Energy Efficient Algorithms for k-Sink Minimum Movement Target Coverage Problem in Mobile Sensor Network

Xiaofeng Gao, *Member, IEEE*, Zhiyin Chen, Fan Wu, *Member, IEEE*, and Guihai Chen, *Senior Member, IEEE*

***Abstract*— Energy consumption is a fundamental and critical issue in wireless sensor networks. Mobile sensors consume much more energy during the movement than that during the communication or sensing process. Thus how to schedule mobile sensors and minimize their moving distance, while keeping the coverage requirement has great significance to researchers. In this paper, we study the target coverage problem in mobile sensor networks where all the targets need to be covered by sensors continuously. Our goal is to minimize the moving distance of sensors to cover all targets in the surveillance region, which is in Euclidean space. Here initially all the sensors are located at k base stations. Thus, we define this problem as *k-Sink Minimum Movement Target Coverage*. To solve this problem, we propose a polynomial-time approximation scheme, named *Energy Effective Movement Algorithm* (EEMA). We prove that the approximation ratio of EEMA is $1+\varepsilon$ and the time complexity is $n^{O(1/\varepsilon^2)}$. We also propose a distributed version of EEMA (D-EEMA) for large-scale networks where EEMA is not efficient enough in practice. Finally, we conduct experiments to validate the efficiency and effectiveness of EEMA and D-EEMA.**

***Index Terms*— Mobile sensor networks, target coverage, PTAS algorithm.**

I. Introduction

WIRELESS sensor networks have been widely applied in different fields, such as military applications, environmental applications, and industrial applications [1], [2]. Since the battery energy of sensors is usually limited and has a great impact on the lifetime of wireless sensor networks, one of the most fundamental and critical issues in wireless sensor networks is how to develop an energy efficient sensor schedule to satisfy some coverage requirements [3], [4].

Many previous works save energy by scheduling the sensor state. When a sensor is idle and turn to sleep mode. It will be active again if needed. However, with the development of micro-electro-mechanical technologies, mobile sensors become more and more popular. A mobile sensor could detect the surveillance region periodically when moving along a predefined trajectory, which greatly reduces the number of sensors needed and thus becomes an economical method for coverage requirements. Since for each sensor, the energy consumption during the movement is much more higher than that during the communication and sensing process, we need to develop efficient sensor scheduling strategies to save energy and extend network lifetime.

Several previous works studied the movement scheduling problem in mobile sensor networks. Liu *et al.* [5] and He et.al. [6] studied the movement scheduling problem for barrier coverage. Ammari *et al.* [7] proposed a heuristic algorithm for k-coverage. By selecting and placing sensors appropriately, they tried to minimize sensor movement length and thus save energy. Wang *et al.* [8] also studied this problem when k is not fixed. Some other works [9], [10] discussed this problem in hybrid networks, which have both mobile sensors and static sensors. However, few literatures studied the problem for target coverage problem. Zorbas *et al.* [11] tried to prolong the network lifetime under probabilistic coverage model. Mini *et al.* [12] proved that this problem is NP-hard. Most of them only developed heuristic algorithms, without any performance guarantee and theoretical analysis.

In this paper, we try to minimize sensor movement for target coverage problem. Our goal is to minimize the moving distance of sensors to cover all targets in the surveillance region which is in Euclidean space. Here initially all the sensors are located at k base stations and all the targets need to be covered by sensors continuously. Thus we define this problem as *k-Sink Minimum Movement Target Coverage* (k-MMTC). We prove that k-MMTC is an NP-hard problem and propose a *polynomial-time approximation scheme* (PTAS), named EEMA, to solve k-MMTC. Its approximation ratio is $1+\varepsilon$.

EEMA can be divided into two phases. In the first phase, we partition the surveillance region into several subareas according to the detection cycle of targets. As shown in Figure 1, t_i is a target and r is the sensing range of mobile sensors. The detection cycle of t_i means if a sensor locates inside this cycle, then t_i can be successfully detected. The detection cycles for each target divide the surveillance region into many subareas, one of which is shown as the shadow area in Figure 1. In the second phase, we calculate the weight of each subarea. The weight represents the minimum distance when a sensor moves from a station to the subarea. Then we

Manuscript received January 6, 2017; revised July 11, 2017; accepted August 30, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P.-J. Wan. Date of publication October 24, 2017; date of current version December 15, 2017. This work was supported in part by the China 973 Project under Grant 2014CB316200, in part by the Program of International S&T Cooperation under Grant 2016YFE0100300, and in part by CCF-Tencent Open Research Fund, the National Natural Science Foundation of China under Grant 61472252, Grant 61133006, Grant 61272443, and Grant 61422208. *(Corresponding author: Xiaofeng Gao.)*

The authors are with the Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: gao-xf@cs.sjtu.edu.cn; chenzhiyin@sjtu.edu.cn; fwu@cs.sjtu.edu.cn; gchen@cs.sjtu.edu.cn).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors, including Appendices A to D.

Digital Object Identifier 10.1109/TNET.2017.2756925

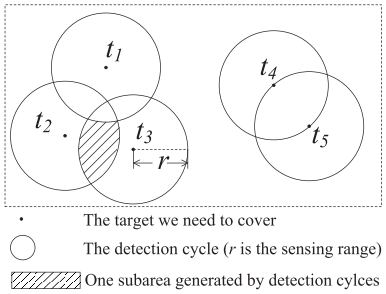

Fig. 1. Illustration of subareas.

just need to select enough subareas with minimum weight to cover the target set. In this phase, we also consider the obstacles in the surveillance region.

We also propose a distributed version of EEMA (D-EEMA) for large-scale networks and conduct experiments to validate the efficiency and effectiveness of EEMA and D-EEMA.

To summarize, our contributions in this paper are as follows:

1) We define the problem *k-sink minimum movement target coverage* (k-MMTC) and propose a novel polynomial-time approximation scheme EEMA.
2) We also consider obstacles in the surveillance region. EEMA can solve k-MMTC problem no matter what shape, or size the obstacle is.
3) We analyze the performance of EEMA and prove its approximation ratio of $1+\varepsilon$. We also prove that the time complexity of EEMA is $n^{O(1/\varepsilon^2)}$ where n is the number of targets and ε is a predefined parameter.
4) We propose a distributed version of EEMA (D-EEMA) for large scale network.
5) We compare EEMA with the optimal solution and the algorithm in [12] to validate the effectiveness and efficiency of EEMA and D-EEMA.

In all, we are the first to design PTAS for k-sink minimum movement target coverage in mobile sensor networks.

The rest of this paper is organized as follows: Section II discusses the related works. Section III describes the preliminary, basic model, problem assumption, problem statement and some definitions. In Section IV, we present the detailed design of EEMA. Furthermore we demonstrate the complexity and approximation ratio of EEMA. In Section VI, we compare EEMA with other solutions. Section VII summarizes our work and states the future research direction. In additional, we give some subtle related proofs in Appendix.[1]

II. Related Works

In majority scenarios, to achieve a best coverage in wireless sensor network is NP-hard. Thus we can hardly find the optimal solution for coverage problem in polynomial time. Hence, previous works usually designed heuristic algorithms or approximations to solve coverage problem.

Compared with approximation algorithms, heuristic algorithms are more popular in previous works. References [13], [14] adopted simulated annealing algorithms. Reference [13] applied simulated annealing to design periodical mobile coverage schedules. The scheduling objective was to distribute the coverage time of a mobile sensor in proportion according to the importance levels of targets when mobile sensors move around the targets. Reference [14] also utilized simulated annealing for a coverage restoration scheme, which could find the best neighboring sensor to replace the failed sensor and extend the network lifetime. References [15]–[19] adopted genetic algorithms. For area coverage maximization and energy conservation, [17] determined the speed and direction of each mobile node based on genetic algorithm. Reference [19] proposed a new network coverage and optimization control strategy to solve 3D coverage problem in wireless sensor networks. With mobile sensors, [18] proposed a coverage holes healing algorithm. Reference [15] designed a sensors deployment strategy, which could meet desired coverage requirement and maintain connectivity. Reference [16] described two exact approaches to maximize the network lifetime under a coverage ratio constraint and to maximize the coverage ratio under a lifetime constraint.

However, unlike approximation algorithms, heuristic algorithms cannot give any performance guarantee. Sometimes, heuristic algorithms may have a bad performance and determining the parameters of heuristic algorithms is usually difficult. Some papers designed approximation algorithms for coverage problem. For target coverage problem, the best approximation ratio is $4+\xi$ [20]. Reference [21] proposed an approximation algorithm for area coverage problem. Their algorithm could maximize the spatial-temporal coverage by scheduling the sensors activity, with approximation ratio 0.5. Reference [22] studied k-coverage problem and proposed a 3-approximation algorithm. Reference [23] designed an $O(\rho)$-approximation algorithm, where ρ was the density of sensors. Some works designed approximation algorithms for variations of coverage problems. References [24], [25] studied sweep coverage problem and proposed a 2-approximation. Reference [26] studied camera coverage problem and gave a 2-approximation algorithm to minimize the number of cameras and maintain the connectivity of network. Though approximation algorithms have performance guarantee, it is usually difficult to design constant-approximations and analyze the approximation ratios. Thus some coverage problems still do not have good approximation algorithms.

Nowadays, with the popularity of mobile sensors, coverage problems with mobile sensors become more and more important. Reference [27] gave a survey of movement strategies, including healing coverage hole, optimizing area coverage, and improving event coverage. Reference [28] studied a similar problem in a specific kind of wireless sensor networks which was referred to wireless sensor and actuator networks. They proposed three efficient heuristics to plan the path of the actuators and reduce the workload of the actuators, including the traveling cost and on-spot cost. References [29], [30] focused on minimizing movement of mobile sensors for target coverage, which was called MMTC. They reduced from the set cover problem to a special case of MMTC, and proved its NP-completeness. However, the best approximation ratio for set cover problem is $\ln n$. Above all, we also find that the proof in [29] and [30] are wrong.

[1]can be downloaded online from http://ieeexplore.ieee.org

III. Preliminary and Problem Statement

In this section, we give the network model and formulate the problem. Then we prove the NP-hardness of our problem, according to which we can hardly develop a polynomial time optimal algorithm for the problem.

A. Network Model

We study the problem in homogeneous network and the surveillance region is in Euclidean space. We model the network as $G = (T, P, S, O, r)$.

T represents the target set that we need to cover continuously. Assume that there are n targets $T = \{t_1, t_2, \cdots, t_n\}$ in the surveillance region. The surveillance region is flat and has some obstacles against sensor movement. Each target is static with a known location.

P represents the station set which can provide mobile sensors. Assume that there are k stations $P = \{p_1, p_2, \cdots, p_k\}$ around the surveillance region. All mobile sensors must start from a sink station in P. At the beginning, enough mobile sensors are static at the sink stations. From each sink station, we can send arbitrary number of mobile sensors to cover the targets in T.

S represents the mobile sensor set. Assume we use m sensors, $S = \{s_1, s_2, \cdots, s_m\}$, to cover all targets in T. All sensors can move continuously in any direction and have the same sensing range.

O represents the obstacle set. Assume that the number of obstacles is q, $O = \{o_1, o_2, \cdots, o_q\}$, and we have known the shape and position of all obstacles. Mobile sensors cannot pass through the obstacles directly. What is more, the sensors cannot stay on the obstacles.

r represents the sensing range of mobile sensors in S. Disk model is adopted for coverage. Namely, the target t_i is covered by the sensor s_j, if the distance between t_i and s_j is less than r.

B. Problem Definition

As now well known, limited energy in sensors is a fundamental issue in wireless network and the energy consumption for sensor movement is much more than that for sensing and communication. The typical energy consumption is 50mW/m for movement and 1mW/m for transmitting a packet [31]. Thus we define the problem to minimize the total moving distance. Before we propose the problem, we give some definitions.

Definition 1 (Moving Distance): For each pair of stations and mobile sensors, (p_i, t_j), a weight denoted as $w(p_i, t_j)$ is given. The value of $w(p_i, t_j)$ represents the cost when a sensor s_k moves from the station p_i to somewhere near the target t_j bypassing the obstacles, such that s_k can cover t_j. The cost is defined as the moving distance of s_k, denoted as $d(s_k)$, and $d(s_k) = w(p_i, t_j)$.

In order to minimize the sensor movement and reduce the energy consumption, we define *k-Sink Minimum Movement Target Coverage* (k-MMTC) problem as follows:

Definition 2 (k-Sink Minimum Movement Target Coverage Problem): We have k sink stations to send mobile sensors and to cover all targets in T. k-MMTC is to schedule the sensor movement trajectories and minimize the sum of moving distance, denoted as $d_{sum} = d(s_1) + d(s_2) + \cdots + d(s_m)$.

The previous work in [29] and [30] have proved that the decision version of 1-MMTC problem can be reduced to set cover problem which is NP-complete. However, the reduction direction is not right in [29] and [30]. When we prove a problem is NP-hard, we need to reduce from a known NP-complete problem to this problem. In Appendix A, we give a new proof to prove the computational complexity of k-MMTC.

Theorem 1: The decision version of the k-MMTC problem is NP-complete.

Proof: Please refer to Appendix A. □

Next, we design a PTAS for the k-MMTC problem.

IV. Energy Effective Movement Algorithm

In this section, we proposed a PTAS, named as Energy Effective Movement Algorithm (EEMA) for k-MMTC.

A. Overview

We divide EEMA into two phases. In the first phase, we propose a novel method to divide the surveillance region into some subareas according to the locations of targets. The sensors in the same subarea can cover the same target set. In the second phase, we schedule the mobile sensors and move the sensors to cover all targets. Finally, we also analyze the time complexity and the approximation ratio of EEMA.

B. Phase One: Region Partition

Before we introduce the detail of EEMA in the first phase, we give some definitions.

Definition 3 (Detection-Cycle): Define $\odot t_i$ as the detection cycle of target t_i, with t_i as its center and sensing range r as its radius.

Apparently, a sensor s_j can detect t_i as long as s_j is in $\odot t_i$.

Definition 4 (Key-Point): When two targets, t_i and t_j, are close enough, $\odot t_i$ and $\odot t_j$ may intersect. The intersection points on $\odot t_i$ are called as the key-points of t_i and denoted as $KP(t_i)$, where $KP(t_i) = \{kp_1(t_i), kp_2(t_i), \cdots\}$.

For a key-point $kp_j(t_i)$, we can record the location of $kp_j(t_i)$ and the two targets which intersect and generate $kp_j(t_i)$. We find that the $KP(t_i)$ divide $\odot t_i$ into some arcs, $\{\overset{\frown}{kp_1(t_i)kp_2(t_i)}, \overset{\frown}{kp_2(t_i)kp_3(t_i)}, \cdots\}$. Those arcs divide the surveillance region and generate some subarea. We define the arcs as follows:

Definition 5 (Curved-Boundary): We can sort $KP(t_i)$ in clockwise order, and get a sequence of key-points $\{kp_1(t_i) \rightarrow kp_2(t_i) \rightarrow \cdots \rightarrow kp_1(t_i)\}$. The neighboring key-points can generate an arc $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$, which is defined as curved-boundary.

Definition 6 (Covered-Set of Curved-Boundary): The sensors on the same curved-boundary, $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$, can cover the same target set. We define this target set as the covered-set of the curved-boundary $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$, denoted as $T(t_i, j)$.

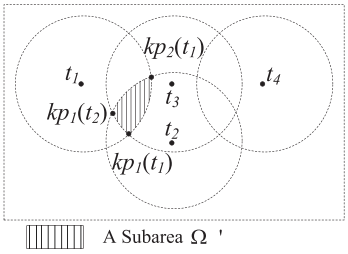

Fig. 2. Illustration of Definitions.

Obviously, $t_i \in T(t_i, j)$. Besides, the distance between the sensors on $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$ and t_i is equal to r. For the other targets which are in $T(t_i, j) \setminus t_i$, the distance must be less than r. Thus we classify the targets in $T(t_i, j)$ into two categories: t_i which is marked as grey target for $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$, and $T(t_i, j) \setminus t_i$ which are marked as black targets for $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$.

Sometimes, $T(t_i, j) \setminus t_i$ may be $\emptyset$. Thus we classify $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$ into two types: grey curved-boundary when $T(t_i, j) \setminus t_i = \emptyset$, black curved-boundary when $T(t_i, j) \setminus t_i \neq \emptyset$. A sequence of closed curved-boundary generate a subarea. We give the definition of covered-set.

Definition 7 (Covered-Set of Subarea): We can get some curved-boundaries from target set T. The curved-boundaries generate some subareas. Each subarea Ω' represents a target set which is named as covered-set. The covered-set is denoted by $T(\Omega')$. A sensor in Ω' can cover all targets in $T(\Omega')$.

Next, We give an instance to further illustrate the definitions above. As shown in Figure 2, there are four targets in the surveillance region. We draw the detection-cycles of all targets, and divide the surveillance region into 11 disjoint subareas. The intersection point of $\odot t_1$ and $\odot t_3$, $kp_1(t_1)$ is the first key-point of t_1. Of course, we can also denote $kp_1(t_1)$ as $kp_1(t_3)$. A subarea Ω' is enclosed by three curved-boundaries, $\overset{\frown}{kp_1(t_1)kp_2(t_1)}$, $\overset{\frown}{kp_1(t_1)kp_1(t_2)}$ and $\overset{\frown}{kp_1(t_2)kp_2(t_1)}$. We can find that $T(\Omega') = \{t_1, t_2, t_3\}$. $T(t_1, 1) = \{t_1, t_2, t_3\}$ and t_1 is a grey target in $T(t_1, 1)$.

In fact, we can easily prove the lemmas below.

Lemma 1: If $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$ is a curved-boundary of a subarea Ω', then $T(\Omega') \subseteq T(t_i, j)$ and $T(t_i, j) \setminus T(\Omega') = \{t_i\}$ or $\emptyset$.

Lemma 2: If $T(t_i, j)$ only contains grey target t_i, then $T(\Omega') = \{t_i\}$ or $\emptyset$.

Lemma 3: If $\overset{\frown}{kp_q(t_p)kp_{q+1}(t_p)}$ is also a curved-boundary of Ω', then $T(t_p, q) \setminus T(t_i, j) = \{t_p\}$ or $\emptyset$.

Lemma 4: $\overset{\frown}{kp_{j+1}(t_i)kp_{j+2}(t_i)}$ and $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$ cannot be the curved-boundaries of Ω' at the same time.

Please refer to Appendix B for the proof of these lemmas.

Based on the lemmas above, we can find all the subareas by search the curved-boundaries. We select a curved-boundary $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$, and then search the neighboring curved-boundary $\overset{\frown}{kp_q(t_p)kp_{q+1}(t_p)}$. Obviously, if both $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$ and $\overset{\frown}{kp_q(t_p)kp_{q+1}(t_p)}$ are the curved-boundaries of a subarea Ω', we can infer that $i \neq p$, $T(\Omega') = T(t_i, j+1) \cap T(t_p, q)$ and $T(t_p, q) \setminus T(\Omega') \subset \{t_p\}$. This the main idea to calculate the subareas. Next, we give the detail to calculate the subareas.

Given the set of all targets, firstly we can easily get all key-points and all curved-boundaries. We need to sort these key-points and curved-boundaries in clockwise order.

We divide the sorting process into three phases. In the first phase, we classify the key-points of the target t_i into two types according to the horizonal coordinates of key-points. If the horizonal coordinate of key-point is less than the horizonal coordinate of t_i, the key-point belongs to the first type. Otherwise, the key-point belongs to the second type. In the second phase, we sort the two types of key-points respectively. For the first type of key-points, we sort them in ascending order by the vertical coordinates of key-points. For the second type of key-points, we sort them in descending order by the vertical coordinates of key-points. In the third phase, we combine these two sequences of key-points directly.

The sorting algorithm is shown in Algorithm 1.

Algorithm 1 Sorting Algorithm

Input: The set of targets, T

Output: The ordered sequence of key-points $\{kp_1(t_i), kp_2(t_i), \cdots\}$ of every target

1: Calculate the key-points set of t_i, $\forall t_i \in T$
2: $\forall t_i \in T$, Classify the key-points of t_i into two types according to the horizonal coordinates
3: $\forall t_i \in T$, Sort the two types of key-points respectively
4: $\forall t_i \in T$, Combine these two ordered sequence of key-points directly
5: **return** The ordered sequence of key-points

After we get all key-points and all curved-boundaries, we can construct a new graph $G = (V, E)$ to calculate the subareas generated by all curved-boundaries. To construct G, we propose Graph Conversion(GC) algorithm which is shown in Algorithm 2. In G, each node $v(t_i, j)$ represents a curved-boundaries $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$. There is an edge between $v(t_i, j)$ and $v(t_p, q)$, if $p \neq i$, $T(t_p, q) \cap T(t_i, j) \neq \emptyset$ and $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$, $\overset{\frown}{kp_q(t_p)kp_{q+1}(t_p)}$ are connected. We also need to calculate $T(t_i, j)$. In this paper, we select the mid-point of the curved-boundary and calculate the distance from the mid-point to the targets. If the distance is no more than r, then the target is in $T(t_i, j)$. Then a subarea corresponds to a cycle L in G which satisfies a specific condition as follows: if $v(t_i, j)$ is in L, then $\forall v(t_p, q)$ in L, $T(t_i, j) \setminus t_i \subseteq T(t_p, q)$.

To find this kind of cycle, we propose Variant Depth First Search (VDFS) algorithm. When we need to find all subareas involved with the node $v(t_i, j)$, we start from $v(t_i, j)$ and search the graphs until we get a cycle corresponding to Ω'. At the first step of search, we move from $v(t_i, j)$ to $v(t_{p1}, q_1)$ and check weather $t_i \in T(t_{p1}, q_1)$. If $t_i \notin T(t_{p1}, q_1)$, $T(\Omega') = T(t_i, j) \setminus t_i$. Otherwise, $T(\Omega') = T(t_i, j)$. Then we move forward to the next node $v(t_{p2}, q_2)$, only if $T(\Omega') \subseteq T(t_{p2}, q_2)$ and $T(t_{p2}, q_2) \setminus t_{p2} \subseteq T(\Omega')$. When we return back to $v(t_i, j)$, we get a cycle. Obviously, a curved boundary is involved with

Algorithm 2 Graph Conversion (GC) Algorithm

Input: The set of targets, T
Output: A new graph, $G = (V, E)$
1: Apply the sorting algorithm and get the ordered key-points set of t_i, $\forall t_i \in T$
2: Calculate the covered set of all curved-boundaries
3: Generate $G = (V, E)$. Each node in G represents a curved-boundaries. Two nodes have an edge if the corresponding curved-boundaries are connected and have different centers
4: **return** $G = (V, E)$

two subareas. Thus we can only find two cycles when we start from $v(t_i, j)$ and the two cycles have no public edges. The detail of VDFS is shown in Algorithm 3 and Algorithm 4.

Algorithm 3 Variant Depth First Search (VDFS) Algorithm

Input: $G = (V, E)$
Output: The paths involved with $v(t_i, j)$
1: **for** all $v \in V$ **do**
2: visited[v]=$false$
3: visited[$v(t_i, j)$]=$true$
4: **if** $T(t_i, j) \setminus t_i \neq \emptyset$ and the $path$ corresponding to $T(t_i, j) \setminus t_i$ is not found **then**
5: EXPLORE($G, v(t_i, j), T(t_i, j) \setminus t_i$)
6: **if** We have not find $path$ corresponding to $T(t_i, j)$ **then**
7: EXPLORE($G, v(t_i, j), T(t_i, j)$)
8: **return**

Algorithm 4 EXPLORE($G,\ v(t_i, j),\ T'$)

Input: $G = (V, E)$, $v(t_i, j)$, T'
Output: The paths involved with $v(t_i, j)$
1: **if** T' is explored **then**
2: **return**
3: $path$.add($v(t_i, j)$)
4: **if** $T' \subseteq T(t_p, q)$ and $T(t_p, q) \setminus t_p \subseteq T'$ and $(v(t_i, j), v(t_p, q)) \in E$ and visited[$v(t_p, q)$] = false **then**
5: visited[$v(t_p, q)$] = true
6: EXPLORE($G, t(t_p, q), T'$)
7: **if** No such $v(t_p, q)$ exists **then**
8: Output $path$
9: $path$=null
10: **return**
11: **return**

To further illustrate GC and VDFS, we give a simple example as shown in Figure 3. There are two targets and four curved-boundaries in this example. Thus we can generate a new graph $G = (V, E)$ and $|V| = 4$. There is no edge between $v(t_1, 1)$ and $v(t_1, 2)$, since they have the same center t_1. There is no edge between $v(t_1, 1)$ and $v(t_2, 2)$, since $T(t_1, 1) \cap T(t_2, 2) = \emptyset$.

Then we try to find all three subareas. we can search from $v(t_1, 1)$. Obviously, $T(t_1, 1) = t_1$, then $T' = t_1$ in algorithm 4. We add $v(t_1, 1)$ and $v(t_2, 1)$ to the path in turn.

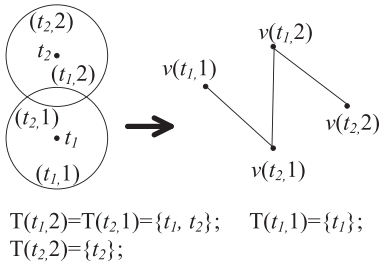

Fig. 3. Illustration for GC and VDFS.

Since $T(t_1, 2) \setminus t_1 \nsubseteq T'$, $v(t_1, 2)$ cannot add to the path. Finally, we get a cycle $v(t_1, 1) \to v(t_2, 1) \to v(t_1, 1)$ and the cycle represents the subarea Ω_1 such that $T(\Omega_1) = t_1$. Next, we search from $v(t_2, 1)$. Since $T(t_2, 1) \setminus t_2 = T(\Omega_1) = t_1$ and we have found Ω_1, we only find the subarea Ω_2 such that $T(\Omega_2) = T(t_2, 1) = t_1, t_2$. We add $v(t_2, 1)$ and $v(t_1, 2)$ to the path, and get Ω_2. In this way, we can easily find all three subareas Ω_1, Ω_2, Ω_3. $T(\Omega_1) = t_1$, $T(\Omega_2) = T(t_2, 1) = t_1, t_2$, $T(\Omega_3) = T(t_2, 1) = t_2$.

Theorem 2: The time complexity of the first phase is $O(n^2)$.

Proof: We have n targets in total. Each pair of targets have 2 key-points at most. Thus we have $2(n-1)$ key-points at most for each target, and we have $n(n-1)$ key-points at most in total. The time complexity of GC is $O(n^2)$. There are no more than $2n(n-1)$ curved-boundaries. Thus in the new graph that we construct, the number of vertices is less that $2n(n-1)$. Obviously, each vertex averagely has no more than 4 edges and the number of edges is less than $4n(n-1)$ in total. When we search all subareas, we visit a vertex at most twice, since a curved-boundaries is involved with two sub areas at most. Thus the time complexity of VDFS is also $O(n^2)$. Therefore, the theorem is proved. □

C. Phase Two: Trajectory Schedule

After all the preprocessing in the first phase, we can schedule the mobile sensors in the second phase. In the scheduling, we need select stations where the mobile sensors start and the destinations to cover the targets. Obviously, we just need select subareas generated in the first phase as the destinations and calculate the minimized sensor moving distance.

Firstly, we study how to calculate the minimized moving distance when a mobile sensor moves from the station p_i to the subarea Ω_j. Apparently, to minimize the moving distance, the mobile sensor must stop at the curved-boundaries of Ω_j. Thus we just need to calculate the distance between the station and all curved-boundaries of Ω_j respectively. Then the minimum distance is the sensor moving distance. When no obstacles exist in the surveillance region, we calculate the distance between p_i and a curved-boundaries according to Theorem 3.

Theorem 3: We have a curved-boundary $\overset{\frown}{kp_j(t_i)kp_{j+1}(t_i)}$ and a station p_i. If the segment p_it_i intersects with the curved-boundary and the intersection point is O, the minimum distance is $|p_iO|$; otherwise, the minimum distance is $|kp_j(t_i)p_i|$ or $|kp_{j+1}(t_i)p_i|$.

Proof: Please refer to Appendix C. □

Based on the Theorem 3, we further propose a method to calculate the distance between p_i and a curved-boundaries when some obstacles exist in the surveillance region. Assume that we have the vertices of the obstacles, and the the obstacles are the convex hulls generated by the vertices. The vertices of obstacle o_i is $\{vo_{i1}, vo_{i2}, \cdots\}$. Firstly, we calculate the shortest path without considering the obstacles and then judge whether the shortest path pass any obstacles. If any segments that connecting two vertices of an obstacle o_i intersect with the shortest path, then the path pass the obstacles and we update the shortest path bypassing the obstacles. Otherwise, the path does not pass any obstacles and the length of path is the minimum moving distance. We update the shortest path bypassing the obstacles according to Theorem 4.

Theorem 4: When the shortest path of a mobile sensor intersects with the boundaries of an obstacle, the first intersection point and the last intersection point must be the vertices of the obstacle boundaries.

Proof: Please refer to Appendix D. □

Based on Theorem 4, we find that we can divide the shortest path bypassing the obstacle into three parts. The first part is from the station to a vertex of the obstacle. The second part is from the vertex in the first part to another vertex of the obstacle. The third part is from the vertex in the second part to a point of the curved-boundary. Both the first part and the third part are segments. The second part is composed of some boundaries of the obstacles which connect the two vertices. Thus we can apply Algorithm 5 to calculate the minimum distance with obstacles.

The detail of the algorithm for calculating the minimum distance is shown in Algorithm 5.

According to the discussion above, we can easily calculate the minimum distance between subareas and stations. When we select a subarea, we just need to ask the nearest station send a mobile sensor. Thus each subarea σ_i has a covered target set and a weight $w(\sigma_i)$ which represents the minimum distance.

Next, we discuss how to select subareas with minimum total weight. The main idea of the selection is partition and shifting strategy [33].

In advance, we give a simple algorithm, called Partition algorithm. Assume that Q is a square which contains exactly all the targets in the surveillance region. Firstly, we divide the square Q into a grid of squares, named as cells. The size of each cell is equal to $2mr \times 2mr$, where m is a constant and $m \in N^+$. Then we can use brute-force search and find the optimal solution for each cell. Finally, we combine the solutions of all cells and get a solution of the original problem. The detail is shown in Algorithm 6.

Theorem 5: The time complexity of Algorithm 6 is $n^{O(m^2)}$.

Proof: We analyze the time complexity for each cell e firstly.

Assume that the number of targets in e is n_e. Then the number of the curved-boundaries is less than n_e^2, since each target correspond to n_e curved-boundaries at most. Each subarea has at least two curved-boundaries and each curved-boundary is shared by only two neighboring subareas. Thus the number of subareas is less than n_e^2.

Algorithm 5 Distance Calculation Algorithm

Input: A subarea σ_i, a station p_j and a obstacle set O
Output: The minimum distance d between p_j and σ_i bypassing all obstacles

```
Calculate the minimum distance between p_j and σ_i without considering obstacles O
if the shortest path passes through some obstacles then
  for each vertex of all these obstacles do
    if the segment connecting the station and the vertex passes through any obstacles then
      Define the distance between the station and the vertex as +∞
    else
      Calculate the distance between the station and the vertex of the obstacles
    if the shortest path from the vertex to the curved-boundary passes through any obstacles then
      Define the distance between the station and the vertex as +∞
    else
      Calculate the distance between the curved-boundary and the vertex of the obstacles
  For any vertices v_i, v_j, define the distance between two vertices v_i, v_j as |v_i v_j| if v_i v_j is a boundary of obstacle. Otherwise, define the distance as +∞
  Update the minimum distance d by Dijkstra's Algorithm
return The minimum distance d
```

Algorithm 6 Partition Algorithm

Input: T, A grid of squares that contain all targets, Q
Output: The subareas that we select to cover T

```
Divide Q into cells which denoted as cell(Q), and the size of each cell is 2mr × 2mr
for each e ∈ cell(Q) do
  Select the subareas which can cover all targets in e, such that the sum of all selected subareas weight is minimum
return All the selected subareas
```

Note that when we select a subarea and send a mobile sensor to the subarea, the sensor can cover a $\sqrt{2}r \times \sqrt{2}r$ square. The size of e is $2mr \times 2mr$. We need to send at most $\left\lceil \sqrt{2}m \right\rceil^2$ sensors. Each sensor select one subarea as the destination.

In conclusion, the upper bound for the number of subareas is less than n_e^2 and the upper bound for the number of mobile sensors is less than $\left\lceil \sqrt{2}m \right\rceil^2$.

We apply brute-force algorithm to find the optimal solution for e and each sensor select only one subarea as the destination. Thus the number of possible solutions for e is at most $n_e^{2\left\lceil \sqrt{2}m \right\rceil^2}$.

According to the discussion above, we can infer that the time complexity of Partition algorithm is as follows:

$$\sum_{e \in cell(Q)} n_e^{O(m^2)} \leq \Big(\sum_{e \in cell(Q)} n_e \Big)^{O(m^2)} = n^{O(m^2)}$$

□

Theorem 6: The approximation ratio of Algorithm 6 is 4.

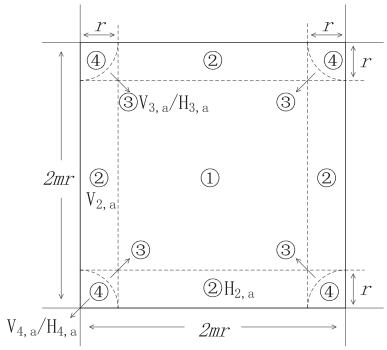

Fig. 4. Illustration for the intersection.

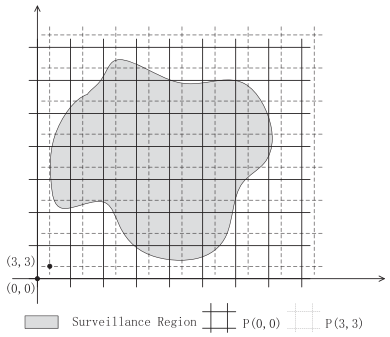

Fig. 5. Illustration for the shifting.

Proof: Assume the optimal solution of selection is S^*, and the feasible solution we get from Partition algorithm is S. $S^*(e)$ is the set of subareas which are contained by S^* and $\sigma_i \in S^*(e)$ if and only if some targets in the covered-set of σ_i are distributed in the cell e. $S(e)$ is the set of subareas which are contained by S and intersect with the cell e.

Obviously, $S^*(e)$ is a feasible solution for e, and $S(e)$ is the optimal solution for e. Thus $\sum_{\sigma_i \in S^*(e)} w(\sigma_i) \geq \sum_{\sigma_i \in S(e)} w(\sigma_i)$.

Assume that S_k^* is the set of subareas which are contained by S^*, and $\sigma_i \in S_k^*$ if and only if the covered-set of σ_i is distributed in k cells. Apparently, $1 \leq k \leq 4$. Thus we can infer that:

$$\begin{aligned}&\sum_{\sigma_i \in S} w(\sigma_i)\\ &= \sum_{e \in cell(Q)} \sum_{\sigma_i \in S(e)} w(\sigma_i) \leq \sum_{e \in cell(Q)} \sum_{\sigma_i \in S^*(e)} w(\sigma_i)\\ &= \sum_{k=1}^{4} \sum_{\sigma_i \in S_k^*} k \cdot w(\sigma_i) \leq \sum_{k=1}^{4} \sum_{\sigma_i \in S_k^*} 4w(\sigma_i) = 4 \sum_{\sigma_i \in S^*} w(\sigma_i)\end{aligned}$$

Then, Theorem 6 is proved. □

In the proof of Theorem 6, we can find that the gap between the optimal solution and our solution is generated by the subareas which intersect more than one cell. Indeed, when the stations and the targets are uniformly distributed in the surveillance region, the expected approximation ratio of Partition Algorithm is much better than 4. As shown in Figure 4, when the sensors stop in the area labeled as k $(1 \leq k \leq 4)$, the covered-set of subareas is distributed in k cells.

Assume that $S_k^*(e)$ is the set of subareas which are contained by $S^*(e)$, and $\sigma_i \in S_k^*(e)$ if and only if the covered-set of σ_i is distributed in k cells. According to Figure 4, we can infer that:

$$\begin{aligned}|S_1^*(e)| &= (m-1)^2/m^2 |S^*(e)|;\\ |S_2^*(e)| &= 2(m-1)/m^2 |S^*(e)|;\\ |S_3^*(e)| &= (4-\pi)/4m^2 |S^*(e)|;\\ |S_4^*(e)| &= \pi/4m^2 |S^*(e)|;\end{aligned}$$

Then we can conclude that $\sum_{e \in cell(Q)} |S^*(e)| \leq \sum_{e \in cell(Q)} \sum_{k=1}^{4} k|S_k^*(e)| = (1 + \frac{2}{m} + \frac{\pi}{4m^2})|S^*|$. When the stations and the targets are uniformly distributed in the surveillance region and we denote the average weight of subareas as $\bar{w}$, we can infer that:

$$\begin{aligned}&\sum_{\sigma_i \in S} w(\sigma_i)\\ &= \sum_{e \in cell(Q)} \sum_{\sigma_i \in S(e)} w(\sigma_i) \leq \sum_{e \in cell(Q)} |S^*(e)| \bar{w}\\ &= (1 + \frac{2}{m} + \frac{\pi}{4m^2})|S^*|\bar{w} = (1 + \frac{2}{m} + \frac{\pi}{4m^2}) \sum_{\sigma_i \in S^*} w(\sigma_i)\end{aligned}$$

Thus we can conclude that the expected approximation ratio of Partition Algorithm is $1 + \frac{2}{m} + \frac{\pi}{4m^2}$ when the stations and the targets are evenly distributed. However, as mentioned in Section III, the distribution of the stations and the targets can be not uniform.

We can apply shifting method for derandomization and improve the algorithm performance no matter what the distribution of the targets and the stations are.

As shown in Figure 5, we define the partition according to the lower-left corner. $P(3,3)$ denotes the partition which has a lower-left corner at $(3,3)$. $P(0,0)$ denotes the partition which has a lower-left corner at $(0,0)$. Obviously, the partition $P(0,0)$ and the partition $P(2mr, 2mr)$ is the same partition. The main idea of the shifting algorithm is to shift the partition from $P(0,0)$ to $P(2r(m-1), 2r(m-1))$, and apply Algorithm 6 to find a solution of k-MMTC. Finally, we chose the optimal solution and the algorithm terminates. The detail of shifting algorithm is shown in Algorithm 7.

Algorithm 7 Shifting Algorithm

Input: T

Output: The subareas that we select to cover T

1: **for** $(0 \leq i < m)$ **do**
2: Apply Algorithm 6 in the partition $P(2ri, 2ri)$ and get a set of selected subareas
3: Select the set with minimum total weight
4: **return** The selected set

Theorem 7: The time complexity of Algorithm 7 is $mn^{O(m^2)}$.

Proof: We invoke Algorithm 6 for m times and the time complexity of Algorithm 6 is $n^{O(m^2)}$. Correspondingly, the time complexity of Algorithm 7 is $mn^{O(m^2)}$. □

Theorem 8: The approximation ratio of Algorithm 7 is $1 + 3/m$.

Proof: We assume that Sen^* denotes the location of sensors in the optimal solution. Then each sensor is assigned to a selected subarea and we can infer that $|Sen^*| = |S^*|$. S_a denotes the solution of Algorithm 6 in the partition $P(2ra, 2ra)$. As shown in Figure 4, $H_{k,a}$ denotes a subset of S^*. $SenH_{k,a}$ denotes the sensor set assigned to $H_{k,a}$. $\forall s \in SenH_{k,a}$, the detection cycle of s intersects the horizontal line of $P(2ra, 2ra)$, and intersects with k cells. Similarly, we define $V_{k,a}$ and $SenV_{k,a}$. $\forall s \in SenV_{k,a}$, the detection cycle of s intersects the vertical line of $P(2ra, 2ra)$, and intersects with k cells. Then we can infer that: $H_{3,a} = V_{3,a}$, $H_{4,a} = V_{4,a}$.

We define that $H_a = \sum_{k=2}^{4} H_{k,a}$ and $V_a = \sum_{k=2}^{4} V_{k,a}$. Then we can have:

$$\begin{aligned}
&\sum_{\sigma_i \in S_a} w(\sigma_i) \\
&= \sum_{e \in cell(Q)} \sum_{\sigma_i \in S(e)} w(\sigma_i) \leq \sum_{e \in cell(Q)} \sum_{\sigma_i \in S^*(e)} w(\sigma_i) \\
&= \sum_{\sigma_i \in S^*} w(\sigma_i) + \sum_{k=2}^{4} \sum_{\sigma_i \in H_{k,a}} (k-1) w(\sigma_i) + \sum_{\sigma_i \in V_{2,a}} w(\sigma_i) \\
&= \sum_{\sigma_i \in S^*} w(\sigma_i) + \sum_{\sigma_i \in H_a} w(\sigma_i) + \sum_{\sigma_i \in V_a} w(\sigma_i) + \sum_{\sigma_i \in H_{4,a}} w(\sigma_i)
\end{aligned}$$

We can conclude that:

$$\sum_{\sigma_i \in S_a} w(\sigma_i) \leq \sum_{\sigma_i \in S^*} w(\sigma_i) + 2 \sum_{\sigma_i \in H_a} w(\sigma_i) + \sum_{\sigma_i \in V_a} w(\sigma_i)$$

Obviously, $\forall s \in S^*$, the detection cycle of s cannot intersect both the horizontal line of $P(2ra, 2ra)$ and the horizontal line of $P(2rb, 2rb)$ at the same time if $a \neq b$. We can infer that:

$$\sum_{a=0}^{m-1} \sum_{\sigma_i \in H_a} w(\sigma_i) \leq \sum_{\sigma_i \in S^*} w(\sigma_i)$$

Similarly,

$$\sum_{a=0}^{m-1} \sum_{\sigma_i \in V_a} w(\sigma_i) \leq \sum_{\sigma_i \in S^*} w(\sigma_i)$$

Therefore, we can infer that: $\sum_{a=0}^{m-1} \sum_{\sigma_i \in S_a} w(\sigma_i) \leq \sum_{a=0}^{m-1} (\sum_{\sigma_i \in S^*} w(\sigma_i) + 2 \sum_{\sigma_i \in H_a} w(\sigma_i) + \sum_{\sigma_i \in V_a} w(\sigma_i))$.

Simplifying the equation above, we can get that:

$$\sum_{a=0}^{m-1} \sum_{\sigma_i \in S_a} w(\sigma_i) \leq (m+3) \sum_{\sigma_i \in S^*} w(\sigma_i)$$

$$\frac{1}{m} \sum_{a=0}^{m-1} \sum_{\sigma_i \in S_a} w(\sigma_i) \leq (1 + \frac{3}{m}) \sum_{\sigma_i \in S^*} w(\sigma_i)$$

The average value of all solutions that we get from the partitions $P(0,0), P(2r, 2r), \cdots, P(2(m-1)r, 2(m-1)r)$ is $(1 + \frac{3}{m}) \sum_{\sigma_i \in S^*} w(\sigma_i)$. In Algorithm 7, we select the solution with minimum value. Thus the approximation ratio of Algorithm 7 is less than $1 + 3/m$. Then Theorem 8 is proved. □

Algorithm 8 EEMA Algorithm

Input: k stations which can send mobile sensors to cover the targets in the surveillance region, P, T

Output: The schedule of mobile sensors to cover T

1: Apply Algorithm 2 to preprocess the targets T
2: Apply Algorithm 3 to get all subareas generated by T
3: Calculate the weight of all subareas according to the position of P
4: Apply Algorithm 7 to select subareas
5: Choose sensors from P to the selected subareas
6: **return** The solution of k-MMTC

The detail of EEMA is shown in Algorithm 8.

According to Theorem 7 and Theorem 8, we can conclude that $\forall \varepsilon > 0$, EEMA can be an $(1+\varepsilon)$-approximation algorithm for k-MMTC problem that runs in time $n^{O(1/\varepsilon^2)}$, when we set $m = \lceil 3/\varepsilon \rceil$.

V. Distributed EEMA for Large Scale Networks

In large scale networks, it is impractical to solve k-MMTC problem by a centralized algorithm. Thus we propose a distributed version of EEMA (D-EEMA) in this section.

To keep the connectivity of the sensors, we use some mobile sensors for communication. We call these sensors as communication sensors which do not have sensing tasks. The communication sensors just need to move around the targets and the stations to collect sensing data [34]–[36]. D-EEMA is divided into two phases. In the first phase, we divide the surveillance region into some subareas and get the positions of the targets. In the second phase, we group the targets and deal with different groups respectively by Algorithm 8.

A. Assumptions

We make the following assumptions in large scale network. The targets and stations are randomly placed in the surveillance area. Every station knows the boundary and the obstacles in the surveillance region. A sink node can collect and combine information from stations. However the sensors cannot access the positions of the nearby targets until the targets are sensed by the sensors. All sensors can keep moving with a constant speed. The communication range of the sensors is at least double of the sensing range.

B. The First Phase

In the first phase, we try to get the locations of all targets, since the positions of targets is unknown initially. We apply the Voronoi algorithm [32] and divide the surveillance region into some subareas according to the station locations. Then we send one mobile sensor from each station to detect all the targets in the corresponding subarea.

In this paper, we let the mobile sensors travel along the Peano Curve in the subareas and get the positions of targets in their subarea. When the sensors need to pass some obstacles, they can move along the boundary of the obstacles and store the positions and the shape of the obstacles. The sensor store

Algorithm 9 The First Phase of D-EEMA

Input: The station set P
Output: The targets T in the surveillance region
In the first round, for every station p:
Search the neighboring stations positions of p
Calculate the subarea corresponding to p according to the Voronoi division
Send out a mobile sensor to detect and collect the information of obstacles and targets in the corresponding subarea
In the ith ($i > 1$) round, for every sensor s:
Travel along the Peano Curve in the corresponding subarea
if s travel across a obstacle **then**
  Move along the boundary of the obstacle and store the information of the obstacle
if s detect a target **then**
  Update the targets information table and add the detected target position to the table
if s returns back to the corresponding station **then**
  Send the information about the targets and obstacles to the corresponding station
In the ith ($i > 1$) round, for every station p:
When receive the information about the targets and obstacles:
if The information is new **then**
  Combine the information into table and transmit it to the sink node

the positions of targets in a table and return to the stations. The communication sensors move around the surveillance region and collect the information about the positions of targets. Then the communication sensors can combine the information and send it to the sink node.

The detail of D-EEMA in the first phase are presented in Algorithm 9.

C. The Second Phase

In the second phase, we firstly group the detected targets in the surveillance region into some clusters, then we deal with each cluster respectively.

In this phase, we proposed a novel algorithm, named Grouping Algorithm (GA). Obviously, we hope that the number of targets in each cluster is not too large. Since the time complexity is high when a cluster contains too many targets. At the same time, we also hope that the solutions in different clusters do not interfere with each other. Thus we can deal with each cluster respectively. Apparently, if and only if the distance between two targets, which are in different clusters, is less than $2r$, the solution in a cluster interfere the optimal solution in the other cluster. This is the main idea of GA.

In the algorithm, we firstly construct a unweighted and undirected graph based on the targets set. Each vertex in the graph represents a target and there is an edge between two vertices when the distance between two corresponding targets is less than $2r$. Then we find all the connected components, denoted by c_1, c_2, $\cdots$, c_k in the graph. Set two thresholds

Algorithm 10 Grouping Algorithm (GA)

Input: The target set T, num_{min}, num_{max}
Output: The clusters to group T
Construct the graph corresponding to the targets set. Each vertex in the graph represents a target and there is an edge between two vertices when the distance between two corresponding targets is less than $2r$
Find all the connected components of the graph, denoted by denoted by $C = \{c_1,\ c_2,\ \cdots, c_k\}$;
for all c_i in C **do**
  if The number of vertices is more than num_{max} **then**
    Apply Karger-Stein algorithm to cut c_i into $\lceil \frac{num_i}{num_{max}} \rceil$ connected components where num_i denotes the number of vertices in c_i
    Select the solution with the minimum cut where each component has more than num_{min} vertices.
    Add the generated components to C
  else
    The targets corresponding to the vertices in c_i are in a new cluster.
return The clusters of T

num_{min} and num_{max}. num_{min} denotes the minimum number of targets in a cluster which is generated by the minimum cut algorithm. The minimum cut algorithm will be applied in the next step. num_{max} denotes the maximum number of targets in a cluster. If the number of vertices in c_i ($0 \leq i \leq k$) is less than num_{max}, then c_i represents a cluster. Otherwise, we iteratively cut c_i into some connected components until the number of vertices in the connected components is less than num_{max}. In this paper, we apply a modified Karger-Stein algorithm to cut the components. In the original Karger-Stein algorithm [37], we get some different cut solutions and select the minimum cut. In the modified Karger-Stein algorithm, we select the minimum cut such that the number of vertices of each generated connected components is more than num_{min}.

The detail of this grouping algorithm are presented in Algorithm 10.

After we group the targets by the GA, we deploy a node for calculation in each cluster. The calculation node can schedule the mobile sensors to cover targets in the subarea according to EEMA. When the node schedule the sensors by EEMA, it only needs to consider the stations which are close to the cluster. Thus we can reduce the time complexity effectively.

The detail of D-EEMA in the second phase are presented in Algorithm 11.

D. Performance Analysis

Apparently, the time complexity of D-EEMA is dominated by the sink node, since the sink node needs to apply GA and group the detected targets into clusters. In this section we will analyze the time complexity of GA.

Theorem 9: The time complexity of GA is $O(n^3 \ln^3\ n)$ where n is the number of targets.

Algorithm 11 The Second Phase of D-EEMA

Input: P, T

Output: The schedule of mobile sensors to cover T

In the ith round, for the sink node:

1: Collect the information about the detected targets positions from the communication sensors
2: After receiving information from all communication sensors, group the detected targets by Algorithm 10
3: Allocate a calculation node for each group

In the ith ($i > 1$) round, for every calculation node:

When receive message from the sink node:

4: Select the stations which is close to the targets cluster
5: Schedule the mobile sensors to cover T by Algorithm 8
6: Send the schedule to the involved stations

In the ith ($i > 1$) round, for every station p:

When receive message from the calculation node:

7: Send the mobile sensors to cover targets

TABLE I
SIMULATION PARAMETERS

Simulation	Surveillance Region	$\|T\|$	$\|P\|$	r	m
A	50×50	20	10	1	3
B	500×500	50-230	20-400	1	9
C	10000×10000	1000-3000	30-300	1	9

Proof: When we construct the graph, we need to calculate the distance between two targets. Thus the time complexity of this step is $O(n^2)$.

The time complexity of the modified Karger-Stein algorithm is $O(n^2 \ln^3 n)$. When we apply the modified Karger-Stein algorithm to get a cut solution, the number of vertices in the original connected component will reduce at least num_{min}. Thus we apply the modified Karger-Stein algorithm at most n/num_{min} which is less than n.

We can conclude that the time complexity of GA is $O(n^3 \ln^3 n)$. □

VI. EVALUATION

In this section, to confirm the effectiveness and efficiency of EEMA, we compare the numerical results with the optimal solution Firstly. We can hardly calculate the optimal solution when the number of targets is huge. Thus we compare EEMA and D-EEMA with TV-greedy [29], [30] in this situation. The time complexity of TV-greedy is $O(n \log n)$ which is better than our algorithms obviously. Thus we only compare the results in the different algorithms.

A. Comparison Between EEMA and the Optimal Solution

The detailed parameters in the evaluation are listed in the simulation A of Table I. The surveillance region is $50\text{m} \times 50\text{m}$. The number of targets is 20. The number of stations which can send sensors is 10. The sensing range of sensors is 1m. In Algorithm 6, we set the value of m as 3.

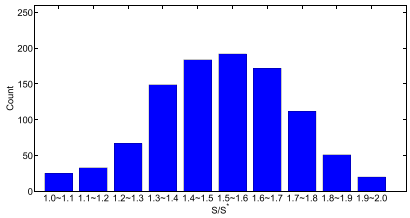

Fig. 6. Comparison with the optimal solution.

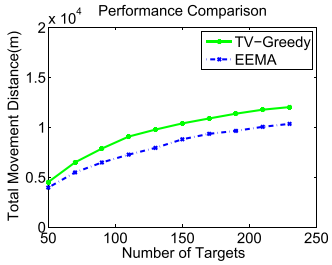

Fig. 7. EEMA: Comparison with different targets density.

We run the algorithm for 1000 times with different initial deployment of sensors and targets.

If $m = 3$, we can get that the approximation ratio of EEMA is 2. In Figure 6, we can find that the experiment result is consistent with Theorem 8. We can conclude that EEMA works well for the random small scale inputs.

B. Comparison Between EEMA and TV-Greedy

In the simulation B, we compare EEMA with TV-greedy [29], [30]. The detailed parameters in the evaluation are listed in Table I. The surveillance region is $500\text{m} \times 500\text{m}$. We change the number of targets from 50 to 200. The number of stations which can send sensors changes from $20 - 400$. The sensing range of sensors is $1m$. In Algorithm 6, we set the value of m as 9.

In this subsection, we also run EEMA and TV-greedy for 1000 times with regard to the same number of targets and stations. Then we calculate the average value.

Firstly, we set $|P| = 100$ and change the number of targets from 50 to 230. The numerical result is shown as Figure 7. We can see that EEMA works more effectively than TV-Greedy. Moreover, EEMA can save more energy than TV-Greedy especially when the number of targets is large.

Then we set $|T| = 100$ and change the number of stations from 20 to 400. The numerical result is shown as Figure 8. We can see that EEMA still works well and is more energy efficient than TV-Greedy. When the number of stations increases, EEMA saves less energy than TV-Greedy. The reason is that when the number of stations is large, each target has many stations nearby and we can easily get an energy efficient schedule.

C. Comparison Between D-EEMA and TV-Greedy

In this subsection, we compare D-EEMA with TV-greedy in large scale networks.

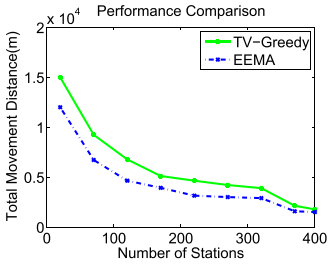

Fig. 8. EEMA: Comparison with different stations density.

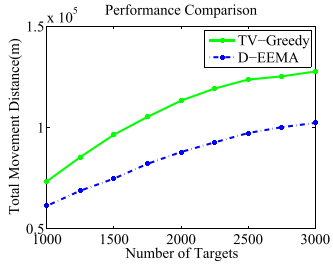

Fig. 9. D-EEMA: Comparison with different targets density.

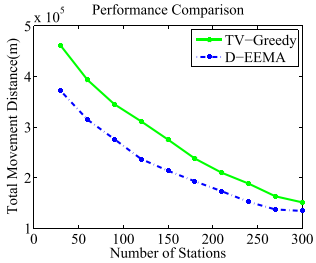

Fig. 10. D-EEMA: Comparison with different stations density.

The detailed parameters in the evaluation are listed in the simulation C of Table I. We set $num_{max} = 200$ and $num_{min} = 10$. The surveillance region is 1000m×1000m. We change the number of targets from 1000 to 3000. The number of stations which can send mobile sensors change from $30 - 300$. The sensing range of sensors is $1m$ and we also set the value of m as 9.

Firstly, we set $|P| = 100$ and change the number of targets from 1000 to 3000. The numerical result is shown as Figure 9. We can see that D-EEMA is more effective than TV-Greedy, and D-EEMA can save about ten percent energy.

Then we set $|T| = 2000$ and change the number of stations from 30 to 300. The numerical result is shown as Figure 10. We can see that D-EEMA is still more energy efficient than TV-Greedy, though the time complexity of TV-Greedy is better.

VII. Conclusion

In this paper, we consider a variation of target coverage problem in mobile sensor networks named *k-Sink Minimum Movement Target Coverage* (k-MMTC). To solve this problem, we propose a polynomial-time approximation scheme (PTAS), named *Energy Effective Movement Algorithm* (EEMA). For large scale networks, we propose a distributed version named D-EEMA. We also provide experiments to validate the effectiveness and efficiency of EEMA and D-EEMA. In all, EEMA is the first PTAS for sensor movement scheduling for target coverage problem.

In the future work, we will study the problem where each station p_i can only send q_i $(q_i > 0)$ sensors.

References

[1] B. Wang, *Coverage Control in Sensor Networks*. London, U.K.: Springer, 2010.
[2] R. Yang, X. Gao, F. Wu, and G. Chen, "Distributed algorithm for full-view barrier coverage with rotatable camera sensors," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.
[3] Y. Wang, S. Wu, X. Gao, F. Wu, and G. Chen, "Efficient line k-coverage algorithms in mobile sensor network," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl. (WASA)*, 2015, pp. 581–591.
[4] C. Lou, X. Gao, F. Wu, and G. Chen, "Energy-aware clustering and routing scheme in wireless sensor network," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl. (WASA)*, 2015, pp. 386–395.
[5] B. Liu, O. Dousse, P. Nain, and D. Towsley, "Dynamic coverage of mobile sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 2, pp. 301–311, Feb. 2013.
[6] S. He, J. Chen, X. Li, X. Shen, and Y. Sun, "Cost-effective barrier coverage by mobile sensor networks," in *Proc. Int. Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 819–827.
[7] H. M. Ammari, "On the problem of k-coverage in mission-oriented mobile wireless sensor networks," *Comput. Netw.*, vol. 56, no. 7, pp. 1935–1950, 2012.
[8] Y.-C. Wang and Y.-C. Tseng, "Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 9, pp. 1280–1294, Sep. 2008.
[9] T. Wimalajeewa and S. K. Jayaweera, "A novel distributed mobility protocol for dynamic coverage in sensor networks," in *Proc. IEEE Global Telecommun. Conf. (GTC)*, Dec. 2010, pp. 1–5.
[10] P. K. Sahoo and W.-C. Liao, "HORA: A distributed coverage hole repair algorithm for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 7, pp. 1397–1410, Jul. 2015.
[11] D. Zorbas and T. Razafindralambo, "Prolonging network lifetime under probabilistic target coverage in wireless mobile sensor networks," *Comput. Commun.*, vol. 36, no. 9, pp. 1039–1053, 2013.
[12] S. Mini, S. K. Udgata, and S. L. Sabat, "Sensor deployment for probabilistic target k-coverage using artificial bee colony algorithm," in *Swarm, Evolutionary, and Memetic Computing*. Heidelberg, Germany: Springer, 2011, pp. 654–661.
[13] D. K. Yau, N. K. Yip, C. Y. Ma, N. S. Rao, and M. Shankar, "Quality of monitoring of stochastic events by periodic and proportional-share scheduling of sensor coverage," *ACM Trans. Sensor Netw.*, vol. 7, no. 2, pp. 2019–2021, 2010.
[14] S. J. Habib and P. N. Marimuthu, "A coverage restoration scheme for wireless sensor networks within simulated annealing," in *Proc. Int. Conf. Wireless Opt. Commun. Netw. (WOCN)*, Sep. 2010, pp. 1–5.
[15] Y. Li, Y.-Q. Song, Y.-H. Zhu, and S. Rene, "Deploying wireless sensors for differentiated coverage and probabilistic connectivity," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2010, pp. 1–6.
[16] R. Andreacute, S. Alok, and M. Sevaux, "Column generation algorithm for sensor coverage scheduling under bandwidth constraints," *Networks*, vol. 60, no. 3, pp. 141–154, 2012.
[17] C. Ş. Şahin, M. Ü. Uyar, S. Gundry, and E. Urrea, "Self organization for area coverage maximization and energy conservation in mobile ad hoc networks," *Trans. Comput. Sci.*, vol. 15. Heidelberg, Germany: Springer, 2012, pp. 49–73, 2012.
[18] X. Yu, C. Wu, D. Chen, and N. Hu, "Level set based coverage holes detection and holes healing scheme in hybrid sensor network," *Int. J. Distrib. Sensor Netw.*, vol. 41, no. 4, pp. 128–134, 2013.
[19] F. Lin, Z. Sun, and T. Qiu, "Genetic algorithm-based 3D coverage research in wireless sensor networks," in *Proc. Int. Conf. Complex, Intell., Softw. Intensive Syst. (CISIS)*, Jul. 2013, pp. 623–628.
[20] L. Ding *et al.*, "Constant-approximation for target coverage problem in wireless sensor networks," in *Proc. Int. Conf. Comput. (INFOCOM)*, Mar. 2012, vol. 131. no. 5, pp. 1584–1592.

[21] C. Liu and G. Cao, "Spatial-temporal coverage optimization in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 5, pp. 465–478, Apr. 2011.
[22] X. Xu and M. Song, "Restricted coverage in wireless networks," in *Proc. Int. Conf. Comput. (INFOCOM)*, Apr. 2014, pp. 558–564.
[23] J. Chen, J. Li, and T. H. Lai, "Energy-efficient intrusion detection with a barrier of probabilistic sensors: Global and local," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4742–4755, Sep. 2013.
[24] G. Barun and M. P. Sarathi, "Point and area sweep coverage in wireless sensor networks," in *Proc. Int. Symp. Modeling Optim. Mobile, Ad Hoc Wireless Netw. (WiOpt)*, May 2013, pp. 140–145.
[25] B. Gorain and P. S. Mandal, "Approximation algorithms for sweep coverage in wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 74, no. 8, pp. 2699–2707, 2014.
[26] H. Huang *et al.*, "Connected wireless camera network deployment with visibility coverage," in *Proc. Int. Conf. Comput. (INFOCOM)*, Apr. 2014, pp. 1204–1212.
[27] B. Wang, H. B. Lim, and D. Ma, "A survey of movement strategies for improving network coverage in wireless sensor networks," *Comput. Commun.*, vol. 32, nos. 13–14, pp. 1427–1436, 2009.
[28] C. Wu, G. S. Tewolde, W. Sheng, B. Xu, and Y. Wang, "Distributed multi-actuator control for workload balancing in wireless sensor and actuator networks," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2462–2467, Oct. 2011.
[29] Z. Liao, S. Zhang, J. Cao, W. Wang, and J. Wang, "Minimizing movement for target coverage in mobile sensor networks," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2012, pp. 194–200.
[30] Z. Liao, J. Wang, S. Zhang, J. Cao, and G. Min, "Minimizing movement for target coverage and network connectivity in mobile sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 7, pp. 1971–1983, Jul. 2015.
[31] R. Rao and G. Kesidis, "Purposeful mobility for relaying and surveillance in mobile ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 3, pp. 225–231, Jul. 2004.
[32] M. H. Alsuwaiyel, *Design Techniques and Analysis*. Singapore: World Scientific, 2003.
[33] D.-Z. Du, K.-I. Ko, and X. Hu, *Design and Analysis of Approximation Algorithms*. New York, NY, USA: Springer, 2012.
[34] Y. Feng, X. Gao, F. Wu, and G. Chen, "Shorten the trajectory of mobile sensors in sweep coverage problem," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.
[35] X. Gao, X. Zhu, Y. Feng, F. Wu, and G. Chen, "Data ferry trajectory planning for sweep coverage problem with multiple mobile sensors," in *Proc. IEEE Int. Conf. Sens., Commun. Netw. (SECON)*, Jun. 2016, pp. 1–9.
[36] Z. Chen *et al.*, "Efficient scheduling strategies for mobile sensors in sweep coverage problem," in *Proc. IEEE Int. Conf. Sens., Commun. Netw. (SECON)*, Jun. 2016, pp. 1–4.
[37] D. R. Karger and C. Stein, "A new approach to the minimum cut problem," *J. ACM*, vol. 43, no. 4, pp. 601–640, 1996.
[38] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Inf. Process. Lett.*, vol. 12, no. 3, pp. 133–137, 1981.

Xiaofeng Gao received the B.S. degree in information and computational science from Nankai University, China, in 2004, the M.S. degree in operations research and control theory from Tsinghua University, China, in 2006, and the Ph.D. degree in computer science from The University of Texas at Dallas, USA, in 2010. She is currently an Associate Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. She has authored over 100 peer-reviewed papers in the related area, including well-archived international journals, such as the TC, TPDS, TMC, TKDE, ENGINEERING, and also in well-known conference proceedings, such as INFOCOM, SIGKDD, and ICDCS. Her research interests include wireless communications, data engineering, and combinatorial optimizations. She has served on the Editorial Board of *Discrete Mathematics, Algorithms and Applications*.

Zhiyin Chen received the B.S. degree in computer science and technology from the School of Computer and Information Engineering, Henan University, in 2011, and the M.S. degree in computer application from the University of Chinese Academy of Sciences in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include coverage problem in wireless sensor works, combinatorial optimization, and approximation algorithm.

Fan Wu received the B.S. degree in computer science from Nanjing University in 2004, and the Ph.D. degree in computer science and engineering from The State University of New York at Buffalo in 2009. He has visited the University of Illinois at Urbana–Champaign as a Post-Doctoral Research Associate. He is currently an Associate Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He has authored over 100 peer-reviewed papers in leading technical journals and conference proceedings. His research interests include wireless networking and mobile computing, algorithmic game theory and its applications, and privacy preservation. He has served as the Chair of CCF YOCSEF Shanghai and a member of technical program committees of over 40 academic conferences. He has served on the Editorial Board of *Computer Communications*.

Guihai Chen received the B.S. degree from Nanjing University in 1984, the M.E. degree from Southeast University in 1987, and the Ph.D. degree from The University of Hong Kong in 1997. He had been invited as a Visiting Professor by many universities, including the Kyushu Institute of Technology, Japan, in 1998, The University of Queensland, Australia, in 2000, and Wayne State University, USA, from 2001 to 2003. He is currently a Distinguished Professor with Shanghai Jiao Tong University, China. He has authored over 400 peer-reviewed papers, and over 200 of them are in well-archived international journals or conference proceedings. He has a wide range of research interests on sensor network, peer-to-peer computing, high-performance computer architecture, and combinatorics.