Approximating Sweep Coverage Delay

Gokarna Sharma(✉) and Jong-Hoon Kim

Department of Computer Science, Kent State University, Kent, OH 44242, USA
sharma@cs.kent.edu, jkim72@kent.edu

Abstract. We consider the following fundamental sweep coverage problem that arises in mobile wireless sensor networks: Given a set of k mobile sensors and a set of m points of interests (POIs) in the Euclidean plane, how to schedule the mobile sensors such that the maximum delay between two subsequent visits to a POI by any sensor is minimized. We study two scenarios of this problem: (i) start positions of the sensors are fixed such that they must return to their start positions between subsequent traversals to POIs that fall in their trajectories, and (ii) sensor positions are not fixed and they are not required to return to their start positions between subsequent traversals. Scenario (i) models battery-constrained sensors which need to be recharged frequently, whereas scenario (ii) models sensors that have no constraint on battery and hence frequent recharging is not necessary. We present two constant factor approximation algorithms for each scenario. The problem we consider is NP-hard and, to the best of our knowledge, these are the first algorithms with guaranteed approximation bounds for this problem.

1 Introduction

Tremendous work in the literature of wireless sensor networks (WSNs) has established that one of the major applications of sensor networks is on surveillance problems [1,3,4,12,15,16,20,21,23–29,31]. These surveillance problems require specific coverage requirements for different proposes. The vast majority of work on surveillance problems using static and mobile WNSs focused on providing two kinds of coverage: *full coverage* and *barrier coverage*. In full coverage, sensors deployed over the given field continuously monitor the entire area. Any point within the area is ensured to be covered by at least one sensor. A full coverage is usually required when users need to fully monitor the entire environment. In barrier coverage, sensors are deployed to form a barrier for detecting any intruders crossing the given barrier area, which is generally a line segment or a strip. The sensors then guard the barrier by guarding the crossing paths. The κ-full coverage and κ-barrier coverage variations of these problems were also studied [1,12,16,20,26,28,29,31].

Both the full and barrier coverage problems can be classified as the *static* coverage problems since the given area (or barrier) needs to be covered at all times by the sensors. In contrast, some applications may require that coverage be provided for the specific given points periodically, i.e., the points do not need

N. Boudriga et al. (Eds.): UNet 2018, LNCS 11277, pp. 14–27, 2018.
https://doi.org/10.1007/978-3-030-02849-7_2

to be covered at all times and they only need to be visited within a specific period. One immediate application of such setting is in patrolling where certain points of interests (POIs) are visited within a specific time period. This problem is called *sweep coverage* and it differs from the static coverage problems as POIs do not need to be covered at all times and only the specific time requirement for inspecting the POI needs to be satisfied. Li *et al.* [18] were the first to study this problem from the objective of minimizing the number of sensors given the sweep period. We denote this problem as MinSensorSweep – given a set of m POIs and the (global) sweep period t, the goal is to schedule sensors such that the sweep coverage requirement is satisfied with the minimum number of sensors.

Contributions. In this paper, we consider the sweep coverage problem with the objective of minimizing the coverage delay. That is, given a set of m POIs and a set of k mobile sensors, the goal is to schedule the given sensors such that the sweep period t is minimized. We denote this problem as MinDelaySweep. MinDelaySweep is different than MinSensorSweep since we deal with the problem of minimizing the time period between two subsequent visits of the POIs. The only previous work that studies this problem is due to Chen *et al.* [5] where they provided several algorithms to minimize the coverage delay. However, their algorithms were evaluated only through experimentally and no approximation bounds were given. Therefore, we focus on designing algorithms and proving achievable approximation bounds for MinDelaySweep.

We consider two different scenarios of MinDelaySweep. The first scenario, called Predefined-Start, covers the case in which start positions of sensors are fixed and after every traversal of the POIs in their trajectories, the sensors go back to their start positions. This scenario is useful when sweep coverage is provided by the battery-constrained mobile sensors which need to be recharged quite frequently at their base stations. The second scenario, called Not-Predefined-Start, covers the case in which sensor positions are not fixed and they do not need to go back to their base stations after every traversal of the POIs as they are assumed of having sufficient energy (i.e., they are not battery-constrained) to provide the sweep coverage for a very long time.

Someone may say that existing techniques and algorithms for MinSensorSweep can be used to solve MinDelaySweep. The idea is to take a solution of MinSensorSweep and see which value of t minimizes the sweeping period for all the POIs. However, this process needs to be repeated at least $\mathcal{O}(\log t)$ times to figure out the right value of t (since the values starting from 1 upto the right value t need to be checked and at least a binary search is needed). Moreover, the solution depends on the solution of MinSensorSweep used. Furthermore, it is worth to note that MinDelaySweep is a NP-Hard problem. Therefore, we focus on the algorithms that provide good approximation of the exact solution and run in polynomial time. We give two algorithms each for Not-Predefined-Start and Predefined-Start scenarios of MinDelaySweep.

– We provide a 2δ-approximation algorithm for the Not-Predefined-Start scenario of MinDelaySweep and a $(\delta + 2 - \frac{1}{k})$-approximation algorithm for

the Predefined-Start scenario of MinDelaySweep, where δ is the approximation ratio of an algorithm for the *traveling salesman* problem (TSP).
– We provide a 2γ-approximation algorithm each for both the Not-Predefined-Start and Predefined-Start scenarios of MinDelaySweep, where γ is the approximation ratio of an algorithm for the *tree cover* problem (TC).

Using Christofides's algorithm [6] to compute the solution for TSP tour, we obtain 3-approximation for the Not-Predefined-Start scenario of MinDelaySweep and $(\frac{7}{2} - \frac{1}{k})$-approximation for the Predefined-Start scenario of MinDelaySweep (Christofides's algorithm has the approximation ratio of 1.5 for the TSP tour). Using the algorithms of Even *et al.* [8] to compute the solution for TC tours in both Not-Predefined-Start and Predefined-Start scenarios of MinDelaySweep, we obtain 8-approximation for both of our algorithms (Even *et al.*'s algorithm has the approximation ratio of 4 for both the rooted and unrooted versions of TC tours). These bounds can be improved if we have better approximation factors for both δ and γ. From recent work on tree and cycle cover problems [14,19], we can obtain 6-approximations for our algorithms based on TC tours. To our best knowledge, these bounds are the first approximation bounds for the NP-hard MinDelaySweep.

Although our solutions look like direct extensions of the existing results on TSP and TC, no such approximation bounds were known in the literature for MinDelaySweep (except experimental study with no approximation bounds in [5]). Our study is interesting since it shows that MinDelaySweep is related to the problem of finding k TSP and TC tours of equal lengths in graphs. Given a solution with k equal length tours, a solution for MinDelaySweep is no more than the factor of 2 times more than the k equal length tour solution used to solve MinDelaySweep.

For the TSP based solution for Not-Predefined-Start scenario, we take a TSP tour and divide that tour into k-subtours in such a way that the approximation obtained from the division is no more than 2 times the approximation of the one single tour. For the Predefined-Start scenario, we use the k-splitour concept of Frederickson *et al.* [9] (details later) and obtain k trajectories such that the claimed approximation bound is still satisfied after the trajectories are modified to include the start positions of sensors. For the TC based solution for both Not-Predefined-Start and Predefined-Start scenarios, we use the concept of Even *et al.* [8] to build k trees such that the approximation obtained is no more than 2 times the approximation of each tree.

Related Work. There is a vast literature on coverage problems in mobile WSNs, which can be divided into three main categories: full coverage, barrier coverage, and sweep coverage. The full and barrier coverage problems are static coverage problems, whereas the sweep coverage problem is a dynamic coverage problem. The full coverage problem is heavily studied under *area* and *point* coverage [3,25,27,28]. Several papers studied how mobile sensors can be used to assist static coverage under a hybrid setting of mobile and static sensors. The k-coverage problem through mobile sensors is also studied in both mobile WSNs

and in hybrid setting in [1,12,16,20,26,28,29,31]. Howard *et al.* [13] proposed a potential-field based algorithm and ensured that the initial configuration of the nodes quickly spreads out to maximize coverage area. A virtual-force-based sensor movement strategy to enhance network coverage is considered in [32].

Kumar *et al.* [15,16] studied barrier coverage where the sensors need to form a barrier to prevent intruders from crossing the barrier. They contributed significantly on theoretical foundations and provided several local algorithms that work based on limited neighborhood information. They also studied density requirements for achieving barrier coverage preserving connectivity requirements. These papers [4,21,23,24] studied several different aspects of barrier coverage. *Target coverage* problem is considered in [3,7,17] for tracking both static and moving targets.

As we mentioned above, most of the existing works focus on static coverage (full and barrier) with stationary configurations of sensors. Even with mobile sensors, they focus mostly on achieving an optimized deployment through their mobility without exploring dynamic coverage [18]. The concept of this kind of coverage was originally studied in the context of robotics, e.g. [2], focusing mainly on the coverage frequency. Li *et al.* [18] were the first to study sweep coverage which necessitates the dynamic coverage in the context of mobile WSNs. They studied sweep coverage with the objective of minimizing the number of mobile sensors (i.e., MinSensorSweep) for required sweep coverage time period. The sweep time period is given for any POI and the objective was to fulfill that timing requirement minimizing the number of sensors. MinSensorSweep is further studied by [10,11,18,22,30].

Roadmap. We discuss model and preliminaries in Sect. 2. We then present and analyze two approximation algorithms for Not-Predefined-Start scenario of MinDelaySweep in Sect. 3. We repeat this process for Predefined-Start scenario in Sect. 4. We conclude in Sect. 5 with a short discussion.

2 Model and Preliminaries

We consider a set $\mathcal{M} = \{s_1, s_2, \ldots, s_k\}$ of k mobile sensors and a set $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ of m static POIs in the Euclidean plane $\mathbb{R}^2$. We denote by c_i the position of the POI $p_i \in \mathcal{P}$ in $\mathbb{R}^2$, which is fixed. Each POI $p_i \in \mathcal{P}$ has a unique identifier (UID). We denote by $\mathsf{dist}(p_i, p_j)$ the Euclidean distance between two POIs p_i and p_j. We assume that mobile sensors $s_i \in \mathcal{M}$ move at a constant speed v in $\mathbb{R}^2$. If speed is not the same, then the speed can be taken as a ratio to compute the length of the trajectory that the mobile sensor should traverse. Each sensor s_i has the limited sensing range and the POIs are said to be *covered* (i.e., visited or scanned) only when the mobile sensors *pass through* the positions of the POIs. We assume that the time is divided into time units and the unit distance corresponds to one time unit.

We consider two scenarios of the problem. In the first scenario, called Predefined-Start, all the mobile sensors $s_i \in \mathcal{M}$ start from the predefined positions to scan specific POIs along their trajectories and they go back to their

predefined start positions after one complete traversal of their trajectories. They then recharge their battery and start their next traversal. We ignore the time to recharge mobile sensors assuming that it is negligible; if it is not the case then, the recharging delay can also be taken into account while computing trajectories. This represents a large class of mobile WSN applications where sensors are highly power-constrained. In the second scenario, called Not-Predefined-Start, the mobile sensors $s_i \in \mathcal{M}$ have no constraint on (battery) power and can scan POIs in their trajectories for a quite long time without recharging. Therefore, the mobile sensors do not need to go back to the predefined positions.

We study the sweep coverage problem with the objective of minimizing the coverage delay, which we denote by MinDelaySweep. More precisely, we aim to schedule k mobile sensors in $\mathcal{M}$ to scan the m POIs in $\mathcal{P}$ such that they delay between subsequent visits to POIs by sensors is minimized and each POI is scanned at least once in one traversal of any sensor. We have the following definitions for MinDelaySweep.

Definition 1 ([18]). *A POI is said to be* t-sweep covered *by a sweep algorithm* $\mathcal{F}$ *if and only if it is covered at least once every* t *time units by the sensors scheduled by* $\mathcal{F}$.

Definition 2 ([18]). *A set of POIs are said to be* globally sweep covered *by a sweep algorithm* $\mathcal{F}$ *if and only if every POI* p_i *is* t_i*-sweep covered under* $\mathcal{F}$.

Definition 3. *The* sweep coverage delay *is the maximum* t_i *among POIs under* $\mathcal{F}$.

Definition 4. *Given a set of* k *mobile sensors and a set of* m *POIs, the* sweep coverage delay minimization *problem,* MinDelaySweep*, is to schedule* k *mobile sensors to globally sweep cover the POIs such that the sweep coverage delay is minimized.*

Given $\mathcal{M}$ and $\mathcal{P}$ in the Euclidean plane $\mathbb{R}^2$ and Not-Predefined-Start scenario, the deployment of the POIs in $\mathcal{P}$ can be represented by an undirected weighted complete graph $G = (V, E, \mathfrak{w})$, where V is the set of all POIs in $\mathcal{P}$ and, for any two POIs p_i and p_j, there is an edge between them, i.e., $(p_i, p_j) \in E$. Moreover, there is a weight function $\mathfrak{w} : E \rightarrow \mathbb{R}^+$ such that $\mathfrak{w}(e) = \mathsf{dist}(p_i, p_j)$ for an edge $e = (p_i, p_j) \in E$. We denote by $c_{\max} := \max_{e \in E} \mathfrak{w}(e)$, the maximum weight edge in E.

In Predefined-Start scenario, the deployment of the POIs in $\mathcal{P}$ and the sensors in $\mathcal{M}$ can be represented by an undirected weighed graph $G' = (V, V', E', \mathfrak{w})$, where V is the set of all POIs in $\mathcal{P}$, V' is the set of predefined start positions of the mobile sensors in $\mathcal{M}$, for any two POIs p_i and p_j, there is an edge $e = (p_i, p_j) \in E'$, and $\mathfrak{w}(e) = \mathsf{dist}(p_i, p_j)$. Furthermore, for any start position s_i and any POI p_j, there is an edge (s_i, p_j) between them such that $e' = (s_i, p_j) \in E$ and $\mathfrak{w}(e') = \mathsf{dist}(s_i, p_j)$.

Given G or G' and k mobile sensors in $\mathcal{M}$, the goal in MinDelaySweep is to find a set of k trajectories to scan all m POIs in $\mathcal{P}$ such that the maximum length

among k trajectories in minimized. MinDelaySweep is NP-hard. Take the Not-Predefined-Start scenario and $|\mathcal{M}| = 1$ (there is only one sensor in $\mathcal{M}$ such that $k = 1$). This setting is equivalent to finding the minimum length Hamiltonian path that passes through all POIs in $\mathcal{P}$ which is a well-known NP-hard problem. Therefore,

Theorem 1. MinDelaySweep *problem is NP-hard.*

Since MinDelaySweep is NP-Hard, we look for approximation algorithms. We use the existing literature on traveling salesman problem (TSP) and tree cover problem (TC) and derive four approximation algorithms. Two approximation algorithms are for Not-Predefined-Start scenario and the rest two are for Predefined-Start scenario.

We now provide several definitions which are useful later in the algorithms. A *tour* is a path that visits all the POIs starting from some initial vertex (POI) v_1 and ends at the same vertex v_1 in G after visiting all the nodes of G exactly once, i.e., $R = \{v_1, v_2, \ldots, v_m, v_1\}$. Note that two subsequent nodes in R are connected by an edge. A *subtour* is a tour that is obtained by dividing the tour R into more than one segments such that a segment contains all the vertexes in the tour R in the same order starting from some initial vertex of the subtour to the ending vertex of that subtour. For example, if R is divided into two tours R_1 and R_2 starting from v_1, then $R_1 = \{v_1, \ldots, v_t, v_1\}$ and $R_2 = \{v_{t+1}, \ldots, v_m, v_{t+1}\}$.

A tree cover of a graph G is a set of trees $\mathcal{T} = \{T_1, \ldots, T_k\}$ such that $V = \bigcup_{i=1}^{k} V(T_i)$. The cost of the tree T_i is defined by $\mathsf{Cost}(T_i) = \sum_{e \in T_i} \mathfrak{w}(e)$. The cost of a tree cover $\mathcal{T}$ is $\max_{T_i \in \mathcal{T}} \mathsf{Cost}(T_i)$. An r-rooted tree cover of a graph G is a tree cover $\mathcal{T}$, where each tree $T_i \in \mathcal{T}$ has a distinct root $r \in \mathcal{Z}$, where $\mathcal{Z} \subset V$ denotes a set of root nodes. Note that the roots of T_i and T_j for $i \neq j$ must be distinct. However, trees may share some nodes and edges to other trees.

3 TSP Tour Based MinDelaySweep Algorithms

We present two algorithms, Not-Predefined-Start-TSP and Predefined-Start-TSP. Not-Predefined-Start-TSP is suitable for Not-Predefined-Start scenario of MinDelaySweep and Predefined-Start-TSP is suitable for Predefined-Start scenario.

Not-Predefined-Start-TSP Algorithm. The pseudocode of Not-Predefined-Start-TSP is given in Algorithm 1. The basic idea behind Not-Predefined-Start-TSP is to find a trajectory for each sensor and ask that sensor to cover (scan) the POIs that are in that trajectory. To find the trajectories, we use the the well-known ideas on constructing a TSP tour and dividing the tour to obtain k trajectories. For the TSP tour construction in G, Not-Predefined-Start-TSP selects a node, say v_1, among the POIs as a starting vertex and uses a known algorithm for TSP (say Christofides [6]).

Denote the TSP tour obtained through this construction by $R := \{v_1, v_2, \ldots, v_m, v_1\}$ and let $\mathsf{Cost}(R) = L$. R is then divided into k-subtours (or

Algorithm 1. Not-Predefined-Start-TSP

1 Pick a POI v_1;
2 Use an algorithm for TSP and find a TSP tour $R = (v_1, v_2, v_3, \ldots, v_n, v_1)$ with $Cost(R) = L$;
3 Let $c_{\max}$ be the longest edge in R. Remove $c_{\max}$ from R such that $L = L - c_{\max}$;
4 **For** $j \leq 1$ to $j < k$ **do**
5 Find the last POI $v_{l(j)}$ such that the cost of the path from v_1 to $v_{l(j)}$ along R is not greater than $\frac{j}{k}L$;
6 Obtain k subtours as
7 $R_1 = (v_1, \ldots, v_{l(1)})$, $R_2 = (v_{l(1)+1}, \ldots, v_{l(2)}), \ldots, R_k = (v_{l(k-1)+1}, \ldots, v_n)$,
8 Add an edge from the last node in each subtour to its first node such that
9 $R_1 = (v_1, \ldots, v_{l(1)}, v_1)$, $R_2 = (v_{l(1)+1}, \ldots, v_{l(2)}, v_{l(1)+1}), \ldots,$
$R_k = (v_{l(k-1)+1}, \ldots, v_n, v_{l(k-1)+1})$,
10 Assign one sensor to each $R_j, 1 \leq j \leq k$;

trajectories), say $R_j, 1 \leq j \leq k$, of almost equal length starting from v_1. The division process works as follows. Let $c_{\max}$ be the longest edge in R. Then, $c_{\max}$ is removed from R such that $L = L - c_{\max}$. Now, starting from v_1, the POIs that fall in R upto length L/k are assigned to R_1 such that $R_1 = \{v_1, \ldots, v_{l(1)}\}$, where $v_{l(1)}$ the last vertex in R_1. Similarly, starting from v_1, the POIs that fall in R upto length $2L/k$ except the POIs that are already in R_1 are assigned to R_2 such that $R_2 = \{v_{l(1)+1}, \ldots, v_{l(2)}\}$, where $v_{l(2)}$ is the last vertex in R_2. According to this division, $R_k = \{v_{l(k-1)+1}, \ldots, v_n\}$ and we have k sub-tours. Moreover, we have that $\mathsf{Cost}(R_i)$ and $\mathsf{Cost}(R_j)$, $1 \leq i, j \leq k, i \neq j$ are at most the factor of 2 from each other. The reasoning is that R is divided in to equal fragments and the length of the edge between the last POI of one sub-tour and the first POI of next subtour changes the length of each fragment only by the factor of 2. These k subtours are updated by adding the starting vertex of each subtour at the end of that tour to obtain one trajectory such that $R_1 = \{v_1, \ldots, v_{l(1)}, v_1\}, R_2 = \{v_{l(1)+1}, \ldots, v_{l(2)}, v_{l(1)+1}\}$, and so on. As sensors do not have predefined start positions and there are k sensors, these sensors are randomly assigned to traverse k trajectories computed. We prove the following theorem for the approximation ratio of Not-Predefined-Start-TSP.

Theorem 2. *The approximation ratio of* Not-Predefined-Start-TSP *is at most* 2δ*, where* δ *is the approximation ratio of an algorithm for TSP.*

Proof. Let L_{OPT} be the length of the optimal tour for TSP in G. Moreover, let L be the length of the TSP tour obtained using an algorithm for TSP. We have that $L = \delta \cdot L_{OPT}$, where δ be the approximation ratio of an algorithm used to compute the TSP tour. Since the tour R is divided into k subtours, we have that the time t_{TSP} required to sweep each subtour R_j is such that $t_{TSP} \leq \frac{2\delta \cdot L_{OPT}}{v}$. This is because the lengths of the subtours are within the factor of 2 from each other. Let t_{OPT} be the time period in the optimal solution. In other words, there is a sweep algorithm $\mathcal{A}$ in which if we use k sensors moving at constant speed

Algorithm 2. Predefined-Start-TSP

Pick a POI v_1;
Use an algorithm for TSP and find a TSP tour $R = (v_1, v_2, v_3, \ldots, v_n, v_1)$ with $Cost(R) = L$;
For $j \leq 1$ to $j < k$ **do**
    Find the last POI $v_{l(j)}$ such that the cost of the path from v_1 to $v_{l(j)}$ along R is not greater than $\frac{j}{k}(L - 2c_{\max}) + c_{\max}$;
Obtain k-tour by forming k subtours as
    $R'_1 = (v_1, \ldots, v_{l(1)})$, $R'_2 = (v_{l(1)+1}, \ldots, v_{l(2)}), \ldots$. $R'_k = (v_{l(k-1)+1}, \ldots, v_n)$,
Assign each subtour R'_j to mobile sensors $s_i \in \mathcal{M}$ such that $c(s_i, R'_j) \leq c(s_m, R'_j), s_i \neq s_m$;
For $j = 1$ to $j = k$ **do**
    Update the subtour R'_j by adding sensor s_j assigned to it in its beginning and end and denote it by R_j;

v each sensor will be visited in minimum time units. As L_{OPT} is the length of the shortest route for the corresponding TSP, we get $t_{OPT} \geq \frac{L_{OPT}}{v}$ for one mobile sensor. Therefore, the approximation ratio of Not-Predefined-Start-TSP is bounded by $\frac{t_{TSP}}{t_{OPT}} \leq \frac{\frac{2\delta \cdot L_{OPT}}{v}}{\frac{L}{v}} \leq 2\delta$. □

Since Christofides's algorithm [6] has approximation 1.5, we obtain:

Corollary 1. *Using Christofides's algorithm [6] for TSP tour,* Not-Predefined-Start-TSP *achieves the approximation ratio of at most* 3 *for* MinDelaySweep.

Predefined-Start-TSP Algorithm. The pseudocode of Predefined-Start-TSP algorithm is given in Algorithm 2. The main idea of this algorithm is to find a TSP tour R similar to Not-Predefined-Start-TSP. However, due to the predefined start positions of the sensors, R need to be carefully split into k subtours and also the sensors needs to be carefully assigned to cover the POIs in those subtours. We use the approach of Frederickson *et al.* [9] to divide the tour R into k-subtours. Moreover, after the tour is divided into k-subtours, the mobile sensors that minimizes the cost $\mathsf{Cost}(s_i, R_j)$ is assigned to R_j to provide the coverage for the POIs in R_j, where $\mathsf{Cost}(s_i, R_j)$ is the minimum distance from the position of any sensor $s_i \in \mathcal{M}$ to any node in subtour R_j.

The k-SPLITOUR algorithm of Frederickson *et al.* [9] starts from some vertex, say v_1, and finds the last POI $v_{l(j)}$ in R such that the cost of the path from v_1 to $v_{l(j)}$ is not greater than $\frac{j}{k}(L - 2c_{\max}) + c_{\max}$, where $c_{\max}$ is the maximum weight of an edge in E. Then it forms k subtours as $R'_1 = \{v_1, \ldots, v_{l(1)}\}, R'_2 = \{v_{l(1)+1}, \ldots, v_{l(2)}\}, \ldots, R'_k = \{v_{l(k-1)+1}, \ldots, v_n\}$. Each subtour R'_j is assigned to a sensor $s_i \in \mathcal{M}$ which minimizes the cost $\mathsf{Cost}(s_i, R'_j)$. Finally, each subtour R'_j is updated by adding the sensor that assigned to cover it in the beginning and end to get R_j, i.e., if a sensor s_i is assigned to R'_j, then we have that $R_j = \{s_i, v_{l(j-1)+1}, \ldots, v_{l(j)}, s_i\}$. That is, R_j is the trajectory for sensor s_i. We prove the following results for the approximation ratio achieved by Predefined-Start-TSP.

Lemma 1. *Let C_k be the cost of the largest of the k-subtours generated by Algorithm 2. Algorithm 2 guarantees that $C_k \leq \frac{L}{k} + 2c_{\max}(2 - \frac{1}{k})$.*

Proof. We have that $\mathsf{Cost}(R'_1) \leq \frac{1}{k}(L - 2c_{\max}) + c_{\max}$. Similarly, $\mathsf{Cost}(R'_k) \leq \frac{1}{k}(L - 2c_{\max}) + c_{\max}$. For each j, $1 \leq j \leq k-2$, $\mathsf{Cost}(R'_j) \leq \frac{1}{k}(L - 2c_{\max})$.

Now, while updating the tours by adding the sensors that are assigned to the subtour, we have that $\mathsf{Cost}(R_1) \leq \mathsf{Cost}(R'_1) + \mathsf{Cost}(v_1, s_1) + \mathsf{Cost}(v_{l(1)}, s_1)$. We have that $\mathsf{Cost}(s_1, v_1) + \mathsf{Cost}(v_{l(1)}, s_1) \leq 3c_{\max}$ due to triangle equality, since $\mathsf{Cost}(v_1, v_{l(1)}) \leq c_{\max}$. Therefore, $\mathsf{Cost}(R_1) \leq \frac{1}{k}(L - 2c_{\max}) + 4c_{\max}$. Similarly, $\mathsf{Cost}(R_k) \leq \frac{1}{k}(L - 2c_{\max}) + 4c_{\max}$. For each $j, 1 \leq j \leq k-2$, $\mathsf{Cost}(R_j) \leq \mathsf{Cost}(R'_j) + \mathsf{Cost}(s_j, v_{l(j)+1}) + \mathsf{Cost}(v_{l(j+1)}, s_j)$. Moreover, we have that $\mathsf{Cost}(s_j, v_{l(j)+1}) + \mathsf{Cost}(v_{l(j+1)}, s_j) \leq 4c_{\max}$.

Thus, $C_k = \max_j \mathsf{Cost}(R_j) \leq \frac{1}{k}(L - 2c_{\max}) + 4c_{\max} \leq \frac{L}{k} + 2c_{\max}(2 - \frac{1}{k})$. □

We immediately have the following lemma for the optimal cost.

Lemma 2 ([9]). *Let C^*_k be the cost of the largest subtour in an optimal solution for the k-subtours. We have that $C^*_k \geq \frac{1}{k}C^*$, where C^* is the cost of an optimal TSP tour.*

Theorem 3. Predefined-Start-TSP *achieves the approximation ratio of at most $\delta + 2 - \frac{1}{k}$, where δ is the approximation ratio of an algorithm for TSP.*

Proof. We have that $L \leq \delta C^*$, where C^* is the cost of the optimal solution for TSP. Moreover, we have that $C^*_k \geq \frac{1}{k}C^*$, and due to traingle inequality, $c_{\max} \leq \frac{1}{2}C^*_k$ [9]. Therefore, combining Lemmas 1 and 2, and substituting L, $c_{\max}$, and C^*_k by their values, the theorem follows. □

Corollary 2. *Using Christofides's algorithm [6] for TSP tour,* Predefined-Start-TSP *achieves the approximation ratio of at most $\frac{7}{2} - \frac{1}{k}$ for* MinDelaySweep.

4 Tree Cover Based MinDelaySweep Algorithms

We present two algorithms, Not-Predefined-Start-TC and Predefined-Start-TC. Not-Predefined-Start-TC is suitable for Not-Predefined-Start scenario of MinDelaySweep and Predefined-Start-TC is suitable for Predefined-Start scenario of MinDelaySweep.

Not-Predefined-Start-TC Algorithm. The pseudocode of Not-Predefined-Start-TC is given in Algorithm 3. Not-Predefined-Start-TC uses an unrooted tree cover construction algorithm Unrooted-TC(G, k, B) to compute a set of k trees $\mathcal{T} = \{T_1, \ldots, T_k\}$. These k trees are then converted to k tours and assign one sensor in each tree to scan the POIs that belong to those trees.

We discuss here the Unrooted-TC(G, k, B) algorithm of Even *et al.* [8] to compute a set $\mathcal{T}$ of k trees given as input these three parameters: the graph G, the number of trees k (which is equal to the number of mobile sensors in $\mathcal{M}$),

Algorithm 3. Not-Predefined-Start-TC

1 Use an algorithm Unrooted-TC(G, k, B) for the unrooted version of TC problem and find a set of k trees $\mathcal{T} = \{T_1, \ldots, T_k\}$;
2 Transform each tree $T_i \in \mathcal{T}$ into a tour P_i using an appropriate tour construction algorithm given a tree;
3 Assign a mobile sensor to each tour P_i to cover the POIs that are in that tour;

and a bound on the cost of each tree B. Unrooted-TC(G, k, B) then either returns that the bound B chosen for the cost of the tree is too small or finds a tree cover $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of cost at most $4B$ for each $T_i, 1 \leq i \leq k$.

Unrooted-TC(G, k, B) of Even *et al.* [8] works as follows. It first removes the edges of G with weight larger than B. This may divide G into a set of connected components which are denoted by $\{G_i\}$. Then a minimum spanning tree MST_i is computed for each G_i. After that Cost(MST_i) is computed and this cost is divided by $2B$ to determine the number of trees k_i required to cover the vertices in G_i. If $\sum_i (k_i + 1) > k$ for k_i determined for every G_i, then it gives more than k trees which means that the estimate of B is small and Unrooted-TC(G, k, B) has to repeat this process with larger B such that $\sum_i (k_i + 1)$ is equal to k. We need exactly k trees since we have k sensors in $\mathcal{M}$. When $\sum_i (k_i + 1) = k$, then each MST_i is decomposed to $k_i + 1$ trees T_i^j such that Cost$(T_i^j) \in [2B, 4B)$, where $1 \leq j \leq k_i$. The leftover of MST_i after constructing k_i trees is assigned to L_i which is called the leftover tree. Therefore, Unrooted-TC(G, k, B) returns in total k trees and Even et al. [8] showed that the cost of each tree is at most $4B$.

Not-Predefined-Start-TC then transforms the k trees obtained using Unrooted-TC(G, k, B) into k tours as follows. For each edge $(i, j) \in T$, we ask Not-Predefined-Start-TC to add another edge between i and j with the same weight $w(ij)$. Note that the subgraph consisting only of the edges in T and these new duplicate edges provides an Euler cycle. Note also that the total cost of the Euler cycle is 2 times Cost(T). Let P be that cycle. Then the tour is obtained as follows. If P has a sequence like $i, j, l, \ldots, o, i, p$, then we replace it by $i, j, l, \ldots, o, p$ (removing the second i in the sequence). The difference here in the total cost of P is only due to the deletion of the second i. As the edge weights in G satisfy triangle inequality, we have that $\mathfrak{w}(op) \leq \mathfrak{w}(oi) + \mathfrak{w}(ip)$. Therefore, this shortcut process does not increase the cost of P and it is within 2 times the cost of T.

Even *et al.* [8] proved the correctness of Unrooted-TC(G, k, B) in the sense that it returns a set of k trees with desired properties if proper cost bound B is provided as an input. Our discussion of the Unrooted-TC(G, k, B) algorithm of [8] for $\mathcal{T}$ construction is for an illustration purpose and other available algorithms for the unrooted tree cover problem can also be used in Line 1 of Algorithm 3 to compute $\mathcal{T}$. Therefore, we focus here on the general approximation ratio achieved by Not-Predefined-Start-TC.

Theorem 4. *The approximation ratio of* Not-Predefined-Start-TC *is at most* 2γ*, where* γ *is the approximation ratio of an algorithm for the unrooted version of TC.*

Proof. Let γ be the approximation ratio of the algorithm used to compute the solution for the unrooted version of TC. In the tour construction process, we increased the cost of each tree by a factor of at most 2. Therefore, the approximation of Not-Predefined-Start-TC is at most 2γ. □

Since Even *et al.*'s algorithm [8] has the approximation ratio of 4 for the unrooted version of TC, we obtain the following corollary.

Corollary 3. *Using the Even et al.'s algorithm [8]* Unrooted-TC(G, k, B), Not-Predefined-Start-TC *achieves the approximation ratio of* 8 *for* MinDelaySweep.

Predefined-Start-TC Algorithm. The pseudocode of Predefined-Start-TC is given in Algorithm 4. Predefined-Start-TC uses a rooted tree cover construction algorithm Rooted-TC(G, k, B) to compute a set of k trees $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ such that each tree T_i is rooted at a start position of a sensor. These k trees are then converted to k tours using an appropriate tour construction algorithm and the sensor that is in the start position (i.e., the root of the tree) is asked to to scan the POIs that fall in those trees.

We discuss here the Rooted-TC(G, k, B) algorithm of [8] which takes as input the same three parameters as in Unrooted-TC(G, k, B). Rooted-TC(G, k, B) then either returns that the bound B chosen for the cost of the tree is too small or finds a tree cover $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of cost at most $4B$ for each $T_i, 1 \leq i \leq k$.

Rooted-TC(G, k, B) removes edges with weights greater than B and compute k different minimum spanning trees T_i with k different roots as the starting positions of the sensors. Some of the trees in T_i can be empty in the sense that they may contain only the root node. Rooted-TC(G, k, B) then decomposes each tree T_i into j trees such that Cost$(T_i^j) \in [B, 2B)$, for every j, and assign the leftover of the tree T_i after dividing it into j trees to the leftover tree L_i. According to the construction Cost$(L_i) < B$. Each tree Cost(T_i^j) is then matched to the roots that are at distance at most B from it. If not all tree are matched, then it is the case that the bound B chosen is too small and Rooted-TC(G, k, B) repeats this k tree construction and matching by choosing a larger value of B. If all trees are matched then Rooted-TC(G, k, B) returns k trees rooted at the start positions of k sensors.

The Predefined-Start-TC algorithm then transforms the k trees obtained using Rooted-TC(G, k, B) into k tours similar to the technique we discussed in Not-Predefined-Start-TC. Each tour P_i constructed for the tree T_i has the cost that is at most 2 times the cost of T_i.

Even *et al.* [8] proved the correctness of Rooted-TC(G, k, B) in the sense that it returns a set of k tree with desired properties of proper cost B is provided as an input. Similar to Not-Predefined-Start-TC, our discussion of the Rooted-TC(G, k, B) algorithm of [8] for $\mathcal{T}$ construction is for an illustration purpose and other available algorithms for the rooted tree cover problem can also be used in Line 1 of Algorithm 3 to compute $\mathcal{T}$. Therefore, we prove the following theorem.

Algorithm 4. Predefined-Start-TC

1 Use an algorithm Rooted-TC(G, k, B) for the rooted version of TC problem and find a set of k trees $\mathcal{T} = \{T_1, \ldots, T_k\}$ rooted at the start positions of k sensors (the roots are different for each tree);
2 Transform each tree $T_i \in \mathcal{T}$ into a tour P_i using an appropriate tour construction algorithm given a tree;
3 Ask the mobile sensor in the start position that is in that tour P_i to cover the POIs that are in P_i;

Theorem 5. *The approximation ratio of* Predefined-Start-TC *is at most* 2γ*, where* γ *is the approximation ratio of an algorithm for the rooted version of TC.*

Proof. Let γ be the approximation ratio of the algorithm used to compute the solution for the rooted version of TC. Predefined-Start-TC modifies the tree cover that is obtained by Rooted-TC(G, k, B) to form a tour in the expense of factor 2 increase in the cost of each tree. Therefore, the approximation of Predefined-Start-TC is 2γ. □

Since Even *et al.*'s algorithm [8] has the approximation ratio of 4 for the rooted version of TC, we obtain the following corollary.

Corollary 4. *Using the Even et al.'s algorithm [8]* Rooted-TC(G, k, B) Predefined-Start-TC *achieves the approximation ratio of* 8 *for* MinDelaySweep.

5 Concluding Remarks

We considered the fundamental problem of sweep coverage in mobile WSNs. We studied this problem with the objective of minimizing the coverage delay given the limited set of k sensors to cover a set of m POIs in the Euclidean plane. For the future work, it is interesting to improve the approximation ratios of our algorithms. For the practical aspect, it is interesting to experimentally evaluate our algorithms, especially the tree cover based algorithms, on the performance they achieve in real world scenarios.

References

1. Bai, X., Xuan, D., Yun, Z., Lai, T.H., Jia, W.: Complete optimal deployment patterns for full-coverage and k-connectivity in wireless sensor networks. In: MobiHoc, pp. 401–410 (2008)
2. Batalin, M.A., Sukhatme, G.S.: Multi-robot dynamic coverage of a planar bounded environment. Technical report (2002)
3. Cardei, M., Thai, M.T., Li, Y., Wu, W.: Energy-efficient target coverage in wireless sensor networks. In: INFOCOM, pp. 1976–1984 (2005)
4. Chen, A., Kumar, S., Lai, T.H.: Designing localized algorithms for barrier coverage. In: MobiCom, pp. 63–74 (2007)

5. Chen, W., Chen, S., Li, D.: Minimum-delay pois coverage in mobile wireless sensor networks. EURASIP J. Wirel. Commun. Netw. **2013**, 262 (2013)
6. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report (1976)
7. Ding, L., Wu, W., Willson, J., Wu, L., Lu, Z., Lee, W.: Constant-approximation for target coverage problem in wireless sensor networks. In: INFOCOM, pp. 1584–1592, March 2012
8. Even, G., Garg, N., KöNemann, J., Ravi, R., Sinha, A.: Min-max tree covers of graphs. Oper. Res. Lett. **32**(4), 309–315 (2004)
9. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: FOCS, pp. 216–227 (1976)
10. Gorain, B., Mandal, P.S.: Approximation algorithms for sweep coverage in wireless sensor networks. J. Parallel Distrib. Comput. **74**(8), 2699–2707 (2014)
11. Gorain, B., Mandal, P.S.: Line sweep coverage in wireless sensor networks. In: COMSNETS, pp. 1–6 (2014)
12. Hefeeda, M., Bagheri, M.: Randomized k-coverage algorithms for dense sensor networks. In: INFOCOM, pp. 2376–2380 (2007)
13. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem. In: Asama, H., Arai, T., Fukuda, T., Hasegawa, T. (eds.) Distributed Autonomic Robotic Systems 5, pp. 299–308. Springer, Tokyo (2002). https://doi.org/10.1007/978-4-431-65941-9_30
14. Khani, M.R., Salavatipour, M.R.: Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM -2011. LNCS, vol. 6845, pp. 302–314. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22935-0_26
15. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. In: MobiCom, pp. 284–298 (2005)
16. Kumar, S., Lai, T.H., Balogh, J.: On k-coverage in a mostly sleeping sensor network. In: MobiCom, pp. 144–158 (2004)
17. Li, D., Cao, J., Liu, M., Zheng, Y.: K-connected target coverage problem in wireless sensor networks. In: Dress, A., Xu, Y., Zhu, B. (eds.) COCOA 2007. LNCS, vol. 4616, pp. 20–31. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73556-4_5
18. Li, M., Cheng, W., Liu, K., He, Y., Li, X.Y., Liao, X.: Sweep coverage with mobile sensors. IEEE Trans. Mob. Comput. **10**(11), 1534–1545 (2011)
19. Liang, W., Lin, X.: Approximation algorithms for min-max cycle cover problems. IEEE Trans. Comput. **64**(3), 600–613 (2014)
20. Liu, B., Brass, P., Dousse, O., Nain, P., Towsley, D.: Mobility improves coverage of sensor networks. In: MobiHoc, pp. 300–308 (2005)
21. Liu, B., Dousse, O., Wang, J., Saipulla, A.: Strong barrier coverage of wireless sensor networks. In: MobiHoc, pp. 411–420 (2008)
22. Lu, X., Chen, S., Chen, W., Li, D.: Sweep coverage with mobile sensors on two-way road. In: Wang, R., Xiao, F. (eds.) CWSN 2012. CCIS, vol. 334, pp. 335–345. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36252-1_31
23. Saipulla, A., Westphal, C., Liu, B., Wang, J.: Barrier coverage with line-based deployed mobile sensors. Ad Hoc Netw. **11**(4), 1381–1391 (2013)
24. Wan, P.J., Yi, C.W.: Coverage by randomly deployed wireless sensor networks. IEEE/ACM Trans. Netw. **14**(SI), 2658–2669 (2006)

25. Wang, B.: Coverage problems in sensor networks: a survey. ACM Comput. Surv. **43**(4), 32:1–32:53 (2011)
26. Wang, D., Liu, J., Zhang, Q.: Probabilistic field coverage using a hybrid network of static and mobile sensors. In: IWQoS, pp. 56–64 (2007)
27. Wang, J.: Efficient point coverage in wireless sensor networks. J. Comb. Optim. **11**, 291–304 (2006)
28. Wang, W., Srinivasan, V., Chua, K.-C.: Trade-offs between mobility and density for coverage in wireless sensor networks. In: MobiCom, pp. 39–50 (2007)
29. Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R., Gill, C.: Integrated coverage and connectivity configuration in wireless sensor networks. In: SenSys, pp. 28–39 (2003)
30. Xi, M., Wu, K., Qi, Y., Zhao, J., Liu, Y., Li, M.: Run to potential: sweep coverage in wireless sensor networks. In: ICPP, pp. 50–57 (2009)
31. Zhou, Z., Das, S., Gupta, H.: Connected k-coverage problem in sensor networks. In: ICCCN, pp. 373–378 (2004)
32. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. In: INFOCOM (2003)