Shorten the Trajectory of Mobile Sensors in Sweep Coverage Problem

Yuchen Feng, Xiaofeng Gao[§], Fan Wu, Guihai Chen
Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
sjtufyc@gmail.com, {gao-xf, fwu, gchen}@cs.sjtu.edu.cn

***Abstract*—Sweep coverage is the problem of scheduling mobile sensors to cover a set of PoIs (Points of Interest) periodically. To improve the monitoring efficiency, multiple mobile sensors can cooperate with each other together to finish the task. With limited unrenewable battery power, how to schedule multiple mobile sensors to visit all PoIs with minimum energy consumption is a challenging problem. In this paper, we introduce an optimization problem to minimize the makespan of mobile sweep routes (M^3SR) and prove its NP-hardness on 2D plane. We then design a greedy algorithm named GD-Sweep and an approximation named BS-Sweep to solve M^3SR. We prove theoretically that BS-Sweep is a constant-factor approximation with approximation ratio of 6, while exhibit numerically that GD-Sweep performs better according to various simulation results. Compared with previous literature, our design can reduce the length of sweep routes more efficiently.**

I. Introduction

Wireless sensor networks (WSNs) have attracted great attention in last decades. Coverage problem is a fundamental problem in WSN [1]–[5]. As a newly-born coverage problem, sweep coverage become more and more popular [6]–[13]. Different from traditional coverage problems, the goal of sweep coverage is to schedule mobile sensors to monitor all PoIs periodically. For example, in Fig. 1, the red dots represent the PoIs and there are 6 distinct routes with 10 mobile sensors. Mobile sensors in each route need to scan all PoIs along this route once within a sweep period. With limited unrenewable battery power, how to schedule multiple mobile sensors to visit all PoIs within shortest possible time is a challenging problem.

Many literatures have studied the problem of sweep coverage [6]–[13]. Most of those studies consider the sweep coverage problem from two aspects: finding the minimum number of mobile sensors required [6]–[8], [12]–[14] and finding the minimum sweep period to cover all PoIs [15]–[18]. No matter what the problems are, the length of the route for mobile sensors is critical and we always expect to find the shortest route. In the previous research, using single mobile sensor to achieve sweep coverage is equivalent to solving the Euclidean Traveling Salesman Problem (TSP). However, when using multiple mobile sensors to accomplish the sweep coverage, most of previous researches adopted the following two ideas:

This work has been supported in part by the National Natural Science Foundation of China (Grant number 61202024, 61472252, 61133006, 61422208), China 973 project (2012CB316200), the Natural Science Foundation of Shanghai (Grant No.12ZR1445000), Shanghai Educational Development Foundation (Chenguang Grant No.12CG09), Shanghai Pujiang Program 13PJ1403900, and in part by Jiangsu Future Network Research Project No. BY2013095-1-10 and CCF-Tencent Open Fund.

[§]X.Gao is the corresponding author.

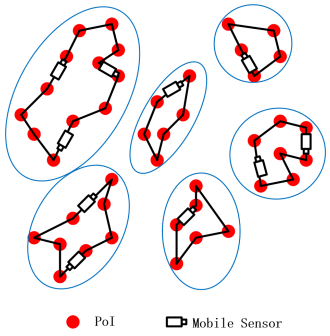


Fig. 1. An example sweep coverage with 44 PoIs, 10 mobile sensors, and 5 routes.

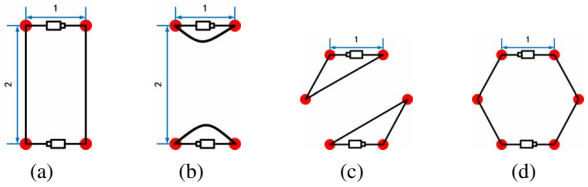


Fig. 2. Two counterexamples to illustrate the disadvantages of two commonly used methods for multiple sensor routes.

1) Partition the PoIs according to their positions and compute distinct TSP cycles for each mobile sensor [8], [9], [13], [19], [20].
2) Compute a globe TSP cycle according to the PoIs and then split it into several loops, each for a mobile sensor distinctively [11], [15], [16].

After careful investgation, we find out that both of the two ideas have their disadvantages, and neither of them can get the optimal solutions for sweep coverage. Fig. 2 exhibits two counterexamples. In Fig. 2(a), originally two mobile sensors work together to cover one TSP cycle with a total length of 6. However, if they work separately with two cycles, then the TSP length can be reduced to 4 as shown FIg. 2(b). On the other hand, in Fig. 2(c), two mobile sensors sweep cover 6 PoIs with 2 distinct cycles, while a better way is to work collaboratively with only one TSP cycle in Fig. 2(d). From this figure, we can see that either working together with one TSP cycle or working separately with different cycles may not bring the optimal solution. We can thereby combine the two ideas together to reduce the route length just as Fig. 1.

In this paper, we propose a novel and general method for sweep coverage in which mobile sensors can move along the same trajectory or isolated trajectory separately. If every

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

mobile sensor works simultaneously, then the longest route determines the scanning period, which is also called the makespan. If we could reduce the length of the makespan route, then we could shorten the scanning period in each round. Correspondingly, our goal is to *minimize the makespan of the mobile routes*, and we refer it as M^{3}SR problem. To solve the M^{3}SR problem, we design a greedy algorithm named GD-Sweep and an approximation named BS-Sweep. We theoretically prove that BS-Sweep is a constant-factor approximation with approximation ratio 6. We also provide simulations to compare our designs with previous works. The numerical experiments validate the advantages of our algorithms and GD-Sweep performs the best among these algorithms.

The remaining of our paper is organized as follows: We briefly survey the related literature in Sec. II. We formally define our model and problem in Sec. III. In Sec. IV, we present a greedy algorithm named GD-Sweep and an approximation named BS-Sweep with approximation ratio analysis. The comparisons of our algorithms with previous works are further exhibited by extensive numerical experiments reported in Sec. V. We finally conclude our paper in Sec. VI.

II. Related Works

Previous researchers have made many efforts to solve problems about sweep coverage. After N. Bisnik [10] first proposed the concept of sweep coverage, the researches regarding this topic become more vast [6]–[13]. Due to the limited battery energy and excessive producing cost of the mobile sensors, related literatures usually focused on these problems: finding the minimum sweep period and finding the minimum number of mobile sensors required with a fixed sensor velocity.

To solve these problems, [8], [9], [13], [19], [20] partitioned the targets according to the position and the access cycle of the targets. [11], [15], [16] splitted a TSP-based path into several loops distinctively.

In [7], the authors proposed two centralized algorithm for two scenarios about Global T-Sweep Coverage respectively. The first centralized algorithm named CSWEEP is to compute a global TSP cycle of PoIs and split it into several loops each for a mobile sensor.

In [8], the authors designed a centralized algorithm named MinExpand to find the minimum number of mobile sensors required for sweep coverage. MinExpand gradually deploys more mobile sensors and schedules distinct sweep route for new mobile sensor until all PoIs are Global T-Sweep Coverage.

In [9], the authors proposed a heuristic, called the perpendicular-distance-based algorithm (PDBA), to minimize the number of employed mobile sensors and the total energy consumed by the mobile sensors. The authors chose the PoI to join the route of a mobile sensor whose perpendicular distance to the bottom was small. The algorithm deploys more mobile sensors when the scan path of the existing mobile sensors exceeds the length constraint.

There are some other types of sweep coverage. In [11], the authors presented a new problem in which every mobile sensor should not only scan the PoI periodically but also visit a base station periodically and proposed an algorithm ITSP to solve it. In [12], the authors discussed a new problem, which is called Area Sweep Coverage, and provided an approximation to solve it. Above all, sweep coverage is a promising topic.

III. Problem Formulation

Assume that there are n PoIs $P = \{p_1, p_2, \cdots, p_n\}$ and m mobile sensors $S = \{s_1, s_2, \cdots, s_m\}$ in a sensing field and the coordinates of each PoI are given. We define that a PoI p_i is covered if and only if a mobile sensor s_j gets to p_i. With these assumptions, We define the *Global T-Sweep Coverage* as follows:

Definition 1. ***(Global T-Sweep Coverage):*** *A set of PoIs is said to be* Global T-sweep Covered *iff. all PoIs are scanned at least once by the mobile sensors within T time units.*

In this work, we assume that mobile sensors can move along the same or different trajectories with other sensors and the speed of all mobile sensors are the same. We also assume that when several sensors move along the same trajectory, the length of sweep route will be evenly divided for these sensors. Our goal in this paper is to shorten the sweep coverage duration, which is equivalent to shorten the longest sweep route. We consider it as the makespan of all mobile sweep routes. Then we are able to define the M^{3}SR problem as follows:

Definition 2. ***(M^{3}SR problem):*** *Given the locations of n PoIs and m mobile sensors, the Minimum Makespan of Mobile Sweep Routes problem (M^{3}SR) is to schedule mobile sweep routes to minimize the makespan.*

We can formulate M^{3}SR formally into a programming:

$$\textbf{min} \quad weight_q(P) = \max_{i=1\cdots q}\left(\frac{Tsp(P_q^i)}{|S_q^i|}\right), \quad q \in \{1, 2, \cdots, m\}$$

$$\textbf{s.t.} \quad \bigcup_{i=1}^{q} S_q^i = S, \quad S_q^i \neq \emptyset$$

$$S_q^i \bigcap_{i \neq j} S_q^j = \emptyset, \quad (i, j \in \{1, 2, \cdots, q\})$$

$$\bigcup_{i=1}^{q} P_q^i = P, \quad P_q^i \neq \emptyset$$

$$P_q^i \bigcap_{i \neq j} P_q^j = \emptyset, \quad (i, j \in \{1, 2, \cdots, q\})$$

In this programming, we partition all PoIs and mobile sensors into q classes ($q \in \{1, 2, \cdots, m\}$). P_q^i and S_q^i denote the PoI set and the mobile sensor set that belong to i^{th} class. Let $Tsp(P_q^i)$ denote the length of the TSP cycle for PoIs in the i^{th} group, and $weight_q(P)$ denote the makespan of mobile sweep routes when all PoIs and mobile sensors are assigned to q distinct routes. For each q, we can get one $weight_q(P)$ and the minimum of these $weight_q(P)$ is the answer to M^3SR.

Theorem 1. *The 2D version of M^{3}SR is NP-hard.*

Proof: We consider a special case when $m = 1$. Then M^{3}SR in 2D case requires to find a shortest sweep route for a single mobile sensor to visit all the PoIs. Obviously, it is a well known Euclidean Traveling-Salesman-Problem (TSP) which is NP-hard. The TSP is a special case of the M^{3}SR in 2D case. Thus M^{3}SR is NP-hard. ■

Usually it is not easy to find a TSP in polynomial time with good performance bound, but if we consider the relationship between Traveling Salesman Problem and Minimum Spanning

Tree Problem, we can replace the $Tsp(P_q^i)$ with $Mst(P_q^i)$. After that we can convert the MST into TSP cycle. We then define a new problem named MST-M^{3}SR as follows:

Definition 3. ***(MST-M^{3}SR):*** *MST-M^{3}SR is a little different from M^{3}SR. The difference between them is that* $weight(P) = \max_{i=1\cdots q}\left(\frac{Mst(P_q^i)}{|S_q^i|}\right)$ *in MST-M^{3}SR, where* $Mst(P_q^i)$ *denotes the weight of the minimum spanning tree for the PoIs in* P_q^i *and the distance between two PoIs is the Euclidean distance.*

IV. Algorithm Design

In this section, we assume that n PoIs are located on 2D plane. Then we design a greedy algorithm named GD-Sweep and an approximation named BS-Sweep to solve M^{3}SR problem. We prove theoretically that BS-Sweep is a constant-factor approximation with approximation ratio of 6.

It should be noticed that we use Christofides algorithm [21] to solve the TSP problem with approximation ratio of 1.5.

A. GD-Sweep

GD-Sweep is a greedy algorithm. The input of GD-Sweep is all coordinates of the PoIs on 2D plane and the output of GD-Sweep is a partition of PoIs and mobile sensors. The algorithm can be shown in two parts:

1) Divide all PoIs into m groups to make the longest length of all TSP cycles for these m groups as short as possible and assign one mobile sensor for each PoI group.
2) Combine some PoI groups and mobile sensors together to make the length of sweep routes shorter.

The detail of GD-Sweep algorithm is in Algorithm 1.

Firstly, GD-Sweep evenly divides all PoIs for each mobile sensor to narrow the difference between the length of sweep routes for all mobile sensors (in Line 1-8); Secondly, it combines the PoI groups and mobile sensors from the first step to assign multiple mobile sensors move along the same trajectory to reduce the length of sweep route (in Line 9-23). For first step, GD-Sweep begins with n groups for n PoIs each. Initialy, the TSP length of each group, of course, is 0. Then we calculate the length of TSP cycle for the union groups of any two PoI groups and pick out the union group that has the minimum length of TSP cycle among all pairs (in Line 3-6). After that, we combine these two groups together to produce a new group and replace the original two groups with this new group (in Line 7). We iterate these four steps until there only exists m nonempty groups.

For second step, GD-Sweep assigns one mobile sensor to each nonempty group from first step (in Line 9-13). Then we combine several PoI groups and mobile sensors together. To simplify the explanation, we define P_i, P_j as one pair of the PoI groups. In each iteration, we should calculate $\frac{weight(Tsp(P_i))}{|S_i|}$, $\frac{weight(Tsp(P_j))}{|S_j|}$ and $\frac{weight(Tsp(P_i\cup P_j))}{|S_i\cup S_j|}$ (in Line 16). Then in Line 17, we pick the pair of PoI groups whose $\frac{weight(Tsp(P_i\cup P_j))}{|S_i\cup S_j|}$ is the minimum among all pairs of PoI groups and meanwhile satisfies the inequation, $\max\left(\frac{weight(Tsp(P_i))}{|S_i|}, \frac{weight(Tsp(P_j))}{|S_j|}\right) < \frac{weight(Tsp(P_i\cup P_j))}{|S_i\cup S_j|}$. Then we combine this pair of PoI groups together and replace the original two PoI groups with the union group, so as to mobile sensors in Line 20. We do the iteration again and again until there does not exist such pair of PoI groups. The answer is the maximum $\frac{weight(Tsp(P_i))}{|S_i|}$ of all PoI groups whose $S_i \neq \emptyset$.

From above analysis, GD-Sweep finally give a partition of all PoIs and mobile sensors.

Algorithm 1 GD-Sweep

Input:
The number of mobile sensors,m and PoIs, n;
All PoIs and mobile sensors, $p_1, \cdots, p_n$ and $s_1, \cdots, s_m$;
Output:
The length of mobile sweep route for M^{3}SR, L;

1: Initialize $P_i = p_i$ $(i = 1 \cdots n)$;
2: **for** $it = 1$; $it < n - m$; $it++$ **do**
3: **for** (i, j) that $P_i \neq \varnothing, P_j \neq \varnothing$ and $i \neq j$ **do**
4: $Len = weight(Tsp(P_i \cup P_j))$;
5: Record minimum Len and record its (i, j) to (i', j');
6: **end for**
7: $P_{i'} = P_{i'} \cup P_{j'}$, $P_{j'} = \emptyset$;
8: **end for**
9: **for** $i = 1, j = 1$; $i < n$; $i++$ **do**
10: **if** $P_i \neq \emptyset$ **then**
11: $S_i = s_j, j++$;
12: **end if**
13: **end for**
14: **while** P_i and S_i does not change anymore **do**
15: **for** pair (i, j) that $P_i \neq \varnothing$, $P_j \neq \varnothing$ and $i \neq j$ **do**
16: **if** $\max\left(\frac{weight(Tsp(P_i))}{|S_i|}, \frac{weight(Tsp(P_j))}{|S_j|}\right) < \frac{weight(Tsp(P_i\cup P_j))}{|S_i\cup S_j|}$ **then**
17: Record minimum $\frac{weight(Tsp(P_i\cup P_j)}{|S_i\cup S_j|}$ and its (i, j) to (i', j');
18: **end if**
19: **end for**
20: $P_{i'} = P_{i'} \cup P_{j'}$, $P_{j'} = \emptyset$, $S_{i'} = S_{i'} \cup S_{j'}$, $S_{j'} = \emptyset$;
21: **end while**
22: $L = \max_{S_i \neq \emptyset}\left(\frac{weith(Tsp(P_i))}{|S_i|}\right)$
23: **return** L;

B. BS-Sweep

In this part, we will present BS-Sweep in detail. It is not easy to give an optimal solution for M^{3}SR with polynomial complexity. However we can give an approximation named BS-Sweep with approximation ratio of 6.

Theorem 2. *Given a PoI set P, we can get:*

$$weight(Mst(P)) < weight(Tsp(P)) < 2weight(Mst(P))$$

where $weight(Tsp(P))$ denotes the length of Euclidean TSP cycle for P, $weight(Mst(P))$ denotes sum of edges belong to the minimal spanning tree derived from P [22].

According to Theorem 2, we can find out an approximation named BS-Sweep based on binary search. We briefly present our algorithm as follows:

1) Give an approximate function MSTCheck(L, m, Loc) to check whether MST-M^{3}SR has an answer less than $2L$ or not, the output of which is a boolean variable. If the output is **TRUE**, the function will give the partition of PoIs and mobile sensors too.

2) Use binary search to find the approximation for MST-M^{3}SR and take MSTCheck as the check function.
3) Calculate the length of the TSP cycle for each PoI set and evenly divide the TSP cycle of each PoI set for the mobile sensors to cover this PoI set.

The remaining of this subsection is organize as follows. We will present the MSTCheck and BS-Sweep in detail. Then we will theoretically prove that BS-Sweep is an approximation with approximation ratio of 6.

1) MSTCheck : The input of MSTCheck consists of (1) A bound L about the answer for MST-M^{3}SR; (2) m, the number of mobile sensors; and (3) Loc, the coordinates of PoIs belong to P. The function returns **TRUE** if we can get an answer for MST-M^{3}SR less than $2L$ and also returns a partition of mobile sensors and PoIs. If not, MSTCheck returns **FALSE**. The main idea of MSTCheck is presented as follows:

1) Construct a complete graph G of all PoIs belong to P and the distance between any two PoIs is the Euclidean Distance.
2) Remove the edges of G which are larger than L.
3) Calculate the weight of a Minimal Spanning Tree for each distinct PoI set of G.
4) Check whether $2L$ can be the answer for MST-M^{3}SR with the input of MSTCheck.

The pseudo code of function MSTCheck is presented in Algorithm 2. The MSTCheck function begins by constructing a graph G of the PoIs and the distance between each pair of PoIs is the Euclidean Distance (in Line 1). Then the algorithm removes edges whose weight are bigger than L (in Line 2). The removal of these edges may cause the graph G unconnected. We denote the connected components by $\{G_i\}$ and find $\{G_i\}$ (in Line 3). In Line 4, Minimal Spanning Tree $Tree_i$ is computed for each component G_i. In Line 5, an estimate k_i of the number of mobile sensors to be responsible for G_i is computed and $k_i = \left\lceil \frac{Weight(Tree_i)}{2L} \right\rceil + 1$. In Line 6-7, the algorithm returns **FALSE** if the sum of estimates k_i is bigger than m because there cannot be enough mobile sensors. This means that the algorithm has a proof that it is impossible to have an answer for MST-M^{3}SR smaller than $2L$. If the algorithm returns **TRUE**, the output consists of the partition of PoIs and mobile sensors, which means that we can have an approximation for MST-M^{3}SR.

2) BS-Sweep : Now we present BS-Sweep in detail. BS-Sweep uses binary search to find the minimum L when MSTCheck outputs **TRUE**. It takes the output of MSTCheck as the partition of PoIs and mobile sensors and calculate the length of maximum sweep route for all mobile sensors.

Now we give the whole algorithm BS-Sweep in Algorithm 3. In Line 1, the algorithm initializes the variable Loc of all coordinates of PoIs. Then the algorithm constructs the graph G of the Loc (in Line 2). After that the algorithm finds the answer with binary search (in Line 5-12), the bottom bound of the answer is 0 and the top bound of the answer is $\frac{weight(Tree)}{m}$. The decision of how to adjust answer depends on the output of MSTCheck. If MSTCheck returns **FALSE**, which means that the answer is too small, then the algorithm adjusts Len_{right} to the average of Len_{right} and Len_{left} and otherwise the algorithm adjusts Len_{left} to the average and adds up a small value (in Line 10) until the difference between Len_{right} and Len_{left} is smaller than a pre-defined value. At last, the algorithm calculates the length of TSP cycle for each PoI set $\{G_i\}$. The answer Len is the length of maximum sweep route for all mobile sensors. The partition of PoIs and mobile sensors are given by $\{G_i\}$ and $\{k_i\}$.

Algorithm 2 MSTCheck(L, m, Loc): check whether the answer for *MST-M^{3}SR* is smaller $2L$

Input:
The bound of the answer for *MST-M^{3}SR*, L;
The number of mobile sensors, m;
The coordinates of all PoIs, $Loc_1, Loc_2, \cdots, Loc_n$;
Output:
TRUE: Can find an answer for *MST-M^{3}SR* smaller than $2L$ and give a partition of PoIs and mobile sensors;
FALSE: Cannot find an answer *MST-M^{3}SR* smaller than $2L$;

1: Construct a graph G of the PoIs belong to Loc and the distance is Euclidean Distance;
2: Remove all edges whose weight greater than L from G;
3: Let G_i denote the connected components after deleting the heavy edges;
4: $Tree_i$ = MST of $\{G_i\}$;
5: $k_i = \left\lceil \frac{Weight(Tree_i)}{2L} \right\rceil + 1$;
6: **if** $\sum_i k_i > m$ **then**
7: **return** FALSE;
8: **end if**
9: **return** TRUE, $\{G_i\}$ and $\{k_i\}$;

Algorithm 3 BS-Sweep

Input:
The number of mobile sensors, m;
The coordinates of all PoIs, $Loc_1, Loc_2, \cdots, Loc_n$;
Output:
The answer for the *M^{3}SR* problem, Len;
The partition of PoIs and mobile sensors,$\{G_i\}$, $\{k_i\}$;

1: $Loc = \{Loc_1, Loc_2, \cdots, Loc_n\}$;
2: Construct a graph G of the PoIs belong to Loc;
3: $Tree$ =Minimal Spanning Tree of G;
4: $Len_{right} = 0$, $Len_{left} = \frac{Weight(Tree)}{m}$;
5: **while** $Len_{left} + \epsilon < Len_{right}$ **do**
6: $Mid = \frac{1}{2}(Len_{left} + Len_{right})$;
7: **if** $MSTCheck(Mid, m, Loc)$ **then**
8: $Len_{left} = Mid$;
9: **else**
10: $Len_{right} = Mid + \epsilon$;
11: **end if**
12: **end while**
13: $TSP_i = weight(Tsp(G_i))$; //G_i is output of MSTCheck;
14: $Len = \max_i \{\frac{Weight(TSP_i)}{k_i}\}$;
15: **return** Len, $\{G_i\}$ and $\{k_i\}$;

It is easy to know that Len is an answer of M^{3}SR. In the following part, we will prove that BS-Sweep is an approximation with approximation ratio of 6.

3) Correctness and Approximation: Let L^* denote the optimal answer for MST-M^{3}SR and we assume that $Tree^* = \{Tree_1^*, Tree_2^*, \cdots, Tree_l^*\}$ denote the optimal partition of PoIs for MST-M^{3}SR, $K^* = \{k_1^*, k_2^*, \cdots, k_l^*\}$ denote the

number of mobile sensors to cover each MST tree of $Tree^*$ and l denotes the number of PoI sets for the optimal partition of PoIs. We also assume that $G = \{G_1, G_2, \cdots, G_{l'}\}$ and $K = \{k_1, k_2, \cdots, k_{l'}\}$ are the partitions of PoIs and mobile sensors for a bound L after running $MSTCheck(L, m, Loc)$. Now, we will prove some lemmas and theorems as follows:

Lemma 1. *The edges in the MST of $Tree_i^*$ is no longer than L^*.*

Lemma 1 is obvious because we can remove the edges bigger than L^* and use one less mobile sensor. The answer must be better than before. With Lemma 1, we can imply the following lemma:

Lemma 2. *if $L^* \leq L$, then the PoIs of $Tree_i^*$ belong to the same PoI set G_j.*

Proof: If two PoIs of $Tree_i^*$ in different G_j, then the distance between these two PoIs is larger than L. It produces contradiction with Lemma 1. Therefore Lemma 2 is true. ■

We assume $k_i^{'}$ denotes the sum of k_j^* that PoIs of $Tree_j^*$ belong to G_i. Then we can prove the following theorems:

Theorem 3. *if $L^* \leq L$, then $k_i < k_i^{'}$, for every i.*

Proof: For simplicity, let $Tree_i^1, Tree_i^2, \cdots, Tree_i^{q_i}$ denote the PoI set of $Tree^*$ that belong to G_i. By adding at most $q_i - 1$ edges, we can connect these q_i PoI sets together to get a tree that spans from G_i. According to Lemma 1, the edges among $Tree_i^j$s is smaller than L. Therefore we can get the inenquation

$$\sum_{j=1}^{q_i} weight(Mst(Tree_j^i)) + L * (q_i - 1) \geq weight(Mst(G_i))$$

Since

$$weight(Mst(Tree_j^i)) \leq L^* * k_j^* \leq L * k_j^*$$

and

$$q_i \leq k_i^{'}$$

we get that

$$L * k_i^{'} + L * (k_i^{'} - 1) \geq weight(Mst(G_i))$$

Therefore

$$k_i^{'} \geq \frac{weight(Mst(G_i))}{2L} + 1/2$$

Meanwhile, from the algorithm, we can get

$$k_i = \left\lceil \frac{weight(Mst(G_i))}{2L} \right\rceil + 1$$

Because $k_i, k_i^{'}$ are all integers, we get $k_i < k_i^{'}$. Then the theorem is proved. ■

Theorem 4. *if $MSTCheck$ returns **FALSE**,then $L^* > L$.*

Proof: Obviously, $m = \sum_{i=1}^{l'} k_i^{'}$. If $MSTCheck$ returns **FALSE**, $\sum_{i=1}^{l'} k_i > m$. According to Theorem 3, if $L^* \leq L$, we can know $m = \sum_{i=1}^{l'} k_i^{'} \geq \sum_{i=1}^{l'} k_i$. Then we get a contradiction. Therefore we can know that $L^* > L$. ■

We can find the L, less than which MSTCheck returns **FALSE** and bigger than which MSTCheck returns **TRUE**. Then $L \leq L^* \leq 2L$ and we know the approximation factor of the binary search is 2 for MST-M^{3}SR.

Theorem 5. *The approximation ratio of BS-Sweep is 6 for M^{3}SR on 2D plane.*

Proof: Let P_{MST} and P_{TSP} denote the PoI set that get the optimal answer for M^{3}SR and M^{3}SR respectively.

Let N_{MST} and N_{TSP} denote the number of mobile sensors to cover P_{MST} and P_{TSP} respectively.

Let P_{OUR} denotes the PoI set after running BS-Sweep. N_{OUR} denotes the number of mobile sensors to cover P_{OUR}.

We can conclude from Theorem 2 that

$$\frac{2weight(Mst(P_{MST}))}{N_{MST}} \geq \frac{weight(Tsp(P_{TSP}))}{N_{TSP}} > \frac{weight(Mst(P_{MST}))}{N_{MST}}$$

We can prove it easily with Reduction to Absurdity. Because binary search is an algorithm with an approximation ratio of 2 for MST-M^{3}SR. Therefore we can get

$$\frac{2weight(Mst(P_{MST}))}{N_{MST}} \geq \frac{weight(Mst(P_{OUR}))}{N_{OUR}} \geq \frac{weight(Mst(P_{MST}))}{N_{MST}}$$

and

$$\frac{2weight(Mst(P_{OUR}))}{N_{OUR}} \geq \frac{weight(Tsp(P_{OUR}))}{N_{OUR}} > \frac{weight(Mst(P_{OUR}))}{N_{OUR}}$$

Thus, we can get

$$\frac{4weight(Mst(P_{MST}))}{N_{MST}} \geq \frac{weight(Tsp(P_{OUR}))}{N_{OUR}} \geq \frac{weight(Tsp(P_{TSP}))}{N_{TSP}} > \frac{weight(Mst(P_{MST}))}{N_{MST}}$$

Then we can know that approximate ratio of our algorithm is 4. Because we use an approximation to calculate TSP with approximation ratio of 1.5, the approximation ratio of BS-Sweep is 4×1.5=6. ■

V. Performance Evaluation

In this section, we evaluate and compare the performance of GD-Sweep and BS-Sweep algorithms for M^{3}SR in 2D case on a stand alone C++ simulation platform. We also compare our algorithms with OSweep in [8], MinExpand proposed in [8], and PDBA in [9]. In our simulation, we place different number of PoIs randomly in a square region of 200×200 m^2. The number of mobile sensors is $\frac{1}{5}$ and $\frac{1}{10}$ of the number of PoIs respectively. We run the simulation for 100 times with the number of PoIs between 10 and 200 having a step of 10 and we compute the average results as the answer for each case. Because some problems that previous paper researched are to find the minimum number of mobile sensors to achieve a Global T-Sweep Coverage, we use binary search through changing the variable T for Global T-Sweep Coverage until the output of the previous algorithms equals to the number of mobile sensors we input in our algorithms. Then we use the T we get and the velocity given by the problem to calculate the length of the longest sweep route for M^{3}SR.

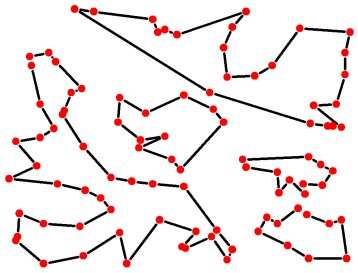
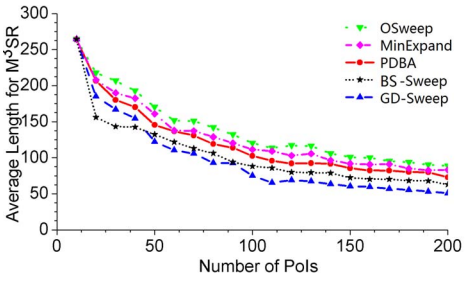
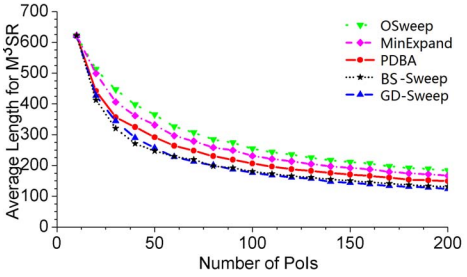


Fig. 3. Example for the partition of PoIs and routes scheduling with 100 PoIs and 10 mobile sensors



(a) Number of mobile sensors is $\frac{1}{5}$ of number of PoIs



(b) Number of mobile sensors is $\frac{1}{10}$ of number of PoIs

Fig. 4. Average length of mobile sweep routes required for M^{3}SR

Fig. 3 shows an example about the partition of PoIs and routes scheduling with 100 PoIs and 10 mobile sensors and the length of mobile sweep routes in Fig. 3 is 185.3. We can abstract from the simulations in Fig. 4 that the average length got by GD-Sweep and BS-Sweep is shorter than others. Thus, our algorithms perform better than other algorithms and GD-Sweep performs best. However the output of BS-Sweep is more stable than GD-Sweep with extreme inputs.

VI. Conclusion

In this paper, we introduce the Minimum Makespan of Mobile Sweep Routes problem (M^{3}SR) and present a novel method to schedule the mobile sensors to achieve a Sweep coverage. We consider whether the mobile sensor should move along the same trajectory. We propose a 6-approximation BS-Sweep and a greedy algorithm GD-Sweep to solve it. The simulations of our algorithms compared to many previous works and validated the efficiency of our designs. Our future work is to add the sink node into the sweep coverage to make a more practical solution.

References

[1] P.-J. Wan and C.-W. Yi, "Coverage by randomly deployed wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2658–2669, 2006.

[2] X. Bai, Z. Yun, D. Xuan, T.-h. Lai, and W. Jia, "Deploying four-connectivity and full-coverage wireless sensor networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, Phoenix, Arizona, USA, 2008, pp. 906–914.

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

[4] A. Chen, S. Kumar, and T. H. Lai, "Designing localized algorithms for barrier coverage," in *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Montreal, Quebec, Canada, 2007, pp. 63–74.

[5] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *ACM International conference on Embedded networked sensor systems (SenSys)*, Los Angeles, California, USA, 2003, pp. 28–39.

[6] W. Cheng, M. Li, K. Liu, Y. Liu, X. Li, and X. Liao, "Sweep coverage with mobile sensors," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, Miami, Florida, USA, 2008, pp. 1–9.

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

[8] 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.

[9] 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.

[10] N. Bisnik, A. A. Abouzeid, and V. Isler, "Stochastic event capture using mobile sensors subject to a quality metric," *IEEE Transactions on Robotics (T-RO)*, vol. 23, no. 4, pp. 676–692, 2007.

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

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

[13] 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.

[14] S. Li, L. Z. Wang Wei, Lin Feng, and Z. Jiliu, "A sweep coverage scheme based on vehicle routing problem," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 4, pp. 2029–2036, 2013.

[15] R. Moazzez-Estanjini and I. C. Paschalidis, "Improved delay-minimized data harvesting with mobile elements in wireless sensor networks," in *IEEE International Symposium on Modeling, Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Princeton, USA, 2011, pp. 49–54.

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

[17] R. Sugihara and R. Gupta, "Optimizing energy-latency trade-off in sensor networks with controlled mobility," in *IEEE International Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, 2009, pp. 2566–2570.

[18] R. Sugihara and K. Gupta, Rajesh, "Optimal speed control of mobile node for data collection in sensor networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 9, no. 1, pp. 127–139, 2010.

[19] S. O. Tiwari and S. kumar Yadav, "Data harvesting with mobile elements in wireless sensor networks," *Journal of Electronics and Communication Engineering (IOSR-JECE)*, vol. 9, no. 1, pp. 104–114, 2014.

[20] Y. Gu, D. Bozdağ, R. W. Brewer, and E. Ekici, "Data harvesting with mobile elements in wireless sensor networks," *Computer Networks*, vol. 50, no. 17, pp. 3449–3465, 2006.

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

[22] D.-Z. Du, K.-I. Ko, and X. Hu, *Design and analysis of approximation algorithms*. Springer Science and Business Media, 2011, vol. 62.