IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 26, 2018, accepted January 14, 2019, date of publication January 22, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894117

Minimum-Cost Heterogeneous Node Placement in Wireless Sensor Networks

YAHUI SUN[1] AND SAMAN HALGAMUGE[2], (Fellow, IEEE)

[1]Research School of Engineering, The Australian National University, Canberra, ACT 2601, Australia
[2]Department of Mechanical Engineering, School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia

Corresponding author: Yahui Sun (yahui.sun@anu.edu.au)

This work was supported by the Australian Research Council under Grant LP140100670.

ABSTRACT The operational cost-effectiveness of wireless sensor networks depends on the placement of heterogeneous nodes: base stations, sensor nodes, and relay nodes. To achieve the global optimality of minimum-cost heterogeneous node placement, we find the locations of base stations, sensor nodes, and relay nodes, simultaneously. The objective is to minimize the sum of node production and placement costs and transmission outage probabilities in the routing tree. First, we formulate this minimum-cost heterogeneous node placement problem as a new NP-hard Steiner tree problem. Then, to solve this problem for both small and large instances, we propose an exponential-time exact algorithm, a polynomial-time heuristic algorithm, and several post-processing algorithms. On a personal computer, our exact algorithm is sufficiently fast to produce optimal solutions for small instances with dozens of vertices, while our heuristic and post-processing algorithms can, respectively, produce and improve suboptimal solutions for large instances with 100 000 vertices and 1 000 000 edges within 6 s. We also compare the heuristic and optimal solutions for small instances with dozens of vertices and show that the ratios between the respective solution costs are generally below 1.35, and our post-processing algorithms can improve this number to 1.1. This indicates that our heuristic and post-processing algorithms are likely to produce near-optimal solutions in practice and are useful for minimum-cost heterogeneous node placement when computational resources are scarce.

INDEX TERMS Steiner tree problem, wireless sensor network, Internet of Things.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are eyes and ears of the emerging Internet of Things (IoTs). There are typically three types of devices in them: base stations, sensor nodes, and relay nodes. Sensor nodes are used to sense targets, while relay nodes help them transmit data to base stations. Given that the locations of devices determine the network topology and operation, node placement is crucial in minimizing the cost of WSNs.

Many node placement approaches and techniques have been developed in the last decade. They generally focus on placing heterogeneous nodes separately to increase the network coverage, connectivity, lifetime, cost-effectiveness, and/or boost the data fidelity (e.g. the base station placement techniques in [1] and [2]; the sensor node placement techniques in [3] and [4]; the relay node placement techniques in [5]–[8]; and the surveys in [9] and [10]). However, in practice, the placements of heterogeneous nodes entangle with each other, and it may be preferable to conduct them simultaneously. For instance, in the separate node placement approach in Figure 1, four sensor nodes are first placed to cover the targets; a base station is then placed in the middle of them to minimize the average transmission distance; this base station is not within the transmission range of any sensor node, thus two relay nodes are required to achieve the network connectivity; while in the simultaneous node placement approach in Figure 1, all the nodes are placed simultaneously; the base station is within the transmission ranges of two left sensor nodes, thus only one relay node is required to perform the same task. Hence, simultaneous node placement has intuitive advantages over separate node placement in minimizing the cost of WSNs. Nonetheless, hardly any work has been done to explore simultaneous node placement to date. In this paper, we address this issue by finding the locations of base stations, sensor nodes, and relay nodes simultaneously for minimum-cost heterogeneous node placement.

On the other hand, Steiner tree problems are well-known NP-hard problems of designing minimum-cost networks. Thus, Steiner tree problems and algorithms may inspire

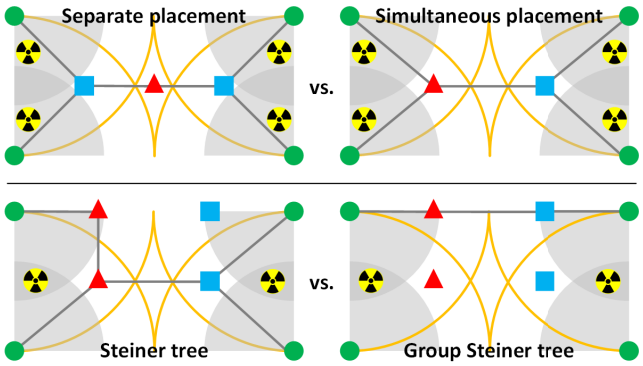

FIGURE 1. The comparison of separate and simultaneous node placement approaches. The trefoil symbols are targets; the red triangles are base stations; the green dots are sensor nodes; the blue boxes are relay nodes; the gray lines are transmission routes; the gray shadows and orange circles are respectively sensing and transmission ranges of sensor nodes.

minimum-cost heterogeneous node placement. There are two classes of Steiner tree problems: Steiner tree problems in geometric spaces (e.g. [11]) and Steiner tree problems in graphs (e.g. [12]). The vertices in Steiner tree problems in geometric spaces can be placed anywhere in the given geometric space, while those in Steiner tree problems in graphs can only be placed at pre-determined candidate locations. Given that node placement regions are usually constrained in reality [13], we focus on Steiner tree problems in graphs in this paper.

The classical Steiner tree problem in graphs [14] is about finding the minimum-cost tree in a graph to connect some compulsory vertices together. If we use compulsory vertices to represent devices that must be connected; use other vertices to represent candidate devices that may not be connected; use edges to represent transmission routes between devices; use node weights to represent node production and placement costs, then we can minimize the sum of node production and placement costs by finding minimum Steiner trees. Furthermore, if we use edge costs to represent outage probabilities of transmission routes, then we can minimize the sum of transmission outage probabilities in the routing tree, and thus enhance the Quality of Service (QoS) [5].

The connectivity and coverage requirements are two basic requirements in designing WSNs. The connectivity requirement, which is about connecting all the devices together, can be naturally met through the Steiner tree approach mentioned above, as Steiner trees are connected. However, the coverage requirement, which is about connecting enough sensor nodes to base stations to cover all the targets, is hard to be met. An easy way to meet this requirement is to make all the candidate base stations and sensor nodes compulsory. This suits separate relay node placement where base stations and sensor nodes are pre-deployed (e.g. [5]–[7]), while it does not suit simultaneous node placement, as not all the candidate sensor nodes and base stations are required. For example, in the Steiner tree in Figure 1, some redundant base stations and sensor nodes are connected to meet the coverage requirement.

We observe that the group Steiner tree problem [15], which is a more general version of the classical Steiner tree problem in graphs, can be exploited to overcome this weakness. It is about finding the minimum-cost tree in a graph to connect at least one vertex in each group of vertices. If we add a group that contains all the candidate base stations; and associate each target with a group that contains all the adjacent candidate sensor nodes, then we can meet the coverage requirement without deploying redundant base stations and sensor nodes. For example, in the group Steiner tree in Figure 1, we add three groups that respectively contain two left sensor nodes, two right sensor nodes and two base stations, then the group Steiner tree connects a single sensor node to cover each target and a single base station to contact users.

Nevertheless, there is no node weight in the existing group Steiner tree problems to represent node production and placement costs (e.g. [16], [17]). Thus, new node-weighted group Steiner tree problems are required for our application, and new Steiner tree algorithms are also required to solve them. Like other NP-hard problems, different Steiner tree algorithms are used for small and large instances: exact algorithms are often used to produce optimal solutions for small instances (e.g. [18]), while heuristic and post-processing algorithms are often combined together to produce fast suboptimal solutions for large instances (e.g. [19]), where heuristic algorithms produce fast suboptimal solutions and post-processing algorithms improve these solutions. In this paper, we will develop all these types of algorithms for our application.

In summary, our major contributions are listed as follows:

- we formulate the minimum-cost heterogeneous node placement problem as the new Node-Weighted Full Group Steiner Tree Problem (NWFGSTP). To our knowledge, this is the first formulation that places base stations, sensor nodes and relay nodes simultaneously.
- we propose an exponential-time exact algorithm to produce optimal solutions to NWFGSTP for small instances.
- we propose a polynomial-time heuristic algorithm to produce heuristic solutions to NWFGSTP for large instances.
- we propose several polynomial-time post-processing algorithms to improve suboptimal solutions to NWFGSTP for large instances.

II. PROBLEM FORMULATION AND DISCUSSION

In this section, we formulate the minimum-cost heterogeneous node placement problem as NWFGSTP. We consider two types of WSNs: single-tiered and two-tiered. Both sensor and relay nodes can relay data in single-tiered WSNs, while only relay nodes can do this in two-tiered ones. As a result, there is no transmission route between sensor nodes in two-tiered WSNs. Suppose that all the devices and targets are stationary. Let B, S, R, Λ be the sets of base stations, sensor nodes, relay nodes, and targets respectively. Let $d(i, j)$ be the euclidean distance between devices i, j, and r_i, r_j be the

transmission ranges of devices i, j respectively. We define the Single-tiered Communication Graph (SCG) and the Two-tiered Communication Graph (TCG) as follows.

Definition 1 (The Single-Tiered Communication Graph): The Single-tiered Communication Graph SCG(V, E, w, c) is an undirected graph, where V is the set of vertices such that $V = B \cup S \cup R$; E is the set of edges such that edge $(i, j) \in E$ if $d(i, j) \leq \min\{r_i, r_j\}$; w is a function which maps each vertex $i \in V$ to a value $w(i)$ that equals the sum of its production and placement costs; and c is a function which maps each edge $e \in E$ to a value $c(e)$ that equals its outage probability.

Definition 2 (The Two-Tiered Communication Graph): The Two-tiered Communication Graph TCG(V, E, w, c) is an undirected graph, where V is the set of vertices such that $V = B \cup S \cup R$; E is the set of edges such that edge $(i, j) \in E$ if $|\{i, j\} \cap S| < 2$ and $d(i, j) \leq \min\{r_i, r_j\}$; w is a function which maps each vertex $i \in V$ to a value $w(i)$ that equals the sum of its production and placement costs; and c is a function which maps each edge $e \in E$ to a value $c(e)$ that equals its outage probability.

The production and placement costs of devices and the outage probabilities of transmission routes are needed to construct SCGs and TCGs (e.g. Figure 2). Since we assume that all the devices are stationary, it may be trivial to acquire the production and placement costs of devices in practice. For example, we can consider the production costs of devices as their selling prices on the market, and their placement costs as the prices of hiring people to install them at certain fixed locations. On the other hand, it may not be easy to acquire the outage probabilities of transmission routes. Recently, Bagaa *et al.* [5] developed a practical model to do this, and they minimized the outage probabilities of transmission routes in the routing tree for relay node placement. In this paper, we mainly focus on the development of node placement algorithms, and assume that the production and placement costs of devices and the outage probabilities of transmission routes are already acquired.

Since minimizing the production and placement costs of nodes allows us to design cheap WSNs; and minimizing the outage probabilities of transmission routes in the routing tree allows us to enhance the QoS [5], we define the minimum-cost relay node placement problems as follows.

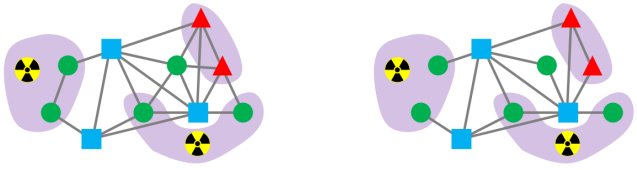

FIGURE 2. Examples of Single-tiered and Two-tiered Communication Graphs. The trefoil symbols are targets; the red triangles are base stations; the green dots are sensor nodes; the blue boxes are relay nodes; and the gray lines are transmission routes. Each target is associated with a purple shadow grouping its adjacent sensor nodes. All the base stations are also grouped in a purple shadow.

Definition 3 (The Single-Tiered Minimum-Cost Heterogeneous Node Placement Problem): Given an SCG(V, E, w, c), the single-tiered minimum-cost heterogeneous node placement problem is to deploy some devices $V' \subseteq V$ in such a way that: 1) each target is covered by a sensor node in V'; 2) there is at least one base station in V'; 3) there is a route between each sensor node and a base station in a tree SCG$'(V', E', w, c)$, $E' \subseteq E$; and 4) $\sum_{e \in E'} c(e) + \sum_{v \in V'} w(v)$ is minimized.

Definition 4 (The Two-Tiered Minimum-Cost Heterogeneous Node Placement Problem): Given a TCG(V, E, w, c), the two-tiered minimum-cost heterogeneous node placement problem is to deploy some devices $V' \subseteq V$ in such a way that: 1) each target is covered by a sensor node in V'; 2) there is at least one base station in V'; 3) there is a route between each sensor node and a base station in a tree TCG$'(V', E', w, c)$, $E' \subseteq E$, and no sensor node is in the middle of this route; and 4) $\sum_{e \in E'} c(e) + \sum_{v \in V'} w(v)$ is minimized.

To solve the minimum-cost relay node placement problems above, we define NWFGSTP as follows.

Definition 5 (The Node-Weighted Full Group Steiner Tree Problem): Let $G(V, E, \Gamma, L, w, c)$ be a connected undirected graph, where V is the set of vertices, E is the set of edges, Γ is a collection of subsets of V called groups, L is a subset of V called leaf vertices, w is a function which maps each vertex in V to a nonpositive value called node weight, and c is a function which maps each edge in E to a nonnegative value called edge cost. The purpose is to find a connected subgraph $G'(V', E', w, c)$, $V' \subseteq V$, $E' \subseteq E$ with the minimum net-cost $c(G') = \sum_{e \in E'} c(e) - \sum_{v \in V'} w(v)$, and for each group $g \in \Gamma$, $g \cap V' \neq \emptyset$; for each leaf vertex $i \in L$, if $i \in V'$, then i is a leaf of G'.

The optimal solution to NWFGSTP is called Full Group Steiner Minimum Tree (FGSMT). NWFGSTP is a more general version of the classical group Steiner tree problem. Since the classical group Steiner tree problem is NP-hard [15], NWFGSTP is also NP-hard. The reason why node weights are nonpositive in NWFGSTP is that, we aim to minimize the node production and placement costs, while positive node weights are often maximized in well-known Steiner tree problems (e.g. [20], [21]). If we associate each target with a group that contains all the adjacent candidate sensor nodes; and add a group that contains all the candidate base stations, then we can solve the single-tiered minimum-cost heterogeneous node placement problem by solving NWFGSTP in SCGs. Given that sensor nodes in two-tiered WSNs cannot relay data and thus are leaves in the routing tree, if we further use leaf vertices to represent sensor nodes, i.e., $L = S$, then we can solve the two-tiered minimum-cost heterogeneous node placement problem by solving NWFGSTP in TCGs. By doing this, the sum of node production and placement costs and transmission outage probabilities in the routing tree can be minimized. Our approach has advantages over the existing ones in that we can find the locations of base stations, sensor nodes, and relay nodes simultaneously to achieve

the global optimality of minimum-cost heterogeneous node placement.

III. SOME THEORETICAL ANALYSES ON NWFGSTP

In this section, we conduct some theoretical analyses on NWFGSTP. First, we prove that the leaf constraint of sensor nodes in two-tiered WSNs, i.e., the constraint of L, increases the node placement cost. Then, we analyze the feasibility of two-tired WSNs by analyzing the solvability of NWFGSTP.

Single-tiered WSNs require less relay nodes to achieve the network connectivity than two-tiered ones, as their sensor nodes can relay data. Consequently, it may be preferable to apply the single-tiered routing topology when sensor nodes are energy-sufficient and their transmission outage probabilities are small. We prove this as follows. First, we remove the constraint of L from NWFGSTP to propose a new problem: the Node-Weighted Group Steiner Tree Problem (NWGSTP), for which the optimal solution is called Group Steiner Minimum Tree (GSMT). Clearly, GSMT and FGSMT ($L \neq \emptyset$) respectively correspond to the minimum-cost single-tiered and two-tiered WSNs. We propose the theorem below to compare their costs.

Theorem 1: If there are feasible solutions to NWFGSTP in graph $G(V, E, \Gamma, L, w, c)$, then $c(\Theta_{GSMT}) \leq c(\Theta_{FGSMT})$, where Θ_{GSMT} and Θ_{FGSMT} are respectively the GSMT and FGSMT in G.

Since a feasible solution to NWFGSTP is also a feasible solution to NWGSTP, this theorem can be easily proven. It indicates that the leaf constraint of sensor nodes in two-tiered WSNs increases the node placement cost.

It may still be preferable to deploy two-tiered WSNs when sensor nodes are not robust or powerful enough to relay data. There are feasible two-tired WSNs only when NWFGSTP ($L \neq \emptyset$) is solvable in TCGs. Here, we check the existence of feasible two-tired WSNs by analyzing the solvability of NWFGSTP ($L \neq \emptyset$). First, we show the solvability of NWFGSTP ($L \neq \emptyset$) in general graphs as follows.

Theorem 2: There is a feasible solution to NWFGSTP in graph $G(V, E, \Gamma, L, w, c)$ if and only if at least one of the three following conditions is met: 1) there is a vertex i such that $i \in g \mid \forall\, g \in \Gamma$; 2) there is an edge (i, j) such that i$\tilde{\text{o}}$r $j \in g \mid \forall\, g \in \Gamma$; 3) there is a vertex $i \in V \setminus L$ such that, for every group $g \in \Gamma$, i is connected with a vertex $j \in g$ in graph $G \setminus (L \setminus j)$.

Proof: If the first condition is met, vertex i is a feasible solution; if the second condition is met, edge (i, j) is a feasible solution; if the third condition is met, $\sum_{g\in\Gamma} P_{ij}$ is a feasible solution, where P_{ij} is a path between i, j such that vertex $k \notin L \mid \forall\, k \in P_{ij}, k \neq j$. Therefore, if at least one of the three conditions is met, there is a feasible solution to NWFGSTP in G. On the other hand, suppose that there is a feasible solution in G, but none of the three conditions is met. In this case, this feasible solution must contain at least three vertices, which means that at least one of them is not a leaf. Without loss of generality, assume that vertex i is in this feasible solution; $i \in V \setminus L$; and it is not a leaf. Then, there is a vertex $j \in g \in \Gamma$ such that a leaf vertex $k \in L$ lies between i and j in this feasible solution, which is not possible. Thus, if there is a feasible solution to NWFGSTP in G, at least one of these three conditions is met. Hence, this theorem holds. □

Since the time complexity to check vertex $i \in V \setminus L$ is $O(|V|)$; to check graph $G \setminus (L \setminus j)$ is $O(|V|)$; to check graph connectivity is $O(|V| + |E|)$ (Tarjan's algorithm [22]), the time complexity to check the solvability of NWFGSTP ($L \neq \emptyset$) using Theorem 2 is $O(|V|^3 + |E||V|^2)$. Therefore, it is tremendously slow to do this in large graphs. Instead, we observe that there is an easier way to check the solvability of NWFGSTP ($L \neq \emptyset$) in TCGs for our application.

Theorem 3: Given a connected Two-tiered Communication Graph $G(V, E, \Gamma, L, w, c)$ where $L = S$. If $G \setminus L$ is connected, then there is a feasible solution to NWFGSTP in G.

Since every sensor node $i \in S = L$ directly connects $G \setminus L$ in TCGs, Theorem 3 can be easily proven. The time complexity to check the solvability of NWFGSTP ($L \neq \emptyset$) using Theorem 3 is $O(|V| + |E|)$. Therefore, it is faster to implement it for our application. In our later computational trials, we will use Theorem 3 to guarantee that there are feasible two-tired WSNs in our generated TCGs (there is no need to do this in SCGs).

IV. THE PROPOSED POST-PROCESSING ALGORITHMS

It may only be possible to produce suboptimal solutions to NWFGSTP when the number of candidate devices in WSNs is large. In this section, we propose some post-processing algorithms to improve suboptimal solutions to NWFGSTP.

A. THE PROPOSED GROUP PRUNING ALGORITHM (GPA)

Here, we propose the Group Pruning Algorithm (GPA; Algorithm 1) to improve suboptimal solutions to NWFGSTP by pruning expensive vertices and edges, i.e., removing expensive devices from WSNs. For example, if there are two sensor nodes that are performing the same task, then we can remove the expensive one from WSNs. First, we formulate this pruning problem as the Node-Weighted Group Steiner Tree Problem in Trees (NWGSTPT), which is a special case of NWFGSTP where the input graph G is a tree and $L = \emptyset$. Consider Θ as a suboptimal solution to NWFGSTP, then solving NWGSTPT in Θ is equivalent to improving this solution by pruning expensive vertices and edges. Note that, this is also true when $L \neq \emptyset$ in NWFGSTP since pruning vertices and edges from a suboptimal solution to NWFGSTP does not turn leaf vertices in L into non-leaf vertices. Ihler *et al.* [15] showed that the group Steiner tree problem in trees is NP-hard. Since NWGSTPT is a more general case of the group Steiner tree problem in trees, NWGSTPT is also NP-hard. Our proposed GPA is a polynomial-time heuristic algorithm for solving it. Before introducing GPA, we propose two theorems as follows. Their proofs are trivial and thus are omitted.

Theorem 4: If a vertex is the only vertex that is in a group, then this vertex is in the FGSMT.

Algorithm 1 The Proposed Group Pruning Algorithm (GPA)

Input: a solution tree $\Theta(V', E', \Gamma, L, w, c)$
Output: an improved solution tree $\Theta_p(V'', E'')$

1: Initialize a max priority queue PQ
2: **if** there is no compulsory vertex **then**
3: Mark all the leaves of Θ as unchecked
4: **while** there is no compulsory vertex **do**
5: **while** there are unchecked leaves **do**
6: **for** unchecked leaf i **do**
7: Mark i as checked
8: **if** i is a group vertex **then**
9: Push i into PQ; priority: $c(i, v_{adj}) - w(i)$
10: **else**
11: Find its closest junction/group predecessor j
12: Prune the branch rooted at j
13: **if** j is a leaf **then**
14: Mark j as unchecked
15: **end if**
16: **end if**
17: **end for**
18: **end while**
19: **if** PQ is not empty **then**
20: Prune edge (v_{top}, v_{adj}) from Θ
21: **if** v_{adj} is a leaf **then**
22: Mark v_{adj} as unchecked
23: **end if**
24: Pop v_{top} out of PQ
25: **end if**
26: **end while**
27: **end if**
28: Select a compulsory vertex as the root r
29: Mark all the vertices as untagged
30: Tag r: $tag(r) = 0$
31: **while** there are untagged vertices **do**
32: **for** untagged j adjacent to tagged v_{adj} **do**
33: **if** v_{adj} is a junction or group vertex **then**
34: Tag j: $tag(j) = c(v_{adj}, j) - w(j)$
35: **else**
36: Tag j: $tag(j) = tag(v_{adj}) + c(v_{adj}, j) - w(j)$
37: **end if**
38: **end for**
39: **end while**
40: Clear PQ
41: Mark all the leaves of Θ as unchecked
42: **while** there are non-compulsory leaves **do**
43: **while** there are unchecked leaves **do**
44: **for** unchecked leaf i **do**
45: Mark i as checked
46: **if** i is a group vertex **then**
47: **if** i is a compulsory vertex **then**
48: Continue
49: **else**
50: Push i into PQ with priority $tag(i)$
51: **end if**
52: **else**
53: Find its closest junction/group predecessor j
54: Prune the branch rooted at j
55: **if** j is a leaf **then**
56: Mark j as unchecked
57: **end if**
58: **end if**
59: **end for**
60: **end while**
61: **if** PQ is not empty **then**
62: **if** v_{top} is not compulsory **then**
63: Prune edge (v_{top}, v_{adj}) from Θ
64: **if** v_{adj} is a leaf **then**
65: Mark v_{adj} as unchecked
66: **end if**
67: **end if**
68: Pop v_{top} out of PQ
69: **end if**
70: **end while**
71: $\Theta_p = \Theta$

Theorem 5: Every leaf of an FGSMT is the only vertex in this FGSMT that is in a group.

Let Θ be the input tree of GPA. First, we initialize a max priority queue PQ (Step 1). Then, we apply Theorem 4 to identify compulsory vertices in Θ. If there is no compulsory vertex, then we implement the following steps (Steps 3-26) to heuristically prune Θ until a compulsory vertex is obtained.

We mark all the leaves of Θ as unchecked (Step 3). For each unchecked leaf i, we distinguish two cases to check it:

Case 1: i is a group vertex, i.e., it is in a group. Since it may be cheaper to keep i than to keep other vertices in the same group, we cannot prune i at this stage. Thus, we push i into PQ for future comparison (Step 9), and its priority is $c(i, v_{adj}) - w(i)$, where v_{adj} is the adjacent vertex of i. Clearly, this priority is a lower-bound of the cost to keep i in Θ.

Case 2: i is not a group vertex. Since $w(i) \leq 0$, we can simply prune edge (i, v_{adj}) from Θ. Nevertheless, v_{adj} may also be a non-group leaf that can be pruned, and it is tremendously slow to prune such vertices one by one. Therefore, we find i's closest junction/group predecessor j (Step 11; a vertex whose degree is larger than 2 is a junction vertex), and then prune the branch rooted at j (Step 12). If j becomes a leaf, then we mark j as unchecked (Step 14).

We iterate the process above until all the leaves have been checked (Step 5), which means that all the leaves are now group vertices that have been pushed into PQ. Note that, since a single vertex may be left after this process as a feasible solution to NWGSTPT, PQ may be empty. If PQ is not empty (Step 19), we prune edge (v_{top}, v_{adj}) from Θ (Step 20), where v_{top} is the top vertex in PQ and v_{adj} is its adjacent vertex. The logic is that it is probably more expensive to keep v_{top} in Θ than to keep other group leaves in PQ. If v_{adj} becomes a leaf, then we mark it as unchecked (Step 22), and check it later

in the same way above. After that, we pop v_{top} out of PQ (Step 24). This pruning process ends when a compulsory vertex is obtained (Step 4).

Note that, this pruning process is heuristic since the pruned v_{top} may be in the FGSMT. Since the time complexities to insert and pop out elements for priority queues are respectively $O(1)$ and $O(log|V'|)$ (Fibonacci heaps [23]), this pruning process has a polynomial-time complexity of $O(|V'|log|V'|)$ (in the extreme scenario, all the vertices in Θ will be pushed into a priority queue, and the time complexity to pop them out of the queue is $O(log|V'|!) \approx O(|V'|log|V'|)$; Stirling's approximation [24]). Since NWGSTPT is NP-hard, it is unlikely to make this pruning process optimal while keeping a polynomial time complexity. Hence, we keep this pruning process as simple as that above to make GPA fast. With the obtained compulsory vertex, we apply a more accurate pruning process as follows.

We randomly select a compulsory vertex as the root r (Step 28; there may be multiple compulsory vertices initially). Then, we calculate the cost of the path from each vertex to the most adjacent group or junction vertex towards the root. To calculate such costs for every vertex in Θ, first, we mark all the vertices as untagged (Step 29); second, we tag r with a value $tag(r) = 0$ (Step 30); third, for each untagged vertex j which is adjacent to a tagged vertex v_{adj}, we iteratively tag j with a value until all the vertices have been tagged (Steps 31-39): if v_{adj} is a junction or group vertex, $tag(j) = c(v_{adj}, j) - w(j)$, otherwise $tag(j) = tag(v_{adj}) + c(v_{adj}, j) - w(j)$. Clearly, for each vertex i, $tag(i)$ is the cost of the path from i to the most adjacent group or junction vertex towards the root. This cost is a tighter lower-bound of the cost to keep i in Θ than $c(i, v_{adj}) - w(i)$ in Step 9. The time complexity of this calculation is $O(|V'|)$. It may be worth mentioning that new tighter lower-bounds of such costs can be explored in the future to improve GPA for applications that are not sensitive to speed.

After tagging all the vertices, we prune Θ using these tagged values (Steps 40-70). First, we clear PQ (Step 40). Then, we mark all the leaves of Θ as unchecked (Step 41). For unchecked leaf i, we distinguish three cases to check it:

Case 1: i is a compulsory vertex. We continue (Step 48).

Case 2: i is a non-compulsory group vertex. We push i into PQ with priority $tag(i)$ (Step 50).

Case 3: i is not a group vertex. We find its closest junction/group predecessor j (Step 53), and then prune the branch rooted at j (Step 54). If j becomes a leaf, we mark it as unchecked (Step 56).

We iterate the process above until all the leaves have been checked (Step 43), which means that all the leaves are now group vertices that are either compulsory or non-compulsory but have been pushed into PQ. It can be seen from Theorem 5 that all the non-compulsory group leaves can be pruned from Θ. The priorities of these non-compulsory group leaves in PQ are the heuristic costs to keep them in Θ. Therefore, we first check v_{top}. If v_{top} is not a compulsory vertex, then we prune edge (v_{top}, v_{adj}) from Θ (Step 63). If v_{adj} becomes a leaf, then we mark v_{adj} as unchecked (Step 65). After that, we pop v_{top} out of PQ (Step 68). Note that, even though we only push non-compulsory group leaves into PQ (Step 50), v_{top} may still be a compulsory vertex. The reason is that pruning edge (v_{top}, v_{adj}) from Θ may induce new compulsory vertices. Thus, we need to check whether v_{top} is a compulsory vertex or not (Step 62) before pruning edge (v_{top}, v_{adj}).

We iterate this new pruning process until all the leaves become compulsory vertices (Step 42), and the resulting subtree is the output of GPA (Step 71). This new pruning process is more accurate than the previous one (Steps 3-26) since $tag(i)$ in Step 50 is a tighter lower-bound of the cost to keep i in Θ than $c(i, v_{adj}) - w(i)$ in Step 9. Moreover, this new pruning process also has a polynomial-time complexity of $O(|V'|log|V'|)$, which means that GPA has a polynomial-time complexity of $O(|V'|log|V'|)$.

B. THE PROPOSED LEAF REPLACING ALGORITHM (LRA)

In WSNs, a sensor usually covers multiple targets, and a target is usually covered by multiple sensors. Consequently, many groups overlap with each other in SCGs and TCGs. Here, we propose the Leaf Replacing Algorithm (LRA; Algorithm 2) to improve suboptimal solutions to NWFGSTP by replacing expensive leaves.

Let $\Theta(V', E', \Gamma, L, w, c)$ be a tree in a connected undirected graph $G(V, E, \Gamma, L, w, c)$. For vertex i that is not in Θ but in G, if there is an edge (i, j) such that vertex j is in Θ, then we say that i is adjacent to Θ. We first mark every vertex i that is adjacent to Θ and shares groups with the leaves of Θ as unchecked (Step 2). Then, for each unchecked vertex i, we find edge (i, j) that corresponds to the value below (Step 5).

$$\Delta c = \min\{c(i, j) - w(i) - \sum_{m}\{c(m, n) - w(m)\}\} \quad (1)$$

where vertex j is in Θ and $j \notin L$, m is a leaf group vertex of Θ; edge (m, n) is in Θ; $\Theta_p = \Theta \setminus \sum_m\{(m, n)\} \cup (i, j)$ is also a feasible solution to NWFGSTP. Clearly, if $\Delta c < 0$, then $c(\Theta_p) < c(\Theta)$. Thus, by replacing edges $\sum_m(m, n)$ with (i, j) (Step 7), Θ can be improved. Subsequently, we mark every removed leaf as unchecked (Step 8) since they may be added back to Θ_p. We iterate this process until all the vertices have been checked.

The reason to only replace group vertices in LRA is that cheap leaves may be replaced by expensive ones if we allow m in Equation (1) to be non-group vertices. Nevertheless, non-group leaves should be removed from Θ anyway. We leave this job to GPA, which can remove non-group leaves more efficiently. Since the time complexity to check every vertex i is $O(|V|)$; to check every vertex j is $O(|V'|)$; to check every vertex m is $O(|V'|)$, the time complexity of LRA is $O(|V'|^2|V|)$. Therefore, LRA is slow when $|V'|$ is large. In our later proposed post-processing procedure for NWFGSTP, we put GPA ahead of LRA to make LRA fast by reducing $|V'|$.

Algorithm 2 The Proposed Leaf Replacing Algorithm (LRA)

Input: graph $G(V, E, \Gamma, L, w, c)$, tree $\Theta(V', E', \Gamma, L, w, c)$
Output: an improved solution tree $\Theta_p(V'', E'')$

1: $\Theta_p = \Theta$
2: Mark every vertex i that is adjacent to Θ_p and shares groups with the leaves of Θ_p as unchecked
3: **while** there are unchecked vertices **do**
4: **for** each unchecked vertex i **do**
5: Find edge (i, j) corresponding to Δc in Eq. (1)
6: **if** $\Delta c < 0$ **then**
7: $\Theta_p = \Theta_p \setminus \sum_m \{(m, n)\} \cup (i, j)$
8: Mark every replaced leaf m as unchecked
9: **end if**
10: **end for**
11: **end while**

C. THE PROPOSED BRANCH REPLACING ALGORITHM (BRA)

Here, we propose the Branch Replacing Algorithm (BRA; Algorithm 3) to improve suboptimal solutions to NWFGSTP by replacing expensive branches, i.e., replacing unreliable transmission routes with reliable ones (note that, relay nodes may be in these routes). First, we randomly select a group vertex as the root r (Step 2). Then, we mark all the vertices as untagged (Step 3). We tag r with two values (Step 4): $tag_1(r) = 0$; $tag_2(r) = r$. For untagged vertex j which is adjacent to tagged vertex i, if j is not a group or junction vertex, then we tag j with two values (Step 21): $tag_1(j) = tag_1(i)+1$; $tag_2(j) = tag_2(i)$. If j is a group or junction vertex, then we distinguish two cases:

Case 1: $tag_1(i) < \alpha$, where α is a constant and $\alpha \geq 0$. We tag j with two values (Step 18): $tag_1(j) = 0$; $tag_2(j) = j$.

Case 2: $tag_1(i) \geq \alpha$. We consider the branch between vertices j and $tag_2(i)$, i.e., $branch(j, tag_2(i))$, as a long branch that may be replaced. $tag_1(i)$ is the number of vertices between j and $tag_2(i)$, and none of these vertices is group or junction vertex. We define the cost of this branch as

$$c_{branch(j,tag_2(i))} = \sum c(k, l) - \sum w(q) \tag{2}$$

where edge (k, l) is in this branch, vertex q is in the middle of this branch. Removing $branch(j, tag_2(i))$ from Θ induces two connected components: Θ_j and $\Theta_{tag_2(i)}$, which respectively contain j and $tag_2(i)$. If there is an edge (m, n) such that $m \in \Theta_{tag_2(i)}$, $n \in \Theta_j$, and

$$c(m, n) = \min\{c(x, y)\} < c_{branch(j,tag_2(i))} \tag{3}$$

where $x \in \Theta_j$, $y \in \Theta_{tag_2(i)}$, $x, y \notin L$, then $\Theta_p = \Theta \setminus branch(j, tag_2(i)) \cup (m, n)$ is also a feasible solution to NWFGSTP, and $c(\Theta_p) < c(\Theta)$. Thus, by replacing $branch(j, tag_2(i))$ with edge (m, n) (Step 10), Θ can be improved. Moreover, when $tag_1(m) > 0$ and m becomes a junction vertex after replacing the branch, we need to update the tagged values of m and its offspring that have been tagged (Step 12). Since the time complexity to tag vertices in Θ is $O(|V'|)$; to find edge (m, n) is $O(|E|)$, the time complexity of BRA is $O(|V'||E|)$.

Algorithm 3 The Proposed Branch Replacing Algorithm (BRA)

Input: graph $G(V, E, \Gamma, L, w, c)$, tree $\Theta(V', E', \Gamma, L, w, c)$, parameter α
Output: an improved solution tree $\Theta_p(V'', E'')$

1: $\Theta_p = \Theta$
2: Randomly select a group vertex in Θ_p as the root r
3: Mark all the vertices as untagged
4: Tag r: $tag_1(r) = 0$; $tag_2(r) = r$
5: **while** there are untagged vertices **do**
6: **for** untagged vertex j adjacent to tagged vertex i **do**
7: **if** j is a group or junction vertex **then**
8: **if** $tag_1(i) \geq \alpha$ **then**
9: **if** edge (m, n) satisfying Equation (3) **then**
10: $\Theta_p = \Theta_p \setminus branch(j, tag_2(i)) \cup (m, n)$
11: **if** $tag_1(m) > 0$ & m is a new junction **then**
12: Update tagged values of m and its offspring
13: **end if**
14: **else**
15: Tag j: $tag_1(j) = 0$; $tag_2(j) = j$
16: **end if**
17: **else**
18: Tag j: $tag_1(j) = 0$; $tag_2(j) = j$
19: **end if**
20: **else**
21: Tag j: $tag_1(j) = tag_1(i) + 1$; $tag_2(j) = tag_2(i)$
22: **end if**
23: **end for**
24: **end while**

D. THE PROPOSED MINIMUM FULL SPANNING TREE ALGORITHM (MFSTA)

The Minimum Spanning Tree (MST) algorithm is widely used to improve suboptimal solutions to some classical Steiner tree problems in graphs, such as the prize-collecting Steiner tree problem [25]. It does this by finding the MST that spans all the vertices in that solution. However, we cannot directly apply it to improve suboptimal solutions to NWFGSTP due to the existence/constraint of leaf vertices. To address this issue, we first propose a new spanning tree problem as follows.

Definition 6 (The Minimum Full Spanning Tree Problem): Let $G(V, E, L, c)$ be a connected undirected graph, where V is the set of vertices, E is the set of edges, L is a subset of V called leaf vertices, and c is a function which maps each edge in E to a nonnegative value called edge cost. The purpose is to find a spanning tree $\Theta(V, E')$, $E' \subseteq E$ with the minimum cost $c(\Theta) = \sum_{e \in E'} c(e)$, and every $i \in L$ is a leaf of Θ.

We can improve a suboptimal solution to NWFGSTP by finding the Minimum Full Spanning Tree (MFST) that spans

all the vertices in this solution, i.e., to find a more reliable routing tree. Note that, there may be no full spanning tree in some instances. However, in our application, we assume that there is always a full spanning tree in TCGs. We prove that the MFST can be found in polynomial time by attaching large costs to edges that are adjacent to leaf vertices.

Theorem 6: Given two graphs $G(V, E, L, w, c)$ *and* $G'(V, E, L, w, c')$. *If there are full spanning trees in them; and for every edge* $(i, j) \in E$, $c'(i, j) = c(i, j) + \tau M$, *where* $\tau = |L \cap \{i, j\}|$, $M \geq \sum_{e \in E} |c(e)|$, *then the MST in* G' *and the MFST in* G *share the same topology.*

Proof: Assume $\Theta_1(V, E_1)$ is a full spanning tree in G'; $\Theta_2(V, E_2)$ is an MST in G', and there is a leaf vertex $i \in L$ such that its degree $\delta(i) = x \geq 1$ in Θ_2; $\Theta_3(V, E_3)$ is an MFST in G'; $\Theta_4(V, E_4)$ is an MFST in G. Since $c(\Theta_1) \geq c(\Theta_2)$, we have

$$\sum_{e \in E_1} c(e) + |L|M \geq \sum_{e \in E_2} c(e) + (|L| + x - 1)M \qquad (4)$$

Thus, $x = 1$; Θ_2 is also a full spanning tree. Therefore, $c(\Theta_2) = c(\Theta_3)$. On the other hand, since $c(\Theta_3) = c(\Theta_4) + |L|M$, every topology that induces Θ_3 also induces Θ_4. Hence, every topology that induces Θ_2 also induces Θ_4, vice versa, which means that the topology of the MST in G' is the same as that of the MFST in G. □

Based on this theorem, we propose the Minimum Full Spanning Tree Algorithm (MFSTA; Algorithm 4) to improve suboptimal solutions to NWFGSTP. Given a solution tree $\Theta(V', E', \Gamma, L, w, c)$ (or a graph G), we first construct $G'(V', E'_2, \Gamma, L, w, c')$, where E'_2 is the set of edges in the initial graph G that connect vertices in V', c' is the set of edge costs as those in Theorem 6. Then, we find the MST of G' as an improved solution tree. Since the time complexity to find the MST on G' is $O(|E'_2| + |V'|log|V'|)$ (Prim's algorithm [26]), MFSTA has a polynomial-time complexity of $O(|E'_2| + |V'|log|V'|)$.

Algorithm 4 The Proposed Minimum Full Spanning Tree Algorithm (MFSTA)

Input: graph $G(V, E, \Gamma, L, w, c)$, tree $\Theta(V', E', \Gamma, L, w, c)$
Output: an improved solution tree $\Theta_p(V'', E'')$

1: Construct $G'(V', E'_2, \Gamma, L, w, c')$
2: $\Theta_p = MST(G')$

E. THE PROPOSED POST-PROCESSING PROCEDURES (POSTP1&2)

There are four post-processing algorithms proposed above: GPA, LRA, BRA and MFSTA. Here, we combine them as two post-processing procedures for NWFGSTP. First, we combine all these four algorithms together as a procedure: PostP1 (Algorithm 5). Different sequences of these algorithms may induce different post-processing effects and running times. Thus, a trade-off between post-processing effect and speed is required to sequence these algorithms. We investigate this trade-off by comparing all the 24 (4!) sequences of these four algorithms through computational trials. The results show that the order between GPA and LRA has the most significant impact on the speed: all the fastest 12 sequences have GPA ahead of LRA. The reason is that GPA reduces the sizes of suboptimal solutions in polynomial time, while the running time of LRA decreases significantly as these sizes decrease. The GMLB sequence, i.e., GPA → MFSTA → LRA→ BRA, achieves a good trade-off between post-processing effect and speed. Hence, we use this sequence in PostP1. We iteratively implement this sequence until the suboptimal solution to NWFGSTP cannot be improved any more. The time complexity of PostP1 is $O(|V'|^2|V| + |V'||E|)$. We will apply it to post-process suboptimal solutions to NWFGSTP in small instances. Furthermore, PostP1 can be modified to achieve stronger post-processing effects in scenarios that are not sensitive to speed. For example, in LRA and BRA, we only use edges to replace leaves and branches. Since there are only nonpositive node weights, we can instead find the lowest-cost paths to replace leaves and branches. By doing this, stronger post-processing effects may be achieved.

On the other hand, PostP1 is slow in large instances due to the large time complexities of LRA and BRA. Thus, we combine GPA and MFSTA as another procedure: PostP2 (Algorithm 6). We use the GMG sequence, i.e., GPA → MFSTA → GPA. Unlike PostP1, we implement this sequence once without iteration. Clearly, the time complexity of PostP2 is $O(|E'_2| + |V'|log|V'|)$. We will apply it to find cheaper node placement solutions in large instances.

Algorithm 5 The First Proposed Post-Processing Procedure (PostP1)

Input: graph $G(V, E, \Gamma, L, w, c)$, tree $\Theta(V', E', \Gamma, L, w, c)$, parameter α
Output: an improved solution tree $\Theta_p(V'', E'')$

1: $\Theta_p = \Theta$
2: **while** Θ_p can be improved **do**
3: $\quad \Theta_p = GPA(\Theta_p)$
4: $\quad \Theta_p = MFSTA(\Theta_p, G)$
5: $\quad \Theta_p = LRA(\Theta_p, G)$
6: $\quad \Theta_p = BRA(\Theta_p, G, \alpha)$
7: **end while**

Algorithm 6 The Second Proposed Post-Processing Procedure (PostP2)

Input: graph $G(V, E, \Gamma, L, w, c)$, tree $\Theta(V', E', \Gamma, L, w, c)$
Output: an improved solution tree $\Theta_p(V'', E'')$

1: $\Theta_p = \Theta$
2: $\Theta_p = GPA(\Theta_p)$
3: $\Theta_p = MFSTA(\Theta_p, G)$
4: $\Theta_p = GPA(\Theta_p)$

V. THE PROPOSED ALGORITHMS FOR NWFGSTP

In this section, we propose an exponential-time exact algorithm and a polynomial-time heuristic algorithm to respectively produce optimal solutions for small instances and suboptimal solutions for large instances.

A. AN EXPONENTIAL-TIME EXACT ALGORITHM

Here, we propose an exact algorithm (ExactA; Algorithm 7) for NWFGSTP. Let $G(V, E, \Gamma, L, w, c)$ be the input graph. We first apply MFSTA on G to produce a feasible solution (Step 1). Then, we use Theorem 4 to identify compulsory vertices (Step 2). For each of the other vertices, there are two possible scenarios, namely that it is in the FGSMT or it is not in the FGSMT. Therefore, the total number of possible scenarios is $2^{|V|-|C|}$, where $|C|$ is the number of identified compulsory vertices. We iteratively check every possible scenario (Step 3). In each of them, a subgraph $G_s(V', E', \Gamma, L, w, c)$ is induced, where V' is the set of included vertices, E' is the set of edges in G that connect vertices in V'. If G_s is connected; and for each group $g \in \Gamma$, $g \cap V' \neq \emptyset$, then we say that G_s is a feasible subgraph, which means that it may contain feasible solutions to NWFGSTP. If G_s is a feasible subgraph, then we apply MFSTA to produce a spanning tree Θ_m on it (Step 5). Since there may be no full spanning tree in G_s, Θ_m may not be a feasible solution to NWFGSTP. On the other hand, if Θ_m is a feasible solution, then we use it to update the best feasible solution we obtained so far (Step 7). The best feasible solution in all these possible scenarios is the FGSMT on G. Clearly, ExactA has an exponential-time complexity of $O(2^{|V|-|C|})$. Unlike the conventional exact algorithms for other Steiner tree problems in graphs, ExactA has a low demand on machines, and thus can be implemented on simple devices. Nevertheless, it is slow for large instances. Faster exact algorithms, such as the ones based on the branch-and-cut [21] or branch-and-bound [27] idea, can be explored in the future.

B. A POLYNOMIAL-TIME HEURISTIC ALGORITHM

Solving NWFGSTP to optimality may be impractical when the number of candidate devices is large in WSNs. Hence, it is preferable to develop fast heuristic algorithms to address this issue. Here, we propose such an algorithm: MGA (MFSTA + GPA; Algorithm 8). Let $G(V, E, \Gamma, L, w, c)$ be the input graph. In MGA, we first construct a new graph $G_1(V, E, \Gamma, L, w, c_1)$ (Step 1). In G_1, the cost of edge (i, j) is $c_1(i, j) = c(i, j) - w(i) - w(j)$. Then, we apply MFSTA on G_1 (Step 2). Ultimately, we use GPA to prune the MFST to produce a feasible solution to NWFGSTP (Step 3). The MFST on G_1 is a better spanning tree for solution generation than the MFST on G, as it considers the existence of negative node weights. There is no approximation guarantee for MGA. However, we will later show that MGA finds high-quality solutions in practice. MGA has a polynomial-time complexity of $O(|E| + |V|log|V|)$. Our computational trials show that it is fast and consumes little memory in practice.

Algorithm 7 The Proposed Exact Algorithm (ExactA)

Input: graph $G(V, E, \Gamma, L, w, c)$
Output: an optimal solution $\Theta_{opt}(V_{opt}, E_{opt})$

1: Initialize $\Theta_{opt} = MFSTA(G)$
2: Identify compulsory vertices using Theorem 4
3: **for** each scenario **do**
4: **if** G_s is a feasible subgraph **then**
5: $\Theta_m = MFSTA(G_s)$
6: **if** Θ_m is feasible and $c(\Theta_m) < c(\Theta_{opt})$ **then**
7: $\Theta_{opt} = \Theta_m$
8: **end if**
9: **end if**
10: **end for**

Algorithm 8 The Proposed Heuristic Algorithm (MGA)

Input: graph $G(V, E, \Gamma, L, w, c)$
Output: a feasible solution $\Theta(V', E')$

1: Construct $G_1(V, E, \Gamma, L, w, c_1)$
2: $\Theta = MFSTA(G_1)$
3: $\Theta = GPA(\Theta)$

VI. EXPERIMENTAL EVALUATION

In this section, we evaluate our complete methodology through experiments. These experiments are conducted on a personal computer from 2016 (Intel Core i7-4790 CPU).[1] Given that the existing node placement strategies and techniques place heterogeneous nodes separately; and it is difficult to integrate and implement them due to their different assumptions and objectives (e.g. [3]–[7]), we only implement ExactA, MGA and PostP1&2 here. We will evaluate the speed of ExactA, and both the speed and solution quality of MGA and PostP1&2. Like the previous work (e.g. [3]–[7]), we randomly generate benchmark instances with different graph features to evaluate our algorithms. There are four graph features: $|V|$, $|E|$, $|\Gamma|$, and $|L|$, in which $|V|$ is the number of vertices, i.e., the number of base stations, sensor nodes, and relay nodes; $|E|$ is the number of edges, i.e., the number of transmission routes; $|\Gamma|$ is number of groups, i.e., the number of targets plus 1 (the sensor nodes covering each target are in a group; the base stations are also in a group); and $|L|$ is the number of leaf vertices, i.e., the number of sensor nodes. We use these benchmark instances to evaluate the performances of our algorithms in different scenarios. Ultimately, we apply a heterogeneous node placement example to show their usefulness in practice.

A. APPLICATION TO BENCHMARK INSTANCES

First, we apply ExactA to solve some small instances to optimality. The average running times of ExactA are shown in Figure 3a. It can be seen that ExactA can solve small

[1]The codes and datasets are available at https://github.com/YahuiSun/minimum-cost-heterogeneous-node-placement.

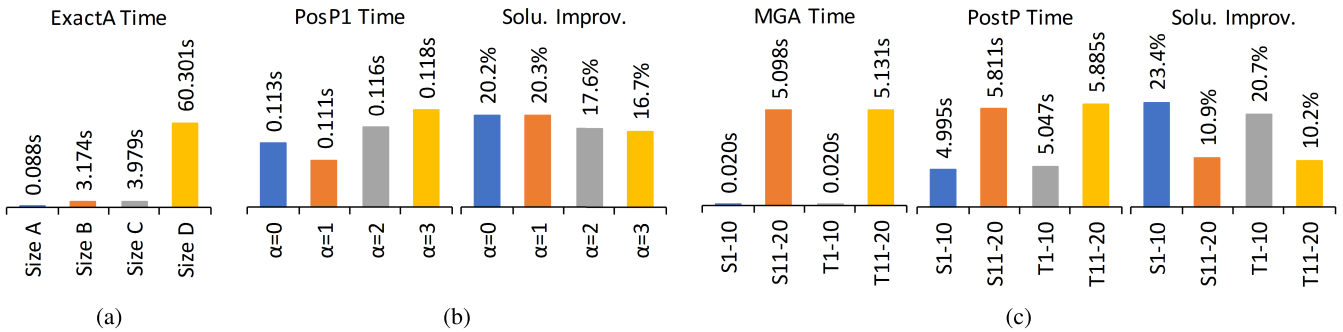

FIGURE 3. Application to benchmark instances. (a) shows the average running times of ExactA for different instances; Size A: $|V| = 20$, $|E| = 190$, $|\Gamma| = 40$; Size B: $|V| = 20$, $|E| = 190$, $|\Gamma| = 20$; Size C: $|V| = 30$, $|E| = 435$, $|\Gamma| = 60$; Size D: $|V| = 30$, $|E| = 435$, $|\Gamma| = 30$; for each $g \in \Gamma$, $1 \leq |g| \leq 3$. (b) show the performance of PostP1 with different values of α for instances where $|V| = 100$, $|E| = 450$, $|\Gamma| = 50$; *Solu. Improv.* shows the improvements of the MGA solutions. (c) shows the performance of MGA and PostP1&2 for the S/T instances (PostP1 in S1-10 and T1-10; PostP2 in S11-20 and T11-20).

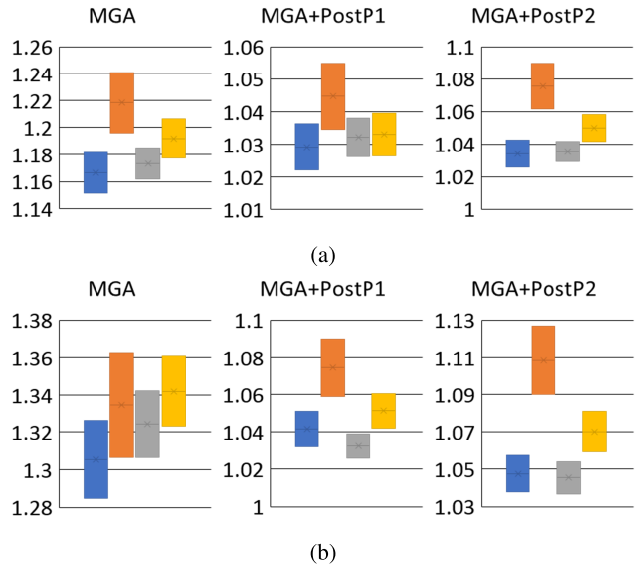

FIGURE 4. The 95% Confidence Intervals of the approximation ratios of MGA and its post-processed solutions. Blue, orange, gray and yellow bars respectively correspond to Size A-D instances in Figure 3a. In (a), $|L| = 0$. In (b), $|L| = 5, 5, 10, 10$.

instances with dozens of vertices within seconds. The reason why ExactA is faster when $|\Gamma|$ is large is that more compulsory vertices can be identified in this case (details in Algorithm 7). Since ExactA has a low demand on computational resources and thus can be implemented on simple devices, it may be preferable to apply ExactA to design small WSNs in some cases. However, ExactA is slow for large instances. Therefore, faster exact algorithms are still recommended for future work.

Subsequently, we analyze the impact of α on PostP1. The performance of PostP1 with different values of α is illustrated in Figure 3b. PostP1 is faster when $\alpha = 0/1$. The reason is that, in this case, BRA has stronger post-processing effects by replacing more branches. As a result, PostP1 needs fewer iterations to finish the post-process. For the same reason, the solution improvements of PostP1 are larger when $\alpha = 0/1$. Based on these results, we set $\alpha = 1$ in the following experiments.

We then apply MGA and PostP1&2 to solve some large instances. There are 40 instances generated: S1-20 and T1-20, which respectively correspond to SCGs and TCGs. Their sizes are listed as follows. S1-10: $|V| = 1000$, $|E| = 10000$, $|\Gamma| = 100$, $|L| = 0$; S11-20: $|V| = 100000$, $|E| = 1000000$, $|\Gamma| = 100000$, $|L| = 0$; T1-10: $|V| = 1000$, $|E| = 10000$, $|\Gamma| = 100$, $|L| = 100$; T11-20: $|V| = 100000$, $|E| = 1000000$, $|\Gamma| = 100000$, $|L| = 10000$. The experimental results are shown in Figure 3. Since MGA is polynomial, it can find fast feasible solutions for these instances. On the other hand, PostP1 is also fast for S/T1-10, and it generally improves the MGA solutions by 20%. However, it is slow for S/T11-20. Therefore, we use PostP2 for S/T11-20, and it generally improves the MGA solutions by 10%. These results show that MPA and PostP1&2 are useful for minimum-cost heterogeneous node placement in large WSNs.

We further compare the heuristic solutions of MGA and PostP1&2 with the optimal ones produced by ExactA for small instances. The 95% Confidence Intervals of the approximation ratios of these heuristic solutions, i.e., the ratios between their costs and those of the optimal ones, are demonstrated in Figure 4. It can be seen that, even though there is no approximation guarantee for MGA, its approximation ratios are generally small in practice, and our post-processing algorithms can improve its solutions to near-optimal. Moreover, the approximation ratios of MGA are generally smaller when $L = \emptyset$ and $|\Gamma|$ is large. The reason is that $L = \emptyset$ induces more efficient edges in the MST by removing the constraints of leaf vertices; and a large number of group constraints increases the sizes of suboptimal solutions and thus makes the errors caused by certain suboptimal components comparatively smaller. This indicates that MGA may have a better performance in designing single-tiered WSNs with a large number of targets.

B. CASE STUDY: MINIMUM-COST HETEROGENEOUS NODE PLACEMENT

Ultimately, we solve the minimum-cost heterogeneous node placement problem in an example of WSNs. This WSN is modified from the one deployed across the first floor of the

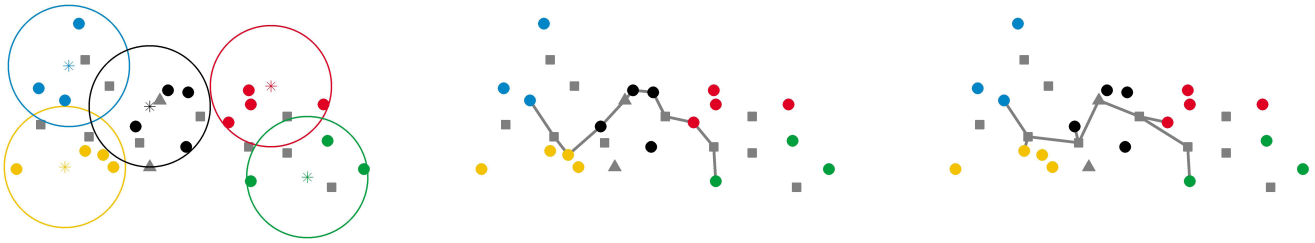

FIGURE 5. Application of our proposed node placement approach in an example of WSNs. The asterisks represent targets; the gray triangles represent candidate base stations; the gray boxes represent candidate relay nodes; the colorful dots represent candidate sensor nodes; and the gray lines represent transmission routes in the routing tree, i.e., the Steiner tree. In (a), each circle encircles a target and the candidate sensor nodes covering this target, and all of them share the same color.

NIMBUS Center for Embedded Systems Research building at Cork Institute of Technology in Ireland [28]. There are 5 targets, 2 candidate base stations, 18 candidate sensor nodes and 10 candidate relay nodes (see Figure 5a). For the sake of simplicity, we assume that the outage probabilities of transmission routes are proportional to their distances. We associate devices with random node weights to represent their production and placement costs. We construct an SCG and a TCG as those in Definitions 1 and 2. Then, we apply MGA and PostP1 to solve NWFGSTP in them, and the designed single-tiered and two-tiered WSNs are respectively shown in Figures 5b and 5c. Clearly, a small number of devices with short transmission routes between them are deployed to meet the coverage and connectivity requirements, which means that the sum of node production and placement costs and transmission outage probabilities in the routing tree is minimized through our approach. This indicates the usefulness of our approach for minimum-cost heterogeneous node placement in practice.

VII. CONCLUSION

In this paper, we meet the challenge of minimum-cost heterogeneous node placement by finding the locations of base stations, sensor nodes, and relay nodes simultaneously to minimize the sum of node production and placement costs and transmission outage probabilities in the routing tree, while meeting the coverage and connectivity requirements. First, we formulate both single-tiered and two-tiered minimum-cost heterogeneous node placement problems as NWFGSTP. Subsequently, we propose an exponential-time exact algorithm, a polynomial-time heuristic algorithm and several post-processing algorithms for NWFGSTP. Ultimately, we evaluate their performances through experiments, and demonstrate their usefulness for minimum-cost heterogeneous node placement in practice.

REFERENCES

[1] O. M. Alia, "Dynamic relocation of mobile base station in wireless sensor networks using a cluster-based harmony search algorithm," *Inf. Sci.*, vols. 385–386, pp. 76–95, Apr. 2017.

[2] X. Li, D. Guo, J. Grosspietsch, H. Yin, and G. Wei, "Maximizing mobile coverage via optimal deployment of base stations and relays," *IEEE Trans. Veh. Technol.*, vol. 65, no. 7, pp. 5060–5072, Jul. 2016.

[3] J. L. Bredin, E. D. Demaine, M. T. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed fault tolerance," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 216–228, Feb. 2010.

[4] W.-C. Ke, B.-H. Liu, and M.-J. Tsai, "Efficient algorithm for constructing minimum size wireless sensor networks to fully cover critical square grids," *IEEE Trans. Wireless Commun.*, vol. 10, no. 4, pp. 1154–1164, Apr. 2011.

[5] M. Bagaa, A. Chelli, D. Djenouri, T. Taleb, I. Balasingham, and K. Kansanen, "Optimal placement of relay nodes over limited positions in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2205–2219, Apr. 2017.

[6] L. Liu, M. Ma, C. Liu, and Y. Shu, "Optimal relay node placement and flow allocation in underwater acoustic sensor networks," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 2141–2152, May 2017.

[7] S. Misra, S. D. Hong, G. Xue, and J. Tang, "Constrained relay node placement in wireless sensor networks: Formulation and approximations," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 434–447, Apr. 2010.

[8] Y. Sun, "Classical, prize-collecting node-weighted Steiner tree problems graphs," Ph.D. dissertation, Dept. Mech. Eng., Univ. Melbourne, Melbourne, Australia, 2018.

[9] S. Abdollahzadeh and N. J. Navimipour, "Deployment strategies in the wireless sensor network: A comprehensive review," *Comput. Commun.*, vols. 91–92, pp. 1–16, Oct. 2016.

[10] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 6, no. 4, pp. 621–655, 2008.

[11] M. Brazil, R. L. Graham, D. A. Thomas, and M. Zachariasen, "On the history of the Euclidean Steiner tree problem," *Arch. History Exact Sci.*, vol. 68, no. 3, pp. 327–354, 2014.

[12] Y. Sun, C. Ma, and S. Halgamuge, "The node-weighted Steiner tree approach to identify elements of cancer-related signaling pathways," *Bioinformatics*, vol. 18, no. 16, pp. 1–13, 2017.

[13] D. Yang, S. Misra, X. Fang, G. Xue, and J. Zhang, "Two-tiered constrained relay node placement in wireless sensor networks: Computational complexity and efficient approximations," *IEEE Trans. Mobile Comput.*, vol. 11, no. 8, pp. 1399–1411, Aug. 2012.

[14] S. E. Dreyfus and R. A. Wagner, "The Steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971.

[15] E. Ihler, G. Reich, and P. Widmayer, "Class Steiner trees and VLSI-design," *Discrete Appl. Math.*, vol. 90, nos. 1–3, pp. 173–194, 1999.

[16] C. E. Ferreira and F. M. de O. Filho, "New reduction techniques for the group Steiner tree problem," *SIAM J. Optim.*, vol. 17, no. 4, pp. 1176–1188, 2006.

[17] E. Halperin, G. Kortsarz, R. Krauthgamer, A. Srinivasan, and N. Wang, "Integrality ratio for group Steiner trees and directed Steiner trees," *SIAM J. Comput.*, vol. 36, no. 5, pp. 1494–1511, 2007.

[18] I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti, "An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem," *Math. Program.*, vol. 105, nos. 2–3, pp. 427–449, 2006.

[19] Y. Sun, M. Brazil, D. Thomas, and S. Halgamuge, "The fast heuristic algorithms and post-processing techniques to design large and low-cost communication networks," *IEEE/ACM Trans. Netw.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8604004
[20] A. Segev, "The node-weighted Steiner tree problem," *Networks*, vol. 17, no. 1, pp. 1–17, 1987.
[21] M. Fischetti *et al.*, "Thinning out Steiner trees: A node-based model for uniform edge costs," *Math. Program. Comput.*, vol. 9, no. 2, pp. 203–229, 2016.
[22] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.
[23] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.
[24] G. Marsaglia and J. C. W. Marsaglia, "A new derivation of Stirling's approximation to n!" *Amer. Math. Monthly*, vol. 97, no. 9, pp. 826–829, 1990.
[25] D. S. Johnson, M. Minkoff, and S. Phillips, "The prize collecting Steiner tree problem: Theory and practice," in *Proc. SODA*, 2000, pp. 760–769.
[26] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.*, vol. 36, no. 6, pp. 1389–1401, 1957.
[27] M. Leitner, I. Ljubić, M. Luipersbeck, and M. Sinnl, "A dual-ascent-based branch-and-bound framework for the prize-collecting Steiner tree and related problems," *INFORMS J. Comput.*, vol. 30, no. 2, pp. 402–420, 2018.
[28] A. Guinard, M. S. Aslam, D. Pusceddu, S. Rea, A. McGibney, and D. Pesch, "Design and deployment tool for in-building wireless sensor networks: A performance discussion," in *Proc. IEEE 36th Conf. Local Comput. Netw.*, Oct. 2011, pp. 649–656.

YAHUI SUN received the bachelor's and master's degrees in aerospace engineering from the Harbin Institute of Technology, in 2012 and 2014, respectively, and the Ph.D. degree from The University of Melbourne, in 2018. He is currently a Postdoctoral Fellow with the Research School of Engineering, The Australian National University. His research interests include computer networks and network data analytics, especially Steiner tree problems in node-weighted graphs and their applications to social, biomedical, telecommunication, and wireless sensor networks.

SAMAN HALGAMUGE was the Director of the Research School of Engineering, The Australian National University. He is currently a Professor with the Department of Mechanical Engineering, The University of Melbourne. His research interests include machine learning, big data analytics, and optimization, with applications to smart grids, sustainable energy generation, bioinformatics, and neuro-engineering. His fundamental research contributions are big data analytics with unsupervised and near unsupervised type learning and transparent deep learning and bioinspired optimization. He was a member of the Australian Research Council College of Experts for Engineering, Information and Computing Sciences. He is a Fellow of the IEEE.