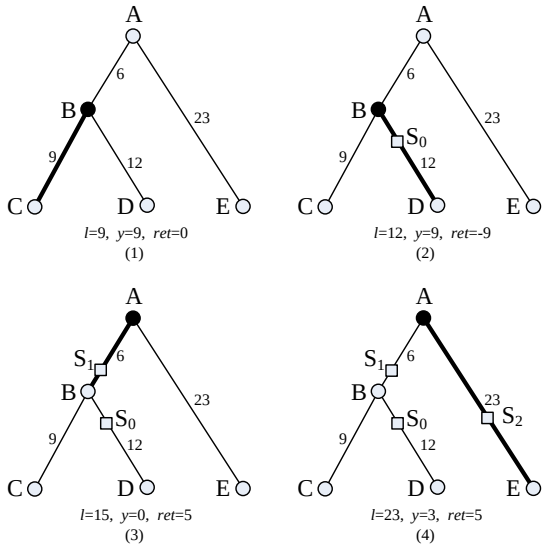

Fig. 3. A running example of the SDT algorithm, in which $|AB| = 6$, $|BC| = 9$, $|BD| = 12$, and $|AE| = 23$. The sub-figures show four successive steps. In each sub-figure, the P and PC referred in Algorithm 1 are marked black and thick, resp.

By employing procedure named Sensor_on_Tree, the algorithm computes $\mathbb{S}(P)$ in a bottom-up way, i.e., $\mathbb{S}(C_i)$ is computed before $\mathbb{S}(P)$ for each child C_i ($1 \leq i \leq p$) of P. $\mathbb{S}(C_i)$ is guaranteed to be part of an optimal answer, as proved in Theorem 2, hence in computing $\mathbb{S}(P)$, the algorithm only deploys sensors on each PC_i. First, it calculates l, the length of the fragment on PC_i that cannot be covered by $\mathbb{S}(C_i)$. Next, it deploys $\lfloor l/(2d) \rfloor k$ sensors on PC_i, and reduces l to $l - \lfloor l/(2d) \rfloor 2d$. After that, if $l \in (d, 2d)$, k sensors are deployed on the uncovered fragment, otherwise no sensor is deployed no matter $l \geq 0$ or not. When the deployment operations finish for all PC_i, the algorithm computes the longest distance between P and any uncovered point in $Tree(P)$, and the longest extra coverage provided by the sensors on $Tree(C_i)$, denoted as y and ret, resp. If $y \leq ret$, $\mathbb{S}(P)$ is optimal, and ret is returned as the extra coverage, otherwise $\mathbb{S}(P)$ is partial-optimal, and the algorithm returns $-y$ and leave the uncovered fragments to Sensor_on_Tree(Q) where Q is an ancestor of P. If P is the root, k sensors are deployed at P. The pseudo code of the SDT algorithm is shown in Algorithm 1.

Figure 3 depicts a running example of Algorithm 1 with $d = 10$, $k = 1$, and the tree rooted at A. Sensor_on_Tree(P) traverses the tree and takes each vertex as its parameter in pre-depth-first order. Sensor_on_Tree(C), Sensor_on_Tree(D), and Sensor_on_Tree(E) return 0 since C, D, and E are all leaf vertices. When executing Sensor_on_Tree(B), the algorithm deploys no sensor on BC since $|BC| < d$, and deploys S_0 at $(DB : 10)$, as shown in Figure 3(1)-(2). Because S_0 cannot cover all the points on BC, Sensor_on_Tree(B) returns $-|BC|$. When executing Sensor_on_Tree(A), the algorithm deploys S_1 at $(BA : 1)$ so as to cover all the points on BC, and deploys S_2 at $(EA : 10)$, as shown in Figure 3(3)-(4). Since S_1 can cover all the points on AE that are uncovered by S_2, the algorithm terminates and returns $\mathbb{S} = \{S_0, S_1, S_2\}$, otherwise it should deploy another sensor at A.

Theorem 2. *Algorithm 1 returns an optimal answer to the Sensor Deployment problem on tree topologies.*

Proof: Clearly, Algorithm 1 returns an answer such that all the points on $Tree(R_r)$ are k-covered by the sensors. By imposing induction on the root of a tree, we prove an equivalent claim that $\forall P \in \mathbb{V}$, $\mathbb{S}(P)$ derived by the algorithm is either optimal or partial-optimal with the corresponding extra coverage maximized.

First, consider an edge $PC \in \mathbb{E}$ where C is a leaf, and let $|PC| = (2x + \delta)d$, in which x is a non-negative integer and $0 \leq \delta < 2$. Since any optimal answer for PC can be converted by moving the sensors to eliminate the fragments covered by more than k sensors, the number of sensors on PC is at least $k(x + \lfloor \delta \rfloor)$, as calculated by Algorithm 1 in line 6-13. Obviously, the extra coverage provided by the sensors on PC is maximized, and the answer is optimal to cover PC if $\delta \geq 1$, otherwise it is partial-optimal because an optimal answer requires additional k sensors at P.

For any $Tree(P)$ of height 1, if all the points on $Tree(P)$ are k-covered, $\mathbb{S}(P)$ derived by Sensor_on_Tree(P) is optimal for $Tree(P)$, because the number of sensors on each edge is the lower bound in any optimal answer. Otherwise, since the extra coverage provided by the sensors on each edge is maximized and 0-covered points remain, an optimal answer requires at least k additional sensors to cover $Tree(P)$. And if k sensors are deployed at P, $Tree(P)$ can be fully covered since $\langle PQ \rangle \leq d$, $\forall Q$ uncovered by $\mathbb{S}(P)$. Therefore, $\mathbb{S}(P)$ is partial-optimal. In both cases, the extra coverage returned by Sensor_on_Tree(P) is maximized since the extra coverage provided by the sensors on each edge is maximized.

Next, assume $\mathbb{S}(C_i)$ computed by the algorithm are optimal or partial-optimal for P's children $\{C_i | 1 \leq i \leq p\}$ with their extra coverage maximized. Denote an optimal or partial-optimal answer for $Tree(P)$ as $\mathbb{S}_{opt}$, and replace the sensors on $Tree(C_i)$ $(1 \leq i \leq p)$ in $\mathbb{S}_{opt}$ with those in $\mathbb{S}(P)$ to obtain another sensor set, denoted as $\mathbb{S}_{temp}$. According to the assumption, $|\mathbb{S}_{temp}| \leq |\mathbb{S}_{opt}|$.

If $\mathbb{S}_{temp}$ covers less points than $\mathbb{S}_{opt}$, $\exists C_i$ such that the number of sensors on $Tree(C_i)$ in $\mathbb{S}_{opt}$ is at least k larger than that in $\mathbb{S}_{temp}$, because no sensor set on $Tree(C_i)$ of size $|\mathbb{S}(C_i)|$ can cover more than $\mathbb{S}(C_i)$. In this case, we add k sensors at each of such C_i into $\mathbb{S}_{temp}$, and the obtained extra coverage is no less than that in $\mathbb{S}_{opt}$. Thus all the points covered by $\mathbb{S}_{opt}$ are covered by $\mathbb{S}_{temp}$, and $|\mathbb{S}_{temp}| \leq |\mathbb{S}_{opt}|$.

Let the sensors on $\{PC_i | 1 \leq i \leq p\}$ in $\mathbb{S}(P)$ and $\mathbb{S}_{temp}$ be $\mathbb{S}_1$ and $\mathbb{S}_2$, resp. Since $\mathbb{S}_1$ is optimal or partial-optimal to cover each edge of length $|PC_i| - e_1(i)$, and $e_1(i) \geq e_2(i)$, where $e_1(i)$ and $e_2(i)$ are the extra coverages on $Tree(C_i)$ in $\mathbb{S}(P)$ and $\mathbb{S}_{temp}$, resp., $|\mathbb{S}_1| \leq |\mathbb{S}_2|$. Replace $\mathbb{S}_2$ in $\mathbb{S}_{temp}$ with $\mathbb{S}_1$, and $\mathbb{S}_{temp} = \mathbb{S}(P)$ while the extra coverage on $Tree(P)$ is maximized. Therefore $\mathbb{S}(P)$ is optimal or partial-optimal with the extra coverage maximized. If Sensor_on_Tree(R_r) in line 4-17 computes a partial-optimal answer, the algorithm deploys k sensors at R_r so that $Tree(R_r)$ is fully covered, and the derived $\mathbb{S}(R_r)$ is optimal. Therefore, Algorithm 1 computes an optimal answer for $Tree(R_r)$. ■

Algorithm 2 The SDG Algorithm

INPUT: A graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$ with weighted edges, k and d.
OUTPUT: $\mathbb{S} = \{S_i | 0 \leq i < t\}$ with $Loc(S_i)$ for $0 \leq i < t$.

1: $t = 0$ and $\mathbb{S} = \emptyset$;
2: **for** each edge $AB \in \mathbb{E}$ **do**
3: $\quad |AB| = |AB| - \lfloor |AB|/(2d) \rfloor 2d$;
4: **for** each vertex $A \in \mathbb{V}$ and each edge $BC \in \mathbb{E}$ **do**
5: $\quad$ **if** $d - \langle AB \rangle \in (0, |BC|)$ **then**
6: $\quad\quad \mathbb{P} = \mathbb{P} \cup \{P\}$ with $Loc(P) = (BC, d - \langle AB \rangle)$;
7: $\quad$ **if** $d - \langle AC \rangle \in (0, |BC|)$ **then**
8: $\quad\quad \mathbb{P} = \mathbb{P} \cup \{Q\}$ with $Loc(Q) = (CB, d - \langle AC \rangle)$;
9: $\quad$ **if** $P, Q \in \mathbb{P}$ and $|BP| + |CQ| > |BC|$ **then**
10: $\quad\quad$ remove P and Q from $\mathbb{P}$;
11: compute $\mathbb{L}$ according to $\mathbb{V}$ and $\mathbb{P}$;
12: solve the set cover problem $(\mathbb{V}, \mathbb{L})$ to obtain $\mathbb{S}_0 \subseteq \mathbb{V}$;
13: **for** each vertex $A \in \mathbb{S}_0$ **do**
14: $\quad \mathbb{S} = \mathbb{S} \cup \{S_i | Loc(S_i) = Loc(A), 1 \leq i \leq k\}$;
15: $\quad t = t + k$;
16: restore the lengths of all the edges in $\mathbb{E}$;
17: **for** each edge $AB \in \mathbb{E}$ with $|AB| \geq 2d$ **do**
18: $\quad$ find P at $(BA : l)$, $\langle PC \rangle = d$, $\langle BC \rangle \leq d$ for some $C \in \mathbb{S}_0$;
19: $\quad$ **while** $|AB| \geq 2d$ **do**
20: $\quad\quad \mathbb{S} = \mathbb{S} \cup \{S_i | Loc(S_i) = (AB : |AB| - l - d), t \leq i < t + k\}$;
21: $\quad\quad t = t + k$ and $|AB| = |AB| - 2d$;
22: **return** $\mathbb{S}$;

B. An Approximation Algorithm for Graph Topologies

When the network structure is a general graph, the problem seems much more complicated: multiple paths should be considered when determining whether a point is covered by a sensor, and there may be even no leaf vertex that can be used to initiate a greedy deployment. Enlightened by the Set Cover problem and the related algorithms, we propose the SDG (Sensor Deployment on Graphs) algorithm, as shown in Algorithm 2. The algorithm finds out a set of *split points*, denoted as $\mathbb{P}$. $\forall P \in \mathbb{P}$, $\exists A \in \mathbb{V}$, such that $\langle PA \rangle = d$. The edges are split by the vertices in $\mathbb{V}$ and the points in $\mathbb{P}$ into a set of line fragments, denoted as $\mathbb{L}$. Then the algorithm computes a subset of $\mathbb{V}$ by greedy selections to cover all the lines in $\mathbb{L}$, and deploys sensors on the subset of vertices.

Specifically, the SDG algorithm deploys $k\lfloor |AB|/(2d) \rfloor$ sensors on edge AB, $\forall AB \in \mathbb{E}$ and $|AB| \geq 2d$, so that a fragment on AB with length $\lfloor |AB|/(2d) \rfloor 2d$ can be covered, and the uncovered parts have a total length less than $2d$. The set of such sensors is denoted as $\mathbb{S}_e$. Next, the algorithm finds out another set of sensors called $\mathbb{S}_v$ in which A is at a vertex in $\mathbb{V}$, $\forall A \in \mathbb{S}_v$, so that the uncovered parts can be covered by the sensors in $\mathbb{S}_v$. Finally, the exact locations of the sensors in $\mathbb{S}_e$ are determined according to $\mathbb{S}_v$, as shown in line 17-21.

To obtain $\mathbb{S}_v$, the algorithm computes $\mathbb{S}_0 \subseteq \mathbb{V}$ in which the vertices can cover all the lines in $\mathbb{L}$, and then let k sensors in $\mathbb{S}_v$ locate at A for each $A \in \mathbb{V}$. To obtain $\mathbb{P}$, the algorithm assigns $|AB| = |AB| - \lfloor |AB|/(2d) \rfloor 2d$, $\forall AB \in \mathbb{E}$, computes $\langle AB \rangle$ $\forall A, B \in \mathbb{V}$ by shortest-path algorithms, and then adds P or Q into $\mathbb{P}$ if $|BP| = d - \langle AB \rangle \in (0, |BC|)$ or $|CQ| = d - \langle AC \rangle \in (0, |BC|)$, resp., $\forall A \in \mathbb{V}$ and $\forall BC \in \mathbb{E}$. Note that P and Q should be removed from $\mathbb{P}$ if $|BP| + |CQ| > |BC|$, because in this case, both $\langle AP \rangle$ and $\langle AQ \rangle$ are less than d.

Figure 4 shows a running example of Algorithm 2 with $d = 10$, $k = 1$ on a graph with five vertices. Initially, 14 split points are computed, denoted as $P_0, \ldots, P_{13}$. For example,

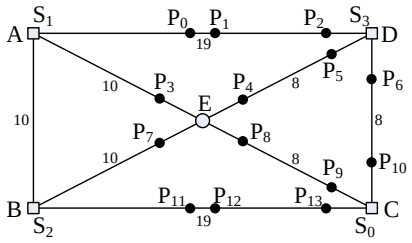

Fig. 4. A running example of the SDG algorithm, in which $|AB| = |AE| = |BE| = 10$, $|CD| = |DE| = |CE| = 8$, and $|AD| = |BC| = 19$. The locations of the split points are determined by $|P_2D| = |P_3E| = |P_4E| = |P_5D| = |P_6D| = |P_7E| = |P_8E| = |P_9C| = |P_{10}C| = |P_{13}C| = 2$, $|P_0A| = |P_{11}B| = |P_{12}C| = |P_1D| = 9$.

P_1 is computed by $\langle P_1A\rangle = 10$, and P_2, P_5, P_6, P_9, P_{10}, P_{13} are computed by $\langle P_2E\rangle = \langle P_5E\rangle = \langle P_6E\rangle = \langle P_9E\rangle = \langle P_{10}E\rangle = \langle P_{13}E\rangle = 10$. Next, 22 line fragments and the relations between the 5 vertices and the 22 fragments are computed. For example, A covers AB, AP_3, P_3E, AP_0, and P_0P_1. Then the greedy algorithm selects C, A, B, and D in order. The algorithm returns $\mathbb{S} = \{S_i | 0 \le i \le 3\}$ in which S_i $(0 \le i \le 3)$ locates at C, A, B, and D, resp.

Theorem 3. *The SDG algorithm returns an answer to the Sensor Deployment problem within an approximation ratio of* $2k\left[\ln\left(2|\mathbb{V}|+1\right)+\ln\left(|\mathbb{E}|\right)+2\right]$.

Proof: Consider $\mathbb{G}^*(\mathbb{V}^*, \mathbb{E}^*)$ with $|AB| \le 2d$, $\forall AB \in \mathbb{E}^*$, and denote the answer computed by Algorithm 2 and an optimal answer as $\mathbb{S}^*_{app}$ and $\mathbb{S}^*_{opt}$, resp. Then we replace each sensor in $\mathbb{S}^*_{opt}$ at $(AB : x)$ $(x > 0)$ by two sensors at A and B, resp., $\forall AB \in \mathbb{E}^*$. The derived set of vertices, denoted as $\mathbb{S}'$, satisfies $|\mathbb{S}'| \le 2|\mathbb{S}^*_{opt}|$. Let $\mathbb{S}_{cov}$ be an optimal answer to the set cover problem, and $|\mathbb{S}'| \ge |\mathbb{S}_{cov}|$ since $\mathbb{S}'$ is an answer to the set cover problem. Using greedy algorithms, Algorithm 2 can compute $\mathbb{S}^*_{alg}$ with $|\mathbb{S}^*_{alg}| \le k\left[\ln|\mathbb{L}|+1\right]|\mathbb{S}_{cov}|$. Therefore, $|\mathbb{S}^*_{alg}| \le k\left[\ln|\mathbb{L}|+1\right]|\mathbb{S}'| \le 2k\left[\ln|\mathbb{L}|+1\right]|\mathbb{S}^*_{opt}|$.

For arbitrary graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$ in which $\mathbb{E}' = \{AB | AB \in \mathbb{E}, |AB| > 2d\}$, denote the answer computed by the algorithm and an optimal answer as $\mathbb{S}_{app}$ and $\mathbb{S}_{opt}$, resp. In any optimal answer, at least $k\lfloor|AB|/(2d)\rfloor$ sensors must be deployed on AB, $\forall AB \in \mathbb{E}'$. Let the set of such sensors be $\mathbb{S}_{ext}$, and $|\mathbb{S}_{opt}| \ge |\mathbb{S}_{ext}|$. According to the algorithm, $\mathbb{S}_{alg} = \mathbb{S}_{fix} \cup \mathbb{S}_{ext}$, where $\mathbb{S}_{fix}$ is computed by the algorithm on the converted graph $\mathbb{G}^*(\mathbb{V}^*, \mathbb{E}^*)$ from $\mathbb{G}$, in which $|AB| \le 2d$, $\forall AB \in \mathbb{E}^*$. Let an optimal answer on $\mathbb{G}^*$ be $\mathbb{S}^*_{opt}$, $|\mathbb{S}^*_{opt}| \le |\mathbb{S}_{opt}|$, and $|\mathbb{S}_{alg}| \le 2k\left[\ln|\mathbb{L}|+1\right]|\mathbb{S}^*_{opt}| + |\mathbb{S}_{opt}| \le 2k\left[\ln|\mathbb{L}|+2\right]|\mathbb{S}_{opt}|$.

Since there are at most two points having network distance d from each vertex in $\mathbb{V}$ on each edge in $\mathbb{E}$, $|\mathbb{L}| \le (2|\mathbb{V}|+1)|\mathbb{E}|$, hence $|\mathbb{S}_{alg}| \le 2k\left[\ln\left(2|\mathbb{V}|+1\right)+\ln\left(|\mathbb{E}|\right)+2\right]|\mathbb{S}_{opt}|$. ■

VI. Sink Deployment and Routing Problem

Suppose the user have m sinks to be deployed on the intersections of the network structure, and at least one sink must be deployed on one of the two ends of each edge for data collection, i.e., m sinks form a vertex cover of the network structure. With such constraint, the sink deployment and routing problem is formulated as follows.

Problem 2 (Sink Deployment and Routing)**.** *Given a network structure* $\mathbb{G}(\mathbb{V}, \mathbb{E})$ *with the deployed sensor set* $\mathbb{S}$*, and a user-assigned integer* m*, the goal is to find a vertex cover* $\mathbb{H} \subseteq \mathbb{V}$ *of size* m*,* $\forall P \in \mathbb{H}$*, a sink is deployed on* P*, and a routing path from each* $S_i \in \mathbb{S}$ *to* $H(S_i) \in \mathbb{H}$*, denoted as* $Path(S_i, H(S_i)) = S_i^0 S_i^1 \ldots S_i^{p_i}$ *where* $S_i^0 = S_i$*,* $S_i^{p_i} = H(S_i)$*,* $S_i^j \in \mathbb{S}\ \forall 0 \le j < p_i$*, and all the points in a path are on the same edge, such that the network communication cost,* $Cost = \sum_{i=0}^{|\mathbb{S}|-1}\sum_{j=0}^{p_i-1} c(S_i^j S_i^{j+1})$*, is minimized.*

The problem is NP-Hard if the communication cost is considered as infinite when the m sinks cannot be deployed as a vertex cover, because in this case, the problem is equivalent to the VC problem, i.e., the cost is less than infinite *iff* a vertex cover of size m exists. In Section VII, we provide the lower bounds of m so that the proposed algorithms can always derive valid vertex covers in polynomial time.

We provide a simple algorithm to compute optimal routing paths for the sensors on a single edge, which is the basis to solve the problem on trees and graphs. The key observation is that each optimal path $Path(S_i, H(S_i))$ is a shortest path from sensor S_i to sink $H(S_i)$, where the weight of $Path(S_i, H(S_i))$ is the total communication cost for sending a unit data from S_i to $H(S_i)$ along it. Since $\mathbb{H}$ is a vertex cover, $\forall AB \in \mathbb{E}$, there may be one sink at A or B, or two sinks at A and B, and the related optimal costs are referred as $cost(A\overline{B})$, $cost(\overline{A}B)$, and $cost(AB)$, resp. The above optimal costs and related optimal paths then can be computed by shortest-path algorithms, such as Dijkstra's algorithm. Since no cross-edge transmission exists, the optimal paths on a single edge are optimal on trees or graphs, and we omit the computation for the routing paths in the following algorithms for simplicity.

VII. Algorithms for Sink Deployment and Routing Problem

In this section, we propose an algorithm named SDRT (Sink Deployment and Routing on Trees) to derive an optimal sink deployment and routing strategy for tree-structured environment, and a heuristic algorithm called SDRG (Sink Deployment and Routing on Graphs) for the problem on general graph-structured environment.

A. An Optimal Algorithm for Tree Topologies

Given a tree topology, a natural question is: how many sinks are necessary to construct a vertex cover, which can be answered by applying a greedy algorithm. We refer this lower bound to λ, and suppose m lies in the range of $[\lambda, |V|]$.

The optimal vertex cover $\mathbb{H}$ $(|\mathbb{H}| = m)$ and the corresponding $Cost$ can be computed by a dynamic programming algorithm. Consider a vertex P with its children $\{C_i | 0 \le i < p\}$ in the network, and a subtree rooted at P consists of the edges $\{PC_j | 0 \le j \le i\}$ and the subtrees $\{Tree(C_j) | 0 \le j \le i\}$. Suppose there are x sinks deployed on $\cup_{j=0}^{i} Tree(C_j)$, and let $g(P, i, x)$ and $g(\overline{P}, i, x)$ be the minimum cost for data transmission on the subtree, where P is selected to deploy a sink or not, resp. The minimum costs can be computed by

$$\begin{aligned} g(P,i,x) = \min\{&cost(PC_i) + g(C_i, f(C_i), y-1) \\ &+g(P, i-1, x-y), cost(P\overline{C_i}) + g(\overline{C_i}, f(C_i), y) \\ &+g(P, i-1, x-y)\}, 0 \le y \le x \end{aligned} \quad (1)$$

Algorithm 3 The SDRT Algorithm

INPUT: A tree rooted at R_r, $\mathbb{S}$, and m.
OUTPUT: A vertex cover $\mathbb{H}$ of size m, and $Cost$.

1: **for** $\forall AB \in \mathbb{E}$ **do**
2: compute $cost(AB)$, $cost(A\overline{B})$ and $cost(\overline{A}B)$;
3: MinCost_on_Tree(R_r,m);
4: **if** $g(R_r, f(R_r), m-1) < g(\overline{R_r}, f(R_r), m)$ **then**
5: $Cost = g(R_r, f(R_r), m-1)$;
6: Sink_on_Tree(R_r,m);
7: **else**
8: $Cost = g(\overline{R_r}, f(R_r), m)$;
9: Sink_on_Tree($\overline{R_r}$,m);
10: **return** $\mathbb{H}$ and $Cost$;
procedure MinCost_on_Tree(P,x)
11: **for** each child C_i of P **do**
12: MinCost_on_Tree(C_i,x);
13: **for** $z = 0$ to x **do**
14: initialize $g(P, i, z) = +\infty$ and $g(\overline{P}, i, z) = +\infty$;
15: **for** $y = 0$ to z **do**
16: update $g(P, i, z)$ according to Equation (1);
17: update $g(\overline{P}, i, z)$ according to Equation (2);
18: let $g(P, i-1, z-y_1)$ and (T, y_1) point to $g(P, i, z)$;
19: let $g(P, i-1, z-y_2)$ and (T, y_2) point to $g(\overline{P}, i, z)$;
procedure Sink_on_Tree(T,x)
20: **if** $T = P$ **then**
21: add P into $\mathbb{H}$;
22: **for** each child C_i of P in reverse order **do**
23: obtain $g(T, i-1, x-y)$ and (T', y) that point to $g(T, i, x)$;
24: Sink_on_Tree(T',y);
25: $x = x - y$;

$$g(\overline{P}, i, x) = \min\{cost(\overline{P}C_i) + g(C_i, f(C_i), y-1) + g(\overline{P}, i-1, x-y)\}, 0 \le y \le x \quad (2)$$

where $f(C_i)$ refers to the maximum index of C_i's children.

To see the correctness of the two recursion functions, consider any optimal deployment with cost $g(P, i, x)$ and suppose $g(P, i-1, z)$, $g(C_i, f(C_i), z)$ and $g(\overline{C_i}, f(C_i), z)$ $(0 \le z \le x)$ are already computed as the optimal costs for the corresponding cases. If there are y sinks in $Tree(C_i)$, $x - y$ sinks must be deployed in the previous i subtrees of P, and the cost for $x - y$ sinks in these subtrees must be $g(P, i-1, x-y)$. Furthermore, if C_i is selected as a sink, $y - 1$ sinks must be deployed in $Tree(C_i)$ with cost $g(C_i, f(C_i), y-1)$, otherwise y sinks must be deployed in $Tree(C_i)$ with cost $g(\overline{C_i}, f(C_i), y)$. Hence Equation (1) enumerates all the possible cases that lead to an optimal $g(P, i, z)$, and finally selects the minimum cost in the cases. Equation (2) obtains $g(\overline{P}, i, x)$ in a similar way, however, C_i is always selected as a sink because P is not a sink and PC_i must be covered by P or C_i.

According to Equation (1) and Equation (2), the algorithm must compute $g(C_i, f(C_i), x)$ and $g(\overline{C_i}, f(C_i), x)$ for all P's children before computing $g(P, i, x)$ and $g(\overline{P}, i, x)$ $\forall 0 \le i \le f(P)$. Furthermore, $g(P, i, z)$ and $g(\overline{P}, i, z)$ for $0 \le z \le x$ must be computed before $g(P, i, x)$ or $g(\overline{P}, i, x)$. Therefore, three nested loops on i, z, and y, resp., are required in the dynamic programming.

The pseudo code of the dynamic programming algorithm is illustrated in Algorithm 3, in which line 1-2 compute the optimal costs for each edge. Line 3 evokes the MinCost_on_Tree procedure to compute the intermediate costs, and line 4-9 derive $Cost$ which stems from $g(R_r, f(R_r), m-1)$ or $g(\overline{R_r}, f(R_r), m)$, and evoke the Sink_on_Tree procedure to obtain $\mathbb{H}$. In order to determine whether C_i is in $\mathbb{H}$, the algorithm needs to record the status of C_i (denoted as T, $T = C_i$ or $\overline{C_i}$) and the number of sinks (denoted as y_1 or y_2) deployed on $Tree(C_i)$ that lead to an optimal answer to $Tree(P)$ having z sinks, as shown in line 18-19. Then the Sink_on_Tree procedure retrieves the status of each vertex in a top-down manner, and update $\mathbb{H}$ accordingly.

Algorithm 4 The SDRG Algorithm

INPUT: A graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$, $\mathbb{S}$, and m.
OUTPUT: a vertex cover $\mathbb{H}$ of size m, and $Cost$.

1: $\mathbb{H} = \emptyset$ and $Cost = 0$;
2: **for** $\forall AB \in \mathbb{E}$ **do**
3: compute $cost(AB)$, $cost(A\overline{B})$ and $cost(\overline{A}B)$;
4: **while** $\exists AB \in \mathbb{E}$, $A, B \notin \mathbb{H}$ **do**
5: **for** all $AB \in \mathbb{E}$, $A, B \notin \mathbb{H}$ **do**
6: compute Edge_Selection_Cost(AB);
7: select AB with the minimum Edge_Selection_Cost(AB);
8: $\mathbb{H} = \mathbb{H} \cup \{A, B\}$, $m = m - 2$;
9: **while** $m > 0$ **do**
10: **for** all $A \in \mathbb{V}$, $A \notin \mathbb{H}$ **do**
11: compute Vertex_Selection_Cost(A);
12: select A with the minimum Vertex_Selection_Cost(A);
13: $\mathbb{H} = \mathbb{H} \cup \{A\}$, $m = m - 1$;
14: **for** each edge $AB \in \mathbb{E}$ **do**
15: add $cost(AB)$ or $cost(A\overline{B})$ or $cost(\overline{A}B)$ to $Cost$;
16: **return** $\mathbb{H}$ and $Cost$;
procedure Vertex_Selection_Cost(A)
17: $ret = 0$;
18: **for** each neighbor B of A **do**
19: $ret+ = cost(AB)$ if $B \in \mathbb{H}$, otherwise $ret+ = cost(A\overline{B})$;
20: **return** ret;
procedure Edge_Selection_Cost(AB)
21: ret = Vertex_Selection_Cost(A)+Vertex_Selection_Cost(B);
22: **return** $ret + cost(AB) - cost(A\overline{B}) - cost(\overline{A}B)$;

B. A Heuristic Algorithm for Graph Topologies

It is difficult for the user to provide a sufficient small m for the sink deployment and routing problem on graphs, because if m is too small, the sinks may even be unable to form a vertex cover, and the minimum Vertex Cover problem is NP-Complete. However, $m \ge w/2$, where w is the number of vertices in a maximum matching of the graph. Hence, m is sufficient large for constructing a maximal matching (and also a vertex cover) if it is no less than the number of vertices in a maximum matching. In the following, we assume that $m \in [w/2, n]$, and propose a heuristic algorithm, denoted as the SDRG (Sink Deployment and Routing in Graphs) algorithm.

The SDRG algorithm constructs a maximal matching, and adds the vertices in the matching into $\mathbb{H}$, and then it continues to add a vertex not in the matching into $\mathbb{H}$ until $|\mathbb{H}| = m$. To reduce the communication cost, the algorithm performs the selection of an edge/vertex in a greedy manner. Specifically, in each step of constructing the matching, it selects edge AB from all the candidates with the minimum bring-in cost calculated by $cost(AB) + \sum_{C \in \mathbb{H}} cost(AC) + \sum_{D \notin \mathbb{H}} cost(A\overline{D}) + \sum_{E \in \mathbb{H}} cost(BE) + \sum_{F \notin \mathbb{H}} cost(B\overline{F})$. Similarly, it selects vertex A from all the candidates with the minimum bring-in cost computed by $\sum_{B \in \mathbb{H}} AB + \sum_{C \notin \mathbb{H}} A\overline{C}$.

The pseudo code of the SDRG algorithm is shown in Algorithm 4, which initiates $\mathbb{H}$ and $Cost$, and computes the

communication costs on each edge as Algorithm 3 does. Then in line 4-8, it computes a maximal matching of the graph by greedy selections, and adds the vertices in the matching into $\mathbb{H}$. If $|\mathbb{H}| < m$, it selects $m - |\mathbb{H}|$ vertices and put them into $\mathbb{H}$, as shown in line 9-13. The algorithm then computes $Cost$ according to $\mathbb{H}$, and returns $\mathbb{H}$ and $Cost$.

VIII. Performance Evaluation

We conduct extensive simulation experiments to evaluate the performance of the proposed algorithms. In this section, we first briefly introduce the experiment setup, and then reveal the results in terms of the number of the deployed sensors and the communication cost.

A. Experiment Setup

The topologies of the underlying environment, either trees or graphs, are randomly generated, which are also guaranteed to be connected. For graph generation, a parameter p is given as the probability of any pair of intersections having an edge, which is used to control the density of the edges. The length of each edge is randomly generated within a given range, measured by meters. We consider the instances with the coverage degree $k = 1$ in our experiments for the Sensor Deployment problem, because the answer given by our algorithms to an instance with $k > 1$ is k times the answer to that instance with $k = 1$. In the communication cost model, we assume $\alpha = 1.0$, $\beta = 2.0$, and $\gamma = 0.0$. We run each algorithm for every parameter setting on 10 different topologies with different edge lengths, and show the average/max/min experiment results.

B. Number of Deployed Sensors

The number of the sensors deployed to ensure network coverage is the primary measurement to evaluate the performance of Algorithm 1 and Algorithm 2. For comparison, we also implement a naive algorithm referred to as the baseline, which deploys $\lceil l/(2d) \rceil$ sensors on an edge of length l when given distance threshold d, and the distance between any adjacent sensors is $2d$. The results under variant number of intersections and distance threshold are shown in Figure 5-8.

When $d = 10$, $p = 0.5$ for graph topologies, and the length of the edges is in the range of $[10, 100]$, the required numbers of sensors obtained by the algorithms keep linear growth as the number of intersections increases both in tree topologies (Figure 5) and in graph topologies (Figure 6). This result meets our intuition that the required number of sensors is linear to the scale of the monitored environment. Furthermore, Algorithm 1 outperforms the baseline algorithm by about 15% as $|V|$ increases from 200 to 1000, while Algorithm 2 obtains less sensors than the baseline algorithm by about 30% as $|V|$ raises from 100 to 200.

To further evaluate the performance of Algorithm 1 and Algorithm 2, we fix $|\mathbb{V}| = 500$, let the length of the edges be in $[10, 100]$ and $p = 0.5$, and let the distance threshold d vary from 10 to 50. The result in Figure 7 implies that compared with the naive algorithm, the advantage of Algorithm 1 is more obvious as d increases in tree topologies, since the ratio of the answer returned by Algorithm 1 to that returned by the naive algorithm keeps decreasing from about 85% to about 55%. In graphs, the performance gap between Algorithm 2 and the baseline algorithm is even larger than that between Algorithm 1 and the baseline algorithm in trees. As illustrated in Figure 8, when $d = 50$, the baseline algorithm returns 615.6 number of sensors on average, which is about 12 times lager than the answer provided by Algorithm 2.

C. Communication Cost

In the experiments, the sensor deployment input of the two algorithms are computed by Algorithm 1 and 2, resp. The communication costs in data collection are computed by Algorithm 3 and 4. The results with variant number of sinks and distance threshold are shown in Figure 9-12.

The relation between the communication cost and the number of sinks in tree topologies is shown in Figure 9. We let $n = 50$, $d = 10$, and the length of the edges be in $[10, 100]$. Algorithm 3 is performed on ten generated networks, in each of which at least 21 sinks are required to cover all the edges. From the figure we can see that the optimal communication cost derived by Algorithm 3 decreases as the number of sinks increases from 21 to 50, and the decreasing speed becomes lower with more sinks. In graph topologies with $n = 100$, $d = 10$, $p = 0.01$ and the length of the edges in $[10, 50]$, the communication cost computed by Algorithm 4 also decreases as the number of sink raises. However, the decreasing speed is higher when there are more sinks, as shown in Figure 10. Algorithm 4 runs on ten graphs with $m = 81$ sinks deployed on vertices of their maximal matchings, and it derives a cost no more than 149% of the lower bound with $m = 100$.

Finally, the experiment reveals the relation between the communication cost and the distance threshold d (Figure11-12). In trees with $n = 100$, $d = 10$, and the length of the edges in $[10, 100]$, and let m be the size of a minimum vertex cover, the communication cost derived by Algorithm 1 keeps linear decreasing as d increases, which is also in accordance with the decreasing of the number of sensors in Figure 7. The result for graph topologies is illustrated in Figure 12, where $n = 100$, $d = 10$, the length of edges is in $[10, 100]$, $p = 0.01$, and m is the size of a maximal matching. The communication cost with all the intersections having sinks is also computed and used as a lower bound for comparison, since the communication cost is minimized when a sink is deployed on every intersection. Figure 12 shows that the cost computed by Algorithm 4 does not exceed 51.3% of the lower bound.

IX. Conclusion

In this paper, we investigate sensor deployment problem in network-structure environments. Specifically, we address the problem of sensor deployment to achieve k-coverage of the network structure while minimizing the number of sensors. We also study the joint optimization problem of sink deployment and routing strategies with the goal to minimize the total communication cost of data collection. We show the hardness of the problems and propose polynomial-time algorithms to obtain optimal solutions in the network structure of tree topology as well as approximation algorithms for the general network structure. The performance of proposed algorithms

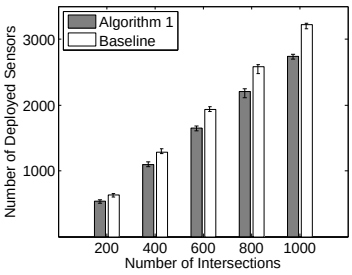

Fig. 5. Number of deployed sensors with different number of intersections in tree topologies.

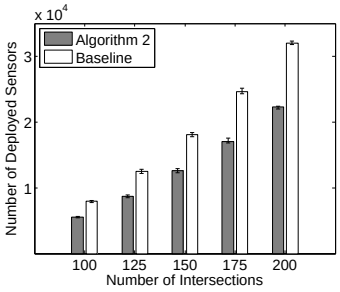

Fig. 6. Number of deployed sensors with different number of intersections in graph topologies.

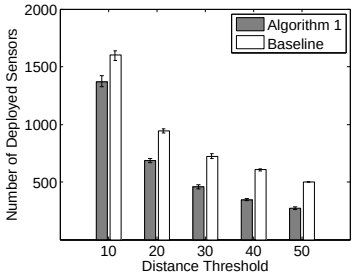

Fig. 7. Number of deployed sensors with different distance threshold in tree topologies.

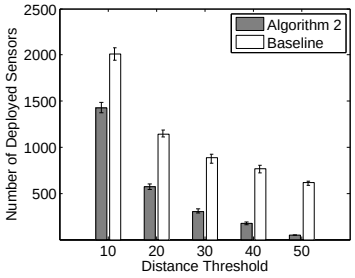

Fig. 8. Number of deployed sensors with different distance threshold in graph topologies.

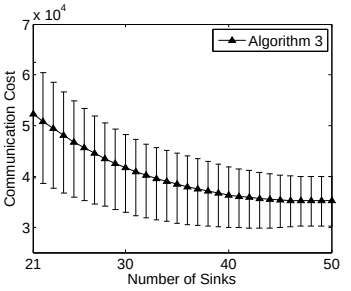

Fig. 9. Communication cost with different number of sinks in tree topologies.

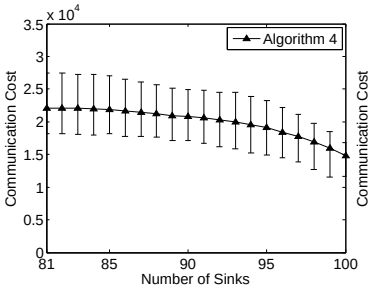

Fig. 10. Communication cost with different number of sinks in graph topologies.

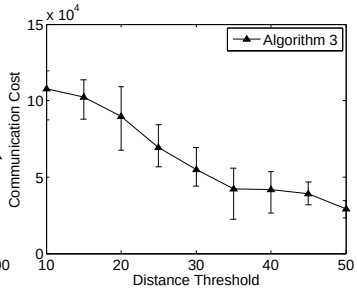

Fig. 11. Communication cost with different distance threshold in tree topologies.

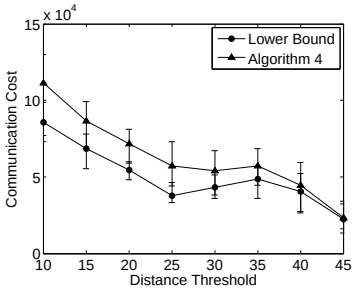

Fig. 12. Communication cost with different distance threshold in graph topologies.

are analyzed. Extensive simulations show that the number of sensors can be significantly reduced, and the communication cost does not exceed the lower bound too much.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants NSF-CSR 1025649,OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

REFERENCES

[1] S. Coleri, S. Y. Cheung, and P. Varaiya, "Sensor Networks for Monitoring Traffic," in *Allerton Conference on Communication, Control and Computing*, 2004.

[2] B. O'Flyrm, R. Martinez, J. Cleary, C. Slater, F. Regan, D. Diamond, and H. Murphy, "SmartCoast: A Wireless Sensor Network for Water Quality Monitoring," in *Proc. of IEEE LCN*, 2007, pp. 815–816.

[3] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "PIPENET: A Wireless Sensor Network for Pipeline Monitoring," in *Proc. of ACM/IEEE IPSN*, 2007, pp. 264–273.

[4] Y. Guo, F. Kong, D. Zhu, A. c. Tosun, and Q. Deng, "Sensor Placement for Lifetime Maximization in Monitoring Oil Pipelines," in *Proc. of ACM/IEEE ICCPS*, 2010, pp. 61–68.

[5] M. Younis and K. Akkaya, "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621–655, 2008.

[6] T. Andersen and S. Tirthapura, "Wireless Sensor Deployment for 3D Coverage with Constraints," in *Proc. of IEEE INSS*, 2009, pp. 78–81.

[7] Y.-C. Wang and Y.-C. Tseng, "Distributed Deployment Schemes for Mobile Wireless Sensor Networks to Ensure Multilevel Coverage," *IEEE TPDS*, vol. 19, no. 9, pp. 1280–1294, 2008.

[8] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces," in *Proc. of INFOCOM*, 2003, pp. 1293–1303.

[9] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal Sensor Placements: Maximizing Information while Minimizing Communication Cost," in *Proc. of ACM/IEEE IPSN*, 2006, pp. 2–10.

[10] X. Zhang and S. Wicker, "How to Distribute Sensors in a Random Field?" in *Proc. of ACM/IEEE IPSN*, 2004, pp. 243–250.

[11] S. Dhillon and K. Chakrabarty, "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks," in *Proc. of IEEE WCNC*, 2003, pp. 1609–1614.

[12] D. Ganesan, R. Cristescu, and B. Beferull-Lozano, "Power-efficient Sensor Placement and Transmission Structure for Data Gathering under Distortion Constraints," in *Proc. of IPSN*, 2004, pp. 142–150.

[13] S. Toumpis and G. Gupta, "Optimal Placement of Nodes in Large Sensor Networks under a General Physical Layer Model," in *Proc. of IEEE SECON*, 2005, pp. 275–283.

[14] X. Liu and P. Mohapatra, "On the Deployment of Wireless Data Backhaul Networks," *IEEE TWC*, vol. 6, no. 4, pp. 1426–1435, 2007.

[15] P. Cheng, C.-N. Chuah, and X. Liu, "Energy-aware Node Placement in Wireless Sensor Networks," in *Proc. of IEEE GLOBECOM*, 2004, pp. 3210–3214.

[16] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a Mostly Sleeping Sensor Network," in *Proc. of ACM MOBICOM*, 2004, pp. 144–158.

[17] M. Hefeeda and M. Bagheri, "Randomized k-coverage Algorithms for Dense Sensor Networks," in *Proc. of INFOCOM*, 2007, pp. 2376–2380.

[18] W. Youssef and M. Younis, "Intelligent Gateways Placement for Reduced Data Latency in Wireless Sensor Networks," in *Proc. of IEEE ICC*, 2007, pp. 3805–3810.

[19] J. Pan, Y. Hou, L. Cai, Y. Shi, and S. Shen, "Locating Base-stations for Video Sensor Networks," in *Proc. of IEEE VTC*, 2003, pp. 3000–3004.

[20] Y. Shi, Y. T. Hou, and A. Efrat, "Algorithm Design for Base Station Placement Problems in Sensor Networks," in *Proc. of QShine*, 2006.

[21] J. Park and S. Sahni, "An Online Heuristic for Maximum Lifetime Routing in Wireless Sensor Networks," *IEEE TOC*, vol. 55, no. 8, pp. 1048–1056, 2006.

[22] J.-H. Chang and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks," *ACM/IEEE TON*, vol. 12, no. 4, pp. 609–619, 2004.

[23] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Efficient Algorithms for Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, 2003.

[24] S. Xiong, J. Li, and L. Yu, "Maximize the Lifetime of a Data-gathering Wireless Sensor Network," in *Proc. of SECON*, 2009, pp. 306–314.

[25] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[26] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.