This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3008348, IEEE Transactions on Mobile Computing

Cooperative Sweep Coverage Problem with Mobile Sensors

Xiaofeng Gao, *Member, IEEE,* Jiahao Fan, Fan Wu, *Member, IEEE,*
Guihai Chen, *Senior Member, IEEE*

Abstract—Sweep coverage plays an important role in many applications such as data gathering, sensing coverage and devices control. In this paper, we deal with the cooperative sweep coverage problem with multiple mobile sensors to periodically cover all positions of interest (PoIs) in the surveillance region. Different from traditional sweep coverage scenarios, the cooperative sweep coverage (CSC) problem allows the deployment of multiple sensors on the same trajectory to further reduce the sweep period or detection delay. We also consider the multi-sink sweep coverage (MSSC) problem where each mobile sensor must periodically transmit its collected data to a base station due to the limited storage capacity and power supply. Correspondingly, we propose two constant-factor approximations, namely *CoCycle* and *SinkCycle*, to minimize the maximum sweep period for these two problems. The approximation ratios of *CoCycle* and *SinkCycle* are proved to be 4 and 6 respectively. As far as we know, *SinkCycle* is the first approximation for the sweep coverage problem with multiple sinks. We also provide two optimal algorithms for the CSC problem in one dimensional case and a useful insight regarding the MSSC problem with only one available sink. Finally, we conduct various numerical experiments to validate the effectiveness and efficiency of our designs.

Index Terms—Sweep Coverage, Wireless Sensor Network, Traveling Salesman Problem, Approximation

1 Introduction

Wireless sensor network (WSN) consists of numerous wireless sensors which are usually low-priced and work cooperatively to form an ad-hoc network [1]. For these wireless sensor networks, coverage problems have been studied extensively under various models. In general, these problems are mainly about monitoring positions of interest (PoIs) and collecting data from a given area by deploying a certain number of sensors. Some studies assume the coverage area of the sensor as a unit disk [2]. Others adopt the probabilistic model to compute the probability of covering the whole region [3].

While many studies focus on continuous monitoring, such as *Target Coverage* problem [4], *Area Coverage* problem [5], [6] and *Barrier Coverage* problem [7], [8], there are some other application scenarios in which only periodic patrol inspections are required for a certain set of PoIs. Typical examples may include police patrolling, message ferrying and device control. In these scenarios, a mobile sensor is capable of moving along some certain trajectory and collecting data from PoIs. The objective in these scenarios can be minimizing the number of sensors under some time constraint or minimizing the detection period with a given number of sensors. We refer to such problems as *Sweep Coverage* [9], [10], [11], [12], [13], [14], [15]. Similar models have also been studied under the context of autonomous robots, vehicle routing, and data collection.

In a typical sweep coverage scenario, each mobile sensor follows a predetermined trajectory to collect data from PoIs on its route. Based on whether multiple mobile sensors can work together on the same trajectory, we can categorize sweep coverage problems into the non-cooperative version and the cooperative version as shown in Figure 1. Under non-cooperative settings, there is exactly one sensor on each trajectory. While under cooperative settings, one or more sensors may be assigned to the same trajectory. In this paper, we mainly focus on the cooperative sweep coverage problem with multiple mobile sensors. Assume that there are n PoIs (or targets) in the surveillance region and we have m mobile sensors to cover them. Each sensor serves as a data ferry to collect information from PoIs. A mobile sensor detects a PoI by approaching its exact location, and a PoI is said to be t-sweep covered if it is detected by some mobile sensor at least once every t time units (we call t its sweep period). The main objective considered in this paper is to minimize the sweep period for all PoIs. We consider two variations of this problem and their detailed descriptions are as follows.

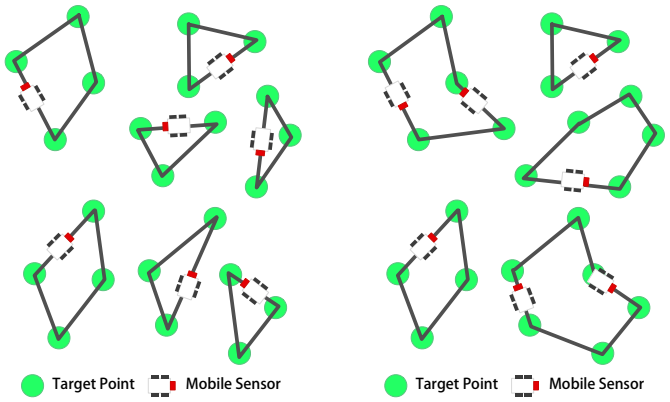
First, we would like to consider the *Cooperative Sweep Coverage* (CSC) problem in its basic form. Assume that all mobile sensors have the same velocity v. If we deploy m mobile sensors evenly on a cycle $\mathcal{C}$ and make them move towards the same direction to cover the PoIs on $\mathcal{C}$, then the sweep period for each PoI on this cycle should be $\frac{d(\mathcal{C})}{mv}$, where $d(\mathcal{C})$ is the length of $\mathcal{C}$. Since the velocity of sensors has no influence on the final outcome in this scenario, we just denote the sweep period as $\frac{d(\mathcal{C})}{m}$ for simplicity. The CSC problem aims to find a coverage scheme that minimizes the maximum sweep period among all PoIs.

Next, we consider a more realistic version of the cooperative sweep coverage problem named *Multi-Sink Sweep Coverage* (MSSC). In this scenario, the data storage capacity and battery power of each mobile sensor is limited and it

- *X. Gao, J. Fan, F. Wu, and G. Chen are with the Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. E-mails: gaoxf@cs.sjtu.edu.cn, j.h.fan@sjtu.edu.cn, {fwu, gchen}@cs.sjtu.edu.cn.*
- *Xiaofeng Gao is the corresponding author.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3008348, IEEE Transactions on Mobile Computing

has to visit one of the data sinks to upload its collected data in each sweep cycle. The MSSC problem shares the same optimization objective as the basic CSC problem (i.e., to minimize the maximum sweep period among all PoIs). However, there must be at least one data sink on each cycle of the solution to the MSSC problem.



(a) Non-cooperative version (b) Cooperative version

Fig. 1. An illustration for sweep coverage problem

Correspondingly, in this paper we propose two constant-factor approximations, namely *CoCycle* and *SinkCycle*, to solve the CSC problem and the MSSC problem respectively. The first approximation *CoCycle* deals with the CSC problem with the approximation ratio of 4. Its main idea is to first find a tree cover, then determine the optimal sensor allocation among these trees and finally transform trees to distinct cycles to construct a desired cycle cover. The second approximation *SinkCycle* is a 6-approximation for the MSSC problem. It integrates the design idea from *CoCycle* together with the modified Prim's algorithm to first find a tree cover for all PoIs and some data sinks, then determine the optimal sensor allocation among these trees and finally transform trees to distinct cycles to construct a desired cycle cover with at least one sink on each cycle. As far as we know, *SinkCycle* is the first approximation algorithm with a guaranteed performance ratio for the sweep coverage problem with multiple sinks.

In addition, we also give two optimal algorithms, *LineSplit-DP* and *LineSplit-Greedy*, for the CSC problem in one dimensional case (CSC1D) and a useful insight regarding the MSSC problem with only one available sink. The analysis of these two special cases further reveals the essence and hardness of the considered problems.

Finally, we provide various comparative experiments to validate the effectiveness and efficiency of our designs. We also conduct several parametric analysis experiments to show the performance of our algorithms under different parametric settings.

To sum up, the contributions of our paper are as follows.

- We formulate two variations of the cooperative sweep coverage problem with multiple mobile sensors, namely *Cooperative Sweep Coverage* (CSC) and *Multi-Sink Sweep Coverage* (MSSC). The objective is to have sensors work cooperatively and minimize the maximum sweep period among all PoIs.
- We propose two constant-factor approximations to solve these two problems respectively. We also give detailed theoretical analysis regarding their approximation ratios and complexity. Furthermore, *SinkCycle* is the first approximation algorithm with a guaranteed performance ratio for the sweep coverage problem with multiple sinks.
- We provide two optimal algorithms for the CSC problem in one dimensional case (CSC1D) and a useful insight regarding the MSSC problem with only one available sink.
- We compare our algorithms with several previous works by simulations. Both theoretical analysis and numerical experiments validate the effectiveness and efficiency of our designs.

The rest of this paper is organized as follows. Section 2 discusses some related work. Section 3 introduces some preliminaries for later sections. In Section 4 and Section 5, we propose algorithms for *Cooperative Sweep Coverage* (CSC) and *Multi-Sink Sweep Coverage* (MSSC) respectively with theoretical analysis. In Section 6, we conduct experiments to evaluate the performance of our algorithms. Section 7 is the final conclusion and future work.

2 Related Work

As for the sweep coverage problem, previous papers mainly focused on three different scenarios.

- *Message Ferry (or Data Gathering)* [16], [17], [18]. In this scenario, there are some static target points that produce desired data, and the goal is to enable data sharing among these target points or data collection to a base station. We can use mobile sensors to successively visit these points to collect data. This is closely related to the Internet of Things (IoT).
- *Sensing Coverage* [11], [19], [20], [21], [22]. In a large-scale sensor network, covering all targets with traditional static sensors imposes a high implementation cost. Thus, if possible, we prefer to use mobile sensors to periodically cover each target by having them move along certain trajectories.
- *Device Control* [23]. In this scenario, in order to work properly, each facility locating in a discrete address needs to periodically receive instructions from a mobile device such as a mobile phone.

Based on these scenarios, researchers also considered various optimization objectives and the most common ones are as follows.

- Find the minimum number of sensors with a fixed velocity under the constraint of the sweep period for all targets [12], [21], [22], [24], [25].
- Find the minimum sweep period for all targets with a fixed number of sensors [15], [18], [26], [27]. In such problem formulations, the velocities of sensors can be the same or not, while different velocities could make the analysis more complicated.
- Find the minimum velocity with a fixed number of sensors under the constraint of the sweep period for all targets [20], [27], [28].

According to our survey, related works mainly used the following approaches to solve the sweep coverage problem:

- *Trajectory planning*. Obviously, in order to reduce energy consumption and the number of sensors, we need to find a trajectory to visit all target points, which is usually converted to a Traveling Salesman Problem (TSP) instance. In [29], the considered problem was converted to an integer programming which is also NP-hard, and the authors proposed an approximation solution by LP relaxation and rounding.
- *Vertex partition*. Since different target points may have various sweep period constraints, having the same sweep period for all targets may be unnecessary for vertices that do not require frequent visits. In [22], [28], authors first partitioned vertices based on their locations and sweep period constraints, and then planned trajectories accordingly.
- *Velocity control*. There are works assuming heterogeneous velocities of sensors, which is related to latency and power consumption. In [27], authors discussed its influence on data collection latency.

The cooperative sweep coverage problem considered in this paper is close related to the multiple Traveling Salesman Problem (TSP) with a min-max objective (denoted as min-max m-TSP). The main difference is that in cooperative sweep coverage, multiple "salesmen" can work together in one cycle to further decrease the longest trajectory length. The TSP problem is one of the most intensively studied problems in the area of combinatorial optimization. A simple algorithm based on minimum spanning tree (MST) gives a 2-approximation solution. By a clever construction, Christofides [30] improved the approximation ratio from 2 to $\frac{3}{2}$. It has been proved that the metric TSP is inapproximable within a ratio of $\frac{123}{122}$, unless P = NP [31]. Obviously, the min-max m-TSP problem and the cooperative sweep coverage problem are at least as hard as the original TSP.

To minimize the number of mobile sensors under the sweep period requirement, Li et al. [12] proposed a 3-approximation with bounded time constraint. However, their approximation analysis has a serious flaw, which has been notified by Gorain et al. [13]. They mistakenly compared the result of their algorithm with the optimal solution of the m-TSP problem instead of their original problem. Thus their approximation ratio is considered to be incorrect. Zhao et al. [20] designed a simulated annealing algorithm to schedule the paths, but their algorithm has no guaranteed performance ratio. Shu et al. [22] discussed this problem with the single-sink constraint, to which mobile sensors must return back in each detection period. They also proposed a heuristic algorithm without theoretical bounds.

Since mobile sensors have sensing ranges, some researchers took the neighborhood effect into consideration to further decrease the trajectory length. In fact, mobile sensors do not need to travel to the exact positions of targets [32]. Any position in the neighborhoods of targets is acceptable as long as the sensor can collect the data as required. He et al. [33] formulated such a problem as Traveling Salesman Problem with Neighborhoods (TSPN), in which there is only one mobile sensor. Kim et al. [34] proposed solutions for min-max m-TSP with neighborhood effect. Xue et al. [35] relaxed the assumption to allow mobile sensors with different velocities.

It is worth mentioning that most previous papers did not consider having mobile sensors work in a cooperative manner. OSweep [24] is a simple TSP-based cooperative sweep coverage algorithm, which computes a single TSP cycle covering all PoIs and allocates all sensors evenly on this cycle. MinExpand [24] and PDBA [36] are two heuristic algorithms for non-cooperative sweep coverage problems, both of which find the minimum path increment based on different criteria during the iteration process. In our simulations, we choose OSweep, MinExpand and PDBA as baselines to validate the effectiveness and efficiency of our cooperative sweep coverage scheme *CoCycle*.

Due to limited storage capacity and battery power, some works [37], [38], [39] also require mobile sensors to periodically visit some base stations (data sinks) in order to upload data and recharge the battery. Yang et al. [37] designed a heuristic algorithm named SCOPe-M-Solver to satisfy both the sweep period for PoIs and the base station visiting period for sensors. It first assigns each PoI to the cluster of its nearest base station, and then constructs cycles from the base station in each cluster by expanding in a similar way to MinExpand. Liang et al. [38] considered the maximum travel distance before visiting some base station in addition to the sweep period constraint. They also gave analysis of their solutions under tree metric specifically. In our simulations, we compare our multi-sink cooperative sweep coverage scheme *SinkCycle* with SCOPe-M-Solver under different parameter settings and give detailed discussions about how the clustering step of SCOPe-M-Solver may affect cross-sink cooperation considered in *SinkCycle*.

Recently, with the development of mobile smart devices, researchers tend to investigate the sensing coverage problem under more complex scenarios such as obstacle avoidance [40] and mobile crowdsensing [41]. Different from the fundamental problems in this paper, these works may consider more complex constraints such as coverage persistence and sensing cost.

3 PRELIMINARY

In this section, we define some basic concepts and introduce some primitive methods which will be used in later sections.

3.1 Metric Space

For any given complete graph $G(V, E)$, we will use d to represent a *metric* on V such that $d : V \times V \to \mathbb{R}^+$. Triangle inequality is the most important property that a *metric space* holds, i.e.,

$$d(x, y) \leq d(x, z) + d(z, y) \text{ for any } x, y, z \in V.$$

Suppose e is the edge between vertices x and y, we will use $d(x, y)$ and $d(e)$ to denote the same thing in context without ambiguity, which represents the distance between vertices x and y or the length (cost) of edge e. For an edge set $E' \subseteq E$, define

$$d(E') = \sum_{e \in E'} d(e).$$

Unless explicitly mentioned, we will assume graphs considered in this paper are complete graphs such that the distances between vertices form a metric.

3.2 Cycle Cover and Tree Cover

Now we will give the definitions of Cycle Cover and Tree Cover. Briefly speaking, Cycle Cover is to cover all vertices in a graph with multiple cycles. Tree Cover is similar to Cycle Cover in the sense of covering. Different from Cycle Cover, Tree Cover uses multiple spanning trees to cover all vertices in a graph. The formal definitions of *Cycle Cover* and *Tree Cover* are as follows.

Definition 1 (Cycle Cover)**.** *Given a graph $G = (V, E)$ and a vertex set $V' \subseteq V$, a cycle cover for V' is a set of cycles $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r\}$, which are subgraphs of G, and the union of their vertices is V'.*

Definition 2 (Tree Cover)**.** *Given a graph $G = (V, E)$ and a vertex set $V' \subseteq V$, a tree cover for V' is a forest $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$, which are subgraphs of G, and the union of their vertices is V'.*

3.3 Constructing Cycle from Tree

In our algorithm design, we first obtain some trees spanning over all PoIs, and then construct cycles from these trees. Here we would like to illustrate the transformation process from trees to cycles.

Suppose we have a spanning tree. The first step is to duplicate all its edges, after which every vertex will have an even degree. Then we can construct an Eulerian cycle. Finally we obtain a Hamiltonian cycle by traversing this Eulerian cycle and removing the repeated vertices, which is called shortcutting. Due to the triangle inequality, shortcutting does not increase the overall edge cost. Algorithm 1 explains the whole process in detail and runs in $O(|V|)$ time, where V is the vertex set of the given tree $\mathcal{T}$.

Algorithm 1: Constructing cycle from tree

input : a tree $\mathcal{T}$ with the vertex set V
output: a cycle $\mathcal{C}$ with the same vertex set V

1 Duplicate all the edges in $\mathcal{T}$ to get a graph G';
2 Find a Eulerian cycle in G';
3 Construct a Hamiltonian cycle $\mathcal{C}$ by removing the repeated vertices from the previous Eulerian cycle;
4 **return** $\mathcal{C}$;

With Algorithm 1, we have the following lemma:

Lemma 1. *For any tree $\mathcal{T}$, we can construct a cycle $\mathcal{C}$ that contains exactly the same set of vertices. Moreover, it holds that*

$$d(\mathcal{C}) \leq 2d(\mathcal{T}).$$

Remark. *We could use Christofides' improvements [30] to construct cycles from trees in our algorithm, but we cannot trivially get a better approximation ratio. We briefly explain how Christofides' algorithm works as follows.*

1) *Let $\mathcal{T}$ be a spanning tree of G and E' be the set of vertices whose degree is odd in $\mathcal{T}$. By the handshaking lemma, E' has an even number of vertices.*
2) *Find a minimum-weight perfect matching $\mathcal{M}$ in the induced subgraph given by the vertices from E'.*
3) *Combine the edges of $\mathcal{M}$ and $\mathcal{T}$ to form a connected multigraph G' in which every vertex has an even degree.*
4) *Form a Eulerian cycle in G' and make the cycle found in previous step into a Hamiltonian cycle by removing the repeated vertices from the cycle, which is called shortcutting.*

This algorithm does not duplicate all the edges to guarantee an even degree for every vertex. Therefore, the overall edge cost of its generated cycle is supposed to be smaller. According to [30], the running time of this algorithm is $O(|V|^3)$.

3.4 Global t-Sweep Coverage

Sweep coverage, unlike traditional area coverage or barrier coverage, does not require static and continuous coverage all the time. In sweep coverage, we only need to cover every PoI at least once every certain time interval to guarantee event detection within a certain delay bound. With this idea, we define *t-Sweep Coverage* as follows.

Definition 3 (t-Sweep Coverage)**.** *A PoI is said to be t-sweep covered by a coverage scheme $\mathcal{F}$ if and only if it is scanned at least once every t time units by the mobile sensors allocated by $\mathcal{F}$.*

If a PoI is t-sweep covered, time interval t is called the *sweep period* of the PoI. When there is a set of PoIs, different PoIs may have different sweep periods. In order to unify the requirements, we define *Global t-Sweep Coverage* as follows.

Definition 4 (Global t-Sweep Coverage)**.** *A set of PoIs is said to be global t-sweep covered by a coverage scheme $\mathcal{F}$ if and only if all PoIs are scanned at least once every t time units by the mobile sensors allocated by $\mathcal{F}$.*

4 Cooperative Sweep Coverage

In this section, we will formally define the Cooperative Sweep Coverage (CSC) problem, and then design an approximation named *CoCycle* for this problem. We also propose two optimal algorithms for Cooperative Sweep Coverage problem in one dimensional case (CSC1D). Besides, we will prove the correctness of our algorithms.

4.1 CSC: General Case

In this subsection, we will talk about the Cooperative Sweep Coverage problem in general case. We firstly give the formal formulation of the problem and then propose an approximation for it, the approximation ratio of which is 4.

4.1.1 Problem Formulation

The basic idea of our algorithm to solve the Cooperative Sweep Coverage problem follows these steps:

1) Find a tree cover $\mathscr{T}$.
2) Determine a sensor allocation scheme for $\mathscr{T}$.
3) Construct a cycle cover $\mathscr{C}$ from $\mathscr{T}$.

To formally describe the process in the second step, we further introduce the concept of *Sensor Allocation*.

Definition 5 (Sensor Allocation)**.** *Given m sensors and a cycle cover $\mathscr{C}$ (or tree cover $\mathscr{T}$), a sensor allocation for $\mathscr{C}$ (or $\mathscr{T}$) is a*

number set $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ *such that* $r = |\mathscr{C}|$ *(or* $|\mathscr{T}|$*),* $m_i \geq 1$ *for* $1 \leq i \leq r$*, and* $\sum_{i=1}^{r} m_i = m$.

Throughout this paper, we consider the sweep coverage scenario where mobile wireless sensors allocated to the same cycle work cooperatively to scan PoIs (as shown in Figure 1b). We define it as *Cooperative Sweep Coverage* (CSC).

Definition 6 (Cooperative Sweep Coverage (CSC))**.** *Given input triple* (G, m, d)*, where* $G = (V, E)$ *is a complete graph,* m *is the number of sensors and* $d : E \to \mathbb{R}^+$ *is a metric, Cooperative Sweep Coverage (CSC) aims to find a coverage scheme* $\mathcal{F} = (\mathscr{C}, \mathcal{A})$*, consisting of a cycle cover* $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r\}$ *for* V *and a sensor allocation* $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ *for* $\mathscr{C}$*, such that* $\max\limits_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\}$ *is minimized.*

Note that we use $\frac{d(\mathcal{C}_i)}{m_i}$ and $\max\limits_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\}$ to represent the *sweep period* of the PoIs in $\mathcal{C}_i$ and the *global sweep period* of all PoIs respectively for simplicity considering that all sensors have the same speed in our assumptions.

4.1.2 CoCycle (An Approximation for CSC)

Now we design the *CoCycle* algorithm to solve the CSC problem. The main idea is to first find a tree cover, then determine the optimal sensor allocation among these trees and finally transform trees to distinct cycles to construct a desired cycle cover.

To start with, we need to search for a relatively good tree cover in the original graph. Thus we define a variant problem called *Cooperative Tree Coverage* (CTC).

Definition 7 (Cooperative Tree Coverage (CTC))**.** *Given input triple* (G, m, d)*, where* $G = (V, E)$ *is a complete graph,* m *is the number of sensors and* $d : E \to \mathbb{R}^+$ *is a metric, Cooperative Tree Coverage (CTC) aims to find a coverage scheme* $\mathcal{F} = (\mathscr{T}, \mathcal{A})$*, consisting of a tree cover* $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$ *for* V *and a sensor allocation* $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ *for* $\mathscr{T}$*, such that* $\max\limits_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$ *is minimized.*

Our interest in the CTC problem arises from the fact that a connection between CSC and CTC can easily be established as follows.

Lemma 2. *If there is an* α*-approximation algorithm for CTC, then there is a* 2α*-approximation algorithm for CSC.*

Proof. Suppose $(\{\mathcal{T}_i^*\}, \{m_i^*\})$ is an optimal solution for CTC, and $(\{\mathcal{C}_i^*\}, \{w_i^*\})$ is an optimal solution for CSC. We have an α-approximation algorithm which returns a solution $(\{\mathcal{T}_i\}, \{m_i\})$ for CTC, i.e.,

$$\max_i \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\} \leq \alpha \times \max_i \left\{ \frac{d(\mathcal{T}_i^*)}{m_i^*} \right\}. \tag{1}$$

Deleting one edge from each $\mathcal{C}_i^*$ leads to a solution for CTC, thus we have

$$\max_i \left\{ \frac{d(\mathcal{T}_i^*)}{m_i^*} \right\} \leq \max_i \left\{ \frac{d(\mathcal{C}_i^*)}{w_i^*} \right\}. \tag{2}$$

We can use Algorithm 1 to transform $\{\mathcal{T}_i\}$ to $\{\mathcal{C}_i\}$, and according to Lemma 1, it holds that

$$d(\mathcal{C}_i) \leq 2 \times d(\mathcal{T}_i). \tag{3}$$

Combining (1), (2) and (3), we have

$$\max_i \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\} \leq 2\alpha \times \max_i \left\{ \frac{d(\mathcal{C}_i^*)}{w_i^*} \right\}. \tag{4}$$

Thus, we obtain a 2α-approximation algorithm for CSC which returns the solution $(\{\mathcal{C}_i\}, \{m_i\})$. □

Guided by the insight from the above lemma, we first design the *CoTree* algorithm to solve the CTC problem. Then we can get the *CoCycle* algorithm for the CSC problem by constructing cycles from the trees returned by *CoTree*. Therefore, we now focus on the description and analysis of our proposed *CoTree* algorithm.

Recall Kruskal's algorithm for constructing a minimum spanning tree. We add edges to the empty graph $G_0 = (V, \emptyset)$ one by one in an increasing order of the length (i.e., $d(\cdot)$). In each stage i of Kruskal's algorithm, G_i will have a number of connected components, and each subgraph induced by a connected component is a spanning tree on its vertices. In other words, G_i is a spanning forest of the original graph G and contains a tree cover for V. The main idea of *CoTree* is to utilize these intermediate tree covers and deliver a performance-guaranteed solution for the CTC problem.

Algorithm 2 describes *CoTree* in detail. *CoTree* will choose the best of all feasible tree covers found in the algorithm.

Algorithm 2: CoTree

input : $G = (V, E)$, $d : E \to \mathbb{R}^+$ and m sensors
output: A tree cover $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$ and a sensor allocation $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ for $\mathscr{T}$

1 $i \leftarrow 0$; $E_0 \leftarrow \emptyset$; $G_0 \leftarrow (V, E_0)$;
2 **foreach** $e \in E$ *(chosen in ascending order by* $d(\cdot)$*)* **do**
3 **if** *adding* e *to* G_i *does not produce a cycle* **then**
4 $i \leftarrow i + 1$; $e_i \leftarrow e$;
5 $E_i \leftarrow E_{i-1} \cup \{e\}$; $G_i \leftarrow (V, E_i)$;
6 **if** # *of* G_i*'s connected components* $\leq m$ **then**
7 Obtain a tree cover $\mathscr{T}_i$ directly from G_i;
8 Find an optimal sensor allocation $\mathcal{A}_i$ for $\mathscr{T}_i$;
9 **return** *the best coverage scheme in* $\{(\mathscr{T}_i, \mathcal{A}_i)\}$ (with minimum global sweep period $\max_i \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$);

The initialization step is done in Line 1. In Line 2-8, we continuously add edges into the edge set E_i in an ascending order of edge length. We check if the adding edge will produce a cycle. In Line 9, we find the best coverage scheme among all the considered schemes.

For Line 8 in Algorithm 2, we could design an efficient allocation strategy for this step. Before this step, we have already got a tree cover, so the task here is just determining how many sensors should be assigned to each tree. An optimal solution can be achieved by a simple greedy algorithm described in Algorithm 3, whose optimality is proved in Lemma 3.

Lemma 3. *Algorithm 3 finds an optimal sensor allocation for any given tree cover* $\mathscr{T}$ *efficiently. Here the optimality means that* $\max\limits_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$ *is minimized when fixing* $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$.

Algorithm 3: Greedy Allocation

input : A tree cover $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$, a metric d and m sensors

output: A sensor allocation $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ for $\mathscr{T}$

$m_i \leftarrow 1$ for $1 \le i \le r$;
while $m > \sum_{i=1}^{r} m_i$ **do**
  $\hat{k} \leftarrow \arg\max_{1 \le i \le r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$;
  $m_{\hat{k}} \leftarrow m_{\hat{k}} + 1$;
return $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$;

Proof. We prove this lemma by contradiction. Assume on the contrary there is a different sensor allocation $\mathcal{A}' = \{m'_1, m'_2, \ldots, m'_r\}$ such that

$$\max_{1 \le i \le r} \left\{ \frac{d(\mathcal{T}_i)}{m'_i} \right\} < \max_{1 \le i \le r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}. \tag{5}$$

Since $\mathcal{A}'$ is different from $\mathcal{A}$, then there must exist a $\hat{k}$ such that

$$m'_{\hat{k}} < m_{\hat{k}} \quad \text{and} \quad \frac{d(\mathcal{T}_{\hat{k}})}{m'_{\hat{k}}} \ge \frac{d(\mathcal{T}_{\hat{k}})}{m_{\hat{k}}}. \tag{6}$$

In Algorithm 3, we did increase the value of $m_{\hat{k}}$ from $m'_{\hat{k}}$ to $m'_{\hat{k}} + 1$, which means that $\frac{d(\mathcal{T}_{\hat{k}})}{m'_{\hat{k}}}$ is the maximum among $\left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$ at some iteration. Notice that the value of $\max_{1 \le i \le r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$ is non-increasing during the iterations. Therefore, we have

$$\frac{d(\mathcal{T}_{\hat{k}})}{m'_{\hat{k}}} \ge \max_{1 \le i \le r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}, \tag{7}$$

which contradicts the assumption. □

Based on the coverage scheme $\mathcal{F}_{\text{CTC}} = (\mathscr{T}, \mathcal{A})$ returned by *CoTree*, *CoCycle* just uses Algorithm 1 (with or without the Christofides' improvements) on each tree in $\mathscr{T}$ respectively to transform the tree cover $\mathscr{T}$ to a cycle cover $\mathscr{C}$, and returns a corresponding coverage scheme $\mathcal{F}_{\text{CSC}} = (\mathscr{C}, \mathcal{A})$ as its solution for the CSC problem.

4.1.3 Performance Analysis

Now we will prove the approximation ratios of the proposed *CoTree* and *CoCycle* algorithms. Assign $n = |V|$ as the number of PoIs in this subsection. From the for-loop in Algorithm 2, we have $d(e_1) \le d(e_2) \le \cdots \le d(e_{n-1})$, and it is easy to get the following fact.

Lemma 4. *G_i is a minimum spanning forest with $(n - i)$ connected components. For each connected component CC in G_i, the subgraph induced by CC is actually a minimum spanning tree on the vertex set of CC.*

Proof. For some connected component CC of G_i, we know that the induced subgraph is an MST of CC by

- This subgraph is **connected** since CC exactly means connected component.
- This subgraph is a **tree**, which is guaranteed by Line 3 in Algorithm 2.
- This tree is **minimized**, which is guaranteed by the correctness of Kruskal's algorithm (the edge set selected by Algorithm 2 in CC is exactly the same as what would be returned by Kruskal's algorithm).

Thus this lemma holds. □

Suppose we have an optimal coverage scheme $\mathcal{F}^*_{\text{CTC}} = (\mathscr{T}^*, \mathcal{A}^*)$ for the CTC problem, where $\mathscr{T}^* = \{\mathcal{T}^*_1, \mathcal{T}^*_2, \ldots, \mathcal{T}^*_{r^*}\}$ and $\mathcal{A}^* = \{m^*_1, m^*_2, \cdots, m^*_{r^*}\}$. Set

$$\text{OPT} = \max_{1 \le i \le r^*} \left\{ \frac{d(\mathcal{T}^*_i)}{m^*_i} \right\} \tag{8}$$

and

$$\sigma = \arg\max_{\substack{1 \le i \le n-1 \\ d(e_i) \le \text{OPT}}} \{d(e_i)\}. \tag{9}$$

Suppose the connected components in G_σ are $CC^\sigma_1, CC^\sigma_2, \ldots, CC^\sigma_{n-\sigma}$, and we use $\mathcal{T}^\sigma_1, \mathcal{T}^\sigma_2, \ldots, \mathcal{T}^\sigma_{n-\sigma}$ to denote their corresponding spanning trees.

Lemma 5. *There is an optimal coverage scheme $\mathcal{F}^*_{\text{CTC}} = (\mathscr{T}^*, \mathcal{A}^*)$ such that all vertices of $\mathcal{T}^*_i$ belong to the same connected component of G_σ, for $1 \le i \le r^*$. In other words, any edge used in this optimal solution has a cost no more than OPT.*

Proof. Suppose $\mathcal{T}^*_i$ uses an edge e' to connect two different connected components of G_σ, then we have $d(e') > \text{OPT}$. If we do not use e', $\mathcal{T}^*_i$ will be partitioned into two trees, $\mathcal{T}_L$ and $\mathcal{T}_R$. Reallocating the m^*_i sensors between $\mathcal{T}_L$ and $\mathcal{T}_R$ accordingly gives a coverage scheme with a smaller sweep period for PoIs in $\mathcal{T}^*_i$. More specifically, assign $m_L = \lceil \frac{d(\mathcal{T}_L)}{\text{OPT}} \rceil$ sensors to $\mathcal{T}_L$, and $m_R = m^*_i - m_L$ sensors to $\mathcal{T}_R$. Notice that

$$\frac{d(\mathcal{T}_L)}{m_L} \le \text{OPT}, \tag{10}$$

and

$$\frac{d(\mathcal{T}_R)}{m_R} = \frac{d(\mathcal{T}^*_i) - d(\mathcal{T}_L) - d(e')}{m^*_i - m_L} \tag{11}$$

$$\le \frac{(m^*_i - (m_L - 1) - 1) \times \text{OPT}}{m^*_i - m_L} \tag{12}$$

$$= \text{OPT}. \tag{13}$$

Thus we can eliminate all such edges to get an optimal coverage scheme satisfying the property described in this lemma. For the rest of this subsection, we will assume that the considered optimal coverage scheme $\mathcal{F}^*_{\text{CTC}} = (\mathscr{T}^*, \mathcal{A}^*)$ has this property. □

Now we prove the approximation ratio of *CoTree*.

Theorem 1. CoTree *is a 2-approximation for CTC.*

Proof. Suppose CC^σ_i contains μ^σ_i trees from the optimal coverage scheme $\mathcal{F}^*_{\text{CTC}}$, and these μ^σ_i trees use totally λ^σ_i ($\ge \mu^\sigma_i$) sensors in the optimal coverage scheme, then

$$d(\mathcal{T}^\sigma_i) \le \lambda^\sigma_i \times \text{OPT} + (\mu^\sigma_i - 1) \times \text{OPT}, \tag{14}$$

which gives

$$\frac{d(\mathcal{T}^\sigma_i)}{\lambda^\sigma_i} \le 2 \times \text{OPT}. \tag{15}$$

Thus for the tree cover $\mathscr{T}_\sigma = \{\mathcal{T}_1^\sigma, \mathcal{T}_2^\sigma, \ldots, \mathcal{T}_{n-\sigma}^\sigma\}$, we already have a sensor allocation $\mathcal{A}'_\sigma = \{\lambda_1^\sigma, \lambda_2^\sigma, \ldots, \lambda_{n-\sigma}^\sigma\}$ such that

$$\max_{1\le i\le n-\sigma}\left\{\frac{d(\mathcal{T}_i^\sigma)}{\lambda_i^\sigma}\right\} \le 2\times \text{OPT}. \tag{16}$$

By Lemma 3, we can find an optimal sensor allocation $\mathcal{A}_\sigma = \{m_1^\sigma, m_2^\sigma, \ldots, m_{n-\sigma}^\sigma\}$ for $\mathscr{T}_\sigma$ whose global sweep period will satisfy

$$\max_{1\le i\le n-\sigma}\left\{\frac{d(\mathcal{T}_i^\sigma)}{m_i^\sigma}\right\} \le \max_{1\le i\le n-\sigma}\left\{\frac{d(\mathcal{T}_i^\sigma)}{\lambda_i^\sigma}\right\} \le 2\times \text{OPT}. \tag{17}$$

Therefore, since *CoTree* returns the best considered coverage scheme, we can conclude that *CoTree* is a 2-approximation for CTC. □

According to Lemma 2, we get the following corollary.

Corollary 1. CoCycle *is a 4-approximation for CSC.*

As the first step of *CoTree*, we have to put all the edges in ascending order. This sorting process can be done in $O(|E|\log|E|))$ steps. Considering that G is a complete graph, we have $|E| = |V|\cdot(|V|-1) = n(n-1)$, thus the running time of this step is $O(n^2\log n))$. Then, we perform Algorithm 3 whose average time complexity is $O(m^2)$ for m times. Therefore, the overall complexity of *CoTree* is $O(n^2\log n + m^3)$, which is also the time complexity of *CoCycle* since Algorithm 1 runs in $O(n)$ time.

4.2 CSC: One Dimensional Case

In this subsection, we consider the Cooperative Sweep Coverage problem in one dimensional case (CSC1D). In practice, CSC1D can be applied in such scenarios that all PoIs are located on a single route, including street patrolling and periodical intrusion detection on borders.

Suppose all PoIs are distributed along a straight line. We mark these n PoIs from left to right as $p_1, p_2, \ldots, p_n$, and we have m mobile sensors to cover them all. Intuitively, we find that in an optimal solution for CSC1D, cycles are just some disjoint line segments visually, each having one or more sensors moving back and forth on it. In fact, this special case for CSC is not NP-hard and can be solved in polynomial time. We introduce two algorithms for this special case, both of which are proved to be optimal.

4.2.1 LineSplit-DP (An Optimal Algorithm for CSC1D)

First, we provide a dynamic programming based algorithm named *LineSplit-DP* to solve this case. Denote t_{ij} $(i, j \ge 1)$ as the minimum global sweep period if we cover the first i PoIs $p_1, p_2, \ldots, p_i$ using just j sensors. After a careful analysis, we obtain the recurrence relation in (18). The first two cases are rather straightforward. If $i = 1$ (i.e., there is only one PoI), the minimum global sweep period is zero; if $j = 1$, then we only have one sensor to cover all PoIs, the minimum global sweep period is $2d(p_1, p_i)$. For the last case, consider dividing the optimal solution into two parts: covering the leftmost u PoIs $\{p_1, p_2, \ldots, p_u\}$ with v sensors and leaving the rest PoIs $\{p_{u+1}, p_{u+2}, \ldots, p_i\}$ as a whole to the remaining $(j - v)$ sensors. Such division must exist based on our observation that cycles in the optimal solution are visually disjoint line segments, each containing several continuous PoIs. Thus, the solution given by (18) is optimal.

$$t_{ij} = \begin{cases} 0 & \text{if } i = 1 \\ 2d(p_1, p_i) & \text{if } j = 1 \\ \min\limits_{\substack{1\le u<i \\ 1\le v<j}} \left\{\max\left\{t_{uv}, \frac{2d(p_{u+1}, p_i)}{j-v}\right\}\right\} & \text{otherwise} \end{cases} \tag{18}$$

Based on this recurrence relation, we implement *LineSplit-DP* in Algorithm 4 using a bottom-up approach.

Algorithm 4: LineSplit-DP

input : n PoIs $P = \{p_1, p_2, \ldots, p_n\}$ in a straight line, $d : P \times P \to \mathbb{R}^+$ and m sensors
output: A set of disjoint PoI groups (the union of which is P) $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r\}$ where $\mathcal{P}_i$ consists of several continuous PoIs, and a sensor allocation $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ for $\mathscr{P}$

```
t_1j ← 0; t_i1 ← 2d(p_1, p_i);
𝒫_1j ← {{p_1}}; 𝒫_i1 ← {{p_1, p_2, ..., p_i}};
𝒜_1j ← {j}; 𝒜_i1 ← {1};
for i ← 2 to n, j ← 2 to m do
    for u ← 1 to i − 1, v ← 1 to j − 1 do
        if t_ij > max{t_uv, 2d(p_{u+1}, p_i)/(j−v)} then
            t_ij ← max{t_uv, 2d(p_{u+1}, p_i)/(j−v)};
            𝒫_ij ← 𝒫_uv ∪ {{p_{u+1}, p_{u+2}, ..., p_i}};
            𝒜_ij ← 𝒜_uv ∪ {j − v};
return (𝒫_nm, 𝒜_nm);
```

After obtaining the partition of PoIs, sensors allocated to each group just move back and forth from the leftmost PoI to the rightmost PoI in the group. Now we prove the optimality of *LineSplit-DP* in the following theorem.

Theorem 2. LineSplit-DP *is an optimal algorithm for the CSC1D problem, which means its solution achieves the minimum global sweep period.*

Proof. Algorithm 4 evaluates t_{ij} in a bottom-up approach. At the time of t_{ij} being evaluated, t_{uv} $(1 \le u < i, 1 \le v < j)$ would all have been computed. As a result, the coverage scheme $(\mathscr{P}_{nm}, \mathcal{A}_{nm})$ returned by Algorithm 4 produces the minimum global sweep period t_{nm} for the CSC1D problem. This finishes the proof of the theorem. □

For a naive implementation as shown in Algorithm 4, *LineSplit-DP* has a time complexity of $O(m^2n^2)$ and a space complexity of $O(m^2n)$. (Note that each group in $\mathscr{P}_{ij}$ can be represented as a constant-size tuple containing only the start and the end PoIs.)

4.2.2 LineSplit-Greedy (An Efficient Algorithm for CSC1D)

As we can see above, the time complexity of *LineSplit-DP* is too high. After a further analysis, we find that the recurrence relation in (18) can be simplified according to Lemma 6.

Lemma 6. *In the optimal solution of CSC1D, no PoI will be covered by two or more sensors.*

Proof. For simplicity, here we denote $d(p_i, p_{i+1})$ as l_i, which is the distance between p_i and p_{i+1} as shown in Figure 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3008348, IEEE Transactions on Mobile Computing

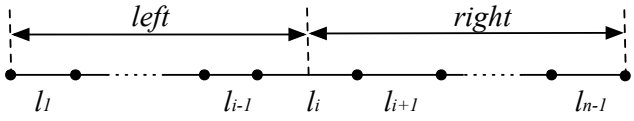


Fig. 2. An illustration for optimal coverage in CSC1D

Suppose initially we have 2 sensors to cover n PoIs, so the global sweep period t should be

$$t = \frac{2\sum_{k=1}^{n-1} l_k}{2} = \sum_{k=1}^{n-1} l_k. \tag{19}$$

If we equally divide the whole segment into two parts, *left* and *right*, as shown in Figure 2, with the division line at somewhere between p_i and p_{i+1}. Then we have

$$2\sum_{k=1}^{i-1} l_k < \sum_{k=1}^{n-1} l_k \quad \text{and} \quad 2\sum_{k=i+1}^{n-1} l_k < \sum_{k=1}^{n-1} l_k. \tag{20}$$

If we place the two sensors separately in the *left* and *right* part, the global sweep period t' satisfies

$$t' = \max\{t_{left}, t_{right}\} \tag{21}$$

$$= \max\left\{2\sum_{k=1}^{i-1} l_k, 2\sum_{k=i+1}^{n-1} l_k\right\} \tag{22}$$

$$< \sum_{k=1}^{n-1} l_k = t. \tag{23}$$

This means that the non-cooperative sweep coverage approach is optimal in CSC1D, which proves this lemma. □

Now that we know in the optimal solution, the coverage areas of sensors have no overlap, we can fix the number of PoIs and only keep the for-loop over the number of sensors. Bearing this idea, Algorithm 5 describes a modified optimal algorithm for CSC1D named *LineSplit-Greedy*.

In Line 1, we only have one sensor, so the optimal coverage scheme is obviously the only scheme. Then in each iteration with a new sensor added, we first find the longest line segment between PoIs in the same group and divide it into two parts in Line 3-6. Naturally, we put PoIs in different parts of the line segment into two different subgroups. When there is a PoI right on the division line, we break the tie by comparing the distances to its left and right neighbors. This strategy is described in Line 7-16. Finally, in Line 17, we replace the old PoI group with two new subgroups and carry on to the next iteration. After obtaining the partition of PoIs in Line 18, we allocate one sensor to each group, and again have it move back and forth from the leftmost PoI to the rightmost PoI in its group.

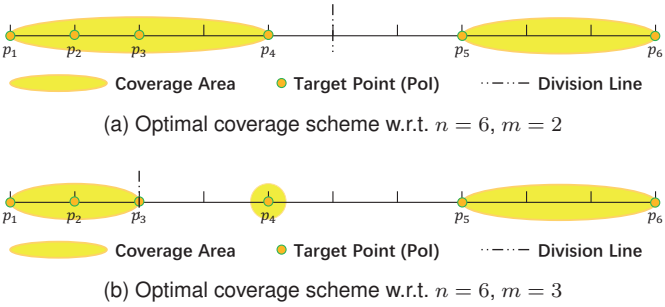
We give an example in Figure 3. In Figure 3a, the optimal coverage scheme with respect to $n = 6$ and $m = 2$ is already given by *LineSplit-Greedy*. If we add a new sensor, we find that there is exactly one PoI p_3 on the division line. According to the strategy discussed above, since the distance between p_3 and p_4 is longer than that between p_2 and p_3, we put p_3 in the left subgroup to give the optimal coverage scheme with respect to $m = 3$ in Figure 3b.

Algorithm 5: LineSplit-Greedy

input : n PoIs $P = \{p_1, p_2, \ldots, p_n\}$ in a straight line, $d : P \times P \to \mathbb{R}^+$ and m sensors

output: A set of disjoint PoI groups (the union of which is P) $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$ where $\mathcal{P}_i$ consists of several continuous PoIs

1 $\mathscr{P}_1 \leftarrow \{P\}$;
2 **for** $j \leftarrow 2$ **to** m **do**
3 $\quad \hat{k} \leftarrow \arg\max_{\mathcal{P}_i \in \mathscr{P}_{j-1}} \left\{ \max_{x,y \in \mathcal{P}_i} \{d(x, y)\} \right\}$;
4 $\quad p_u \leftarrow$ the leftmost PoI in $\mathcal{P}_{\hat{k}}$;
5 $\quad p_v \leftarrow$ the rightmost PoI in $\mathcal{P}_{\hat{k}}$;
6 $\quad$ Divide $\mathcal{P}_{\hat{k}}$ into two subgroups $\mathcal{P}_L$ and $\mathcal{P}_R$ by cutting the line segment connecting p_u and p_v from the middle, the division line of which is ℓ_j;
7 $\quad$ **if** *some PoI* p_c $(u < c < v)$ *is right on* ℓ_j **then**
8 $\quad\quad$ **if** $d(p_{c-1}, p_c) \le d(p_c, p_{c+1})$ **then**
9 $\quad\quad\quad \mathcal{P}_L \leftarrow \{p_u, p_{u+1}, \ldots, p_c\}$;
10 $\quad\quad\quad \mathcal{P}_R \leftarrow \{p_{c+1}, p_{c+2}, \ldots, p_v\}$;
11 $\quad\quad$ **else**
12 $\quad\quad\quad \mathcal{P}_L \leftarrow \{p_u, p_{u+1}, \ldots, p_{c-1}\}$;
13 $\quad\quad\quad \mathcal{P}_R \leftarrow \{p_c, p_{c+1}, \ldots, p_v\}$;
14 $\quad$ **else**
15 $\quad\quad \mathcal{P}_L \leftarrow \mathcal{P}_{\hat{k}} \cap \{$PoIs to the left of $\ell_j\}$;
16 $\quad\quad \mathcal{P}_R \leftarrow \mathcal{P}_{\hat{k}} \cap \{$PoIs to the right of $\ell_j\}$;
17 $\quad \mathscr{P}_j \leftarrow (\mathscr{P}_{j-1} \setminus \{\mathcal{P}_{\hat{k}}\}) \cup \{\mathcal{P}_L, \mathcal{P}_R\}$;
18 **return** $\mathscr{P}_m$;



(a) Optimal coverage scheme w.r.t. $n = 6, m = 2$

(b) Optimal coverage scheme w.r.t. $n = 6, m = 3$

Fig. 3. An example for optimal coverage in CSC1D with $n = 6$

Theorem 3. LineSplit-Greedy *is an optimal algorithm for the CSC1D problem, which means its solution achieves the minimum global sweep period.*

Proof. By Lemma 6, we know that no PoI is covered by two or more sensors in the optimal solution of CSC1D. We also know that if we are to minimize the global sweep period, the optimal division is cutting the original line segment from the middle. These facts prove the optimality of our division strategy for the PoI group.

Next, we prove the original theorem by induction on m:

- When $m = 1$, there is only one possible coverage scheme and it is certainly optimal.
- Assume *LineSplit-Greedy* delivers an optimal coverage scheme when $m = k$ and its global sweep period

is t_k, which is dominated by PoI group $\mathcal{P}_{i_k}$, i.e.,

$$t_k = 2 \times \max_{x,y \in \mathcal{P}_{i_k}} \{d(x,y)\}. \quad (24)$$

- When $m = k+1$, if we follow Algorithm 5 and add the new sensor to $\mathcal{P}_{i_k}$, then t_{k+1} has a chance to be smaller than, if not equal to, t_k. Otherwise, t_{k+1} will remain the same as t_k. Since we have proved the optimality of our division strategy for $\mathcal{P}_{i_k}$ above, we can conclude that *LineSplit-Greedy* also delivers an optimal coverage scheme when $m = k+1$.

This finishes the proof of this theorem. □

From Algorithm 5, we can find that *LineSplit-Greedy* has a time complexity of $O(mn)$ and a space complexity of $O(n)$, which is much more efficient than *LineSplit-DP*. (Note that if we maintain a max-heap for Line 3 and 17, and use the binary search strategy for Line 6, the time complexity can be further reduced to $O(m \log n)$.)

5 Multi-Sink Sweep Coverage

Usually, the limited storage capacity and battery power constraints require mobile sensors to transmit their collected data to base stations (also called "sinks") periodically. For example, drones need to be refueled after several hours of patrolling in a certain region, making the basic CSC formulation not feasible in such real-world applications. In this section, we consider a more realistic sweep coverage problem named Multi-Sink Sweep Coverage (MSSC) which takes base station visiting events into consideration. An approximation is designed for the general case of this problem followed by the theoretical analysis regarding its approximation ratio. We also give an insight regrading a special case of MSSC where there is only one available sink.

5.1 MSSC: General Case

In this subsection, we will talk about the Multi-Sink Sweep Coverage problem in general case. We firstly give a formal definition of MSSC, and then propose a novel approximation algorithm named *SinkCycle* to solve it, the approximation ratio of which is 6.

5.1.1 Problem Formulation

Suppose there are multiple static data sinks and each sensor has to approach at least one of them during each sweep cycle. Then how to compute trajectory cycles for mobile sensors with this new constraint raises a challenging problem. We define it as *Multi-Sink Sweep Coverage* (MSSC).

Definition 8 (Multi-Sink Sweep Coverage (MSSC))**.** *Given input (G, S, m, d), where $G = (V, E)$ is a complete graph, $S \subset V$ is a set of sinks with $|S| \leq m$, m is the number of sensors and $d : E \to \mathbb{R}^+$ is a metric, Multi-Sink Sweep Coverage (MSSC) aims to find a coverage scheme $\mathcal{F} = (\mathcal{V}, \mathscr{C}, \mathcal{A})$, consisting of a vertex set $\mathcal{V}$ with $(V \setminus S) \subset \mathcal{V} \subseteq V$, a cycle cover $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r\}$ for $\mathcal{V}$ with at least one sink in each $\mathcal{C}_i \in \mathscr{C}$, and a sensor allocation $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ for $\mathscr{C}$, such that $\max_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\}$ is minimized.*

Again, we use $\frac{d(\mathcal{C}_i)}{m_i}$ and $\max_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\}$ to represent the *sweep period* of the PoIs in $\mathcal{C}_i$ and the *global sweep period* of all PoIs respectively for simplicity.

Figure 4 is an illustration for MSSC, where 8 mobile sensors work cooperatively to cover 17 PoIs along 4 distinct trajectory cycles with a data sink on each cycle to collect sensing data.

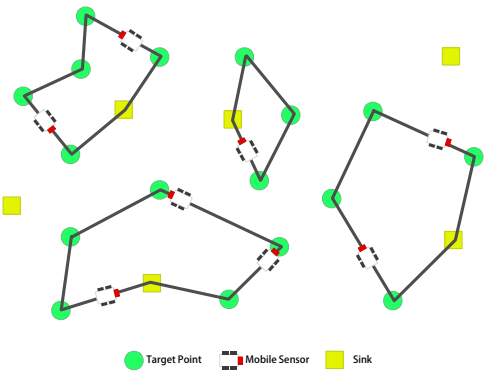


Fig. 4. An illustration for multi-sink sweep coverage problem

5.1.2 SinkCycle (a Novel Approximation for MSSC)

Now we introduce the *SinkCycle* algorithm for the MSSC problem. It integrates the design idea from *CoCycle* together with the modified Prim's algorithm to first find a tree cover for all PoIs and some data sinks, then determine the optimal sensor allocation among these trees and finally transform trees to distinct cycles to construct a desired cycle cover. Therefore, we also define a variant problem called *Multi-Sink Tree Coverage* (MSTC) as follows.

Definition 9 (Multi-Sink Tree Coverage (MSTC))**.** *Given input (G, S, m, d), where $G = (V, E)$ is a complete graph, $S \subset V$ is a set of sinks with $|S| \leq m$, m is the number of sensors and $d : E \to \mathbb{R}^+$ is a metric, Multi-Sink Tree Coverage (MSTC) aims to find a coverage scheme $\mathcal{F} = (\mathcal{V}, \mathscr{T}, \mathcal{A})$, consisting of a vertex set $\mathcal{V}$ with $(V \setminus S) \subset \mathcal{V} \subseteq V$, a tree cover $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$ for $\mathcal{V}$ with at least one sink in each $\mathcal{T}_i \in \mathscr{T}$, and a sensor allocation $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ for $\mathscr{T}$, such that $\max_{1 \leq i \leq r} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$ is minimized.*

Similar to Lemma 2, we can get the following lemma.

Lemma 7. *If there is an α-approximation algorithm for MSTC, then there is a 2α-approximation algorithm for MSSC.*

Next, we design the *SinkTree* algorithm to solve the MSTC problem and the *SinkCycle* algorithm follows naturally. To better describe *SinkTree*, we need the following definition.

Definition 10 (Tree Cover with Roots)**.** *Given a vertex set V and a set of roots $S \subseteq V$, if a tree cover $\mathscr{T}$ for V has exactly one root from S in each tree, then $\mathscr{T}$ is a tree cover with roots S for V, denoted as $\mathscr{T} \uparrow S$.*

Given a graph $G = (V, E)$ and a sink set $S \subset V$, a minimum tree cover with roots S for V can be found through Algorithm 6, whose optimality is guaranteed by the correctness of Prim's algorithm.

Algorithm 6: Modified Prim's Algorithm

input : $G = (V, E)$, $S \subset V$, $d : E \to \mathbb{R}^+$
output: A tree cover $\hat{\mathscr{T}}$ with roots S for V

```
1  foreach v ∈ V do
2      pre(v) ← arg min_{s∈S} {d(s, v)};
3  V' ← S; T̂ ← ∅;
4  while V' ≠ V do
5      u ← arg min_{v∈V\V'} {d(v, pre(v))};
6      V' ← V' ∪ {u}; T̂ ← T̂ ∪ {(u, pre(u))};
7      foreach v ∈ V \ V' do
8          if d(v, u) < d(v, pre(v)) then
9              pre(v) ← u;
10 return T̂;
```

Intuitively, we find that a minimum tree cover $\hat{\mathscr{T}}_0$ with roots S can produce a feasible solution to MSTC. However, trees in $\hat{\mathscr{T}}_0$ can be severely unbalanced such that their weights (i.e., $d(\mathcal{T})$ where $\mathcal{T} \in \hat{\mathscr{T}}_0$) may vary greatly. If we do not have enough mobile sensors to balance them, the global sweep period can be very large. Therefore, after obtaining $\hat{\mathscr{T}}_0$, we need to add more edges to merge some spanning trees. Based on this thought, we introduce the *SinkTree* algorithm as shown in Algorithm 7.

Algorithm 7: SinkTree

input : $G = (V, E)$, $S \subset V$, $d : E \to \mathbb{R}^+$ and m sensors
output: A vertex set $\mathcal{V}$ with $(V \setminus S) \subset \mathcal{V} \subseteq V$, a tree cover $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$ for $\mathcal{V}$ with at least one sink in each $\mathcal{T}_i \in \mathscr{T}$, and a sensor allocation $\mathcal{A} = \{m_1, m_2, \ldots, m_r\}$ for $\mathscr{T}$

```
1  Find a tree cover T̂_0 with roots S by Algorithm 6, whose overall edge cost is minimized.;
2  i ← 0; E_0 ← the set of all edges in T̂_0; G_0 ← (V, E_0);
3  T_0 ← {T | T ∈ T̂_0 and T ∩ (V \ S) ≠ ∅};
4  if |T_0| ≤ m then
5      V_0 ← the set of all vertices in T_0;
6      Find an optimal sensor allocation A_0 for T_0;
7  foreach e ∈ E \ E_0 (chosen in ascending order by d(·)) do
8      if adding e to G_i does not produce a cycle then
9          i ← i + 1; e_i ← e;
10         E_i ← E_{i−1} ∪ {e}; G_i ← (V, E_i);
11         Obtain a tree cover T̂_i directly from G_i;
12         T_i ← {T | T ∈ T̂_i and T ∩ (V \ S) ≠ ∅};
13         if |T_i| ≤ m then
14             V_i ← the set of all vertices in T_i;
15             Find an optimal sensor allocation A_i for T_i;
16 return the best coverage scheme in {(V_i, T_i, A_i)} (with minimum global sweep period max_i {d(T_i)/m_i});
```

In Line 1, we find a minimum tree cover with roots S. Then in Line 2-6, we use this tree cover to do the initialization steps. In Line 7-15, we continuously add edges into the edge set E_i in an ascending order of edge length. During this process, we ensure that the adding edge will not produce a cycle and no sensor will be allocated to a tree without PoIs. In Line 16, we find the best coverage scheme among all the considered schemes.

Based on the coverage scheme $\mathcal{F}_{\text{MSTC}} = (\mathcal{V}, \mathscr{T}, \mathcal{A})$ returned by *SinkTree*, *SinkCycle* just uses Algorithm 1 (with or without the Christofides' improvements) on each tree in $\mathscr{T}$ respectively to transform the tree cover $\mathscr{T}$ to a cycle cover $\mathscr{C}$, and returns a corresponding coverage scheme $\mathcal{F}_{\text{MSSC}} = (\mathcal{V}, \mathscr{C}, \mathcal{A})$ as its solution for the MSSC problem.

5.1.3 Performance Analysis

Now we will prove the approximation ratios of the proposed *SinkTree* and *SinkCycle* algorithms. Suppose we have an optimal solution $\mathcal{F}^*_{\text{MSTC}} = (\mathcal{V}^*, \mathscr{T}^*, \mathcal{A}^*)$ for the MSTC problem, where $\mathscr{T}^* = \{\mathcal{T}^*_1, \mathcal{T}^*_2, \ldots, \mathcal{T}^*_{r^*}\}$ and $\mathcal{A}^* = \{m^*_1, m^*_2, \ldots, m^*_{r^*}\}$. Set

$$\text{OPT} = \max_{1 \le i \le r^*} \left\{ \frac{d(\mathcal{T}^*_i)}{m^*_i} \right\} \tag{25}$$

and

$$\sigma = \underset{\substack{1 \le i \le |S|-1 \\ d(e_i) \le \text{OPT}}}{\arg\max} \{d(e_i)\}. \tag{26}$$

Suppose the connected components in G_σ are $CC^\sigma_1, CC^\sigma_2, \ldots$, and we use $\mathcal{T}^\sigma_1, \mathcal{T}^\sigma_2, \ldots$ to denote their corresponding spanning trees.

Lemma 8. *There is a coverage scheme $\mathcal{F}'_{MSTC} = (\mathcal{V}', \mathscr{T}', \mathcal{A}')$ such that all vertices of $\mathcal{T}' \in \mathscr{T}'$ belong to the same connected component of G_σ and $\max_i \left\{ \frac{d(\mathcal{T}'_i)}{m'_i} \right\} \le 2 \times OPT$.*

Proof. We construct such a coverage scheme $\mathcal{F}'_{\text{MSTC}}$ from the optimal cover scheme $\mathcal{F}^*_{\text{MSTC}}$.

It is obvious that $\hat{\mathscr{T}}_0$ as well as $\mathscr{T}^*$ has only one sink in each tree. Use $\hat{\mathcal{T}}^0_{\uparrow s}$ and $\mathcal{T}^*_{\uparrow s}$ to represent the tree rooted at sink s in $\hat{\mathscr{T}}_0$ and $\mathscr{T}^*$ respectively. Consider an edge $(u, v) \in \mathcal{T}^*_{\uparrow s_1}$ which connects two different connected components in G_σ, then we have $d(u, v) > \text{OPT}$. Suppose $u \in \hat{\mathcal{T}}^0_{\uparrow s_1}$, $v \in \hat{\mathcal{T}}^0_{\uparrow s_2}$ and w is the parent of v in $\hat{\mathcal{T}}^0_{\uparrow s_2}$. Deleting edge (u, v) divides $\mathcal{T}^*_{\uparrow s_1}$ into two parts, $\mathcal{T}_1$ and $\mathcal{T}_2$ ($\mathcal{T}_1$ contains s_1).

Notice that $d(v, w) \le d(u, v)$ (otherwise, replacing (v, w) with (u, v) in $\hat{\mathscr{T}}_0$ will result in a smaller tree cover with roots S, which contradicts the optimality of $\hat{\mathscr{T}}_0$). We replace (u, v) with (v, w) in $\mathscr{T}^*$ to get a new tree cover. Basically, $\mathcal{T}^*_{\uparrow s_1}$ becomes $\mathcal{T}'_{\uparrow s_1} = \mathcal{T}^*_{\uparrow s_1} - \{(u, v)\} - \mathcal{T}_2$, and $\mathcal{T}^*_{\uparrow s_2}$ becomes $\mathcal{T}'_{\uparrow s_2} = \mathcal{T}^*_{\uparrow s_2} + \{(v, w)\} + \mathcal{T}_2$.

Suppose in $\mathcal{A}^*$, the corresponding numbers of sensors for $\mathcal{T}^*_{\uparrow s_1}$ and $\mathcal{T}^*_{\uparrow s_2}$ are $m^*_{\uparrow s_1}$ and $m^*_{\uparrow s_2}$ respectively. If we assign $m'_{\uparrow s_1} = \lceil \frac{d(\mathcal{T}'_{\uparrow s_1})}{\text{OPT}} \rceil$ sensors to $\mathcal{T}'_{\uparrow s_1}$, and $m'_{\uparrow s_2} = m^*_{\uparrow s_2} + m^*_{\uparrow s_1} - m'_{\uparrow s_1}$ sensors to $\mathcal{T}'_{\uparrow s_2}$, we will have

$$\frac{d(\mathcal{T}'_{\uparrow s_1})}{m'_{\uparrow s_1}} \le \text{OPT}, \tag{27}$$

and

$$\frac{d(\mathcal{T}'_{\uparrow s_2})}{m'_{\uparrow s_2}} = \frac{d(\mathcal{T}^*_{\uparrow s_2}) + d(v,w) + d(\mathcal{T}_2)}{m^*_{\uparrow s_2} + m^*_{\uparrow s_1} - m'_{\uparrow s_1}} \tag{28}$$

$$\leq \frac{d(\mathcal{T}^*_{\uparrow s_2}) + d(u,v) + d(\mathcal{T}_2)}{m^*_{\uparrow s_2} + m^*_{\uparrow s_1} - m'_{\uparrow s_1}} \tag{29}$$

$$= \frac{d(\mathcal{T}^*_{\uparrow s_2}) + d(\mathcal{T}^*_{\uparrow s_1}) - d(\mathcal{T}'_{\uparrow s_1})}{m^*_{\uparrow s_2} + m^*_{\uparrow s_1} - m'_{\uparrow s_1}} \tag{30}$$

$$\leq \frac{(m^*_{\uparrow s_2} + m^*_{\uparrow s_1} - (m'_{\uparrow s_1} - 1)) \times \mathrm{OPT}}{m^*_{\uparrow s_2} + m^*_{\uparrow s_1} - m'_{\uparrow s_1}} \tag{31}$$

$$\leq 2 \times \mathrm{OPT}. \tag{32}$$

By replacing all such edges, we get a new coverage scheme $\mathcal{F}'_{\mathrm{MSTC}} = (\mathcal{V}', \mathscr{T}', \mathcal{A}')$ satisfying the property described in this lemma. □

Now we continue to prove the approximation ratio of *SinkTree* using $\mathcal{F}'_{\mathrm{MSTC}} = (\mathcal{V}', \mathscr{T}', \mathcal{A}')$.

Theorem 4. SinkTree *is a 3-approximation for MSTC.*

Proof. Suppose CC^σ_i contains μ^σ_i trees from $\mathscr{T}'$. Then CC^σ_i also contains μ^σ_i sinks, which means CC^σ_i contains μ^σ_i trees from $\hat{\mathscr{T}}_0$. From the optimality of $\hat{\mathscr{T}}_0$, we have

$$\sum_{\mathcal{T} \in (\hat{\mathscr{T}}_0 \cap CC^\sigma_i)} d(\mathcal{T}) \leq \sum_{\mathcal{T} \in (\mathscr{T}' \cap CC^\sigma_i)} d(\mathcal{T}). \tag{33}$$

If these μ^σ_i trees use totally λ^σ_i ($\geq \mu^\sigma_i$) sensors in $\mathcal{F}'_{\mathrm{MSTC}}$. Then after Algorithm 7 adds $(\mu^\sigma_i - 1)$ edges to connect the trees in $(\hat{\mathscr{T}}_0 \cap CC^\sigma_i)$, we have

$$d(\mathcal{T}^\sigma_i) \leq \sum_{\mathcal{T} \in (\hat{\mathscr{T}}_0 \cap CC^\sigma_i)} d(\mathcal{T}) + (\mu^\sigma_i - 1) \times \mathrm{OPT} \tag{34}$$

$$\leq \sum_{\mathcal{T} \in (\mathscr{T}' \cap CC^\sigma_i)} d(\mathcal{T}) + (\mu^\sigma_i - 1) \times \mathrm{OPT} \tag{35}$$

$$\leq 2\lambda^\sigma_i \times \mathrm{OPT} + (\mu^\sigma_i - 1) \times \mathrm{OPT}, \tag{36}$$

which gives

$$\frac{d(\mathcal{T}^\sigma_i)}{\lambda^\sigma_i} \leq 3 \times \mathrm{OPT}. \tag{37}$$

By Lemma 3, we can find an optimal sensor allocation $\mathcal{A}_\sigma = \{m^\sigma_i\}$ for $\mathscr{T}_\sigma$ whose global sweep period will satisfy

$$\max_i \left\{ \frac{d(\mathcal{T}^\sigma_i)}{m^\sigma_i} \right\} \leq \max_i \left\{ \frac{d(\mathcal{T}^\sigma_i)}{\lambda^\sigma_i} \right\} \leq 3 \times \mathrm{OPT}. \tag{38}$$

Therefore, we can conclude that *SinkTree* is a 3-approximation for MSTC. □

According to Lemma 7, we get the following corollary.

Corollary 2. SinkCycle *is a 6-approximation for MSSC.*

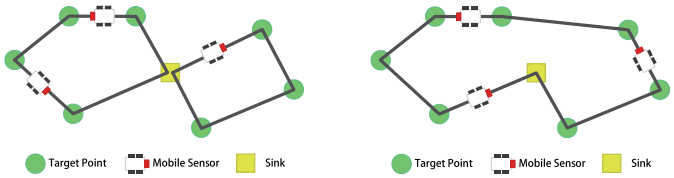
Similar to *CoCycle*, the overall time complexity of *SinkCycle* is also $O(n^2 \log n + m^3)$. Note that Algorithm 6 runs in $O(n^2)$ time as Prim's algorithm.

5.2 MSSC: Single Sink Case

In this subsection, we consider a special case of MSSC where there is only one available sink and different sweep cycles can share this sink together. After a careful investigation, we give the following insight regarding this special case.

Theorem 5. *If there is only one sink in MSSC and sink sharing is allowed, then its optimal solution is to find a minimum Hamiltonian cycle through all PoIs and this sink with all sensors working cooperatively along this cycle.*

Proof. Suppose some sensors work on two different cycles, $\mathcal{C}_1$ and $\mathcal{C}_2$, in the optimal solution, which are respectively allocated m_1 and m_2 sensors as shown in Figure 5a. Since we only have one sink s, $\mathcal{C}_1$ and $\mathcal{C}_2$ must both contain s.



(a) Before reallocation (b) After reallocation

Fig. 5. An illustration for single-sink sweep coverage problem

If we join these two cycles together as $\mathcal{C}$ as shown in Figure 5b, according to triangle inequality, we have

$$d(\mathcal{C}) \leq d(\mathcal{C}_1) + d(\mathcal{C}_2). \tag{39}$$

Now reallocate the $(m_1 + m_2)$ sensors to $\mathcal{C}$. We can get a better global sweep period

$$\frac{d(\mathcal{C})}{m_1 + m_2} \leq \frac{d(\mathcal{C}_1) + d(\mathcal{C}_2)}{m_1 + m_2} \leq \max\left\{ \frac{d(\mathcal{C}_1)}{m_1}, \frac{d(\mathcal{C}_2)}{m_2} \right\}. \tag{40}$$

This finishes the proof of this theorem. □

Theorem 5 also reveals the fact that in the optimal coverage scheme of MSSC, sweep cycles do not share sinks. This justifies our choice of cycle covers (i.e., no sink sharing) as solutions to the MSSC problem in general case.

6 Performance Evaluation

In this section, we evaluate the performances of *CoCycle* and *SinkCycle*, and compare them with other previous works on a simulation platform written in Python 3.6. All simulations are carried out on a standard Intel(R) Core(TM) i7-8750H CPU @ 2.2GHz processor.

6.1 Algorithms in Comparison

We find several previous algorithms for evaluation purpose and their brief descriptions are as follows.

- OSweep [24] is an algorithm for the CSC problem. It first uses a polynomial time approximation scheme (PTAS) to find a TSP cycle, and then allocates the sensors evenly on this cycle, making them move towards the same direction to cooperatively cover all PoIs. Although the approximation ratio of the PTAS is fairly good, its time complexity is too high

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3008348, IEEE Transactions on Mobile Computing

in practice. In our simulations, we use a simpler MST based algorithm to find the TSP cycle for OSweep.

- MinExpand [24] is a heuristic algorithm for the CSC problem. Its objective is to find the minimum number of sensors with a fixed global sweep period. It selects a corner PoI as the starting point of a cycle, and constantly adds the PoI with the smallest path increment to this cycle before it fails to meet the sweep period requirement. Then it requests a new sensor and repeats the above procedure until there are no uncovered PoIs. In our simulations, although MinExpand has a different objective from the definition of CSC, we use the binary search strategy to find the minimum sweep period it can achieve with no more than m sensors.
- PDBA [36] is a randomized heuristic algorithm for the CSC problem. Its objective is to find the minimum number of sensors with a fixed global sweep period as well. It randomly selects a PoI as the starting point of a cycle, and constantly expands this cycle by adding the PoI with the minimum perpendicular distance to the last line segment of the cycle. When this cycle cannot meet the sweep period requirement, it requests a new sensor and repeats this procedure until all PoIs are covered. In our simulations, we again use the binary search strategy to find the minimum sweep period it can achieve with no more than m sensors. Since it is a randomized algorithm, we also run it several times and take its average performance for comparison.
- SCOPe-M-Solver [37] is a heuristic algorithm for the MSSC problem. Its objective is to find the minimum number of sensors satisfying both the sweep period for PoIs and the base station (sink) visiting period for sensors. It first assigns each PoI to the cluster of its nearest base station, and then constructs cycles from the base station in each cluster by expanding in a similar way to MinExpand. By adjusting its period requirements and adopting the binary search strategy, we can find the minimum sweep period it can achieve with no more than m sensors.

Among these algorithms, we use OSweep, MinExpand and PDBA to compare with *CoCycle* and use SCOPe-M-Solver to compare with *SinkCycle* in our simulations.

6.2 Parameter Settings

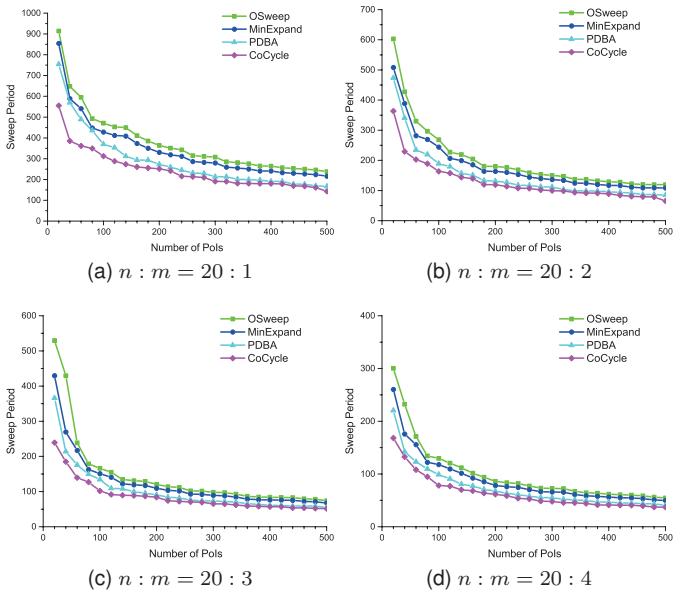
In our simulations, we prepare a 200 $\times$ 200 virtual plane and randomly deploy a number of PoIs. For simplicity, we use the *Euclidean Distance* as the metric $d(\cdot)$. For the CSC problem, in each simulation, we fix the ratio between the number of PoIs n and the number of sensors m, and vary the number of PoIs n from 20 to 500 with a step of 20. The ratio $(n : m)$ is chosen from $[(20 : 1), (20 : 2), (20 : 3), (20 : 4)]$. For the MSSC problem, we first fix the ratio among the number of PoIs n, the number of sinks k and the number of sensors m when varying the number of PoIs n from 20 to 500 with a step of 20. The ratio $(n : k : m)$ is chosen from $[(20 : 1 : 4), (20 : 2 : 4), (20 : 1 : 8), (20 : 2 : 8)]$. Then we fix the number of PoIs n and the number of sensors m, and vary the number of sinks k for 10 times in each simulation. Here, n is fixed to 200 and m is chosen from $[10, 20, 30, 40]$. k will not exceed m in our simulations since SCOPe-M-Solver may not be able to deliver a feasible solution otherwise.

6.3 Simulation Results

In this subsection, we present the simulation results followed by some discussions regarding the performances of the algorithms in comparison.

6.3.1 Results for CSC

Figure 6 shows the simulation results of varying n when fixing the ratio $(n : m)$ to $(20 : 1)$, $(20 : 2)$, $(20 : 3)$, $(20 : 4)$ respectively. As we can see, the global sweep period decreases when we have more PoIs while maintaining the same ratio of $(n : m)$ in the restricted area. The increase in the density of PoIs allows us to cover more of them using almost the same number of sensors without compromising the sweep period requirement. Apparently, we can also decrease the sweep period by using more sensors given the same number of PoIs.



(a) $n : m = 20 : 1$ (b) $n : m = 20 : 2$

(c) $n : m = 20 : 3$ (d) $n : m = 20 : 4$

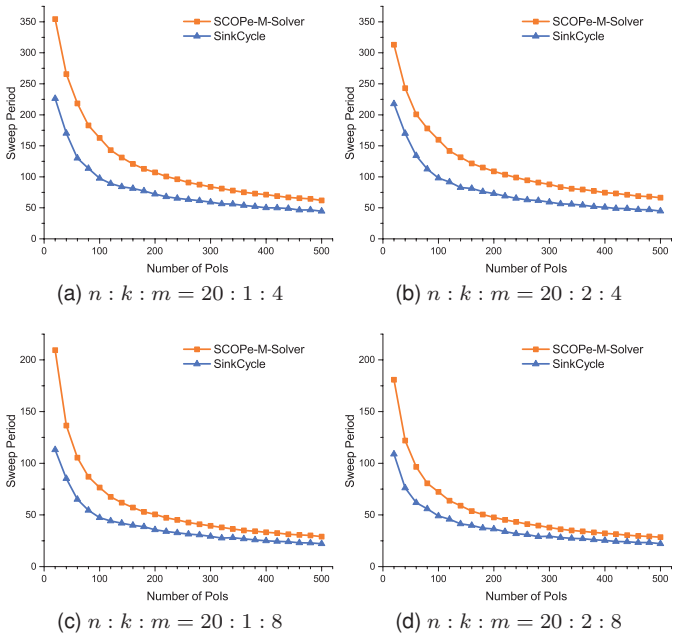
Fig. 6. Results of varying n in the CSC problem ($n = |V|$)

Among these four algorithms, *CoCycle* delivers the best results under most settings, especially when the number of sensors is relatively small. This is due to the cooperation among sensors enabled in *CoCycle*. Although OSweep also allows cooperative sweep coverage, it forces all sensors to move on a single TSP cycle through all PoIs, causing some of them to move extra distances. MinExpand and PDBA are relatively better than OSweep even though they do not enable any cooperation among sensors, and their performances are close to *CoCycle*. However, they have a common time complexity of $O(n^3)$, which makes them less efficient than *CoCycle* when there are a large number of PoIs. In our simulation, by setting $n = 1000$ and $m = 50$, the average running time is 743ms for *CoCycle*, 851ms for MinExpand and 1097ms for PDBA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3008348, IEEE Transactions on Mobile Computing

6.3.2 Results for MSSC

Figure 7 shows the simulation results of varying n when fixing the ratio $(n : k : m)$ to $(20 : 1 : 4)$, $(20 : 2 : 4)$, $(20 : 1 : 8)$, $(20 : 2 : 8)$ respectively. As is shown in the figure, the global sweep period decreases when we have more PoIs while maintaining the same ratio of $(n : k : m)$ in the restricted area. The increase of the number of sensors can lead to the decrease in the sweep period for both *SinkCycle* and SCOPe-M-Solver. However, the increase in the number of sinks only has a noticeable influence on the sweep period for SCOPe-M-Solver while the performance of *SinkCycle* remains almost the same.

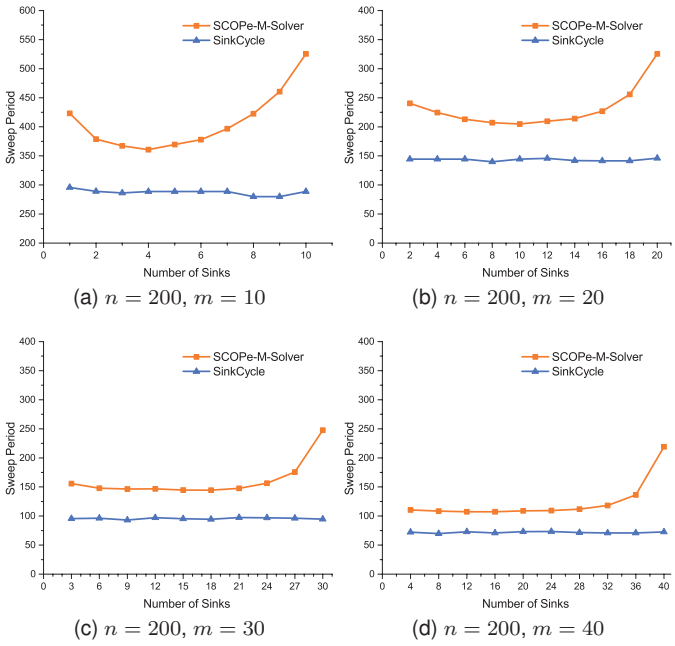


(a) $n : k : m = 20 : 1 : 4$ (b) $n : k : m = 20 : 2 : 4$

(c) $n : k : m = 20 : 1 : 8$ (d) $n : k : m = 20 : 2 : 8$

Fig. 7. Results of varying n in the MSSC problem ($n = |V|$, $k = |S|$)

To further investigate the phenomenon described above, we conduct more simulations by varying the number of sinks when fixing both the number of PoIs and the number of sensors. Figure 8 shows the simulation results of varying k when fixing n to 200 and assigning k to 10, 20, 30, 40 respectively. Since SCOPe-M-Solver only includes one sink in each sweep cycle, the initial increases in the number of sinks can reduce the sweep period effectively by shortening the average edge length in the same cluster. However, with the further increases in the number of sinks, the benefit of including multiple sinks in each sweep cycle begins to reveal, and the performance of SCOPe-M-Solver becomes worse. Such behavior is especially noticeable when the number of sensors is relatively small (e.g., in Figure 8a and 8b). This explains the observation made by Yang et al. [37] that the number of base stations (sinks) only has a limited impact over the number of mobile sensors deployed by SCOPe-M-Solver. Thus in our simulations, its impact over the sweep period is also limited given a fixed number of sensors. On the contrary, the performance of *SinkCycle* is stable during this process and stays ahead of SCOPe-M-Solver all the time due to the consideration of the cross-sink cooperation in each sweep cycle, which is ignored by the clustering step of SCOPe-M-Solver.



(a) $n = 200$, $m = 10$ (b) $n = 200$, $m = 20$

(c) $n = 200$, $m = 30$ (d) $n = 200$, $m = 40$

Fig. 8. Results of varying k in the MSSC problem ($n = |V|$, $k = |S|$)

6.4 Evaluation on Real Dataset

In order to investigate the performance of our proposed algorithms in a more realistic scenario, we find a real-world PoI dataset on Kaggle [42] which includes over 400,000 unique positions of interest mentioned in wikipedia articles and other metadata. We select a region of 20×20 km from somewhere in Asian and filter out 2873 PoIs in this region. The number of sensors is chosen from $[200, 300, 400, 500]$ for both CSC and MSSC, and we randomly initialize 100 sinks for MSSC. The evaluation results are in Table 1.

TABLE 1
Results of sweep period on real dataset

Algorithm	$m = 200$	$m = 300$	$m = 400$	$m = 500$
OSweep	6.33	4.57	3.12	2.46
MinExpand	5.93	4.05	2.91	2.37
PDBA	5.69	3.94	2.72	2.25
CoCycle	5.51	3.59	2.54	2.19
SCOPe-M-Solver	7.28	4.90	3.47	2.94
SinkCycle	5.59	3.66	2.65	2.23

In general, the evaluation results on this real-world PoI dataset are consistent with the previous discussions. This validates the potential benefits of using our designs in practical applications.

7 Conclusion and Future Work

In this paper, we discuss two variations of the cooperative sweep coverage problem to minimize the maximum sweep period for PoIs, and design two constant-factor approximation algorithms respectively.

For the basic Cooperative Sweep Coverage (CSC) problem, We propose a 4-approximation named *CoCycle* to construct a desired cycle cover. Mobile sensors allocated to the same cycle are evenly deployed on this cycle and move towards the same direction to cooperatively cover all PoIs on it. For the Multi-Sink Sweep Coverage (MSSC) problem, we propose a 6-approximation named *SinkCycle* which integrates the design idea from *CoCycle* together with the modified Prim's algorithm to deliver an efficient solution. *SinkCycle* is the first constant-factor approximation for the sweep coverage problem with multiple sinks.

We also investigate the CSC problem in one dimensional case (CSC1D), and provide two optimal algorithms, namely *LineSplit-DP* and *LineSplit-Greedy*, for this special case. They are designed based on dynamic programming and the greedy strategy respectively. Besides, after analyzing a special case of the MSSC problem where there is only one available sink, we find that the optimal coverage scheme of the MSSC problem in general case does not share sinks between different sweep cycles.

Finally, we compare our algorithms with several previous works through extensive simulations. Specifically, we compare *CoCycle* with OSweep [24], MinExpand [24] and PDBA [36], and compare *SinkCycle* with SCOPe-M-Solver [37] to evaluate their performances. Both theoretical analysis and numerical experiments validate the effectiveness and efficiency of our designs.

There is a small possibility, however, that a large sweep cycle with multiple sensors but only one sink may be included in the solution to the MSSC problem, escalating the issue of limited storage capacity and power supply. Although we make sure that each PoI is assigned to its nearest sink at first and try our best to shorten the sweep period for all PoIs, we still cannot guarantee that such an issue will not become a bottleneck in practice. In the future, we will further discuss this issue and take the sink visiting period for sensors into consideration of the MSSC problem.

Acknowledgments

This work was supported by the National Key R&D Program of China [2019YFB2102200]; the National Natural Science Foundation of China [61872238, 61972254, 61672353], and the CCF-Huawei Database System Innovation Research Plan [CCF-Huawei DBIR2019002A]. The authors would like to thank Xudong Zhu, Yuchen Feng and Gehua Qin for their contribution on the early versions of this paper.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[3] N. Ahmed, S. S. Kanhere, and S. Jha, "Probabilistic coverage in wireless sensor networks," in *IEEE Conference on Local Computer Networks (LCN)*, 2005, pp. 672–681.

[4] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, vol. 3, 2005, pp. 1976–1984.

[5] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2002, pp. 299–308.

[6] X. Bai, Z. Yun, D. Xuan, T.-h. Lai, and W. Jia, "Deploying four-connectivity and full-coverage wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2008, pp. 296–300.

[7] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2005, pp. 284–298.

[8] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2008, pp. 411–420.

[9] S. He, J. Chen, X. Li, X. Shen, and Y. Sun, "Cost-effective barrier coverage by mobile sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2012, pp. 819–827.

[10] B. Liu, O. Dousse, P. Nain, and D. Towsley, "Dynamic coverage of mobile sensor networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 2, no. 24, pp. 301–311, 2013.

[11] B. Gorain and P. S. Mandal, "Approximation algorithms for sweep coverage in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 74, no. 8, pp. 2699–2707, 2014.

[12] M. Li, W. Cheng, K. Liu, Y. He, X. Li, and X. Liao, "Sweep coverage with mobile sensors," *IEEE Transactions on Mobile Computing (TMC)*, vol. 10, no. 11, pp. 1534–1545, 2011.

[13] B. Gorain and P. S. Mandal, "Approximation algorithm for sweep coverage on graph," *Information Processing Letters*, vol. 115, no. 9, pp. 712–718, 2015.

[14] N. Bisnik, A. Abouzeid, and V. Isler, "Stochastic event capture using mobile sensors subject to a quality metric," in *ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2006, pp. 98–109.

[15] X. Gao, X. Zhu, Y. Feng, F. Wu, and G. Chen, "Data ferry trajectory planning for sweep coverage problem with multiple mobile sensors," in *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2016, pp. 1–9.

[16] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *IEEE Conference on Computer Communications (INFOCOM)*, vol. 2, 2005, pp. 1407–1418.

[17] M. M. Bin Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006, pp. 37–48.

[18] R. Moazzez-Estanjini and I. C. Paschalidis, "On delay-minimized data harvesting with mobile elements in wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1191–1203, 2012.

[19] M. Xi, K. Wu, Y. Qi, J. Zhao, Y. Liu, and M. Li, "Run to potential: Sweep coverage in wireless sensor networks," in *IEEE International Conference on Parallel Processing (ICPP)*, 2009, pp. 50–57.

[20] D. Zhao, H. Ma, and L. Liu, "Mobile sensor scheduling for timely sweep coverage," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 1771–1776.

[21] B. Gorain and P. S. Mandal, "Point and area sweep coverage in wireless sensor networks," in *IEEE International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt)*, 2013, pp. 140–145.

[22] L. Shu, K.-w. Cheng, X.-w. Zhang, and J.-l. Zhou, "Periodic sweep coverage scheme based on periodic vehicle routing problem," *Journal of Networks*, vol. 9, no. 3, p. 726, 2014.

[23] J.-h. Roh and S. Jin, "Device control protocol using mobile phone," in *International Conference on Advanced Communication Technology (ICACT)*, 2014, pp. 355–359.

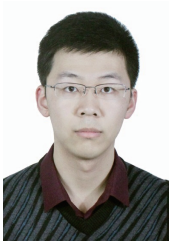
[24] J. Du, Y. Li, H. Liu, and K. Sha, "On sweep coverage with minimum mobile sensors," in *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 2010, pp. 283–290.

[25] Z. Chen, X. Zhu, X. Gao, F. Wu, J. Gu, and G. Chen, "Efficient scheduling strategies for mobile sensors in sweep coverage problem," in *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2016, pp. 1–4.

[26] X. Gao, J. Fan, F. Wu, and G. Chen, "Approximation algorithms for sweep coverage problem with multiple mobile sensors," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 2, pp. 990–1003, 2018.

[27] R. Sugihara and R. K. Gupta, "Optimal speed control of mobile node for data collection in sensor networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 9, no. 1, pp. 127–139, 2010.
[28] Y. Gu, D. Bozdağ, R. W. Brewer, and E. Ekici, "Data harvesting with mobile elements in wireless sensor networks," *Computer Networks*, vol. 50, no. 17, pp. 3449–3465, 2006.
[29] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," *IEEE Transactions on Mobile Computing (TMC)*, vol. 6, no. 4, pp. 395–410, 2007.
[30] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, Tech. Rep., 1976.
[31] M. Karpinski, M. Lampis, and R. Schmied, "New inapproximability bounds for tsp," *Journal of Computer and System Sciences (JCSS)*, vol. 81, no. 8, pp. 1665–1677, 2015.
[32] Z. Chen, S. Wu, X. Zhu, X. Gao, J. Gu, and G. Chen, "A route scheduling algorithm for the sweep coverage problem," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015, pp. 750–751.
[33] L. He, J. Pan, and J. Xu, "A progressive approach to reducing data collection latency in wireless sensor networks with mobile elements," *IEEE Transactions on Mobile Computing (TMC)*, vol. 12, no. 7, pp. 1308–1320, 2012.
[34] D. Kim, B. H. Abay, R. Uma, W. Wu, W. Wang, and A. O. Tokuta, "Minimizing data collection latency in wireless sensor network with multiple mobile elements," in *IEEE Conference on Computer Communications (INFOCOM)*, 2012, pp. 504–512.
[35] L. Xue, D. Kim, Y. Zhu, D. Li, W. Wang, and A. O. Tokuta, "Multiple heterogeneous data ferry trajectory planning in wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2014, pp. 2274–2282.
[36] B. H. Liu, N. T. Nguyen, and V. T. Pham, "An efficient method for sweep coverage with minimum mobile sensor," in *IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, 2014, pp. 289–292.
[37] M. Yang, D. Kim, D. Li, W. Chen, H. Du, and A. O. Tokuta, "Sweep-coverage with energy-restricted mobile wireless sensor nodes," in *International Conference on Wireless Algorithms, Systems, and Applications (WASA)*, 2013, pp. 486–497.
[38] J. Liang, X. Huang, and Z. Zhang, "Approximation algorithms for distance constraint sweep coverage with base stations," *Journal of Combinatorial Optimization*, vol. 37, no. 4, pp. 1111–1125, 2019.
[39] D. Zhang, D. Zhao, and H. Ma, "On timely sweep coverage with multiple mobile nodes," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.
[40] Y. Tang, R. Zhou, G. Sun, B. Di, and R. Xiong, "A novel cooperative path planning for multirobot persistent coverage in complex environments," *IEEE Sensors Journal*, vol. 20, no. 8, pp. 4485–4495, 2020.
[41] L. Wu, Y. Xiong, M. Wu, Y. He, and J. She, "A task assignment method for sweep coverage optimization based on crowdsensing," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 686–10 699, 2019.
[42] E. Hallmark, "Points of interest poi database," https://www.kaggle.com/ehallmar/points-of-interest-poi-database.

Xiaofeng Gao received the B.S. degree in information and computational science from Nankai University, China, in 2004; the M.S. degree in operations research and control theory from Tsinghua University, China, in 2006; and the Ph.D. degree in computer science from The University of Texas at Dallas, USA, in 2010. She is currently a professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Her research interests include distributed system, wireless communications, data engineering, and combinatorial optimizations. She has published more than 160 peer-reviewed papers in the related area, including well-archived international journals such as IEEE TC, TKDE, TMC, TPDS, JSAC, and also in well-known conference proceedings such as SIGKDD, INFOCOM, ICDCS, etc. She has served on the editorial board of Discrete Mathematics, Algorithms and Applications, and as the PCs and peer reviewers for a number of international conferences and journals.



Jiahao Fan is a graduate student from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, P.R.China. He received his B.S. degree in Computer Science and Technology from Shanghai Jiao Tong University in 2018. His research interests include wireless sensor networks and mobile crowdsourcing. He has published several peer-reviewed papers in the related area, some in well-archived international journals such as TON and TMC.

Fan Wu is a professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He received his B.S. in Computer Science from Nanjing University in 2004, and Ph.D. in Computer Science and Engineering from the State University of New York at Buffalo in 2009. He has visited the University of Illinois at Urbana-Champaign (UIUC) as a Post Doc Research Associate. His research interests include wireless networking and mobile computing, algorithmic game theory and its applications, and privacy preservation. He has published more than 150 peer-reviewed papers in technical journals and conference proceedings. He is a recipient of the first class prize for Natural Science Award of China Ministry of Education, NSFC Excellent Young Scholars Program, ACM China Rising Star Award, CCF-Tencent "Rhinoceros bird" Outstanding Award, and CCF-Intel Young Faculty Researcher Program Award. He has served as an associate editor of IEEE Transactions on Mobile Computing and ACM Transactions on Sensor Networks, an area editor of Elsevier Computer Networks, and as the member of technical program committees of more than 90 academic conferences. For more information, please visit http://www.cs.sjtu.edu.cn/~fwu/.

Guihai Chen earned his B.S. degree from Nanjing University in 1984, M.E. degree from Southeast University in 1987, and Ph.D. degree from the University of Hong Kong in 1997. He is a distinguished professor of Shanghai Jiao Tong University, China. He had been invited as a visiting professor by many universities including Kyushu Institute of Technology, Japan in 1998, University of Queensland, Australia in 2000, and Wayne State University, USA during September 2001 to August 2003. He has a wide range of research interests with focus on sensor networks, peer-to-peer computing, high-performance computer architecture and combinatorics. He is a senior member of IEEE and has published more than 250 peer-reviewed papers, and more than 170 of them are in well-archived international journals such as IEEE Transactions on Parallel and Distributed Systems, Journal of Parallel and Distributed Computing, Wireless Networks, The Computer Journal, International Journal of Foundations of Computer Science, and Performance Evaluation, and also in well-known conference proceedings such as HPCA, MOBIHOC, INFOCOM, ICNP, ICDCS, CoNEXT and AAAI. He has won several best paper awards including ICNP 2015 best paper award. His papers have been cited for more than 10000 times according to Google Scholar. He is a CCF fellow.