Data Ferry Trajectory Planning for Sweep Coverage Problem with Multiple Mobile Sensors

Xiaofeng Gao, Xudong Zhu, Yuchen Feng, Fan Wu, Guihai Chen
Department of Computer Science and Technology, Shanghai Jiao Tong University, China
gao-xf@sjtu.edu.com, nongeek.zv@gmail.com, crosslife@sjtu.edu.cn, {fwu, gchen}@cs.sjtu.edu.cn

***Abstract*—Sweep coverage plays an important role in many applications like data gathering, sensing coverage, and devices control, etc. In this paper, we deal with sweep coverage problem with multiple mobile sensors to periodically cover n targets in the surveillance region. We propose three constant-factor approximations, namely *CycleSplit*, *CoCycle*, and *AugPrim*, to minimize the longest trajectory length of mobile sensors under different scenarios respectively. *CycleSplit* deals with non-cooperative sweep coverage problem (NCSC), in which each mobile sensor works independently along different trajectory cycle. It has an approximation ratio of $(5 - \frac{2}{n-m+1})$, which improves the best known approximation ratio of 5. *CoCycle* solves the cooperative sweep coverage problem (CSC), in which multiple sensors work together to cover n targets. Its approximation ratio is 4, which outperforms the state-of-art best approximation ratio of 6. *AugPrim* is also a 6-approximation. It computes the sensor route for multi-sink sweep coverage problem (MSSC), where each sensor has to transmit the gathered data to one of the given sinks in each detection period. It implements the modified Prim algorithm and augment paths in turn according to the sink positions. AugPrim is the first approximation for sweep coverage problem with sinks. Finally, we provide various numerical experiments and comparisons with several previous work to validate the efficiency of our design.**

I. Introduction

Coverage problems in wireless sensor networks have been studied extensively under various models. Briefly speaking, this kind of problems require to collect useful information about a given region by deploying wireless sensors. Some studies focus on covering PoIs (Points of Interest), and formulate the problem as *Target Coverage* problem [1]. Some others consider *Area Coverage* problem, in which the objective is to full cover or partial cover the given area. Another interesting application is *Barrier Coverage*: we want to monitor a belt region, such that any intruder who try to cross the region will be detected [2], [3]. Besides the omni-sensing wireless sensors, researchers also investigate the coverage issues for directional sensor networks [4].

Instead of continuous monitoring discussed above, many applications only require periodic patrol inspections for a certain set of PoIs. Typical examples include police patrolling, message ferrying, and devices control, etc. In such scenario, a mobile sensor can move around to collect data from targets actively, and the objective is usually to minimize the number of detecting sensors under a time constraint or shorten the trajectory length of mobile sensors. We refer such problem as *Sweep Coverage* [5]–[10]. Similar models have also been studied under the context of autonomous robots, vehicle routing.

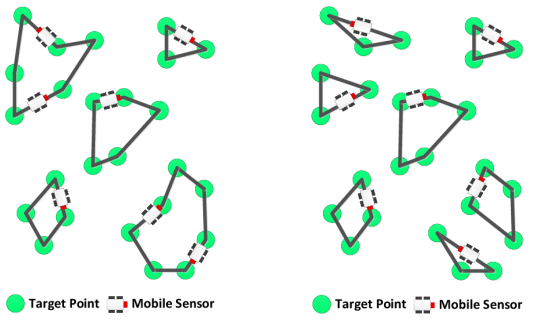
In this paper, we mainly focus on sweep coverage problem with multiple mobile sensors. Assume that there are n targets in the surveillance region and m mobile sensors. Each sensor serves as data ferry to collect information from targets. A target is said to be detected by a mobile sensor if the sensor moves to its position. Imaging that all the mobile sensors move along a predefined trajectory continuously to collect data, and a target is said to be t-sweep covered if it is detected by a mobile sensor at least once every t time units (we call t its sweep period). The objective here is to minimize the sweep period. We consider three variations of this problem. Their detailed descriptions are as follows.

Assume that all mobile sensors have the same velocity v. If m mobile sensors move along a same cycle $\mathcal{C}$, then we could deploy them evenly on $\mathcal{C}$, and schedule them moving toward the same direction along $\mathcal{C}$. An example is shown in Figure 1(a). There are 25 targets (PoIs) marked as green nodes. 7 mobile sensors form 5 disjoint trajectory cycles, three of which are formed cooperatively by multiple sensors. For each target on $\mathcal{C}$, its sweep period is $\frac{Len(\mathcal{C})}{mv}$, which is proportional to $l = \frac{Len(\mathcal{C})}{m}$. Here $Len(\mathcal{C})$ denotes the length of cycle $\mathcal{C}$. If we assign l as the trajectory length for each sensor on $\mathcal{C}$, then minimizing the sweep period for PoIs is equal to minimizing the maximum trajectory length for each sensor. We call it *Cooperative Sweep Coverage* problem (CSC). By setting a constraint that one mobile sensor are only permitted to move along one cycle, we get a variation of *Non-Cooperative Sweep Coverage* problem (NCSC). We further consider a more realistic version of this problem, in which there are a set of sinks and each mobile sensor must transmit the gathered data to one of the sinks in each detection period. We refer it as *multi-sink sweep coverage* (MSSC) problem.

Easy to see, if $m = 1$, the sweep coverage problem becomes a Traveling-Salesman-Problem (TSP), which is a classic NP-complete problem. Therefore CSC, NCSC and MSSC are all NP-complete. In previous literature, the NCSC problem is also known as min-max m-TSP problem, which looks for m

This work has been supported in part by the China 973 project (2012CB316200), National Natural Science Foundation of China (Grant number 61472252, 61133006, 61272443, 61422208), the Opening Project of Key Lab of Information Network Security of Ministry of Public Security (The Third Research Institute of Ministry of Public Security) Grant number C15602, and the Opening Project of Baidu (Grant number 181515P005267).

978-1-5090-1732-4/16/$31.00 ©2016 IEEE



(a) Cooperative Version (b) Non-Cooperative Version

Fig. 1: An Illustration for Sweep Coverage Problem

disjoint TSP cycles to cover the given targets. Correspondingly, Yu [11] proposed a 5-approximation named MMCCP to solve this problem, which is the best approximation up to now. For the CSC problem, the best known approximation is a 6-approximation named M^{3}SR in [12]. Their main idea is a binary search technique to partition the PoIs and sensors into different groups and then compute TSP cycle for each group. To the best of our knowledge, there is no approximation algorithm for the MSSC problem.

Correspondingly, in this paper we propose three constant-factor approximations, namely *CycleSplit*, *CoCycle*, and *AugPrim*, to solve the above mentioned three variations respectively. The first approximation, *CycleSplit*, deals with NCSC problem with approximation ratio of $(5-\frac{2}{n-m+1})$. It improves the best known approximation ratio of 5 in [11]. The main idea is to compute a TSP cycle for several selected connected components of the given graph, split the TSP cycle into several segments, and finally form distinct cycles. The second approximation, *CoCycle*, solves the CSC problem, which outperforms Feng's 6-approximation M^{3}SR [12]. Firstly it computes a Minimum Spanning Tree (MST) for each selected connected component, and then form cycles according to each MST. Finally it assigns sensors to each MST cycle to get the best solution. The third approximation *AugPrim* is a 6-approximation for the MSSC problem. It implements the modified Prim algorithm and augment paths in turn according to the sink positions. It is the first approximation for sweep coverage problem with sinks.

Although sweep coverage problem is naturally defined in Euclidean space, we relax our discussion to metric space, since we only use the triangle inequality property in our analysis. Thus, in the following discussion all the problems are defined in metric space unless explicitly mentioned, which means that our algorithms is applicable to more general cases.

Next, we provide various numerical experiments to validate the efficiency of our design. We compare our algorithms with OSweep in [13], MinExpand proposed in [13], and PDBA in [14]. The performance evaluation shows that our algorithms could achieve better results under the same network scale.

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

- we consider three variations of sweep coverage problem under metric space with multiple mobile sensors, namely *non-cooperative sweep coverage* problem (NCSC), *cooperative sweep coverage* problem (CSC), and *multi-sink sweep coverage* (MSSC) problem. The objective is to shorten the longest trajectory length of mobile sensors so as to reduce the target detection period.
- We propose three constant-factor approximations *CycleSplit*, *CoCycle*, and *AugPrim* to solve these three problems respectively. Both CycleSplit and CoCycle have better approximation ratio than the best state-of-art algorithms, while AugPrim is the first constant-factor approximation for sweep coverage problem with sinks.
- We compare our algorithms with several previous literature by simulations. Both theoretical analysis and numerical experiments validate the efficiency of our design.

The rest of this paper is organized as follows. Section III discusses some related work. Section III introduced some preliminaries for later sections. In Section IV and Section V, we proposed algorithms for NCSC, CSC, and MSSC respectively with approximation analysis. In section VI we give experiments to evaluate the performance of our algorithms. Section VII is the final conclusion.

II. Related Work

The sweep coverage problem with multiple sensors is close related to multiple Traveling-Salesman-Problem (TSP) with min-max objective (denoted as *min-max m-TSP* problem, or *min-max k-cycle* problem). The main difference is that in sweep coverage, multiple "salesmen" could work cooperatively in a single cycle to further decrease the trajectory length of each "salesman". The traditional single TSP is one of the most intensively studied problems in the area of combinational optimization. A simple algorithm based on minimum spanning tree (MST) gives a 2-approximation solution. By a clever construction, Christofides [15] 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 [16]. Obviously, the min-max k-cycle problem and cooperative sweep coverage problem (CSC) are at least as hard as the original TSP. For min-max k-cycle problem with a single root, [17] presented an $(e+1-\frac{1}{k})$-approximation, where e is the approximation ratio for the TSP problem. Arkin [18] gave a 4-approximation for the min-max k-path problem. Kim [19] proposed solutions for min-max m-TSP with neighborhood effect. In their model a mobile sensor can detect a PoI successfully within a distance threshold, so their designed trajectory will pass by each PoI but not touch it.

Following the similar strategy, we could firstly find a min-max k-tree for a given graph, and then construct cycles based on these trees. If the min-max k-tree problem has an α-approximation, then there is a 2α-approximation for min-max k-cycle problem. Unfortunately, min-max k-tree is also an NP-hard problem, which is proved in [20] by a reduction from Bin Packing. G. Even [20] presented a 4-approximation algorithm for min-max k-tree and a 4-approximation algorithm

for its rooted variants. In 2014, Khani [21] improved the approximation ratio from 4 to 3. It has also been proved in [22] that the min-max k-tree problem and its rooted variant have an inapproximability bound of $\frac{3}{2}$. Therefore, following this approach, we cannot design an approximation with ratio better than 3 to min-max k-cycle problem.

To minimize the number of mobile sensors with sweep coverage requirement, Li [8] proposed a 3-approximation with bounded time constraint. However, their approximation analysis has a serious flaw, which has been notified by Gorain in [9]. They mistakenly compared the result of their algorithm with the optimal solution of an m-TSP problem, not the optimal solution of their original problem. Thus the approximation ratio is obviously incorrect. Similarly, Gorain [7] proposed a 2-approximation but their analysis falls into the same flaw. Zhao [23] designed a simulated annealing to schedule the paths, while their algorithm does not have any guaranteed performance ratio. Shu [24] discussed this problem with single-sink constraint, to which mobile sensors must return back to the sink in each detection period. They also proposed a heuristic without theoretical bounds.

III. Preliminary

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

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

Suppose e is the edge between vertices u and v, we will use $d(u, v)$ and $d(e)$ to denote the same thing in context without ambiguity, which represents the distance between points u and v or the length of edge e. For an edge set $E' \subset E$, define $d(E') = \sum_{e \in E'} d(e)$.

B. Constructing Cycle from Tree

By constructing a cycle from a tree, we mean duplicating each edge in the tree, and constructing an Euler cycle from it since each vertex now has an even degree. The cost of the Euler cycle is twice of the cost for the original tree. Then we obtain a Hamiltonian cycle by traversing this Euler cycle and "shortcutting" the already visited vertices. Since the cost of edges satisfy triangle inequality, shortcutting does not increase the overall cost.

Remark. *We could use Christofides' improvements [15] to construct cycles from trees in our algorithm, but we cannot trivially get a better approximation ratio.*

C. Cycle Cover and Tree Cover

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 $\mathscr{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_t\}$, which are subgraphs of G, and the union of their vertices is V'.*

We can define tree cover similarly.

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

IV. Cooperative Sweep Coverage without Sinks

In this section, we will formally define the sweep coverage problem, and then design an approximation named *CycleSplit* for the non-cooperative sweep coverage problem (NCSC). Next, we will analyze the connection between the cooperative version and the non-cooperative version and propose another approximation named *CoCycle* for the cooperative sweep coverage problem (CSC).

A. Problem Formulation

The basic idea of our algorithms to solve the sweep coverage problem follows the following procedures:

1) Find a tree cover $\mathscr{T}$;
2) Contruct a cycle cover $\mathscr{C}$ from $\mathscr{T}$;
3) Determine how many sensors will be allocated to each cycle in $\mathscr{C}$.

To formally describe the process in the third step, we further introduce the concept of sensor allocation.

Definition 3 (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 $\mathcal{A} = \{m_1, \cdots, m_t\}$ such that $t = |\mathscr{C}|$ (or $|\mathscr{T}|$), $m_i \geq 1$ for $1 \leq i \leq t$, and $\sum_{i=1}^{t} m_i = m$.*

With these concepts defined above, we could formally define the problems we concerned in this paper.

Definition 4 (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 $d : V \times V \to \mathbb{R}^+$ is a metric, a cover scheme for (G, m, d) is a cycle cover $\mathscr{C}$ for V and a sensor allocation $\mathcal{A}$ for $\mathscr{C}$. Cooperative Sweep Coverage (CSC) problem aims to find a cover scheme such that $\max_{1 \leq i \leq |\mathscr{C}|} \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\}$ is minimized.*

To tackle CSC, we first consider a simplified version of this problem: sensors do not work cooperatively, and each sensor will traverse a distinct cycle (as shown in Figure 1(b)). We define it as non-cooperative sweep coverage problem (NCSC).

Definition 5 (Non-Cooperative Sweep Coverage (**NCSC**))**.** *Given input triple (G, m, d), where $G = (V, E)$ is a complete graph, m is the number of sensors, $d : V \times V \to \mathbb{R}^+$ is a metric, Non-Cooperative Sweep Coverage ($NCSC$) problem aims to find a cycle cover $\mathscr{C}$ such that $|\mathscr{C}| = m$ and $\max_{1 \leq i \leq |\mathscr{C}|} d(\mathcal{C}_i)$ is minimized.*

As mentioned in Section II, NCSC is actually the multiple Traveling-Salesman-Problem with min-max objective (min-max m-TSP). The following theorem states the relationship between CSC and NCSC.

Theorem 1. *If there is an α-approximation algorithm to $NCSC$, then there is a 2α-approximation algorithm for CSC.*

Proof. Simply notice that if we assign k sensors to a cycle $\mathcal{C}$ in CSC, we can get a solution for NCSC by:

- splitting $\mathcal{C}$ into k disjoint paths, each with length no larger than $\frac{d(\mathcal{C})}{k}$;
- doubling these paths to get cycles, each with length no larger than $\frac{2d(\mathcal{C})}{k}$.

Then we can successfully get a solution for NCSC.

Consider an optimal solution for CSC, whose length is OPT_{CSC}. If we split cycles like this, it becomes a feasible solution for NCSC with length 2OPT_{CSC}. Thus, the optimal solution for NCSC, whose length is OPT_{NCSC}, should be smaller than 2OPT_{CSC}. □

B. CycleSplit: an Approximation for $NCSC$

In this subsection we design *CycleSplit* algorithm to solve the NCSC problem. The main idea is to first select some connected components from the given graph, compute a TSP cycle for each connected component, split the TSP cycle into several segments and finally form distinct cycles.

Recall Kruskal's algorithm for constructing a minimum spanning tree. We add edges to the empty graph $G' = (V, \emptyset)$ one by one, by an increasing order of their length (i.e. $d(\cdot)$). In each stage of Kruskal's algorithm, G' will have a number of connected components, and each subgraph induced by the connected component is a tree. These trees will be used as a basis to construct a cycle cover. After obtaining the cycle cover, we will split some cycles to get exactly m cycles.

Algorithm 1 describes CycleSplit in detail. CycleSplit will choose the best of all feasible cycle covers in algorithm.

Algorithm 1: CycleSplit

input : $G = (V, E)$, $d : E \to \mathbb{R}^+$, m sensors
output: A cycle cover $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$

$E_0 \leftarrow \emptyset$; $G_0 \leftarrow (V, E_0)$; $i \leftarrow 0$;
foreach $e \in E$ *(chosen in ascending order by $d(\cdot)$)* **do**
  if *adding e to G_i does not produce a cycle* **then**
    $i \leftarrow i + 1$; $e_i \leftarrow e$;
    $E_i \leftarrow E_{i-1} \cup \{e\}$; $G_i \leftarrow (V, E_i)$;
    if *# of connected components $> m$* **then**
      Go to next iteration;
    $\mathscr{C}' \leftarrow \emptyset$;
    foreach *connected component CC in G_i* **do**
      Construct a cycle $\mathcal{C}$ by Christofides's Alg. [15];
      $\mathscr{C}' \leftarrow \mathscr{C}' \cup \{\mathcal{C}\}$;
    Split the cycles in $\mathscr{C}'$ to get a new cycle set $\mathscr{C}^{(i)}$ with $|\mathscr{C}^{(i)}| = m$;
Choose the best cycle cover from the above computed cycle covers, and denote it as $\mathscr{C}$;
return $\mathscr{C}$;

For Line 12 in Algorithm 1, we could design an efficient splitting strategy for this step. Before this step, we have already got a cycle cover, so the task here is just determining how to split large cycle into small ones, i.e. how many sensors should be assigned to each large cycle. The solution can be achieved by a simple greedy algorithm described in Algorithm 2, whose optimality is proved in Lemma 1.

Algorithm 2: GreedyAllocation

input : m sensors, a cycle cover $\mathscr{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_t\}$ and a metric d
output: a sensor allocation for $\mathscr{C}$ and a cycle cover $\mathscr{C}'$ with $|\mathscr{C}'| = m$

$m_i \leftarrow 1$ for $1 \le i \le t$;
while $m > \sum_{i=1}^{t} m_i$ **do**
  $k \leftarrow \arg\max_{1 \le i \le t} \frac{d(\mathcal{C}_i)}{m_i}$; $m_k \leftarrow m_k + 1$;
Split each $\mathcal{C}_i$ into m_i segments evenly, connect the two sides of each segment together with straight line and put these m_i new cycles into $\mathscr{C}'$;
return $\mathcal{A} = \{m_1, m_2, \cdots, m_t\}$ and $\mathscr{C}'$;

Lemma 1. *Algorithm 2 can find an optimal sensor allocation for a given cycle cover $\mathscr{C}$ efficiently. Here the optimality means that $\max \frac{2d(\mathcal{C}_i)}{m_i}$ is minimized.*

Proof. Assume on the contrary there is a different sensor allocation $\mathcal{A}' = \{m'_1, m'_2, \cdots, m'_t\}$ such that

$$\max_{1 \le i \le t} \frac{d(\mathcal{C}_i)}{m'_i} < \max_{1 \le i \le t} \frac{d(\mathcal{C}_i)}{m_i}.$$

Since $\mathcal{A}'$ is different from $\mathcal{A}$, then there exists a k such that $m'_k < m_k$ and $\frac{d(\mathcal{C}_k)}{m'_k} \ge \frac{d(\mathcal{C}_k)}{m_k}$. In Algorithm 2, we increased the value of m_k from m'_k to $m'_k + 1$, which means that $\frac{d(\mathcal{C}_k)}{m'_k}$ is the maximum among $\{\frac{d(\mathcal{C}_i)}{m_i}\}$ at some iteration. Notice that the value of $\max_{1 \le i \le t} \frac{d(\mathcal{C}_i)}{m_i}$ is non-increasing during the iterations. Therefore, we have

$$\frac{d(\mathcal{C}_k)}{m'_k} \ge \max_{1 \le i \le t} \frac{d(\mathcal{C}_i)}{m_i}$$

which contradicts the assumption. □

Now we will prove the approximation ratio for the CycleSplit algorithm. Denote the optimal solution as $\mathscr{C}^* = \{\mathcal{C}_1^*, \mathcal{C}_2^*, \cdots, \mathcal{C}_m^*\}$, set $\text{OPT} = \max_i(d(\mathcal{C}_i^*))$. We use d^*_{max} to denote the maximal distance between any two vertices in the same cycle, i.e. $d^*_{max} = \max_{u,v \in \mathcal{C}_i^*}\{d(u, v)\}$, where u, v belongs to the vertices of some $\mathcal{C}_i^*$. We first derive an upper bound for d^*_{max}.

Lemma 2. $d^*_{max} \le \frac{1}{2}OPT$.

Proof. Suppose that $d(u, v) = d^*_{max}$ for u, v in some $\mathcal{C}_i^*$. To construct a cycle, we must travel along a path from u to v, then

back from v to u. Since we are considering metric space, both of the two paths are larger than or equal to $d(u,v)$. Therefore

$$\text{OPT} \geq d(\mathcal{C}_i^*) \geq 2d(u,v) = 2d^*_{max}.$$

Thus Lemma 2 holds. □

As shown in Algorithm 1, denote the edges added to the graph as $e_1, e_2, \cdots, e_{|V|-1}$, we have $d(e_1) \leq d(e_2) \leq \cdots \leq d(e_{|V|-1})$. Let $G_i = (V, E_i)$ where $E_i = \{e_1, e_2, \cdots, e_i\}$. It is easy to get the following fact.

Lemma 3. *G_i is a minimum spanning forest with $|V| - i$ connected components. For each connected component CC, the subgraph induced by CC is actually a minimum spanning tree for CC.*

Set $j = \underset{1 \leq i \leq |V|-1}{\arg\max}\ d(e_i) \leq d^*_{max}$. Suppose the connected components of G_j are $CC_1^{(j)}, CC_2^{(j)}, \cdots, CC_{|V|-j}^{(j)}$. We will use $\mathcal{T}_1^{(j)}, \mathcal{T}_2^{(j)}, \cdots, \mathcal{T}_{|V|-j}^{(j)}$ to denote their corresponding trees. Denote $\mathcal{T}_i^*$ as the MST for the vertices in $\mathcal{C}_i^*$.

Lemma 4. *$d(\mathcal{T}_i^*) \leq (1 - \frac{1}{|V| - m + 1})$OPT, for $1 \leq i \leq m$.*

Proof. Notice that the number of vertices in any $\mathcal{C}_i^*$ is smaller than or equal to $(|V| - m + 1)$. Additionally, there must be an edge e' in $\mathcal{C}_i^*$ such that

$$d(e') \geq \frac{d(\mathcal{C}_i^*)}{\#\text{ of vertices in } \mathcal{C}_i^*} \geq \frac{d(\mathcal{C}_i^*)}{|V| - m + 1}$$

Delete e' from $\mathcal{C}_i^*$ we can get a spanning tree and by $d(\mathcal{C}_i^*) \leq \text{OPT}$, we have

$$d(\mathcal{T}_i^*) \leq d(\mathcal{C}_i^*) - d(e') \leq \frac{(|V| - m)d(\mathcal{C}_i^*)}{|V| - m + 1} \leq \frac{(|V| - m)\text{OPT}}{|V| - m + 1}$$

Thus the lemma holds. □

Lemma 5. *All vertices of $\mathcal{C}_i^*$ belongs to the same connected component of G_j, for $1 \leq i \leq |V| - j$.*

Proof. Suppose that $\mathcal{C}_i^*$ use an edge e' to connect two different connected components of G_j. Recall that $j = \underset{1 \leq i \leq |V|-1}{\arg\max}\ d(e_i) \leq d^*_{max}$. Then by our algorithm, $d(e') > d^*_{max}$, which contradicts the definition of d_{max}. □

Lemma 6. *Suppose $CC_i^{(j)}$ contains k_i^* cycles from the optimal solution, then*

$$d(\mathcal{T}_i^{(j)}) \leq ((\frac{3}{2} - \frac{1}{|V| - m + 1})k_i^* - \frac{1}{2})\textit{OPT}$$

Proof. From Lemma 5, we can find that these k_i^* cycles from the optimal solution. We can use $k_i^* - 1$ edges to connect them, and each edge will cost no more than $d(e_j)$. Since $\mathcal{T}_i^{(j)}$ is an MST, we have

$$\begin{aligned} d(\mathcal{T}_i^{(j)}) &\leq k_i^* d(T_i^*) + (k_i^* - 1)d(e_j) \\ &\leq k_i^* \frac{(|V| - m)\text{OPT}}{|V| - m + 1} + (k_i^* - 1)\frac{\text{OPT}}{2} \\ &\leq \left((\frac{3}{2} - \frac{1}{|V| - m + 1})k_i^* - \frac{1}{2}\right)\text{OPT} \end{aligned}$$

□

Lemma 7. *Denote $TSP_i^{(j)}$ as the optimal TSP cycle for $CC_i^{(j)}$. Suppose $CC_i^{(j)}$ contains k_i^* cycles from the optimal solution, then $d(TSP_i^{(j)}) \leq (2k_i^* - 1)$OPT.*

Proof. Consider the k_i^* cycles from the optimal solution. We can use $k_i^* - 1$ edges to connect them, and duplicating these edges will result a TSP cycle. Each edge will cost no more than $d(e_j)$. Thus we have

$$\begin{aligned} d(TSP_i^{(j)}) &\leq k_i^*\text{OPT} + 2(k_i^{(j)} - 1)d(e_j) \\ &\leq k_i^*\text{OPT} + 2(k_i^{(j)} - 1)d^*_{max} \\ &\leq (2k_i^{(j)} - 1)\text{OPT} \end{aligned}$$

Then the statement holds. □

Lemma 8. *Denote $\mathcal{C}_i'^{(j)}$ as the TSP cycle computed by Christofides Alg. for $CC_i^{(j)}$ at Line 10 in Algorithm 1. Suppose $CC_i^{(j)}$ contains k_i^* cycles from the optimal solution, then*

$$d(\mathcal{C}_i'^{(j)}) \leq \left((\frac{5}{2} - \frac{1}{|V| - m + 1})k_i^{(j)} - 1\right)\textit{OPT}$$

Proof. From Christofides' algorithm, we can get that

$$\begin{aligned} d(\mathcal{C}_i'^{(j)}) &\leq d(\mathcal{T}_i^{(j)}) + \frac{1}{2}d(TSP_i^{(j)}) \\ &\leq \left((\frac{3}{2} - \frac{1}{|V| - m + 1})k_i^* - \frac{1}{2}\right)\text{OPT} + \\ &\quad \frac{1}{2}((2k_i^* - 1)\text{OPT}) \end{aligned}$$

This finishes the proof. □

If we split $\mathcal{C}_i'^{(j)}$ into k_i^* paths, then each path will have length less than $(\frac{5}{2} - \frac{1}{|V|-m+1})$OPT. Thus the cycles obtained from these paths will have length less than $(5 - \frac{2}{|V|-m+1})$OPT. As we have already stated in Theorem 1, the optimal way to split these cycles can be found. Therefore, we have the following theorem.

Theorem 2. *$\mathscr{C}^{(j)}$ is a $(5 - \frac{2}{|V| - m + 1})$-approximation, i.e.*

$$\max_{\mathcal{C} \in \mathscr{C}^{(j)}} d(\mathcal{C}) \leq (5 - \frac{2}{|V| - m + 1})\textit{OPT},$$

and CycleSplit is a (5-$\frac{1}{|V|-m+1}$)-approximation for $NCSC$.

C. CoCycle: an Approximation for CSC

In this subsection, we propose CoCycle algorithm to solve the CSC problem. Firstly it computes an MST for each selected connected component, and then form cycles according to each MST. Finally it assigns sensors to each MST cycle to get the best solution.

To solve CSC, we first solve the *cooperative tree coverage* problem (CTC) instead, whose definition is as follows.

Definition 6 (Cooperative Tree Coverage (CTC))**.** *Given input (G, m, d), where $G = (V, E)$ is a complete graph, m is the number of sensors, d is a metric, a cover scheme for*

(G, m, d) *is a tree cover* $\mathscr{T}$ *for* V *and a sensor allocation* $\mathcal{A}$ *for* $\mathscr{T}$*. Cooperative Tree Coverage (CTC) problem aims to find a cover scheme such that* $\max_{1 \le i \le |\mathscr{T}|} \left\{ \frac{d(\mathcal{T}_i)}{m_i} \right\}$ *is minimized.*

The connection between CSC and CTC can be easily established: if there is an α-algorithm for CTC, then there is a 2α-algorithm for CSC. Thus the rest part of this subsection will focus on CTC. We first design *CoCycle-Tree* algorithm to solve the CTC problem. Then we can get the *CoCycle* algorithm for CSC by constructing cycles from the trees returned by CoCycle-Tree. Algorithm 3 shows the details of CoCycle-Tree.

Algorithm 3: CoCycle-Tree

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

```
G' ← (V, ∅);
foreach e ∈ E (chosen in ascending order by d(·)) do
    if adding e to G' does not produce a cycle then
        Add e to G';
        if G' has less than m connected components then
            Obtain a tree cover 𝒯 from G';
            Find the optimal sensor allocation for 𝒯;
return the best cover scheme computed above;
```

Next, let us analyze the performance of CoCycle algorithm. Denote the optimal cover scheme as $\mathscr{T}^* = \{\mathcal{T}_1^*, \mathcal{T}_2^*, \cdots, \mathcal{T}_t^*\}$ and $\mathcal{A}^* = \{m_1^*, m_2^*, \cdots, m_t^*\}$, set $\text{OPT} = \max(\frac{d(\mathcal{T}_i^*)}{m_i^*})$. Set $j = \arg\max_{1 \le i \le |S|-1} d(e_i) \le \text{OPT}$. Suppose the connected component of G_j is $CC_1^{(j)}, CC_2^{(j)}, \cdots, CC_{n-j}^{(j)}$, and we use $\mathcal{T}_1^{(j)}, \mathcal{T}_2^{(j)}, \cdots, \mathcal{T}_{|V|-j}^{(j)}$ to denote their corresponding trees.

Lemma 9. *There is an optimal solution* $\mathscr{T}^*$ *and* $\mathcal{A}^*$*, such that all vertices of* $\mathcal{T}_i^*$ *belongs to the same connected component of* G_j*, for* $1 \le i \le t$*.*

Proof. Suppose that $\mathcal{T}_i^*$ uses an edge e' to connect two different connected components of G_j, then we have $d(e') > \text{OPT}$. If we do not use e', $\mathcal{T}_i^*$ will be partitioned into two trees P_1, P_2. Reallocating the m_i^* sensors to P_1 and P_2 accordingly will give a better cover scheme. Thus we can eliminate all such edges to get an optimal cover scheme satisfying this described property. In the rest of this subsection, we assume that the optimal cover scheme $(\mathscr{T}^*, \mathcal{A}^*)$ has this property. □

By Lemma 9, we can derive a similar result as Lemma 2.

Lemma 10. *Suppose* $CC_i^{(j)}$ *contains* k_i^* *cycles from the optimal cover scheme, and these* k_i^* *cycles used* n_i^* *sensors in the optimal cover scheme, then*

$$d(\mathcal{T}_i^{(j)}) \le n_i^* \mathit{OPT} + (k_i^{(j)} - 1) d_{max}^*$$

From this lemma, we have $\frac{d(\mathcal{T}_i^{(j)})}{n_i^*} \le 2\text{OPT}$, Then by Lemma 1, we can get the final conclusion.

Theorem 3. *CoCycle-Tree is a 2-approximation for CTC.*

We can construct the CoCycle algorithm for CSC by constructing cycles from the trees returned by CoCycle-Tree.

Corollary 1. *The CoCycle algorithm is a 4-approximation algorithm for the CSC problem.*

V. Multi-Sink Sweep Coverage

In this section we consider a more realistic version of the sweep coverage problem: there are a set of sinks and each mobile sensor must transmit the gathered data to one of the sinks in each detection period. We refer this problem as *multi-sink sweep coverage* (MSSC) problem. We will first give a formal definition to MSSC, and then propose a novel approximation algorithm *AugPrim* to solve it.

A. Problem Formulation

Usually, the limited storage capacity and battery power constraint require mobile sensors to transmit their collected data to base stations (also called "sinks") periodically. Suppose there are multiple static sinks and each sensor has to touch at least one of them during each detection period. Then how to compute new trajectory cycles for m sensors with this new constraint becomes a challenging problem. We define it as *multi-sink sweep coverage* problem (MSSC). Figure 2 is an example scenario, 9 mobile sensors work cooperatively to cover 19 targets along 4 distinct trajectory cycles, while on each cycle there must be a sink to collect and transmit data.

Definition 7 (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* $|S| \le m$*,* m *is the number of sensors,* d *is a metric, a cover scheme for* (G, S, m, d) *is a cycle cover for* $\mathscr{C}$ *for* $V \setminus S$ *where each cycle from* $\mathscr{C}$ *contains at least one sink, and a sensor allocation* $\mathcal{A}$ *for* $\mathscr{C}$*. Multiple Sink Sweep Coverage (MSSC) aims to find a cover scheme which minimize* $\max \left\{ \frac{d(\mathcal{C}_i)}{m_i} \right\}$*.*

We first consider the simplest case with only one sink. Then we will have an important observation in Theorem 4

Theorem 4. *If there is only one sink, then finding an optimal TSP cycle, and assigning all the sensors work cooperatively along this cycle is an optimal solution for MSSC.*

As discussed in the previous section, we could convert the cycle cover requirement in MSSC to tree cover. Then we get a new variation, *Multiple Sink Tree Coverage* problem (MSTC).

B. AugPrim: a Novel Approximation for MSSC

In this subsection we introduce a novel approximation for the MSSC problem, namely *AugPrim*. It implements the modified Prim algorithm and augment paths in turn according to the sink positions. AugPrim is the first approximation for sweep coverage problem with sinks.

The design pattern is the same as before, we will first try to solve MSTC, then the algorithm for MSSC follows naturally. To describe AugPrim, we need the following definition.

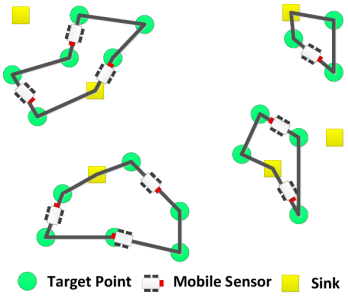
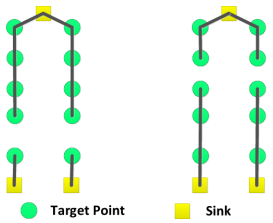


Fig. 2: An illustration for multi-sink cooperative sweep coverage



(a) a MST (b) a better tree cover

Fig. 3: An illustration of Spanning Forest

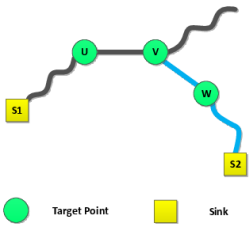


Fig. 4: $d(u,v) > \text{OPT}$

Definition 8 (Spanning Forest with Multiple Roots)**.** *Given a graph $G = (V, E)$, and a set of roots $S \subset V$, a spanning forest $\mathscr{F}$ with S is a set of spanning trees and contains all the vertices of this graph. Moreover, each spanning tree has one root from S.*

Given $G(V, E)$ and a set of roots $S \subset V$, we could try to find the minimum spanning forest with roots S. However, this approach can be arbitrary bad as shown in Figure 3. Suppose we have only 3 sensors. Figure 3(a) is a minimum spanning forest with the 3 sinks, and the upper tree becomes the bottleneck. Figure 3(b) shows a better tree cover where each tree is more balanced.

Therefore, after obtaining the minimum spanning forest with roots S, we need to add more edges into the forest to merge some spanning trees. The details is shown in Algorithm 4.

Algorithm 4: AugPrim-Tree

input : $G = (V, E)$, $S \subset V$, m sensors, $d : E \to \mathbb{R}^+$
output: A tree cover $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_t\}$, each $\mathcal{T}_i$ contains at least one sink from S, and a sensor allocation $\mathcal{A} = \{m_1, m_2, \cdots, m_t\}$ for $\mathscr{T}$

1 Find a minimum spanning forest F with S;
2 Compute a sensor allocation for F;
3 **foreach** $e \in E \setminus F$ *(in ascending order by $d(\cdot)$)* **do**
4 **if** *adding e to F does not produce a cycle* **then**
5 $F = F \cup \{e\}$;
6 Compute a sensor allocation based on F;
7 **return** *the best cover scheme*;

C. Performance Analysis

We first argue that the minimum spanning forest with roots can be efficiently constructed, by a simple modification to Prim's algorithm for minimum spanning tree. The pseudocode is shown in Algorithm 5.

Lemma 11. *Given input (G, S, d), Algorithm 5 can find the minimum spanning forest with S.*

Suppose that the optimal solution for $MSTC$ is $(\mathscr{T}^*, \mathcal{A}^*)$, where $\mathscr{T} = \{\mathcal{T}_1^*, \mathcal{T}_2^*, \cdots, \mathcal{T}_t^*\}$ and $\mathcal{A} = \{m_1^*, m_2^*, \cdots, m_t^*\}$, set $\text{OPT} = \max\left\{\frac{d(\mathcal{T}_i^*)}{m_i^*}\right\}$. Suppose that edges added at Line 5

Algorithm 5: Modified Prim Algorithm

input : $G = (V, E)$, sink set $S \subset V$, $d : E \to \mathbb{R}^+$
output: A spanning forest with S

1 **foreach** $v \in V$ **do**
2 $pre(v) = \arg\min_{s \in S} d(s, v)$; $price(v) = d(r, pre(v))$;
3 $V' = S$; $F = \emptyset$;
4 **while** $V' \neq V$ **do**
5 $u = \arg\min_{v \in V \setminus V'} price(v)$; $V' = V' \cup \{u\}$;
  $F = F \cup \{(u, pre(u))\}$;
6 **foreach** $v \in V \setminus V'$ **do**
7 **if** $d(u, v) < price(v)$ **then**
8 $price(v) = d(u, v)$; $pre(v) = u$;
9 **return** F;

in Algorithm 4 are $e_1, e_2, \cdots, e_{|S|-1}$, such that $d(e_1) \leq d(e_2) \leq \cdots \leq d(e_{|S|-1})$, set $j = \arg\max_{1 \leq i \leq |S|-1} d(e_i) \leq \text{OPT}$, denote the spanning forest after we add e_j as $\mathscr{T}^{(j)}$. $E^{(j)}$, $G^{(j)}$, $\mathscr{T}^{(j)}$, $\mathcal{A}^{(j)}$ are similarly defined.

Lemma 12. *There is a cover scheme $(\mathscr{T}', \mathcal{A}')$ such that for each $\mathcal{T}' \in \mathscr{T}'$, its vertices is belong to the same connected component of $G^{(j)}$ and $\max\left\{\frac{d(\mathcal{T}_i')}{m_i'}\right\} \leq 2OPT$.*

Proof. We construct such cover scheme $(\mathscr{T}', \mathcal{A}')$ from the optimal cover scheme $(\mathscr{T}^*, \mathcal{A}^*)$.

Use $\mathcal{T}_s^{(0)}$ and $\mathcal{T}_s^*$ to represent the tree rooted at a sink s in $\mathscr{T}^{(0)}$ and $\mathscr{T}^*$ respectively. Consider an edge $(u, v) \in \mathcal{T}_{s_1}^*$ such that $d(u, v) > \text{OPT}$ and it connects two different connected components in $G^{(j)}$ as shown in Figure 4. u belongs to the connected component which contains sink s_1, and v belongs to the connected component which contains sink s_2. w is the parent of v in $\mathcal{T}_{s_2}^{(0)}$. Suppose deleting edge (u,v) partitions $T_{s_1}^*$ into two parts, P_1 and P_2 (P_1 contains s_1).

Notice that $d(v, w) \leq d(u, v)$ (if not, replacing the edge (w, v) with (u, v) in $\mathscr{T}^{(0)}$ will result a smaller spanning forest). We could replace edge (u, v) with (v, w) in $\mathscr{T}^*$ to get a new tree cover. Basically, $T_{s_1}^*$ becomes $T_{s_1}' = T_{s_1}^* -$

$\{(u,v)\} - P_2$, $T^*_{s_2}$ becomes $T'_{s_2} = T^*_{s_2} + \{(v,w)\} + P_2$. Let

$$x = \left\lceil \frac{d(T'_{s_1})}{\text{OPT}} \right\rceil$$

If we assign x sensors to T'_{s_1}, and $m^*_{s_2} + m^*_{s_1} - x$ sensors to T'_{s_2}, we will have

$$\frac{d(T'_{s_1})}{x} \leq \text{OPT}; \qquad \text{and}$$

$$\frac{d(T'_{s_2})}{m^*_{s_2} + m^*_{s_1} - x} \leq \frac{d(\{(v,w)\} + P_2)}{m^*_{s_1} - x} \leq 2\text{OPT}$$

Thus, by this replacing process we could get a new cover scheme $(\mathscr{T}', \mathcal{A}')$ satisfying the property described above. □

Theorem 5. *$(\mathscr{T}^{(j)}, \mathcal{A}^{(j)})$ is a 3-approximation cover scheme for the MSTC problem.*

Proof. Suppose that $(\mathscr{T}', \mathcal{A}')$ is a cover scheme that for each $\mathcal{T}' \in \mathscr{T}'$, its vertices is belong to the same connected component of $\mathscr{F}$ and $\max \frac{d(\mathcal{T}'_i)}{m'_i} \leq 2\text{OPT}$. Denote the connected components in $G^{(j)}$ as $CC^{(j)}_1, CC^{(j)}_2, \cdots$, let $\mathcal{T}^{(j)}_1, \mathcal{T}^{(j)}_2, \cdots$ be the corresponding trees.

Suppose $CC^{(j)}_i$ contains k_i trees from $\mathscr{T}'$, denote their union as F'_i. Then $CC^{(j)}_i$ also contains k_i sinks, which means $CC^{(j)}_i$ contains k_i trees from $\mathscr{T}^{(0)}$, denote their union as $F^{(0)}_i$. Assume $(\mathscr{T}', \mathcal{A}')$ allocates n'_i sensors in total for trees in F'_i.

$$\sum_{\mathcal{T} \in F^{(0)}_i} d(\mathcal{T}) \leq \sum_{\mathcal{T} \in F'_i} d(\mathcal{T})$$

Then we add $k_i - 1$ edges to connect the trees in $F^{(0)}_i$,

$$\begin{aligned} d(\mathcal{T}^{(j)}_i) &\leq \sum_{\mathcal{T} \in F^{(0)}_i} d(\mathcal{T}) + (k_i - 1)\text{OPT} \\ &\leq \sum_{\mathcal{T} \in F'_i} d(\mathcal{T}) + (k_i - 1)\text{OPT} \\ &\leq 2n'_i\text{OPT} + (k_i - 1)\text{OPT} \end{aligned}$$

If we allocate n'_i sensors to $\mathcal{T}^{(j)}_i$, then

$$\frac{d(\mathcal{T}^{(j)}_i)}{n'_i} \leq 3\text{OPT}.$$

By Theorem 1, we can compute the optimal sensor allocation. Thus

$$\max \frac{d(\mathcal{T}^{(j)}_i)}{m_i} \leq 3\text{OPT}.$$

Since we get the best cover scheme in Algorithm 4, we will have Theorem 6. □

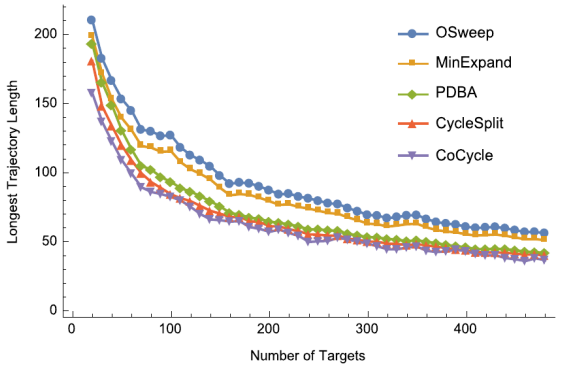
Theorem 6. *The AugPrim-Tree algorithm is a 3-approximation for the MSTC problem.*

We can get the AugPrim algorithm for MSSC by constructing cycles from the trees returned by AugPrim-Tree algorithm.

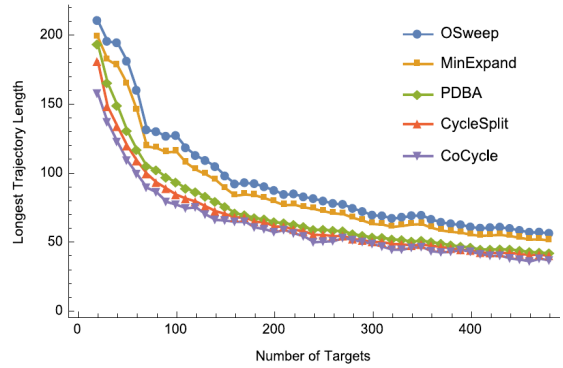
Theorem 7. *The AugPrim algorithm is a 6-approximation.*

VI. Performance Evaluation

In this section, we evaluate and compare the performance of CycleSplit and CoCycle-Tree on a stand alone C++ simulation platform. We also compare our algorithms with OSweep in [13], MinExpand proposed in [13], and PDBA in [14]. In this simulation, we prepare a 200 × 200 virtual 2-D space and randomly deploy a number of targets. The number of targets range from 20 to 200 with a step of 10 and the number of mobile sensors is $\frac{1}{5}$ or $\frac{1}{10}$ of the number of targets. When the numbers of mobile sensors and targets are fixed, we generate 20 problem instances and calculate the average of the costs of the outputs of CoCycle, CycleSplit, OSweep [13], MinExpand [13], and PDBA [14]. Since some previous algorithms are to find the minimum number of mobile sensors to achieve Sweep Coverage with a constraint on the length of mobile sensors' trajectories [13], [14], we use binary search through changing the bound length for sweep coverage until the output of the previous algorithms equals to the number of mobile sensors we input. We use that bound length as the output of these algorithms.



(a) $m = \frac{n}{5}$

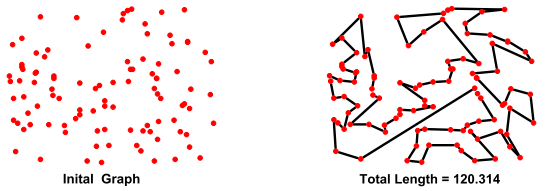


(b) $m = \frac{n}{10}$

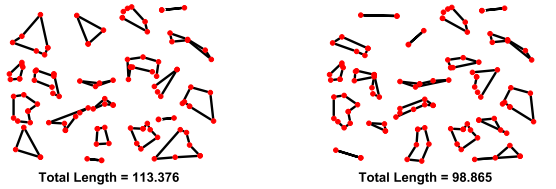
Fig. 5: Simulation Results of Five Algorithms

Figure 5(a) and Figure 5(b) are simulation results for these five algorithms under different parameter settings. The x axis denotes the number of targets to be covered, while the y axis shows the length of the longest trajectory calculated by each algorithm. We can easily abstract from these two figures that CoCycle and CycleSplit algorithms outperform previous algorithms and CoCycle performs best.

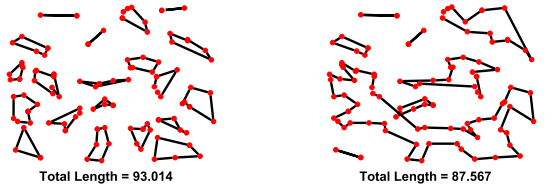
Figure 6(a) exhibits an instance with $n = 100$ targets and we want to cover them by $m = 20$ sensors. Figure 6(b)-(e) show the output trajectories computed by these five algorithms. We can find that the output of CoCycle has the shortest total length. The simulation results indicate that mobile sensors working cooperatively for sweep coverage can efficiently reduce the detection period.



(a) 100 Targets to be Covered (b) Result by OSweep



(c) Result by MinExpand (d) Result by PDBA



(e) Result by NCSC (f) Result by CSC

Fig. 6: Outputs for one instance with n=100 and m=20

VII. Conclusion

In this paper, we discuss three variations of the sweep coverage problem to shorten the longest trajectory length of mobile sensors, and design three constant-factor approximations respectively. For non-cooperative sweep coverage problem (NCSC), we propose a $(5 - \frac{2}{n-m+1})$-approximation named CycleSplit, which improves the the best known approximation ratio of 5; for cooperative sweep coverage problem (CSC), we propose a 4-approximation named CoCycle, which outperforms the best state-of-art approximation ratio of 6. We further consider multi-sink sweep coverage problem (MSSC), and proposed a 6-approximation algorithm for it. This is the first constant-factor approximation with sink constraint. Our analysis is based on the property of metric space, which is more general than the commonly discussed Euclidean space. Finally, we compare our algorithms with several previous works by simulations. Both theoretical analysis and numerical experiments validate the efficiency of our design.

References

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

[2] 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.

[3] 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.

[4] M. A. Guvensan and A. G. Yavuz, "On coverage issues in directional sensor networks: A survey," *Ad Hoc Networks*, vol. 9, no. 7, pp. 1238–1255, 2011.

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

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

[7] 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.

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

[9] B. Gorain and P. S. Mandal, "Approximation algorithm for sweep coverage on graph," *Information Processing Letters*, 2015.

[10] 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.

[11] W. Yu and Z. Liu, "Improved approximation algorithms for min-max and minimum vehicle routing problems," pp. 147–158, 2015.

[12] Y. Feng, X. Gao, F. Wu, and G. Chen, "Shorten the trajectory of mobile sensors in sweep coverage problem," in *IEEE Global Communications Conference (GlOBECOM)*, 2015.

[13] 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)*, Shanghai, China, 2010, pp. 283–290.

[14] 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)*, Kitakyushu, Japan, 2014, pp. 289–292.

[15] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," DTIC Document, Tech. Rep., 1976.

[16] M. Karpinski, M. Lampis, and R. Schmied, "New inapproximability bounds for tsp," pp. 568–578, 2013.

[17] G. N. Frederickson, M. S. Hecht, and C. E. Kim, "Approximation algorithms for some routing problems," in *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 1976, pp. 216–227.

[18] E. M. Arkin, R. Hassin, and A. Levin, "Approximations for minimum and min-max vehicle routing problems," *Journal of Algorithms*, vol. 59, no. 1, pp. 1–18, 2006.

[19] 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 International Conference on Computer Communications (INFOCOM)*, 2012, pp. 504–512.

[20] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha, "Min–max tree covers of graphs," *Operations Research Letters*, vol. 32, no. 4, pp. 309–315, 2004.

[21] M. R. Khani and M. R. Salavatipour, "Improved approximation algorithms for the min-max tree cover and bounded tree cover problems," *Algorithmica*, vol. 69, no. 2, pp. 443–460, 2014.

[22] Z. Xu and Q. Wen, "Approximation hardness of min–max tree covers," *Operations Research Letters*, vol. 38, no. 3, pp. 169–173, 2010.

[23] 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.

[24] 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, pp. 726–732, 2014.