A Physarum-Inspired Algorithm for Minimum-Cost Relay Node Placement in Wireless Sensor Networks

Yahui Sun, Daniel Rehfeldt, Marcus Brazil, Doreen Thomas, *Senior Member, IEEE*, and Saman Halgamuge, *Fellow, IEEE*

***Abstract*— Relay node placement, which aims to connect pre-deployed sensor nodes to base stations, is essential in minimizing the costs of wireless sensor networks. In this paper, we formulate the new Node-Weighted Partial Terminal Steiner Tree Problem (NWPTSTP) for minimum-cost relay node placement in two-tiered wireless sensor networks. The objective is to minimize the sum of heterogeneous production and placement costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree simultaneously. This extends the previous work that considers the costs of relay nodes to be homogeneous. After formulating NWPTSTP for this purpose, we prove that it can be transformed to the existing node-weighted Steiner tree problem. Subsequently, we conduct some theoretical analyses on the emerging Physarum-inspired algorithms to reveal their potential of computing Steiner trees. Based on these analyses, we propose a new Physarum-inspired algorithm for solving NWPTSTP. We conduct computational trials to show that: 1) in comparison to a state-of-the-art approximation algorithm for solving the node-weighted Steiner tree problem, our Physarum-inspired algorithm can produce better solutions in a smaller amount of time; and 2) in comparison to two state-of-the-art relay node placement algorithms, our Physarum-inspired algorithm can design wireless sensor networks with 25% lower relay cost and similar quality of service (specifically, 5% shorter network lifetime, 2% longer delay, and 0% loss of goodput). This indicates the usefulness of our Physarum-inspired algorithm for minimum-cost relay node placement in budget-limited scenarios.**

***Index Terms*— Physarum polycephalum, Steiner tree problem, nature-inspired algorithm, Internet of Things.**

Manuscript received March 28, 2018; revised November 28, 2018 and August 18, 2019; accepted January 21, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor O. Gurewitz. This work was supported in part by the Grant DP150103512. *(Corresponding author: Yahui Sun.)*

Yahui Sun was with the Research School of Engineering, Australian National University, Canberra, ACT 2601, Australia. He is now with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: yahuisun@outlook.com).

Daniel Rehfeldt is with the Department of Mathematics, Technische Universität Berlin, 10623 Berlin, Germany (e-mail: rehfeldt@zib.de).

Marcus Brazil and Doreen Thomas are with the School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: brazil@unimelb.edu.au; doreen.thomas@unimelb.edu.au).

Saman Halgamuge is with the School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia, and also with the Australian National University, Canberra, ACT 2601, Australia (e-mail: saman@unimelb.edu.au).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TNET.2020.2971770

I. INTRODUCTION

WIRELESS Sensor Networks (WSNs), which can be considered as "eyes and ears" of the emerging Internet of Things, consist of spatially distributed autonomous devices using sensors to monitor environmental conditions. These sensors are generally not powerful or energy efficient enough for long distance transmission, and may easily break down and leave the network disconnected in harsh environments (e.g. [1]). Thus, relay nodes that are energy sufficient and can transmit data between devices are often deployed to reduce the energy consumption of sensors or to reconnect sensors to base stations. Given that relay nodes are expensive; and their locations determine the network topology and operation, relay node placement is essential in minimizing the costs of WSNs, while enhancing the Quality of Service (QoS).

Based on the routing topology, relay node placement can be classified into two groups: *single-tiered* and *two-tiered*. Both sensor and relay nodes can relay data in single-tiered relay node placement, while only relay nodes can do this in the two-tiered one. Furthermore, based on the deployment locations, relay node placement can be classified differently into two groups: *constrained* and *unconstrained*. Relay nodes can only be placed at pre-determined candidate locations in constrained relay node placement, while they can be placed anywhere in the unconstrained one. Since two-tiered topology is more energy efficient; and constrained relay node placement permits the capture of possible geographical constraints in reality [2] (e.g. inaccessible private properties), we focus on two-tiered constrained relay node placement in this paper.

The two-tiered constrained relay node placement problem was first studied by Yang *et al.* [3]. Assuming that the costs of relay nodes are homogeneous, their objective was to place a minimum number of relay nodes to achieve the network connectivity and survivability. A lot of work has been done based on their model (e.g. [4], [5]). Most recently, Bagaa *et al.* [6] extended their model by minimizing not only the number of relay nodes but also the sum of outage probabilities of transmission routes in a routing tree. Bagaa *et al.* showed that, by doing this, they can reduce the costs of WSNs, while enhancing the QoS.

Nevertheless, the previous work above may not be cost-aware enough for minimum-cost relay node placement, given that an intuitive fact is ignored: there may be different types of

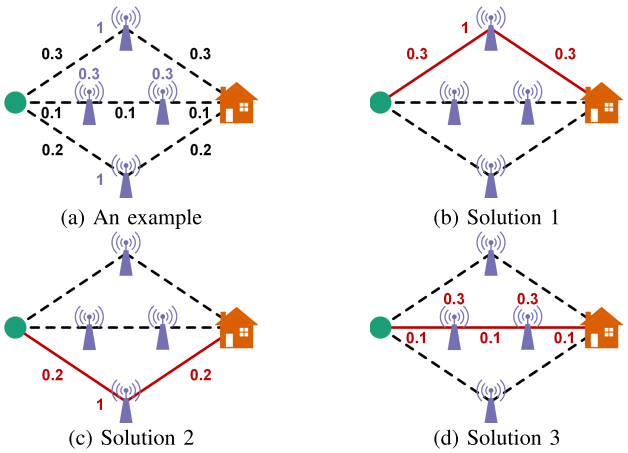


Fig. 1. An instance of placing relay nodes (purple) to connect a sensor node (green) with a base station (orange). (a) shows the instance, where the numbers represent the costs of relay nodes and the outage probabilities of transmission routes. (b)-(d) show three solutions (red). Yang *et al.* [3] may find Solution 1 for placing a minimum number of relay nodes. Bagaa *et al.* [6] finds Solution 2 for placing a minimum number of relay nodes, while minimizing the sum of outage probabilities of transmission routes in a routing tree. We find Solution 3 for minimizing the sum of heterogeneous costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree simultaneously (the cost of Solution 3, i.e., $0.3 \times 2 + 0.1 \times 3$, is the minimum).

relay nodes with different production costs, and the placement costs of the same type of relay nodes may vary between locations. For example, to monitor the vibration of a bridge using wireless sensors, we may need to place different types of relay nodes above and below the water, and the workers may charge different prices for placing them. We consider the costs of relay nodes to be the sums of their production and placement costs. Ignoring the heterogeneity of these costs greatly limits the applicability of the previous work above for designing WSNs with sufficiently low costs. Therefore, more cost-aware models are required for minimum-cost relay node placement in reality.

In this paper, we address this issue by formulating and solving the new Node-Weighted Partial Terminal Steiner Tree Problem (NWPTSTP). Given a connected undirected graph with nonnegative node weights and edge costs, NWPTSTP asks for a tree such that 1) compulsory vertices are connected by this tree; 2) some special compulsory vertices are leaves of this tree; and 3) the sum of node weights and edge costs in this tree is minimized. Since base stations and sensor nodes should be connected by a routing tree; and sensor nodes cannot relay data and thus should be leaves of a routing tree, we can solve NWPTSTP to place relay nodes by 1) using compulsory vertices to represent base stations and sensor nodes; 2) using special compulsory vertices to represent sensor nodes; 3) using non-compulsory vertices with different node weights to represent relay nodes with different costs; and 4) using edges with different costs to represent transmission routes with different outage probabilities. Our approach extends the previous ones by minimizing the sum of heterogeneous costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree simultaneously (e.g. Figure 1).

It is hard to solve NWPTSTP due to the NP-completeness. Like other Steiner tree problems (e.g. [7]), *exact* and *non-exact* algorithms are used in different scenarios. Exact algorithms

Fig. 2. Photograph of Physarum polycephalum (provided by Prof. Toshiyuki Nakagaki in the Hokkaido University, Japan).

can produce optimal solutions, but have exponential time complexities. Moreover, exact algorithms generally demand a lot of computational resources that are often not available. In comparison, non-exact algorithms cannot guarantee optimality, but often have polynomial time complexities, and generally demand much lower computational resources. Since WSNs may be large and computational resources may be limited, we focus on non-exact algorithms in this paper.

The Physarum-inspired Algorithms (PAs) are non-exact algorithms that are promoted by the research on Physarum polycephalum (see Figure 2), which is a slime mold that has exhibited many intelligent behaviors, such as solving mazes and building efficient networks (e.g. [8]–[10]; and a TED talk in 2014 [11]). Some biological experiments have shown that the tubular structures of Physarum polycephalum are often analogous to those of Steiner trees (e.g. [12]), which indicates that PAs may have the potential of computing Steiner trees (e.g. [13]–[15]). To the best of our knowledge, no work has been done to reveal this potential from a theoretical perspective to date [16]. In this paper, we first reveal this potential from a theoretical perspective, and then exploit this potential by proposing a new PA for solving NWPTSTP.

In summary, our major contributions are listed as follows:

- We formulate NWPTSTP for minimum-cost relay node placement in two-tiered WSNs. The objective is to minimize the sum of heterogeneous costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree simultaneously.
- We conduct some theoretical analyses on the solvability of NWPTSTP, which is equivalent to the feasibility of two-tiered WSNs for a given set of devices. We further prove that NWPTSTP can be transformed to the Node-Weighted Steiner Tree Problem (NWSTP) [17].
- We conduct some theoretical analyses on PAs to reveal their potential of computing Steiner trees.
- We propose a new and hybrid PA to solve NWPTSTP by incorporating Steiner tree techniques into the Physarum-inspired optimization process.

Ultimately, we compare our PA with the state-of-the-art Steiner tree and relay node placement algorithms to demonstrate its usefulness for designing cheap WSNs with a high QoS in budget-limited scenarios.

II. Related Work

A. Relay Node Placement in Wireless Sensor Networks

The traditional relay node placement approaches are unconstrained, i.e., relay nodes can be placed anywhere in a given

geometric space (e.g. [18]). These approaches do not permit the capture of possible geographical constraints, and thus may not be applicable in some cases. The recently developed constrained relay node placement approaches can meet this challenge. Misra *et al.* [2] first studied the single-tiered constrained relay node placement problem, while Yang *et al.* [3] first studied the two-tiered one. Both of them placed a minimum number of relay nodes to achieve the network connectivity and survivability. Later, Zheng *et al.* [19] studied the two-tiered constrained relay node placement problem in energy-harvesting WSNs. They placed a minimum number of relay nodes that are powered by green energy, and optimized resource allocation to achieve the network connectivity and a high QoS. Furthermore, Misra *et al.* [20] studied the single-tiered constrained relay node placement problem in energy-harvesting WSNs. They placed a minimum number of relay nodes to achieve the network connectivity and survivability, while ensuring that the deployed relay nodes harvest a large amount of ambient energy. Kimençe and Bekmezci (2014) [21] extended the work on single-tiered constrained relay node placement by weighting all the deployment locations. They minimized the sum of such weights of all the deployed relay nodes. Most recently, Liu *et al.* [22] and Djenouri and Bagaa (2017) [23] further studied the single-tiered constrained relay node placement problem: Liu *et al.* placed a limited number of relay nodes to maximize the network lifetime; and Djenouri *et al.* placed a minimum number of relay nodes to help energy-rich sensor nodes relay data. Chelli *et al.* [4], Bagaa *et al.* [6], and Yuan *et al.* (2017) [5] further studied the two-tiered constrained relay node placement problem: Chelli *et al.* placed a minimum number of relay nodes to achieve the network connectivity; Bagaa *et al.* placed a minimum number of relay nodes to achieve the network connectivity, while minimizing the sum of outage probabilities of transmission routes in a routing tree; and Yuan *et al.* placed a limited number of relay nodes to maximize the network lifetime. Our work extends the previous work by minimizing the sum of heterogeneous costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree simultaneously.

B. Steiner Tree Problems in Graphs

Given an undirected graph with positive edge costs, the classical Steiner tree problem in graphs [24] is about finding the minimum-cost tree to connect compulsory vertices together. Many more complex Steiner tree problems in graphs have been developed based on it, including the Partial Terminal Steiner Tree Problem (PTSTP) [25], where some special compulsory vertices must be leaves of Steiner trees. Since sensor nodes cannot relay data and thus are leaves of a routing tree, PTSTP is in our interest. There is no node weight in PTSTP to represent the heterogeneous costs of relay nodes. This motivates us to later formulate, analyze, and solve NWPTSTP. On the other hand, node weights have already been introduced into some other Steiner tree problems in graphs, including Klein and Ravi's [17] Node-Weighted Steiner Tree Problem (NWSTP), which aims to minimize the sum of nonnegative node weights and edge costs in Steiner trees.

Klein and Ravi (1995) [17] proposed the first approximation algorithm for solving this problem, and it has an approximation guarantee of 2 $ln|T|$, where $|T|$ is the number of compulsory vertices. This is close to the best known lower bound of the approximation guarantee $(1 - o(1))ln|T|$ [26]. Guha and Khuller (1999) [26] later improved Klein and Ravi's algorithm, and the improvement has an approximation guarantee of $(1.35+\varepsilon)ln|T|$, for any constant $\varepsilon > 0$. Both Klein and Ravi's algorithm and Guha and Khuller's improvement are not fast in practice, as we need to repeatedly find many minimum weight matchings in each iteration [26]. Consequently, Guha and Khuller [26] further proposed a simple greedy algorithm that has an approximation guarantee of $1.6103ln|T|$. To our knowledge, this algorithm can be considered as a state-of-the-art one for solving this problem, since the more recent work focuses on solving this problem in special graphs, such as unit disk graphs (2009) [27] and planar graphs (2014) [28]. In our application, $|T|$ represents the number of base stations and sensor nodes, which means that $|T|$ may be large in some cases. As a result, the approximation guarantees above, such as $1.6103ln|T|$, although close to the best known lower bound, may be too large to guarantee a high solution quality. Therefore, it is still worth exploring new algorithms that can produce high-quality solutions. The emerging PAs may be such algorithms (e.g. [13]–[15]). To the best of our knowledge, no work has been done to reveal their potential of computing Steiner trees from a theoretical perspective [16]. This motivates us to later conduct some theoretical analyses on PAs. Based on these analyses, we propose a new PA that can produce high-quality solutions for our application.

III. Problem Formulation

We consider a WSN consisting of three types of devices: base stations (B), sensor nodes (S), and relay nodes (R). Base stations and sensor nodes are pre-deployed and stationary, while relay nodes are fixed at pre-determined candidate locations. We use r_B, r_S and r_R to denote the transmission ranges of base stations, sensor nodes and relay nodes respectively. Like the previous work [2], [3], we assume that there is only one relay node at each candidate location; $r_R > r_S > 0$; and r_B is much larger than r_R such that there is a transmission route between each pair of base stations. The transmission routes between devices have outage probabilities in practice. Like the previous work [6], we consider such outage probabilities for enhancing the QoS of WSNs.

We are interested in the two-tiered topology, where sensor nodes cannot relay data. This means that there is no transmission route between sensor nodes. We define the Cost-aware Communication Graph (CCG) as follows, where $d(i, j)$ is the Euclidean distance between devices i, j, and r_i, r_j are the transmission ranges of devices i, j respectively.

Definition 1 (The Cost-aware Communication Graph): The Cost-aware Communication Graph $CCG(V, E, w, c)$ is a connected 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 $i \in R$ to a nonnegative 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 nonnegative value $c(e)$ that equals its outage probability.

Given that relay nodes are fixed at pre-determined candidate locations, it may be easy to quantify their production and placement costs in practice. For example, we can consider their production costs as their selling prices on the market, and their placement costs as the prices of hiring people to install them at the pre-determined candidate locations. On the other hand, the outage probabilities of transmission routes can be calculated using the model developed by Bagaa *et al.* [6]. We assume that the outage probabilities of transmission routes between base stations are zero, which means that a packet received by a base station is considered received by all the base stations. Consequently, it is sufficient for each sensor node to connect at least one base station, and it is not necessary to make the designed WSN connected. Nevertheless, without loss of generality, we assume that the designed WSN is always connected, as we can easily make a disconnected WSN connected using the zero-outage-probability transmission routes between base stations. For the same reason, we also assume that the designed WSN contains all the base stations; and there is a routing tree that spans all the base stations and sensor nodes.

We focus on scenarios where sensor nodes are often far away from base stations, which means that there are often a large number of relay nodes between sensor nodes and base stations. It is preferable to minimize the sum of costs of these relay nodes for designing cheap WSNs. We consider the costs of relay nodes to be the sums of their production and placement costs. We assume that the outage probabilities of transmission routes are small values. In this case, the outage probability of a routing path between a sensor node and a base station can be approximated as the sum of outage probabilities of transmission routes in this path, i.e., $1-\prod_{e\in E_P}(1-c(e)) \approx \sum_{e\in E_P} c(e)$, where E_P is the set of transmission routes in this path. Therefore, by minimizing the sum of costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree that contains a routing path between each sensor node and a base station, we can approximately find low-cost relay nodes that enable low-outage-probability routing paths between sensor nodes and base stations, and thus design cheap WSNs with a high QoS. Specifically, we define the Minimum-Cost Relay Node Placement Problem as follows.

Problem 1 (The Minimum-Cost Relay Node Placement Problem): Given a $CCG(V, E, w, c)$ such that $V = B \cup S \cup R$, the Minimum-Cost Relay Node Placement Problem is to deploy some relay nodes $R' \in R$ in such a way that: 1) there is a routing path between each sensor node and a base station in a routing tree $CCG'(V', E'), V' = B \cup S \cup R'$, $E' \subseteq E$, and no sensor node is in the middle of this path; and 2) $\sum_{v\in R'} w(v) + \sum_{e\in E'} c(e)$ is minimized.

The first condition above guarantees that CCG' is a feasible routing tree, while the second condition above guarantees that the sum of costs of relay nodes and the sum of outage probabilities of transmission routes in the routing tree are minimized simultaneously. Since sensor nodes cannot relay data, all the sensor nodes are leaves of the routing tree, otherwise we can remove an adjacent edge of a non-leaf sensor node to reduce $\sum_{e\in E'} c(e)$, while still meeting the routing and network connectivity requirements, as no route between a sensor node and a base station is jeopardized by this removal; and all the base stations are connected to each other through zero-outage-probability transmission routes. To find CCG', we formulate the new Node-Weighted Partial Terminal Steiner Tree Problem (NWPTSTP) as follows.

Problem 2 (The Node-Weighted Partial Terminal Steiner Tree Problem): Let $G(V, E, T, T_L, w, c)$ be a connected undirected graph, where V is the set of vertices, E is the set of edges, T is a subset of V that we refer to as compulsory vertices, T_L is a subset of T that we refer to as compulsory leaf vertices, w is a function which maps each vertex in V to a nonnegative value that we refer to as node weight, and c is a function which maps each edge in E to a nonnegative value that we refer to as edge cost. The purpose is to find a tree $G'(V', E'), T_L \subseteq T \subseteq V' \subseteq V$, $E' \subseteq E$ with the minimum net-cost $c(G') = \sum_{v\in V'} w(v) + \sum_{e\in E'} c(e)$, and all the compulsory leaf vertices are leaves of this tree.

G' becomes CCG' when $V = B \cup S \cup R$; $T = B \cup S$; $T_L = S$. The leaf constraints of T_L guarantee that G' is a feasible routing tree, i.e., there is a routing path between each sensor node and a base station in G', and no sensor node is in the middle of this path. Minimizing $\sum_{v\in V'} w(v)+\sum_{e\in E'} c(e)$ further guarantees that G' has a low cost, and enables a high QoS, as discussed above. Our relay node placement approach is more cost-aware than the existing ones in that we consider the heterogeneous costs of relay nodes (e.g. Figure 1). Since the outage probabilities of transmission routes are unit-less values between 0 and 1, it may be preferable to normalize the costs of relay nodes to unit-less values between 0 and 1, and then introduce a regulating weight into the objective function, i.e., to minimize $\alpha \sum_{v\in V'} w_n(v) + \sum_{e\in E'} c(e)$, where α is a positive constant, and w_n is the normalized node weight function. We can do this by solving NWPTSTP in $G(V, E, T, T_L, \alpha w_n, c)$. A large α weights the costs of relay nodes more than the outage probabilities of transmission routes. Therefore, a large α helps us to minimize the costs of WSNs, and a small α helps us to enhance the QoS of WSNs. We will later show the trade-off via computational trials.

IV. Some Theoretical Analyses on NWPTSTP

In this section, we conduct some theoretical analyses on NWPTSTP. We focus on connected undirected graphs. For the sake of simplicity, we refer to a connected undirected graph G as a graph G throughout the following content.

A. The NP-Completeness

Here, we prove the NP-completeness of the decision version of NWPTSTP: given a graph G and a nonnegative value k, is these a solution G' such that $c(G') \leq k$?

Theorem 1: The decision version of NWPTSTP is NP-complete.

Proof: Since a given solution to NWPTSTP can be verified in polynomial time, the decision version of NWPTSTP is in NP. When $T_L = w = \emptyset$ and $c(e) \in \mathbb{R}_{>0} \mid \forall e \in E$, the decision version of NWPTSTP is equivalent to the decision version

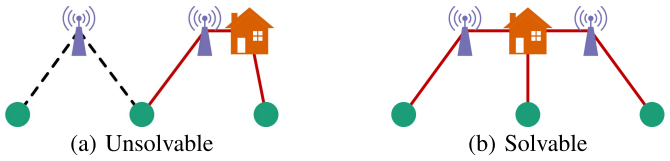


Fig. 3. Two relay node placement examples: placing relay nodes (purple) to connect sensor nodes (green) with a base station (orange). The red solid lines and black dashed lines represent the used and unused transmission routes respectively. In (a), a relay node is disconnected with the base station, as it is far away from the other relay node, and the middle sensor node cannot relay data; as a result, the left sensor node is disconnected. In (b), two relay nodes are connected with the base station; as a result, all the sensor node are connected.

of the classical Steiner tree problem in graphs [24], which is among Karp's original 21 NP-complete problems [29]. Hence, this theorem holds. □

B. The Solvability of NWPTSTP

Two-tiered WSNs may not be available when there are not enough relay nodes between sensor nodes and base stations (e.g. Figure 3). The feasibility of two-tiered WSNs for a given set of devices is equivalent to the solvability of NWPTSTP, i.e., whether feasible solutions exist or not for a given graph. Here, we conduct some theoretical analyses on this solvability.

Theorem 2: There is a feasible solution to NWPTSTP in a graph $G(V, E, T, T_L, w, c)$ if and only if at least one of the following four conditions is met: 1) $|T_L| = 0$; 2) $|T| = |T_L| = 1$; 3) $|T| = |T_L| = 2$, and $(i, j) \in E$ for $i, j \in T_L$; 4) there is a vertex $i \in V \setminus T_L$ such that i is connected with every compulsory vertex $j \in T$ in graph $G \setminus (T_L \setminus j)$.

Proof: Given that G is connected, there is a tree that contains all the compulsory vertices, such as the Minimum Spanning Tree of G. If condition 1 is met, then this tree is a feasible solution. If condition 2 is met, then the single vertex $\{i\}$ for $i \in T_L$ is a feasible solution. If condition 3 is met, then the single edge $\{(i, j)\}$ for $i, j \in T_L$ is a feasible solution. If condition 4 is met, then there is a vertex $i \in V \setminus T_L$ and a path $P(i, j)$ for all $j \in T$ such that no compulsory leaf vertex is in the middle of this path, and a spanning tree of $\cup P(i, j) | \forall j \in T$ is a feasible solution. Thus, if at least one of the four conditions is met, then there is a feasible solution. Subsequently, suppose that there is a feasible solution, but none of the four conditions is met. Since the first three conditions are not met, there are three possible scenarios: 1) $|T| = 2$, $|T_L| = 1$; 2) $|T| = |T_L| = 2$, and $(i, j) \notin E$ for $i, j \in T_L$; and 3) $|T| > 2$, $|T_L| \geq 1$. Suppose that $\Theta(V', E')$ is a feasible solution. In each scenario, there is a vertex $i \in V' \setminus T_L$ in Θ such that i is disconnected with a compulsory vertex $j \in T$ in $\Theta \setminus (T_L \setminus j)$, which is not possible. Therefore, if there is a feasible solution, then at least one of the four conditions is met. Hence, this theorem holds. □

In our application, there is at least one base station, i.e., $|T \setminus T_L| \geq 1$, and there is at least one sensor node, i.e., $|T_L| \geq 1$. As a result, only the fourth condition in Theorem 2 can be met. Since we need to check every i ($O(|V|)$), every j ($O(|T|)$) and the connectivity of $G \setminus (T_L \setminus j)$ ($O(|V| + |E|)$ [30]), the time complexity of implementing Theorem 2 to check the solvability of NWPTSTP for general graphs is $O(|T||V|^2 + |T||V||E|)$. It is slow to do this for large graphs. Thus, we further prove that there is a faster way to identify solvable CCGs for our application.

Theorem 3: Given a $CCG(V, E, T, T_L, w, c)$ such that $V = B \cup S \cup R$; $T = B \cup S$; $T_L = S$, if $CCG \setminus T_L$ is connected, then there is a feasible solution to NWPTSTP.

Proof: Given that CCG is connected and edge $(i, j) \notin E$ for $i, j \in T_L$, if $CCG \setminus T_L$ is connected, then there is a vertex $i \in V \setminus T_L$ such that i is connected with every compulsory vertex $j \in T$ in $CCG \setminus (T_L \setminus j)$. As a result, the fourth condition in Theorem 2 is always met. This theorem holds. □

Since we need to check the connectivity of $CCG \setminus T_L$, the time complexity of implementing Theorem 3 to identify solvable CCGs is $O(|V| + |E|)$. Therefore, it is fast to implement Theorem 3 to identify solvable CCGs for our application. We will do this in our later computational trials for generating a large number of solvable relay node placement instances.

C. Removing the Leaf Constraints in NWPTSTP

Here, we prove that the leaf constraints of T_L in NWPTSTP can be removed by updating edge costs. First, we show that the leaf constraints increase solution costs via the following theorem, where we refer to the optimal solution to NWPTSTP as Terminal Steiner Minimum Tree (TSMT).

Theorem 4: If there are feasible solutions to NWPTSTP in a graph $G(V, E, T, T_L, w, c)$, then $c(\Theta_{opt}) \geq c(\Theta'_{opt})$, where Θ_{opt} and Θ'_{opt} are the TSMTs in $G(V, E, T, T_L, w, c)$ and $G'(V, E, T, \emptyset, w, c)$ respectively.

Given that a feasible solution to NWPTSTP is also a feasible solution to NWPTSTP ($T_L = \emptyset$), this theorem can be easily proven. It is used in the proof of the following theorem that proves that T_L can be removed by updating edge costs.

Theorem 5: If there are feasible solutions to NWPTSTP in graphs $G(V, E, T, T_L, w, c)$ and $G'(V, E, T, \emptyset, w, c')$ such that $c'(i, j) = c(i, j) + \tau M \mid \forall (i, j) \in E$, where $\tau = |T_L \cap \{i, j\}|$, $M \geq \sum_{v \in V} w(v) + \sum_{e \in E} c(e)$, then the TSMTs in G and G' share the same sets of vertices and edges.

Proof: Suppose that $\Theta_{opt}(V_{opt}, E_{opt})$, $\Theta'_{opt}(V'_{opt}, E'_{opt})$, and $\Theta''_{opt}(V''_{opt}, E''_{opt})$ are the TSMTs in $G(V, E, T, T_L, w, c)$, $G'(V, E, T, \emptyset, w, c')$, and $G''(V, E, T, T_L, w, c')$ respectively; and there is a compulsory leaf vertex $i \in T_L$ such that its degree $\delta(i) = x > 1$ in Θ'_{opt}. Then, $c(\Theta'_{opt}) \geq (|T_L| + x - 1)M + \sum_{e \in E'_{opt}} c(e) + \sum_{v \in V'_{opt}} w(v) > c(\Theta''_{opt}) = |T_L|M + \sum_{e \in E''_{opt}} c(e) + \sum_{v \in V''_{opt}} w(v)$, which is not possible (by Theorem 4). Therefore, $\delta(i) = 1$, i.e., Θ'_{opt} is also a feasible solution to NWPTSTP in G''. Since $c(\Theta'_{opt}) \leq c(\Theta''_{opt})$ (by Theorem 4), $c(\Theta'_{opt}) = c(\Theta''_{opt})$. Therefore, every tree that corresponds to Θ'_{opt} also corresponds to Θ''_{opt}, vice versa. Moreover, every tree that corresponds to a feasible solution to NWPTSTP in G also corresponds to a feasible solution to NWPTSTP in G'', vice versa. Given that $c'(i, j) = c(i, j) + \tau M$, these feasible solutions differ in cost by the constant $|T_L|M$. Thus, $c(\Theta''_{opt}) = c(\Theta_{opt}) + |T_L|M$, and every tree that corresponds to Θ''_{opt} also corresponds to Θ_{opt},

vice versa. As a result, every tree that corresponds to Θ'_{opt} also corresponds to Θ_{opt}, vice versa. Hence, this theorem holds. □

This theorem shows that T_L can be removed. NWPTSTP ($T_L = \emptyset$) is equivalent to Klein and Ravi's NWSTP [17]. We will compare a state-of-the-art approximation algorithm for solving NWSTP in our later computational trials.

V. Some Theoretical Analyses on PAs

Physarum-inspired Algorithms (PAs) have the potential of computing Steiner trees. However, to the best of our knowledge, no work has been done to reveal this potential from a theoretical perspective to date. In this section, we first introduce Physarum Solver (PS) [31], which is a well-known PA for solving the shortest path problem, i.e., finding the shortest path between two terminals in an undirected graph. PS is the basis of most PAs (e.g. [13]–[15]). After introducing PS, we conduct some theoretical analyses on these PAs to reveal their potential of computing Steiner trees.

In a biological experiment [8], Physarum polycephalum demonstrated the ability of finding the shortest path between two agar blocks in a maze. This ability is attributed to an underlying physiological mechanism: the tube of Physarum polycephalum thickens as the protoplasmic (or nutritional) flux through it increases. PS is inspired by this mechanism.

In PS, the graph is considered as a network with a value associated with each edge modeling the protoplasmic flux in this edge. The two terminals in the shortest path problem represent two agar blocks containing nutrient, which are food for Physarum polycephalum.

Like Physarum polycephalum, PS finds the shortest path between these two terminals by maximizing the protoplasmic flux in this path. One terminal is called the source node, and the other terminal is called the sink node. The protoplasmic flux flows into the graph from the source node and out of the graph from the sink node (note that, for the real Physarum polycephalum, this flux does not flow out of the body; instead, it periodically reverses the flow direction between two terminals as a result of shuttle streaming [32], [33]). There is pressure at each vertex, and the quantity of flux in each edge is proportional to the pressure difference between two ends of this edge. Specifically, the flux Q_{ij} in edge (i,j) is given by the following Hagen-Poiseuille equation.

$$Q_{ij} = \frac{D_{ij}}{c_{ij}}(p_i - p_j) \tag{1}$$

$$D_{ij} = \frac{\pi r_{ij}^4}{8\xi} \tag{2}$$

where D_{ij} is the edge conductivity, c_{ij} is the edge length/cost, i.e., $c(i,j)$, p_i and p_j are pressures at vertices i and j, r_{ij} is the edge radius, and ξ is the viscosity coefficient. Equation (2) shows that the conductivity increases with the tube thickness, i.e., r_{ij}. Thus, the change of tube thickness can be described by the conductivity update equation as follows.

$$\frac{d}{dt}D_{ij} = \eta|Q_{ij}| - \mu D_{ij} \tag{3}$$

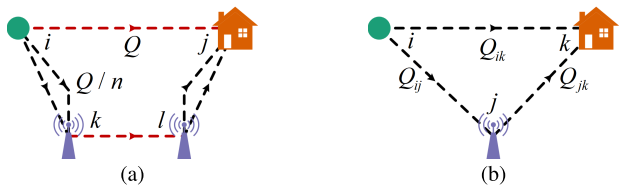


Fig. 4. Two Physarum optimization examples. The flux flows from the source node (green) to the sink node (orange), possibly through the other nodes (purple). In (a), the quantity of flux in (red) edges (i,j) and (k,l) is Q, while that in the other (black) edges is Q/n, where n is the number of disjoint paths between i,k or l,j, and $n > 1$. In (b), the quantities of flux in edges $(i,j),(i,k),(j,k)$ are Q_{ij}, Q_{ik}, Q_{jk} respectively.

where η and μ are two positive constants. The conductivity update equation implies that conductivities tend to increase in edges with large flux. Therefore, PS exploits the physiological mechanism that the tube of Physarum polycephalum thickens as the protoplasmic flux through it increases.

To calculate the flux and update edge conductivities, we first need to calculate the pressures. By considering the conservation law of flux at each vertex, the pressures can be calculated using the network Poisson equation below.

$$\sum_{i \in V(j)} \frac{D_{ij}}{c_{ij}}(p_i - p_j) = \begin{cases} -I_0, & j = source \\ I_0, & j = sink \\ 0, & otherwise \end{cases} \tag{4}$$

where $V(j)$ is the set of vertices linked to vertex j, I_0 is the quantity of flux flowing into the source node and out of the sink node.

Let the pressure at the sink node be 0, and give each edge conductivity an initial value, then the other pressures can be calculated using Equation (4). Subsequently, the flux in each edge can be calculated using Equation (1), and the conductivity of each edge can be updated using Equation (3). Clearly, the pressures (excluding the pressure at the sink node) and the fluxes will also change after the update of edge conductivities. Use ϵ to signify the threshold value of edge conductivity. Edges with conductivities smaller than ϵ will be cut from the network. Ultimately, if there is a unique shortest path between the source and sink nodes, then this path can be found by iteratively updating edge conductivities and cutting edges [34]. Most existing PAs are based on PS. We conduct some theoretical analyses on them to reveal their potential of computing Steiner trees as follows.

First, we observe that isolated subgraphs with no source or sink node inside may exist after cutting edges. For example, in Figure 4a, if $\eta = 1$; $\mu = 1$; and $Q > \epsilon > Q/n$, then all the edges except (i,j) and (k,l) will be cut, and (k,l) becomes such an isolated subgraph. Suppose that G_A, G_B are two connected subgraphs after cutting edges in an iteration, and they are isolated from each other. Without loss of generality, we assume that both the source and sink nodes are in G_A. Then, the network Poisson equation in the next iteration is

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} P_A \\ P_B \end{bmatrix} = \begin{bmatrix} C \\ 0 \end{bmatrix} \tag{5}$$

where A, B, C are non-zero sub-matrices, and P_A, P_B are the sets of pressures of vertices in G_A, G_B. Clearly, $P_B = 0$. Consequently, all the fluxes in G_B are 0 in the next iteration. Suppose (k, l) is an edge in G_B. Then, its conductivity in the next iteration is $(1-\mu)D^0_{kl}$, where μ is in Equation (3), D^0_{kl} is its conductivity in this iteration. If $\mu = 1$, then edge (k, l) will be cut. Thus, the following conclusion can be made.

Conclusion 1: Isolated subgraphs with no source or sink node inside may exist in an iteration of PAs. If $\mu = 1$, then they will disappear in the next iteration.

This conclusion indicates the preference of setting $\mu = 1$, which has been observed in the previous computational trials of employing PAs to compute shortest paths [35].

We next analyze the fluxes between the source and sink nodes. Suppose that there is a triangular graph with three vertices i, j, k (see Figure 4b), where vertices i, k are respectively the source and sink nodes. We have the following equation.

$$Q_{ij} = \frac{D_{ij}}{c_{ij}}(p_i - p_j) = Q_{jk} = \frac{D_{jk}}{c_{jk}}(p_j - p_k) \tag{6}$$

We define

$$A_{ij} : \frac{D_{ij}}{c_{ij}} \tag{7}$$

$$A_{ij} \otimes A_{jk} : \frac{A_{ij}A_{jk}}{A_{ij} + A_{jk}} = \frac{D_{ij}D_{jk}}{c_{ij}D_{jk} + c_{jk}D_{ij}} \tag{8}$$

The flux in path $\{(i, j), (j, k)\}$ is

$$\begin{aligned} Q = Q_{ij} = Q_{jk} &= \frac{A_{ij} + A_{jk}}{A_{ij} + A_{jk}}Q \\ &= \frac{1}{A_{ij} + A_{jk}}(A_{jk}Q_{ij} + A_{ij}Q_{jk}) \\ &= \frac{1}{A_{ij} + A_{jk}}[A_{ij}A_{jk}(p_i - p_j) + A_{ij}A_{jk}(p_j - p_k)] \\ &= \frac{A_{ij}A_{jk}}{A_{ij} + A_{jk}}(p_i - p_k) \\ &= A_{ij} \otimes A_{jk}(p_i - p_k) \end{aligned} \tag{9}$$

The flux in path $\{(i, k)\}$ is

$$Q_{ik} = \frac{D_{ik}}{c_{ik}}(p_i - p_k) \tag{10}$$

If

$$A_{ij} \otimes A_{jk} = \frac{D_{ij}D_{jk}}{c_{ij}D_{jk} + c_{jk}D_{ij}} = \frac{D_{ik}}{c_{ik}} \tag{11}$$

then the fluxes in these two path are equal. If all the initial edge conductivities are equal and $c_{ij} + c_{jk} = c_{ik}$, i.e. both of these two paths are the shortest paths, then the fluxes in them will be equal after each iteration, and both of them will survive to the end. In a more general case where m disjoint paths between i, k exist, we assume that the vertices on path x are $i, v_1, v_2 \cdots, v_n, k$. It can be seen from Equation (9) that the flux in path x is

$$Q_{P_x} = A_{iv_1} \otimes A_{v_1v_2} \otimes \cdots \otimes A_{v_nk}(p_i - p_k) \tag{12}$$

If all the edge conductivities on this path are equal, then

$$Q_{P_x} = \frac{D_{P_x}}{c_{P_x}}(p_i - p_k) \tag{13}$$

where D_{P_x} is the edge conductivity on this path, c_{P_x} is the total cost of this path. If

$$\frac{D_{P_1}}{c_{P_1}} = \frac{D_{P_2}}{c_{P_2}} = \cdots = \frac{D_{P_m}}{c_{P_m}} \tag{14}$$

then the fluxes in these paths will always be equal. Consequently, all these paths will survive to the end. Thus, the following conclusion can be made.

Conclusion 2: The solutions of PAs may not be trees.

This conclusion indicates that, to compute Steiner trees, additional techniques are required to guarantee that the solutions of PAs are always trees. We will use a Minimum Spanning Tree technique in our later proposed PA to guarantee this.

Subsequently, we analyze the ability of PAs of computing shortest paths in an undirected graph. Bonifaci *et al.* [34] have provided the rigorous proof for this ability under the condition that $\eta = 1$ and $\mu = 1$ in Equation (3). Here, we prove it from a more general perspective in that η is not constrained to 1, and we indicate for the first time that multiple shortest paths may survive at the same time in PAs.

Suppose that there are two disjoint paths between the source node i and sink node k (note that, since a sub-path of a shortest path is itself a shortest path, the conclusions below also hold true for joint paths); their edge conductivities and costs are $D_{ik1}, D_{ik2}, c_{ik1}, c_{ik2}$ respectively. Since $p_k = 0$, we have

$$Q^1_{ik1} = \frac{D^1_{ik1}}{c_{ik1}}p^1_i \tag{15}$$

$$Q^1_{ik2} = \frac{D^1_{ik2}}{c_{ik2}}p^1_i \tag{16}$$

$$\frac{Q^1_{ik1}}{Q^1_{ik2}} = \frac{D^1_{ik1}}{D^1_{ik2}} \cdot \frac{c_{ik2}}{c_{ik1}} \tag{17}$$

where $Q^1_{ik1}, Q^1_{ik2}, D^1_{ik1}, D^1_{ik2}$ are respectively the fluxes and edge conductivities of paths $1, 2$ in the first iteration (the superscripts on Q and D correspond to the iteration number), p^1_i is the pressure at the source node i in the first iteration. Suppose that $\mu = 1$ in Equation (3). Since $Q_{ik} > 0$, we have

$$D^2_{ik1} = (1 + \frac{\eta p^1_i}{c_{ik1}} - \mu)D^1_{ik1} = \frac{\eta p^1_i}{c_{ik1}}D^1_{ik1} \tag{18}$$

$$D^2_{ik2} = (1 + \frac{\eta p^1_i}{c_{ik2}} - \mu)D^1_{ik2} = \frac{\eta p^1_i}{c_{ik2}}D^1_{ik2} \tag{19}$$

$$Q^2_{ik1} = \frac{D^2_{ik1}}{c_{ik1}}p^2_i = \frac{\eta D^1_{ik1}}{(c_{ik1})^2}p^1_i p^2_i \tag{20}$$

$$Q^2_{ik2} = \frac{D^2_{ik2}}{c_{ik2}}p^2_i = \frac{\eta D^1_{ik2}}{(c_{ik2})^2}p^1_i p^2_i \tag{21}$$

$$\frac{Q^2_{ik1}}{Q^2_{ik2}} = \frac{D^1_{ik1}}{D^1_{ik2}} \cdot (\frac{c_{ik2}}{c_{ik1}})^2 \tag{22}$$

Similarly, we conclude that in general

$$\frac{Q^m_{ik1}}{Q^m_{ik2}} = \frac{D^1_{ik1}}{D^1_{ik2}} \cdot (\frac{c_{ik2}}{c_{ik1}})^m \tag{23}$$

If there are multiple paths between i, k, then the fluxes in paths x, y in the mth iteration meet the following condition.

$$\frac{Q^m_{ikx}}{Q^m_{iky}} = \frac{D^1_{ikx}}{D^1_{iky}} \cdot (\frac{c_{iky}}{c_{ikx}})^m \tag{24}$$

where Q^m_{ikx}, Q^m_{iky} are the fluxes of paths x, y in the mth iteration, c_{ikx}, c_{iky} are the costs of paths x, y. If $c_{ikx} < c_{iky}$ for any path y ($y \neq x$), then

$$\lim_{m\to\infty} \frac{Q^m_{ikx}}{Q^m_{iky}} = \infty \tag{25}$$

which means that the shortest path x will survive to the end, and all the other paths will be cut. This proves that PAs can compute shortest paths. However, if $c_{ikx} = c_{iky}$, then

$$\lim_{m\to\infty} \frac{Q^m_{ikx}}{Q^m_{iky}} = \frac{D^1_{ikx}}{D^1_{iky}} \tag{26}$$

which means that the ratio of their fluxes will always equal that of their initial edge conductivities. Hence, multiple shortest paths may survive at the same time in PAs. Therefore, the following conclusion can be made.

Conclusion 3: PAs can find the shortest path, and may find multiple shortest paths at the same time.

This conclusion indicates that PAs may have the potential to compute Steiner trees, as Steiner trees are short or low-cost networks. Even though several PAs have already been applied to challenge some Steiner tree problems in graphs (e.g. [13]–[15]), there is currently no rigorous proof that compulsory vertices will always be connected in the solutions of PAs. The theorem below fills this gap by proving that PAs can find a subgraph connecting multiple compulsory vertices together by selecting them to be source and sink nodes.

Theorem 6: Let $G(V, E, c)$ be a connected undirected graph with multiple source nodes and a single sink node. Assume that every source node has an identical flux $I_0 > 0$. Consider a PA acting on G satisfying Equation (3), and with a threshold value ϵ for cutting edges. If $\mu = 1$ and $\epsilon \leq \frac{4\eta I_0}{|V|^2}$, then the source and sink nodes will always remain connected under the action of this PA.

Proof: Consider any partition of G into two subgraphs G_A and G_B such that G_A contains at least one source node and G_B contains the sink node and the other source node(s). Let x be the number of source nodes in G_A and let s be the number of vertices in G_A. We will show that the action of a PA satisfying the conditions of this theorem cannot cut all the edges directly connecting G_A and G_B. Let the number of edges directly connecting G_A and G_B be y. We observe that $y \leq s(|V| - s)$. The net flux from G_A to G_B is xI_0. There is an edge between G_A and G_B with flux at least $\frac{xI_0}{y} \geq \frac{xI_0}{s(|V|-s)}$. Let $g(x, s) = \frac{xI_0}{s(|V|-s)}$. Clearly, the minimum possible value of g occurs when $x = 1$ and $s = |V|/2$, in which case $g(x, s) = \frac{4\ I_0}{|V|^2}$. Let $\mu = 1$ in Equation (3), then $D_{ij}(k+1) = \eta|Q_{ij}(k)|$, where k is the number of conductivity update times, and $k \geq 1$. If $\epsilon \leq \frac{4\eta I_0}{|V|^2}$, then there is at least one edge directly connecting G_A and G_B where the conductivity is not smaller than the threshold value ϵ. Hence, this theorem holds. □

We next show that PAs tend to find low-cost networks to connect source and sink nodes together. Suppose that there are n disjoint paths between a pair of vertices i, j, and the initial edge conductivities on these paths are equal. We have

$$Q^1_x = \frac{D^1_x}{c_x}(p^1_i - p^1_j) \tag{27}$$

$$\sum_{x=1}^{n} Q^1_x = I_1 \tag{28}$$

where Q^1_x, D^1_x are respectively the flux and edge conductivity of path x in the first iteration, c_x is the cost of path x, I_1 is the flux flowing from i to j in the first iteration. Then, the flux in path x in the first iteration is

$$\begin{aligned} Q^1_x &= \frac{\frac{D^1_x}{c_x}}{\frac{D^1_1}{c_1} + \cdots + \frac{D^1_n}{c_n}} I_1 \\ &= \frac{D^1_x \prod_{k\neq x} c_k}{D^1_1 \prod_{k\neq 1} c_k + \cdots + D^1_n \prod_{k\neq n} c_k} I_1 \\ &= \frac{\prod_{k\neq x} c_k}{\prod_{k\neq 1} c_k + \cdots + \prod_{k\neq n} c_k} I_1 \end{aligned} \tag{29}$$

Suppose that $\mu = 1$ in Equation (3). Then,

$$D^2_x = \eta Q^1_x = \eta \frac{\prod_{k\neq x} c_k}{\prod_{k\neq 1} c_k + \cdots + \prod_{k\neq n} c_k} I_1 \tag{30}$$

$$Q^2_x = \frac{\frac{D^2_x}{c_x}}{\frac{D^2_1}{c_1} + \cdots + \frac{D^2_n}{c_n}} I_2 = \frac{(\prod_{k\neq x} c_k)^2}{(\prod_{k\neq 1} c_k)^2 + \cdots + (\prod_{k\neq n} c_k)^2} I_2 \tag{31}$$

Similarly, we have

$$Q^m_x = \frac{(\prod_{k\neq x} c_k)^m}{(\prod_{k\neq 1} c_k)^m + \cdots + (\prod_{k\neq n} c_k)^m} I_m \tag{32}$$

If $c_x = \min(c_k)_{k=1,\cdots,n}$, which means that path x is the shortest path between vertices i, j, then

$$\prod_{k\neq x} c_k = \max(\prod_{k\neq y} c_k)|_{y=1,\cdots,n} \tag{33}$$

Hence,

$$\lim_{m\to\infty} Q^m_x = I_m \tag{34}$$

$$\lim_{m\to\infty} Q^m_k|_{k\neq x} = 0 \tag{35}$$

Thus, PAs tend to find the shortest path between each pair of vertices in the graph (notably, this result is different from that in Equation (25) in that, here, we analyze the flux between each pair of vertices, and we assume that all the initial edge conductivities are equal). Consequently, by selecting multiple compulsory vertices to be source and sink nodes, PAs tend to find a low-cost network to connect them. Hence, the following conclusion can be made.

Conclusion 4: PAs can find low-cost networks to connect multiple compulsory vertices together.

This conclusion shows that PAs have the potential to compute Steiner trees. Some PAs have already been applied to solve various Steiner tree problems in graphs (e.g. [13]–[15]). These PAs compute Steiner trees by iteratively calculating pressures, updating edge conductivities, and cutting edges in

Algorithm 1 The Proposed Physarum-Inspired Steiner Tree Algorithm (PSTA)

Input: A solvable $CCG(V, E, T, T_L, w, c)$, a parameter K
Output: A feasible solution $\Theta_{best}(V_{\Theta_{best}}, E_{\Theta_{best}})$

1: Update c using Theorem 5
2: $\Theta_{best}(V_{\Theta_{best}}, E_{\Theta_{best}}) = MST(CCG(V, E, c))$
3: **for** $k = 1 : K$ **do**
4: Randomize pressures
5: Calculate fluxes using Equation (36)
6: Construct $CCG'(V, E, c')$ using Equation (38)
7: Update c' using Theorem 5
8: $\Theta(V_\Theta, E_\Theta) = MST(CCG'(V, E, c'))$
9: Prune non-compulsory leaves in Θ
10: $\Theta(V_\Theta, E_\Theta) = MST(\Theta(V_\Theta, E_\Theta, c))$
11: Prune non-compulsory leaves in Θ
12: **if** $c(\Theta_{best}) > c(\Theta)$ **then**
13: $\Theta_{best} = \Theta$
14: **end if**
15: **end for**
16: Return $\Theta_{best}(V_{\Theta_{best}}, E_{\Theta_{best}})$

the same way as PS. We observe that such iterations are computationally too expensive for large graphs. Moreover, the quality of their solutions may not be high enough either. In the next section, we propose a hybrid PA to overcome these weaknesses by incorporating Steiner tree techniques into the Physarum-inspired optimization process.

VI. The Proposed PA for Solving NWPTSTP

In this section, we propose the Physarum-inspired Steiner Tree Algorithm (PSTA) for solving NWPTSTP. Suppose that there is a solvable $CCG(V, E, T, T_L, w, c)$ identified using Theorem 3. First, we update c using Theorem 5 (Step 1). Then, we initialize the best produced solution Θ_{best} to be the Minimum Spanning Tree (MST) of CCG (Step 2). Given that there is a spanning tree in CCG that meets the constraints of T_L; and c has been updated using Theorem 5, this initial Θ_{best} is a feasible solution. Subsequently, we produce better solutions by iteratively pruning low-cost spanning trees in K loops (Steps 3-15). The reason why we do not iteratively update edge conductivities and delete edges to form a Steiner tree like traditional PAs is that the solutions produced by doing this may not be trees (see Conclusion 2).

In each loop, we associate each node with a random pressure value (Step 4). The reason why we do not solve the network Poisson equation, i.e., Equation (4), to calculate pressures like traditional PAs is that it is computationally too expensive to do this (specifically, the smallest time complexity of matrix multiplication and inversion is $O(|V|^{2.373})$ [36]). After randomizing pressures, we modify the Hagen-Poiseuille equation, i.e., Equation (1), to consider the existence of node weights in NWPTSTP as follows.

$$Q_{ij} = \frac{D_{ij}}{C_{ij}}(p_i - p_j) \tag{36}$$

$$C_{ij} = c(i, j) + w(i) + w(j) \tag{37}$$

where D_{ij} is constant for all edges. The above equations ensure that there are small fluxes in the adjacent edges of non-compulsory vertices with large node weights. We use Equation (36) to calculate fluxes in each edge (Step 5). To form a low-cost spanning tree, we first construct a new graph $CCG'(V, E, c')$ that has different edge costs with CCG as follows (Step 6).

$$c'(i, j) = \frac{1}{|Q_{ij}|} \tag{38}$$

By doing this, edges with large fluxes have small costs in CCG'. Subsequently, we update c' using Theorem 5 (Step 7), and find the MST of CCG': Θ (Step 8). Since the adjacent edges of non-compulsory vertices with large node weights tend to have small fluxes and large edge costs in CCG', these vertices are more likely to be leaves of Θ. Consequently, we may easily remove these vertices from Θ by pruning non-compulsory leaves (Step 9). Notably, Θ is not the MST for c. Thus, we further update Θ to be the MST of the pruned tree for c (Step 10), and then prune non-compulsory leaves again (Step 11). Since a subtree in an MST is also an MST [37], there is no need to find the MST of Θ anymore. We use Θ to update Θ_{best} (Steps 12-14). After K loops, PSTA returns Θ_{best} as the final solution (Step 16). PSTA has a time complexity of $O(K|E| + K|V|log|V|)$. The rigorous proof of this time complexity is in the supplement. Since Θ_{best} is the MST of CCG when $K = 0$, there is no approximation guarantee for PSTA. Nevertheless, we will later show that PSTA can produce high-quality solutions in practice.

VII. Computational Analyses

Here, we evaluate our methodology via computational trials that are conducted on a personal computer (Intel Xeon E5-1650 CPU). All the reported values have been averaged over 300 trials and are associated with their 95% Confidence Intervals. The codes and datasets are available at [38].

Algorithms: We implement four algorithms:

1) PSTA: It is our PA for solving NWPTSTP.

2) GKA (Guha and Khuller's Algorithm) [26]: It is Guha and Khuller's $1.6103ln|T|$ approximation algorithm for solving NWSTP. It can solve NWPTSTP by removing the leaf constraints via Theorem 5 (notably, GKA incorporates Klein and Ravi's method [17] of finding minimum-ratio spiders to concatenate compulsory vertices, and it can be seen that the large M values in Theorem 5 guarantees that the leaf constraints are met in such concatenations).

3) RRPL [6]: It is a recently proposed relay node placement algorithm. It aims to place a minimum number of relay nodes to achieve the network connectivity, while minimizing the sum of outage probabilities of transmission routes in a routing tree.

4) OSRP [4]: It is another recently proposed relay node placement algorithm. It aims to place a minimum number of relay nodes to achieve the network connectivity. Both RRPL and OSRP incorporate an approximation algorithm for solving the classical Steiner tree problem in graphs [24]. The one incorporated into RRPL is Kou's algorithm [39], while the one incorporated into OSRP has not been specified. Here, we also incorporate Kou's algorithm into OSRP.

Parameters: We vary six parameters:

1) α: It is the regulating weight between the costs of relay nodes and the outage probabilities of transmission routes (details in Section III).

2) $|B|$: It is the number of pre-deployed base stations.

3) $|S|$: It is the number of pre-deployed sensor nodes.

4) $|R|$: It is the number of candidate relay nodes.

5) γ_{th}: It is the receiver sensitivity threshold. The transmission ranges of devices decrease with γ_{th} (details in [6]).

6) M_r: It is the maximum number of retransmissions. A large M_r induces large success probabilities of transmissions between devices, at the cost of long delays (details in [6]).

Metrics: We apply six metrics:

1) $c(G')$: It is the objective function cost of NWPTSTP.

2) t_{run}: It is the running time of algorithms.

3) $cost_{relay}$: It is the sum of production and placement costs of deployed relay nodes.

4) t_{net}: It is the network lifetime. We assume that base stations and relay nodes are energy-sufficient. Consequently, we consider t_{net} to be the average number of times each sensor node can transmit useful information (unit: cycle), which depends on the average energy efficiency of sensor nodes [6]. The reported values of t_{net} are proportional to the assumed number of times each sensor node can transmit data, which has not been specified in [6]. We assume that each sensor node has enough energy to transmit data 10^7 times, i.e., if the outage probabilities of transmission routes are zero, then each sensor node can transmit useful information 10^7 times. Therefore, as the simplified expression of energy efficiency in [6] suggested, we consider $t_{net} = 10^7 \sum_{i \in S}[1 - c(e_i)]/|S|$, where $c(e_i)$ is the outage probability of the transmission route between sensor node i and the adjacent relay node or base station in the transmission paths from sensor nodes to base stations. We assume that sensor nodes transmit data to base stations via the minimum-sum-of-outage-probability paths, i.e., the sums of outage probabilities of transmission routes in these paths are minimized. These paths can be found in polynomial time using Dijkstra's algorithm. Furthermore, in the supplement, we show that the computational conclusions in this paper also hold for a widely-used shortest distance routing strategy (e.g. [40]).

5) t_{delay}: It is the average delay for the transmissions from sensor nodes to base stations, i.e., the average sojourn time of packets in buffers (details in [6]). The reported values of t_{delay} are proportional to the frame duration, which has not been specified in [6]. We set the frame duration to 30 ms.

6) $g_{\%}$: It is the average goodput for the transmissions from sensor nodes to base stations, i.e., the average percentage of useful information received by base stations (details in [6]).

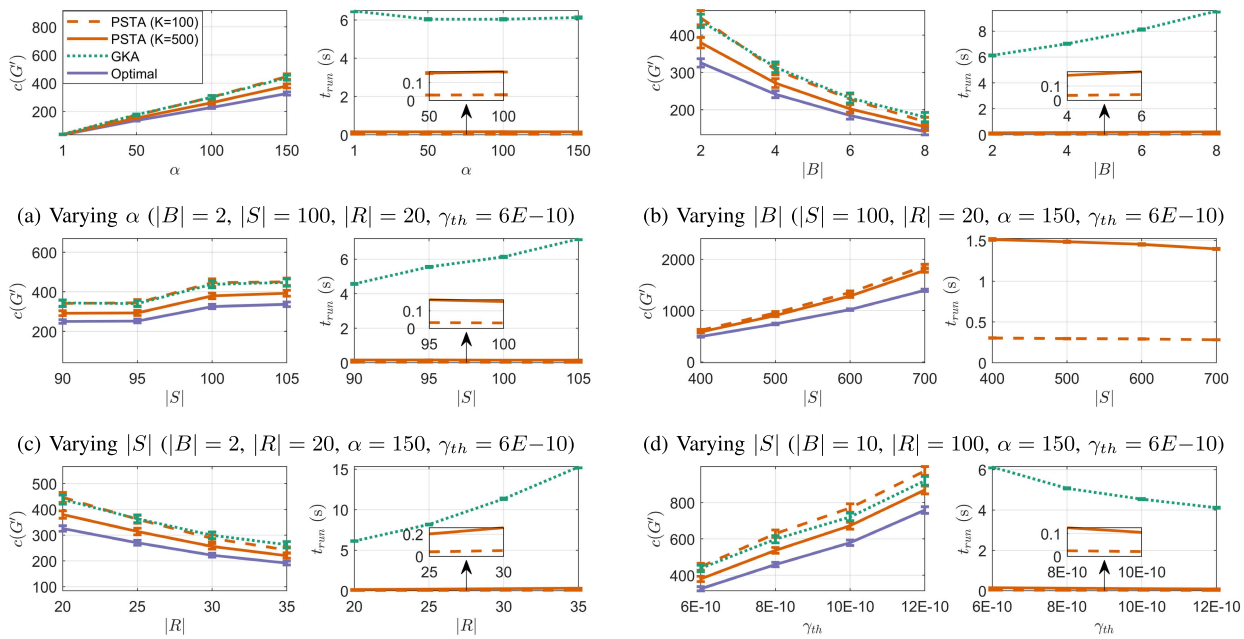
Benchmarks: Like the previous work (e.g. [2], [3], [6]), we generate a large number of relay node placement instances with different graph features as benchmarks. First, we randomly place all the devices in a square region to guarantee that they are non-identically distributed. By doing this, we can apply the model of Bagaa *et al.* [6] to calculate the outage probabilities of transmission routes. The transmission range of a device is proportional to $\sqrt{P/\gamma_{th}}$, where P is the transmission power [6]. We assume that the transmission powers of relay nodes are four times larger than those of sensor nodes, which means that $r_R = 2r_S$, as in [2], [3]; and the transmission powers of base stations are such larger than those of relay nodes that there is a transmission route between each pair of base stations, as in [2], [3]. We vary γ_{th} in a similar range as that in [6], and set the transmission powers of sensor and relay nodes in such a way that r_R and r_S are large enough to easily generate solvable instances in the square region above (notably, the theoretical analyses on the solvability of NWPTSTP, which is equivalent to the feasibility of two-tiered WSNs for a given set of devices, are in Section IV-B). We associate each transmission route e with an outage probability value $c(e)$ that is calculated using the model of Bagaa *et al.* [6]. We associate each relay node i with a random cost value $w(i)$ in the range of $[100, 500]$ (unit: dollar) to represent the sum of its production and placement costs. Then, we normalize $w(i)$ to a unit-less value $w_n(i) = (w(i) - 100)/400$ for bring the costs of relay nodes on the same scale with the outage probabilities of transmission routes. Since base stations and sensor nodes are compulsory vertices, we assume that each base station or sensor node i is associated with a normalized value $w_n(i) = 0$ for guaranteeing that their costs are not reflected in the objective function cost of NWPTSTP. Subsequently, we build a graph $G(V, E, T, T_L, \alpha w_n, c)$ as a benchmark instance (details in Section III).

A. Evaluating the Steiner Tree Solutions

Here, we evaluate the Steiner tree solutions in Figure 5. For a graph $G(V, E, T, T_L, \alpha w_n, c)$, $c(G') = \sum_{v \in V'} \alpha w_n(v) + \sum_{e \in E'} c(e)$. Given that M_r does not affect $c(G')$ and t_{run} (details in [6]), we do not vary M_r here.

Varying α: We vary α in Figure 5a. We observe that GKA finds similar solutions with PSTA ($K = 100$), but worse solutions than PSTA ($K = 500$). We also observe that PSTA is faster than GKA. The reason is that PSTA has a smaller time complexity than GKA. Specifically, PSTA has a time complexity of $O(K|E| + K|V|log|V|)$, while GKA has a time complexity of $O(|T||V||E| + |T||V|^2 log|V|)$, where $|T| = |S| + |B|$ for our application (the rigorous proof for the time complexity of GKA is in the supplement). This indicates that PSTA may be able to produce better solutions than GKA in a smaller amount of time in some cases. Moreover, we observe that $c(G')$ increases with α, as node weights increase with α. For this reason, the gaps between the optimal solutions and the PSTA and GKA solutions increase with α. Nevertheless, we consider both the PSTA and GKA solutions as near-optimal for these instances, as their approximation ratios for these instances are generally below 1.5. Furthermore, GKA is slower when $\alpha = 1$. The reason is that node weights are small in this case. As a result, spiders with the minimum ratios in GKA are less likely to be 3+ spiders, which means that GKA becomes slower by finding more spiders and forests to concatenate compulsory vertices (details in [26]).

Varying $|B|$: We vary $|B|$ in Figure 5b. We observe that $c(G')$ decrease with $|B|$. The reason is that fewer relay nodes are required to achieve the network connectivity when $|B|$



(a) Varying α ($|B|=2$, $|S|=100$, $|R|=20$, $\gamma_{th}=6E{-}10$)

(b) Varying $|B|$ ($|S|=100$, $|R|=20$, $\alpha=150$, $\gamma_{th}=6E{-}10$)

(c) Varying $|S|$ ($|B|=2$, $|R|=20$, $\alpha=150$, $\gamma_{th}=6E{-}10$)

(d) Varying $|S|$ ($|B|=10$, $|R|=100$, $\alpha=150$, $\gamma_{th}=6E{-}10$)

(e) Varying $|R|$ ($|B|=2$, $|S|=100$, $\alpha=150$, $\gamma_{th}=6E{-}10$)

(f) Varying γ_{th} ($|B|=2$, $|S|=100$, $|R|=20$, $\alpha=150$)

Fig. 5. Evaluating the Steiner tree solutions of PSTA and GKA. The optimal solutions are produced for comparison using a state-of-the-art Steiner tree exact solver: SCIP-Jack [41], [42][1]. GKA is too slow to be implemented in (d).

is large. We also observe that t_{run} increases with $|B|$, as $|V|$ increases with $|B|$.

Varying $|S|$: We vary $|S|$ in Figure 5c. We observe that $c(G')$ increases with $|S|$, as more relay nodes are required to achieve the network connectivity when $|S|$ is large. Notably, when generating these instances, we let the size of the deployment square region be proportional to $|S|$. The reason to do this is that increasing $|S|$ without increasing the size of the deployment square region cannot reflect the intuitive fact that more relay nodes are required to connect more sensor nodes, as a small number of relay nodes can connect a large number of sensor nodes in a small region. These instances are sparser when $|S|$ is large, as $|B|$ and $|R|$ are fixed. As a result, t_{run} of PSTA does not increase with $|S|$ for these instances. In comparison, t_{run} of GKA increases significantly with $|S|$ for these instances, as $|S|+|B|=|T|$, and GKA has a time complexity of $O(|T||V||E|+|T||V|^2 log|V|)$. This indicates that PSTA can solve instances that are too large for GKA to solve, such as those in Figure 5d.

Varying $|R|$: We vary $|R|$ in Figure 5e. We observe that $c(G')$ decreases with $|R|$. The reason is that a large $|R|$ provides more and possibly cheaper solutions.

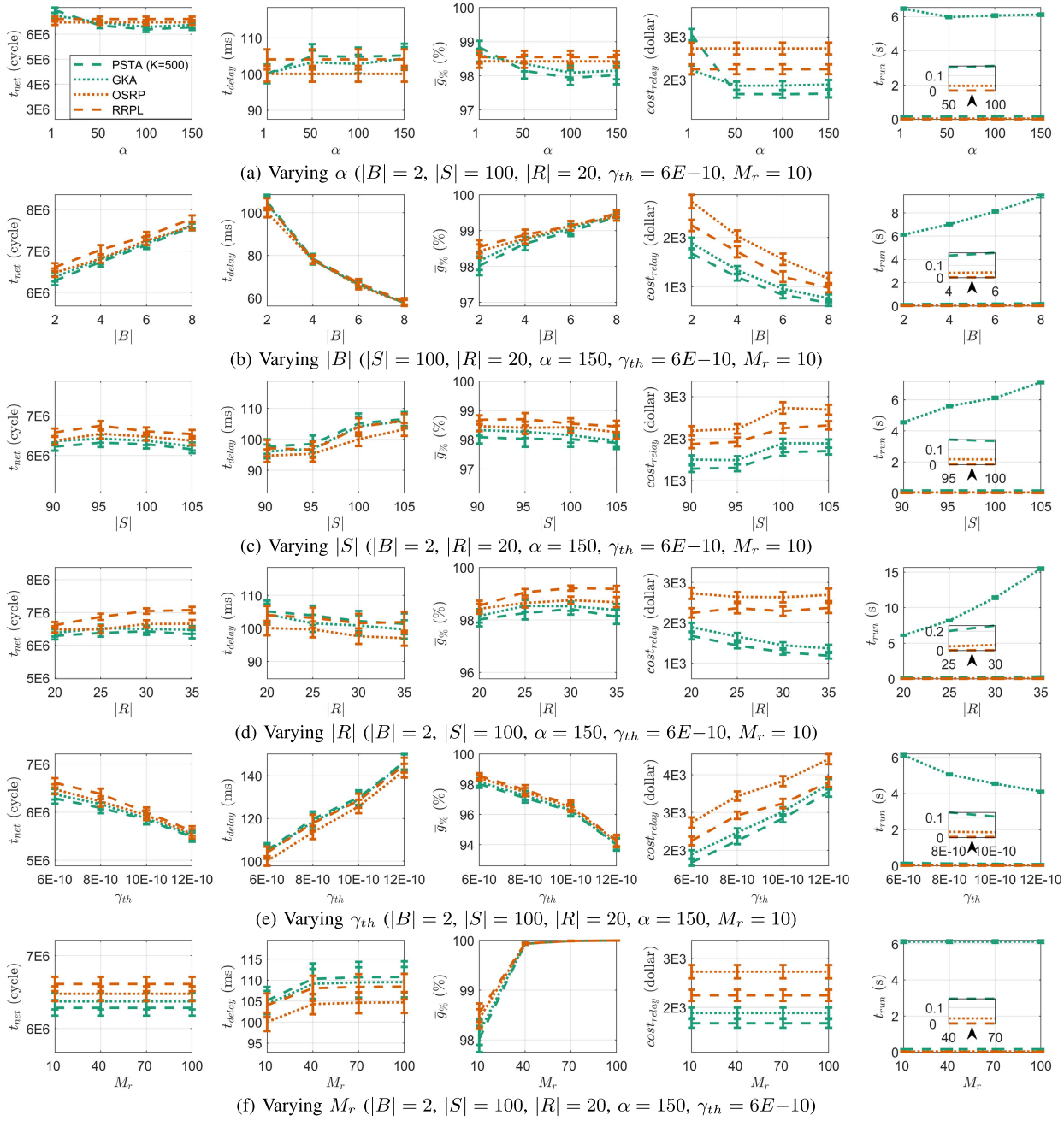
Varying γ_{th}: We vary γ_{th} in Figure 5f. We observe that $c(G')$ increases with γ_{th}. The reason is that the transmission ranges of devices decrease with γ_{th} (details in [6]), and as a result, more relay nodes are required to achieve the network connectivity when γ_{th} is large. For the same reason, t_{run} decreases with γ_{th}.

The Solution Quality of Our PSTA: The above computational results show that PSTA generally produces better solutions when K is larger. Specifically, in comparison to PSTA ($K=100$), PSTA ($K=500$) produces better solutions for 70.75% instances, same solutions for 17.92% instances, and worse solutions for 11.33% instances. The reason is that PSTA finds K low-cost solution trees by randomizing pressure values, and then return the lowest-cost solution tree. As a result, when K is larger, PSTA finds more low-cost solution trees, and is likely to return lower-cost solution trees. The above computational results also show that PSTA generally produces better solutions than GKA when K is large. Specifically, in comparison to GKA, PSTA ($K=500$) produces better solutions for 91.98% instances, same solutions for 0.02% instances, and worse solutions for 8.00% instances. We further show this by varying K between 1 and 500 in the supplement.

B. Evaluating the Designed Wireless Sensor Networks

Varying α: We vary α in Figure 6a. α does not affect OSRP and RRPL, as they do not consider the existence of node weights, i.e., the heterogeneous costs of relay nodes. In comparison, α affects PSTA and GKA, as the heterogeneous costs of relay nodes are considered in our approach. We observe that, when $\alpha=1$, the WSNs designed by PSTA and GKA generally have larger t_{net}, smaller t_{delay}, and larger $g_{\%}$ than those designed by OSRP and RRPL. The reason is that a small α weights the outage probabilities of transmission routes more

[1]Given enough computational resources (e.g. a few GBs memory consumption), these instances can be solved to optimality by SCIP-Jack within seconds when using the commercial linear programming solver CPLEX [43]. In this paper, we do not compare SCIP-Jack, as we focus on non-exact algorithms that have smaller polynomial time complexities (whereas SCIP-Jack does not have a polynomial-time guarantee) and lower demands on computational resources (e.g. less than 10 MBs memory consumption here for PSTA and GKA).



(a) Varying α ($|B| = 2$, $|S| = 100$, $|R| = 20$, $\gamma_{th} = 6E{-}10$, $M_r = 10$)

(b) Varying $|B|$ ($|S| = 100$, $|R| = 20$, $\alpha = 150$, $\gamma_{th} = 6E{-}10$, $M_r = 10$)

(c) Varying $|S|$ ($|B| = 2$, $|R| = 20$, $\alpha = 150$, $\gamma_{th} = 6E{-}10$, $M_r = 10$)

(d) Varying $|R|$ ($|B| = 2$, $|S| = 100$, $\alpha = 150$, $\gamma_{th} = 6E{-}10$, $M_r = 10$)

(e) Varying γ_{th} ($|B| = 2$, $|S| = 100$, $|R| = 20$, $\alpha = 150$, $M_r = 10$)

(f) Varying M_r ($|B| = 2$, $|S| = 100$, $|R| = 20$, $\alpha = 150$, $\gamma_{th} = 6E{-}10$)

Fig. 6. Evaluating the designed wireless sensor networks by PSTA, GKA, OSRP, and RRPL.

than the costs of relay nodes. This indicates that, in comparison to OSRP and RRPL, PSTA and GKA can design WSNs with a higher QoS when α is small.

We also observe that, when $\alpha = 150$, the WSNs designed by PSTA and GKA generally have smaller t_{net}, larger t_{delay}, and smaller $g_{\%}$ than those designed by OSRP and RRPL. The reason is that, when $\alpha = 150$, PSTA and GKA trade large outage probabilities of transmission routes with small costs of relay nodes. On the other hand, both PSTA and GKA are slower than OSRP and RRPL. We consider this as acceptable, as we focus on relay node placement in budget-limited scenarios, where it is more important to design cheap WSNs with a high QoS than to design WSNs fast.

Varying $|B|$: We vary $|B|$ in Figure 6b. We observe that, when $|B|$ is large, the designed WSNs have larger t_{net}, smaller t_{delay}, larger $g_{\%}$, and smaller $cost_{relay}$. The reason is that fewer relay nodes are in the middle of sensor nodes and base stations when $|B|$ is large.

Varying $|S|$: We vary $|S|$ in Figure 6c. We observe that, when $|S|$ is large, the designed WSNs generally have smaller t_{net}, larger t_{delay}, smaller $g_{\%}$, and larger $cost_{relay}$. The reason is that, when $|S|$ is large, the generated instances are sparser (details in Section VII-A), and more relay nodes are in the middle of sensor nodes and base stations.

Varying $|R|$: We vary $|R|$ in Figure 6d. We observe that, when $|R|$ is large, the designed WSNs generally have larger

TABLE I
THE COMPARISON WHEN $|B| = 2$, $|S| = 100$, $|R| = 20$, $\alpha = 150$, $\gamma_{th} = 6E{-}10$, $M_r = 100$ (BASE ALGORITHM: RRPL)

Algorithm	t_{net}	t_{delay}	$g_{\%}$	$cost_{relay}$	t_{run}
PSTA	0.95*base	1.02*base	1.00*base	0.75*base	74.32*base
GKA	0.96*base	1.01*base	1.00*base	0.84*base	2949*base
OSRP	0.98*base	0.97*base	1.00*base	1.22*base	16.13*base
RRPL	1.00*base	1.00*base	1.00*base	1.00*base	1.00*base

t_{net}, smaller t_{delay} and larger $g_{\%}$. The reason is that a large $|R|$ provides more and possibly better solutions. For the same reason, $cost_{relay}$ of PSTA and GKA decreases significantly with $|R|$, as they minimize the relay costs, while $cost_{relay}$ of RRPL and OSRP does not decrease significantly with $|R|$, as they minimize the numbers of relay nodes.

Varying γ_{th}: We vary γ_{th} in Figure 6e. We observe that, when γ_{th} is small, the designed WSNs have larger t_{net}, smaller t_{delay}, and larger $g_{\%}$. There are two reasons for this: 1) a small γ_{th} induces large transmission ranges of devices, and as a result, a large number of transmission routes; and 2) a small γ_{th} induces small outage probabilities of transmission routes (details in [6]). For the first listed reason above, $cost_{relay}$ increases with γ_{th}, and t_{run} decreases with γ_{th}.

Varying M_r: We vary M_r in Figure 6f. M_r does not affect t_{net}, $cost_{relay}$, or t_{run}, as it does not affect CCGs. In comparison, M_r affects t_{delay} and $g_{\%}$. We observe that t_{delay} and $g_{\%}$ increase with M_r, as a large M_r induces large success probabilities of transmissions between devices, at the cost of long delays (details in [6]).

The Usefulness of Our PSTA: In practice, it may be preferable to set M_r to a large value in scenarios that are sensitive to $g_{\%}$ but not to t_{delay}, such as designing WSNs for monitoring forest fires, where correctly reporting every fire is critical and a delay of few milliseconds is tolerable. In this case, PSTA can design cheap WSNs with a high QoS. For example, in comparison to RRPL, our PSTA can design WSNs with 25% lower relay cost and similar quality of service by average (specifically, 5% shorter network lifetime, 2% longer delay, and 0% loss of goodput) when $|B| = 2$, $|S| = 100$, $|R| = 20$, $\alpha = 150$, $\gamma_{th} = 6E{-}10$, $M_r = 100$ (details in Table I). This provides evidence in favor of our PSTA for minimum-cost relay node placement in budget-limited scenarios.

VIII. CONCLUSION

In this paper, we formulate the new Node-Weighted Partial Terminal Steiner Tree Problem (NWPTSTP) for minimum-cost relay node placement. The objective is to minimize the sum of heterogeneous costs of relay nodes and the sum of outage probabilities of transmission routes in a routing tree simultaneously. This extends the previous work that considers the costs of relay nodes to be homogeneous. We prove that NWPTSTP can be transformed to the node-weighted Steiner tree problem. Then, we conduct some theoretical analyses on Physarum-inspired algorithms to reveal their potential of computing Steiner trees. Based on these analyses, we propose a new Physarum-inspired algorithm for solving NWPTSTP. We conduct computational trials to show that: 1) in comparison to a state-of-the-art approximation algorithm for solving the node-weighted Steiner tree problem, our algorithm can produce better solutions in a smaller amount of time; and 2) in comparison to two state-of-the-art relay node placement algorithms, our algorithm can design cheaper WSNs with similar quality of service. This indicates the usefulness of our algorithm for minimum-cost relay node placement.

ACKNOWLEDGMENT

The authors sincerely thank the anonymous reviewers for their valuable comments to improve this work.

REFERENCES

[1] S. Maheswararajah, S. K. Halgamuge, K. B. Dassanayake, and D. Chapman, "Management of orphaned-nodes in wireless sensor networks for smart irrigation systems," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4909–4922, Oct. 2011.

[2] 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.

[3] 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.

[4] A. Chelli, M. Bagaa, D. Djenouri, I. Balasingham, and T. Taleb, "One-step approach for two-tiered constrained relay node placement in wireless sensor networks," *IEEE Wireless Commun. Lett.*, vol. 5, no. 4, pp. 448–451, Aug. 2016.

[5] B. Yuan, H. Chen, and X. Yao, "Optimal relay placement for lifetime maximization in wireless underground sensor networks," *Inf. Sci.*, vols. 418–419, pp. 463–479, Dec. 2017.

[6] 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.

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

[8] T. Nakagaki, H. Yamada, and A. Tóth, "Intelligence: Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 407, p. 470, 2000.

[9] A. Tero *et al.*, "Rules for biologically inspired adaptive network design," *Science*, vol. 327, no. 5964, pp. 439–442, Jan. 2010.

[10] W. Marwan, "Amoeba-inspired network design," *Science*, vol. 327, no. 5964, pp. 419–420, Jan. 2010.

[11] *What Humans Can Learn From Semi-Intelligent Slime*. Accessed: Feb. 13, 2020. [Online]. Available: https://www.ted.com/talks/heather_barnett_what_humans_can_learn_from_semi_intelligent_slime

[12] T. Nakagaki, R. Kobayashi, Y. Nishiura, and T. Ueda, "Obtaining multiple separate food sources: Behavioural intelligence in the Physarum plasmodium," *Proc. Roy. Soc. London B, Biol. Sci.*, vol. 271, no. 1554, pp. 2305–2310, Nov. 2004.

[13] L. Liu, Y. Song, H. Zhang, H. Ma, and A. V. Vasilakos, "Physarum optimization: A biology-inspired algorithm for the Steiner tree problem in networks," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 818–831, Mar. 2015.

[14] Y. Sun, P. N. Hameed, K. Verspoor, and S. Halgamuge, "A Physarum-inspired prize-collecting Steiner tree approach to identify subnetworks for drug repositioning," *BMC Syst. Biol.*, vol. 10, no. S5, pp. 25–38, 2016.

[15] Y. Sun and S. Halgamuge, "Fast algorithms inspired by Physarum polycephalum for node weighted steiner tree problem with multiple terminals," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 3254–3260.

[16] Y. Sun, "Physarum-inspired network optimization: A review," 2017, *arXiv:1712.02910*. [Online]. Available: https://arxiv.org/abs/1712.02910

[17] P. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted Steiner trees," *J. Algorithms*, vol. 19, no. 1, pp. 104–115, Jul. 1995.

[18] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks," *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 134–138, Jan. 2007.

[19] Z. Zheng, L. X. Cai, R. Zhang, and X. S. Shen, "RNP-SA: Joint relay placement and sub-carrier allocation in wireless communication networks with sustainable energy," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3818–3828, Oct. 2012.

[20] S. Misra, N. E. Majd, and H. Huang, "Approximation algorithms for constrained relay node placement in energy harvesting wireless sensor networks," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 2933–2947, Dec. 2014.

[21] Ş. Kimençe and I. Bekmezci, "Weighted relay node placement for wireless sensor network connectivity," *Wireless Netw.*, vol. 20, no. 4, pp. 553–562, May 2014.

[22] 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.

[23] D. Djenouri and M. Bagaa, "Energy-aware constrained relay node deployment for sustainable wireless sensor networks," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 1, pp. 30–42, Jan. 2017.

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

[25] S.-Y. Hsieh and H.-M. Gao, "On the partial terminal Steiner tree problem," *J. Supercomput.*, vol. 41, no. 1, pp. 41–52, May 2007.

[26] S. Guha and S. Khuller, "Improved methods for approximating node weighted Steiner trees and connected dominating sets," *Inf. Comput.*, vol. 150, no. 1, pp. 57–74, 1999.

[27] F. Zou, X. Li, S. Gao, and W. Wu, "Node-weighted Steiner tree approximation in unit disk graphs," *J. Combinat. Optim.*, vol. 18, no. 4, pp. 342–349, Nov. 2009.

[28] E. D. Demaine, M. Hajiaghayi, and P. N. Klein, "Node-weighted Steiner tree and group Steiner tree in planar graphs," *ACM Trans. Algorithms*, vol. 10, no. 3, p. 13, 2014.

[29] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Boston, MA, USA: Springer, 1972, pp. 85–103.

[30] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, Jun. 1972.

[31] A. Tero, R. Kobayashi, and T. Nakagaki, "Physarum solver: A biologically inspired method of road-network navigation," *Phys. A, Stat. Mech. Appl.*, vol. 363, no. 1, pp. 115–119, Apr. 2006.

[32] J. Siriwardana and S. K. Halgamuge, "Fast shortest path optimization inspired by shuttle streaming of Physarum polycephalum," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[33] K. Alim, G. Amselem, F. Peaudecerf, M. P. Brenner, and A. Pringle, "Random network peristalsis in Physarum polycephalum organizes fluid flows across an individual," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 33, pp. 13306–13311, Aug. 2013.

[34] V. Bonifaci, K. Mehlhorn, and G. Varma, "Physarum can compute shortest paths," *J. Theor. Biol.*, vol. 309, pp. 121–133, Sep. 2012.

[35] A. Tero, R. Kobayashi, and T. Nakagaki, "A mathematical model for adaptive transport network in path finding by true slime mold," *J. Theor. Biol.*, vol. 244, no. 4, pp. 553–564, Feb. 2007.

[36] A. M. Davie and A. J. Stothers, "Improved bound for complexity of matrix multiplication," *Proc. Roy. Soc. Edinburgh A, Math.*, vol. 143, no. 2, pp. 351–369, Apr. 2013.

[37] 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.*, vol. 27, no. 1, pp. 375–388, Feb. 2019.

[38] *The Codes and Datasets for the Node-Weighted Partial Terminal Steiner Tree Problem*. Accessed: Feb. 13, 2020. [Online]. Available: https://github.com/YahuiSun/NWPTSTP

[39] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Inf.*, vol. 15, no. 2, pp. 141–145, Jun. 1981.

[40] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 4, no. 4, pp. 1–33, Aug. 2008.

[41] G. Gamrath, T. Koch, S. J. Maher, D. Rehfeldt, and Y. Shinano, "SCIP-jack-a solver for STP and variants with parallelization extensions," *Math. Program. Comput.*, vol. 9, no. 2, pp. 231–296, 2017.

[42] D. Rehfeldt and T. Koch, "SCIP-Jack—A solver for STP and variants with parallelization extensions," in *Operations Research Proceedings*. Cham, Switzerland: Springer, 2018, pp. 191–196.

[43] *IBM ILOG CPLEX Optimizer*. Accessed: Feb. 13, 2020. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

Yahui Sun received the bachelor's and master's degrees from the Harbin Institute of Technology in 2012 and 2014, respectively, and the Ph.D. degree from The University of Melbourne in 2018. He was a Post-Doctoral Fellow with the Research School of Engineering, Australian National University, and a Research Fellow with Nanyang Technological University. His current research interests are graph theory and graph mining, especially Steiner tree problems in graphs and their applications to social, biomedical, telecommunication, and wireless sensor networks.

Daniel Rehfeldt received the B.Sc. and M.Sc. degrees from the Department of Mathematics, Technische Universitat Berlin, where he is currently pursuing the Ph.D. degree. He is part of the SCIP developers group, Zuse Institute Berlin, and is the main developer of SCIP-Jack, which is an exact solver for Steiner tree and related problems, as well as a developer of the linear programming solver SoPlex. His research interests are combinatorial optimization, linear programming, and (parallel) interior-point methods.

Marcus Brazil is currently an Associate Professor with the School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne. His main research interest is optimal network design with applications to telecommunications, wireless sensor networks, VLSI physical design, underground mining, and infrastructure for electric vehicles. He also combines this with more theoretical work, particularly in the area of Steiner trees. His research and consultancy work in optimization for underground mine design has been particularly successful and having received strong support from a number of Industry bodies.

Doreen Thomas (Senior Member, IEEE) is an Emeritus Professor with the School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne. She has an outstanding international reputation for fundamental mathematical research in network optimization. The software encapsulating her work is now used by the largest mining companies in the world to reduce underground mine development and haulage costs. Her research interests are optimal network design, wireless sensor networks, computational neuroscience, and electric vehicle charging and grid constraints.

Saman Halgamuge (Fellow, IEEE) studied at TU Darmstadt, Germany and University of Moratuwa, Sri Lanka. He is currently a Professor with the Department of Mechanical Engineering, School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne. He is also an Honorary Professor of Australian National University, where he was previously the Director of the Research School of Engineering. His research interests are in machine learning, big data analytics and optimization and their applications. These applications vary from sensor networks, smart grids, and sustainable energy to bioinformatics and neuro-engineering. He is a Distinguished Lecturer appointed by the IEEE CIS.