EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Minimum-Delay POIs Coverage in mobile wireless sensor networks

Wenping Chen*, Si Chen and Deying Li

Abstract

In some mobile wireless sensor network applications, it is not necessary to monitor the entire field all the time, and only a number of critical points or points of interest (*POIs*) need to be monitored periodically. In this paper, we address the *minimum-delay POIs coverage* in *mobile wireless sensor networks* problem with cost restriction, which is how to schedule the limited number of mobile sensors monitoring to minimize the service delay of POIs. We study two scenarios of the problem. In the first scenario, the start positions of mobile sensors are determined in advance, we propose the *SSR* algorithm to address this problem. In the second scenario, without pre-defined start positions, we propose two algorithms, the *TSP-S* and the *SSNOR*. By the comprehensive simulations, we evaluate the performance of the proposed algorithms. The simulation results show the efficiency of our algorithms.

Keywords: Sweep coverage; POIs; Mobile sensor; TSP

1 Introduction

Wireless sensor networks (*WSNs*) have been widely studied in recent years. Coverage is one of the most important issues in WSNs. There have been a great number of works on area coverage, target coverage and barrier coverage. Most of them require the targets or monitoring area is covered continuously. However, in some applications, such as patrol inspection, only some critical points are needed to be detected periodically. Employing continuous coverage in such application is undoubtedly wasteful.

To satisfy the demands of the above applications with low system cost, another type of coverage called sweep coverage is proposed [1]. The sweep coverage has been applied to practical application, such as GreenOrbs [1], which is a sustainable and large-scale wireless sensor network system in the forest. GreenOrbs provide all-year-round ecological surveillance in the forest in Tianmu Mountain, collecting various sensory data, such as temperature, humidity, illumination, and content of carbon dioxide. In sweep coverage, a set of Points of Interests *(POIs)* in the monitoring area are periodically detected instead of being continuously monitored. To achieve sweep coverage, a downsized number of mobile sensors are employed to sweep the POIs at regular intervals.

*Correspondence: chenwenping@ruc.edu.cn
School of Information, Renmin University of China, Beijing, China

There have been some efforts on the sweep coverage problem [1,2]. These works focus on how to schedule minimum number of mobile sensors to achieve the sweep coverage within specified sweep period. The existing works assumed that they have enough mobile sensors. However, in real applications, there may have been restrictions on the number of mobile sensors due to the system cost.

In this paper, we investigate POI coverage under the restriction on the number of mobile sensors. We address how to schedule the mobile sensors to minimize the sweep delay of POIs. As shown in Figure 1, there are five mobile sensors in the area. Each mobile sensor scans specific POIs in its trajectory, respectively. The delay of the sweep is defined as the time interval from sensors starting to sweep to all the sensors finishing their monitoring task. Suppose the moving speed of each mobile sensor is the same; since $M4$ has the longest trajectory, the delay of sweep is decided by $M4$ which is several times longer than that of $M1, M2, M3$, and $M5$. We study two scenarios of the problem. In the first scenario, the energy of sensors may not be enough to sustain for a long time, such that the sensors must be recharged periodically. Thus, the start positions of mobile sensors located at the recharging places are determined. We present a greedy-based algorithm named *SSR*. For each mobile sensor's determined start position, we always choose the POI that can minimize the length difference between

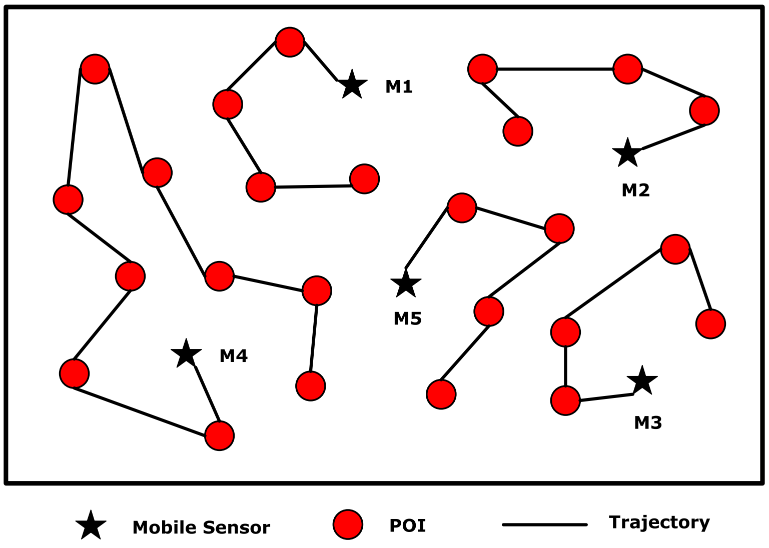

Figure 1 The POIs coverage.

the maximal and minimal trajectories of mobile sensors. The procedure continues until all the POIs have been added to the trajectories of mobile sensors. In the second scenario, the energy of sensor is enough to sustain for a long time, such that the sensors need not return back frequently. Compared with the long sweeping time, the time that the mobile sensor moves from start position to the the first POI to be swept can be ignored. We propose two algorithms. One is constructing a traveling salesman problem (*TSP*) ring using the polynomial-time approximation scheme (*PTAS*) for the TSP [3], then divide the ring into *k* trajectories and let every mobile sensor move along with a trajectory, respectively. The second algorithm is derived from the SSR algorithm.

The rest of this paper is organized as follows: Section 2 reviews the existing works. In Section 3, we introduce the network model and the definition of *Minimum-Delay POIs Coverage* in *mobile wireless sensor networks* problem. In Section 4, we propose the Stretched Searching SSR algorithm for the Minimum-Delay POIs Coverage problem with restriction on start positions of mobile sensors. The *TSP-S* algorithm and the *SSNOR* algorithm are presented for the the Minimum-Delay POIs Coverage problem without restriction on start positions of mobile sensors in Section 5. We evaluate the performance of our algorithms through extensive simulations in Section 6. Finally, we conclude the paper in Section 7.

2 Related work

The coverage problem is one of the fundamental issues in wireless sensor networks, evidenced by many works contribute to this field in recent years. It can be classified into three categories: full coverage, barrier coverage, and sweep coverage.

Many efforts have been made on full coverage problem (including point coverage [4-8], area coverage [9-18]), and barrier coverage problem [19-26]. Full coverage and barrier coverage often require that target or area is covered all the time. To achieve full coverage or barrier coverage, a majority of works studied the static coverage under the requirement of continuous coverage. Although it can achieve the best coverage, the system cost is prohibitive. To improve the coverage quality efficiently, a type of mixed network infrastructure called hybrid network is adopted in some applications, which is composed of mobile sensors as well as static sensors [27-30]. Wang et al. studied the optimized movement of mobile sensors to provide *k*-coverage in both mobile sensor networks and hybrid sensor networks [27]. They also presented a distributed relocation algorithm, such that each mobile sensor can achieve optimal relocation with the local information.

The sweep coverage problem is motivated by the applications without continuous coverage requirement. In such cases, mobile sensors are often used. There are some existing works about sweep coverage [1,2]. The main goal of those works is to minimize the number of mobile sensors so as to cover all the POIs in the region without violating the given time. Li et al. proposed a centralized algorithm CSWEEP [1]. Firstly, the authors take all POIs as input and employ a PTAS algorithm [31] of TSP and get an approximate TSP ring. Secondly, every mobile sensor is scheduled to move along the TSP ring segments back and forth under the requirement of sweep period.

Du et al. [2] proposed two algorithms for different situations. In the first algorithm, they gradually deploy mobile sensors and schedule the mobile sensors to move on the same trajectory in each period. In the second algorithm, named OSWEEP algorithm, the mobile sensors are not required to follow the same trajectory in each period, all the mobile sensors are scheduled to move along the TSP ring which consists of POIs towards the same direction. The OSWEEP algorithm has better performance than CSWEEP.

Both of the works assumed that they can obtain enough mobile sensors. However, in many practical applications, due to high energy consumptions and resource constraints, there may not be enough mobile sensors to accomplish monitoring.

In this paper, we consider the situation that the number of mobile sensors is given. We discuss two scenarios for the problem: (1) the start positions of mobile sensors are determined in advance; (2) the start positions of mobile sensors are not determined, and we ignore the time that mobile sensor moves from start position to the first POI to be swept.

3 Minimum-Delay POIs Coverage in mobile wireless sensor networks problem

3.1 Network model

Suppose there are k given mobile sensors to monitor m POIs deployed in a region R. Each POI has a globally unique ID and a fixed position. We denote the set of mobile sensors as $S = \{s_1, \ldots, s_k\}$ and the set of POIs as $P = \{p_1, \ldots, p_m\}$. Let $d(p_i, p_j)$ be the Euclidean distance between points p_i and p_j. There are two scenarios. In the first one, all mobile sensors start from the determined positions to scan specific POIs along different trajectories and go back to the start position to recharge after a period. In the second one, the mobile sensors have enough energy to sweep POIs for quite a long time, therefore, the distance from recharging place to start position of each mobile sensor is ignored. We assume that all the mobile sensors move at a constant speed v in the region. In fact, if the moving speed of mobile sensor is not the same when generating trajectory for each mobile sensor, we will take its speed as a ratio to compute the distance of the trajectory. We also assume the sensing range of all mobile sensors is very small such that the POIs can be covered only when the mobile sensors pass the position of the POIs.

In this paper, we study the minimum-delay POIs coverage in mobile wireless sensor networks. We aim to schedule k mobile sensors to scan those POIs such that the delay of the mobile wireless sensors network *(MWSN)* is minimized and each POI can be scanned exactly one time in a period.

3.2 Problem definition

Before formalizing the Minimum-Delay POIs Coverage in mobile wireless sensor networks problem, we first introduce a few of definitions as follows:

Definition 1. ***POIs Coverage (PC)****: If all POIs are swept by some mobile sensors, we call it POIs Coverage.*

Definition 2. ***The Delay of POIs Coverage****: The delay of POIs Coverage is the time interval from the mobile sensors starting to sweep to all POIs having been swept.*

Definition 3. ***Minimum-Delay POIs Coverage in mobile wireless sensor networks problem****: Given k mobile sensors and the deployments of m POIs, the Minimum-Delay POIs Coverage problem is to schedule k mobile sensors to scan POIs along different trajectories such that the delay of the POIs coverage is minimized.*

If there is no restriction on the start positions of mobile sensors, given the deployment of POIs, an undirected complete graph $G = (V, E, W)$ is derived from the mobile wireless sensor network, where V is the set of all POIs. For any two POIs p_i, p_j, there is an edge between p_i and p_j (i.e., $(p_i, p_j) \in E$), and $w(e) = d(p_i, p_j)$. If there is restriction on start positions of mobile sensor, given the deployment of POIs, an undirected weight graph $G_{\text{start}} = (V, V_{\text{start}}, E_{\text{start}}, W)$ is derived, where V is the set of all POIs and V_{start} is the set of start positions of mobile sensors. For any two POIs p_i, p_j, there is an edge between p_i and p_j (i.e., $(p_i, p_j) \in E_{\text{start}}$), and $w((p_i, p_j)) = d(p_i, p_j)$; for any start position of mobile sensor s_i and any POI p_j, there must be an edge (s_i, p_j) between s_i and p_j (i.e., $(s_i, p_j) \in E_{\text{start}}$), and $w((s_i, p_j)) = d(s_i, p_j)$.

Based on the given Euclidean distance between any two points, finding a coverage scheme for the Minimum-Delay POIs Coverage problem is equivalent to finding a set of trajectories which can pass by all the POIs.

Apparently, the Minimum-Delay POIs Coverage problem can be transformed to the problem of minimizing the maximal trajectory of mobile sensors: given k mobile sensors, find a set of k trajectories to pass all POIs such that the maximum length among all k trajectories is minimized.

It is easy to know that the problem with no restriction on the start positions of mobile sensors is NP-hard. This is because when $k = 1$, this problem becomes the minimum length Hamilton Path problem which is well known NP-hard.

4 Algorithm for the Minimum-Delay POIs Coverage with restriction

In this section, we study the Minimum-Delay POIs Coverage problem with the restriction on the start positions of mobile sensors, and propose an algorithm for the problem. To minimize the network delay which is decided by the longest monitoring period among all the mobile sensors, we aim to make the sweeping time of each mobile sensor be close to each other.

We first give the following definitions:

Definition 4. ***Max-min difference for a set of k trajectories (2MDkT)****: Given k trajectories, max-min difference is the length difference between the longest trajectory and the shortest trajectory.*

Definition 5. ***Minimum max-min trajectory schedule****: Among the feasible monitoring schedules of k mobile sensors, each of which can be represented as a set of the k mobile sensors' trajectories, minimum max-min trajectory schedule is the one with minimum 2MDkT.*

It is easy to know that the Minimum-Delay POIs Coverage problem is equivalent to the minimum max-min trajectory schedule problem. In this section, we propose an effective algorithm, the SSR algorithm for the Minimum-Delay POIs Coverage problem with the restriction on the start positions of mobile sensors.

The main idea of the SSR algorithm is as follows: in every step, we always choose the pair of POI and trajectory such that the 2MDkT of the current MWSN is minimized.

Before introducing the SSR algorithm, we give some notations used in the algorithm.

We denote C as the current selected trajectories for mobile sensors. $C = \{c_1, c_2, \ldots, c_k\}, c_i$ is the ith mobile sensor's trajectory. Initially, we may set $C = \emptyset$. We use T to represent the set of POIs which have not been covered by C. There are k mobile sensors and n POIs in the region. The input of the algorithm is $G_{\text{start}} = (V, V_{\text{start}}, E_{\text{start}}, W)$. The output of the algorithm is a set $C = \{c_1, c_2, \ldots, c_k\}$ of trajectories which contains all vertices in V and V_{start}. For the convenience to describe the algorithm, we let ΔL denote 2MDkT, $\Delta L_{\min}$ denote the current minimal ΔL in the MWSN. In the following, we present the SSR algorithm in details.

(1) Set $C = \emptyset$ and set all POIs to be *uncovered.*
(2) Select the shortest edge e_t^s in G where s is the start position of a mobile sensor and t is a POI. Then add t to the trajectory of mobile sensor s. Set t to be *covered.*
(3) As for the *POI i* which has not been *covered*, we tentatively add the *POI* to the trajectory of every mobile sensor j ($j = [1, \ldots, k]$) and compute the ΔL_i^j respectively. ΔL_i^j is the 2MDkT, that if the POI i is added to the the trajectory of mobile sensor j. Thus, $\Delta L_i = \{\Delta L_i^1, \Delta L_i^2, \ldots, \Delta L_i^k\}$.
(4) Set $\Delta L_{\text{cur}} = \{\Delta L_1, \Delta L_2, \ldots, \Delta L_m\}$, where m is the number of uncovered POIs. Find the minimal element ΔL_i^j in ΔL_{cur} and add the POI i to the trajectory of mobile sensor j. Then set the POI i to be covered.
(5) Repeat steps (3) to (4) until the trajectories in C contain all vertexes of V.
(6) Check the trajectory of each mobile sensor, if the trajectory intersects with other trajectories and the distance from start position to the point of intersection is the same, there will be collision between mobile sensors. Adjust the depart time of one of the mobile sensors slightly to avoid collision.

The pseudocode of the SSR algorithm is presented in Algorithm 1.

Algorithm 1 SSR($V, V_{\text{start}}, E_{\text{start}}, W, k, C$)
```
1:  Set $C = \phi$, $T = V$;
2:  for ($j = 1$ to $k$) do
3:      $c_j = c_j \cup \{v_j^{\text{start}}\}$;
4:      $v_j^{\text{cur}} = v_j^{\text{start}}$;
5:      $C = C \cup c_j$;
6:      $L_j = 0$;
7:  end for
8:  while $T \neq \emptyset$ do
9:      $\Delta L_{\min} = \text{INFINITE}$;
10:     for each $v_i$ in $T$ do
11:         for ($j = 1$ to $k$) do
12:             $L_j = L_j + |d_{v_i}, v_j^{\text{cur}}|$;
13:             $L_{\min} = Min\{L_1, L_2, \ldots, L_k\}$;
14:             $L_{\max} = Max\{L_1, L_2, \ldots, L_k\}$;
15:             $\Delta L_{\text{temp}} = L_{\max} - L_{\min}$;
16:             $L_j = L_j - |d_{v_i}, v_j^{\text{cur}}|$;
17:             if ($\Delta L_{\text{temp}} < \Delta L_{\min}$) then
18:                 $\Delta L_{\min} = \Delta L_{\text{temp}}$;
19:                 $t = i$;
20:                 $s = j$;
21:             end if
22:         end for
23:     end for
24:     $T = T - v_t$;
25:     $c_s = c_s \cup v_t$;
26:     $v_s^{\text{cur}} = v_t$;
27:     $L_s = L_s + |d_{v_t}, v_s^{\text{cur}}|$;
28: end while
```

Note that if there are more than one minimal elements in ΔL_{cur}, the element with lowest POI label number will be selected. Additionally, if there are two elements with the same POI label number, the mobile sensor with smaller label number is considered. In the next round of sweeping, the mobile sensors will move back along the original trajectories.

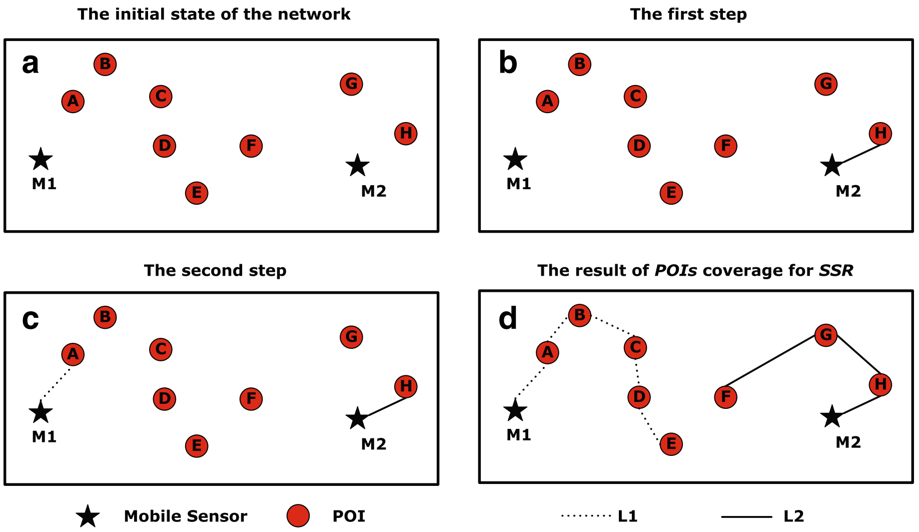
The initial state of the network
The first step
The second step
The result of POIs coverage for SSR
a
b
c
d
A
B
C
D
E
F
G
H
M1
M2
Mobile Sensor
POI
L1
L2

Figure 2 An example of POIs coverage for SSR. (a) There are two mobile sensors and eight POIs in the region. **(b)** POI *H* is added to the trajectory of *M*2. **(c)** POI *A* is added to the trajectory of *M*1. **(d)** The trajectories of *M*1 and *M*2 is constructed after the rest of the POIs are added one by one.

To better describe the SSR algorithm, we give an example as follows: in Figure 2a, there are two mobile sensors and eight POIs in the rectangle region. The weight of edge, which is the distance between two points in the area is illustrated in Table 1. We set $v = 100$ m/min for each mobile sensor. Firstly, as $\overline{\text{M2H}}$ is the smallest edge connected to the mobile sensors, we add POI H to the trajectory of mobile sensor $M2$ as shown in Figure 2b. Secondly, for each of the rest uncovered POIs, we tentatively add these seven POIs to the trajectories of mobile sensors $M1$ and $M2$. Meanwhile, we compute the 2MDkT in the current mobile sensor network after tentatively adding each of the rest uncovered POIs, as shown in Table 2. We find that when we try to add POI A to the trajectory of mobile sensor $M1$, the 2MDkT in the current mobile sensor network is the smallest compared with adding other POIs. Thus, we add POI A to the trajectory of mobile sensor $M1$ as shown in Figure 2c. Similarly, we add the rest POIs one by one until all the POIs are covered, and the result of the POIs coverage is shown in Figure 2d.

We set $M1$ and $M2$ to move on $L1$ and $L2$, respectively. The length of $L1$ is $1,324$ m and the length of $L2$ is $1,300$ m. So the difference between $L1$ and $L2$ in SSR is 24 m. Thus, the end time of one round of monitoring for $M1$ and $M2$ is almost the same. This is mainly because in every step of SSR algorithm, when choosing the next POI to join one of the mobile sensors' trajectory, we always guarantee that the 2MDkT of current schedule is always minimized after this POI is added. It means that the current delay of schedule in every step is minimized. Therefore, those features can effectively achieve the goal of our work.

Theorem 1. *The time complexity of SSR algorithm is* $O(k|V|^2)$, *where k is the number of mobile sensors and* $|V|$ *is the number of the POIs.*

Table 1 The weight of edges

	M1	*M2*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
M1	0	2,900	204	325	605	653	868	1,100	2,000	2,200
M2	2,900	0	2,050	2,000	1,600	1,500	1,400	1,100	313	100
A	204	2,050	0	120	463	595	821	1,000	1,900	2,100
B	325	2,000	120	0	400	586	831	976	1,800	2,000
C	605	1,600	463	400	0	300	507	585	1,400	1,600
D	653	1,500	595	586	300	0	300	440	1,400	1,500
E	868	1,400	821	831	507	300	0	290	1,300	1,400
F	1,100	1,100	1,000	976	585	440	290	0	1,000	1,100
G	2,000	313	1,900	1,800	1,400	1,400	1,300	1,000	0	200
H	2,200	100	2,100	2,000	1,600	1,500	1,400	1,100	200	0

Table 2 The 2MDkT after tentatively add every uncovered POI

POI/Mobile sensor	*M*1	*M*2
A	104	2,200
B	225	2,100
C	505	1,700
D	553	1,600
E	768	1,500
F	1,000	1,200
G	1,900	300

Proof. According to SSR algorithm, firstly, there are $|V|$ POIs needed to be added and only one POI can be added in each iteration. Secondly, after tentatively adding each POI to all the trajectories, the calculation of length difference between the maximal and minimal trajectories in current mobile sensor network costs $O(k|V|)$ time. Therefore, the computational complexity of the SSR algorithm is $O(k|V|^2)$. The proof finishes. □

5 Algorithms for the Minimum-Delay POIs Coverage problem without restriction

In this section, we will propose two algorithms for the Minimum-Delay POIs Coverage problem to deal with the scenario that the mobile sensors are not restricted to start sweeping from the determined positions, which is more flexible in practical applications.

5.1 TSP-based searching algorithm for the Minimum-Delay POIs Coverage without restriction

In this subsection, we propose a TSP-based searching algorithm (*TSP-S* algorithm). The main idea of the TSP-S algorithm is as follows: firstly, we create a weighted completed graph $G = (V, E, W)$ shown as Section 3.2. Secondly, we use the TSP algorithm in [3] to find a TSP ring on G. Thirdly, we divide the TSP ring into k trajectories. Finally, each mobile sensor moves along one trajectory back and forth.

Similar as the notations defined in Section 4, we denote C as a set of trajectories, $C = \{c_1, c_2, \ldots, c_k\}$, c_i is the ith trajectory. Initially, set $C = \emptyset$. T represents the set of POIs which have not been covered by C. The input of the algorithm is $G = (V, E, W)$. The output of the algorithm is a set $C = \{c_1, c_2, \ldots, c_k\}$ of trajectories which contains all vertexes in V. Next, we present the TSP-S algorithm in details.

(1) Set $C = \emptyset$ and set all POIs to be uncovered.
(2) Input G into the PTAS algorithm for TSP, and get an approximate TSP ring. The length of the TSP ring is $|L|$.
(3) Remove the current longest edge $l_{\max}$ in the TSP ring. We denote the left curve of the TSP ring as l_{cur}, $|l_{\text{cur}}| = |L_j| - |l_{\max}|$. Then we obtain a bound l, $l = |l_{\text{cur}}|/k$, k is the number of trajectories we aim to find.
(4) Select one of the ends of l_{cur}, and make it as the start of establishing the current trajectory j.
(5) Add POIs in l_{cur} to the current trajectory j in sequence until the length of j is larger than l, remove the last added POI i from j. Thus, we obtain the trajectory j and add it to C. And set the POIs in j to be covered.
(6) Remove the edge between the POI i and the last POI in trajectory j, which is denoted as l_{del}. Therefore, $|l_{\text{cur}}| = |l_{\text{cur}}| - |l_{\text{del}}|$, $k = k - 1$. Let POI i be the start of establishing the next trajectory and calculate the bound l again.
(7) Repeat steps (5) to (6) until the trajectories in C contains all POIs.
(8) When all POIs have been covered and the number of trajectories that have been found are less than k, then we will adjust the trajectories. We always choose the longest trajectory w in C and split it into two trajectories evenly until we have obtained k trajectories.

The pseudocode of the TSP-S algorithm is presented in Algorithm 2 and Algorithm 3.

Algorithm 2 TSP-S(V, E, W, k, C)

```
1:  Set $C = \phi$;
2:  ring $\leftarrow$ TSP();
3:  $t = k$, $t_{\text{cur}} = 1$;
4:  $l_{\text{cur}}$ = ring.length;
5:  $l_{\max}$ = max edge in ring;
6:  $l_{\text{cur}} = |l_{\text{cur}}| - |l_{\max}|$;
7:  $v_{\text{cur}}$ = POI on the max edge in ring;
8:  $c_{t_{\text{cur}}} = \{v_{\text{cur}}\}$, $l_{t_{\text{cur}}} = 0$, $C = C \cup c_{t_{\text{cur}}}$;
9:  for ($i = 0$ to ring.size()$-1$) do
10:     $v_{\text{next}}$ = ring.next();
11:     if $|l_{t_{\text{cur}}}| + |v_{\text{cur}}, v_{\text{next}}| <= |l_{\text{cur}}|/t$ then
12:         $c_{t_{\text{cur}}} = c_{t_{\text{cur}}} \cup \{v_{\text{next}}\}$;
13:         $l_{t_{\text{cur}}} = l_{t_{\text{cur}}} + |v_{\text{cur}}, v_{\text{next}}|$;
14:     else
15:         $t_{\text{cur}} + +$;
16:         $c_{t_{\text{cur}}} = c_{t_{\text{cur}}} \cup \{v_{\text{next}}\}$;
17:         $C = C \cup c_{t_{\text{cur}}}$, $l_{t_{\text{cur}}} = 0$;
18:         $l_{\text{cur}} = |l_{\text{cur}}| - |v_{\text{cur}}, v_{\text{next}}|$;
19:         $t - -$;
20:     end if
21:     $v_{\text{cur}} = v_{\text{next}}$;
22: end for
23: if $t_{\text{cur}} < k$ then
24:     for $t_{\text{cur}} = t_{\text{cur}} + 1$ to $k$ do
25:         $c_{\text{longest}} \leftarrow$ FindLongestTraj();
26:         $c_{\text{new}} \leftarrow$ SplitTraj($c_{\text{longest}}$);
27:         $C = C \cup c_{\text{new}}$;
28:     end for
29: end if
```

Algorithm 3 SplitTraj(c_{longest})

```
1:  Let $l_{\text{longest}}$ denote the longest trajectory in $C$.
2:  $h = |l_{\text{longest}}|/2$;
3:  $v_{\text{cur}} = c_{\text{longest}}.get()$;
4:  $c_{\text{new}} = v_{\text{cur}}$;
5:  $l_{\text{new}} = 0$;
6:  $c_{\text{longest}} = c_{\text{longest}} - \{v_{\text{cur}}\}$;
7:  while $l_{\text{new}} < h$ do
8:      $v_{\text{next}} = c_{\text{longest}}.\text{next}()$;
9:      $l_{\text{longest}} = |l_{\text{longest}}| - |v_{\text{cur}}, v_{\text{next}}|$;
10:     $c_{\text{longest}} = c_{\text{longest}} - \{v_{\text{next}}\}$;
11:     $c_{\text{new}} = c_{\text{new}} \cup v_{\text{next}}$;
12:     $l_{\text{new}} = l_{\text{new}} + |v_{\text{cur}}, v_{\text{next}}|$;
13:     $v_{\text{cur}} = v_{\text{next}}$;
14: end while
15: $v_{\text{next}} = c_{\text{longest}}.\text{next}()$;
16: $l_{\text{longest}} = |l_{\text{longest}}| - |v_{\text{cur}}, v_{\text{next}}|$;
17: Return $c_{\text{new}}$;
```

In the following, we explain the TSP-S algorithm by two examples. In the first example, there are three mobile sensors and a set of POIs. Firstly, as shown in Figure 3a, we obtain a TSP ring and calculate the bound l. Secondly, as shown in Figure 3b, we remove the longest edge $\overline{AJ}$ in the ring and choose POI A as the start location to establish the first trajectory. We add POI B, C, D, and E to the trajectory in sequence. However, when adding POI E to the first trajectory, we find that the length of current trajectory is larger than the bound l. Then, remove POI E from the trajectory and delete the edge $\overline{DE}$ from the ring as shown in Figure 3b. Thus, the first trajectory is $L1 = \{A, B, C, D\}$. Then, we calculate the bound l again and start from POI E to repeat the above steps to establish the next trajectory. The next removed edge is $\overline{GH}$ as shown in Figure 3c. At last, three trajectories are generated, the second and third trajectories are $L2 = \{E, F, G\}$ and $L3 = \{H, I, J\}$ as illustrated in Figure 3d. Let the three mobile sensors move along with the three trajectories, respectively.

Another example is shown in Figure 4 which is a special case for the TSP-S algorithm. There are four mobile sensors. However, when searching the third trajectory, all POIs in the region have been covered.

To solve this problem, we have to adjust the trajectories as shown in Figure 4a. We choose the longest one $L1$ among the three trajectories, and split $L1$ into two parts $L11$ and $L12$. After the adjustment, as shown in Figure 4b, there are four trajectories for four mobile sensors at last.

Theorem 2. *The time complexity of the TSP-S algorithm is $O(|V|^2)$, where $|V|$ is the number of the POIs.*

Proof. According to the TSP-S algorithm, firstly, creating an approximate *TSP* ring takes $O(|V|^2)$ time. Secondly, it costs $O(|V|)$ to divide and adjust the TSP ring into k trajectories. Therefore, the time complexity of the TSP-S algorithm is $O(|V|^2)$. The proof finishes. □

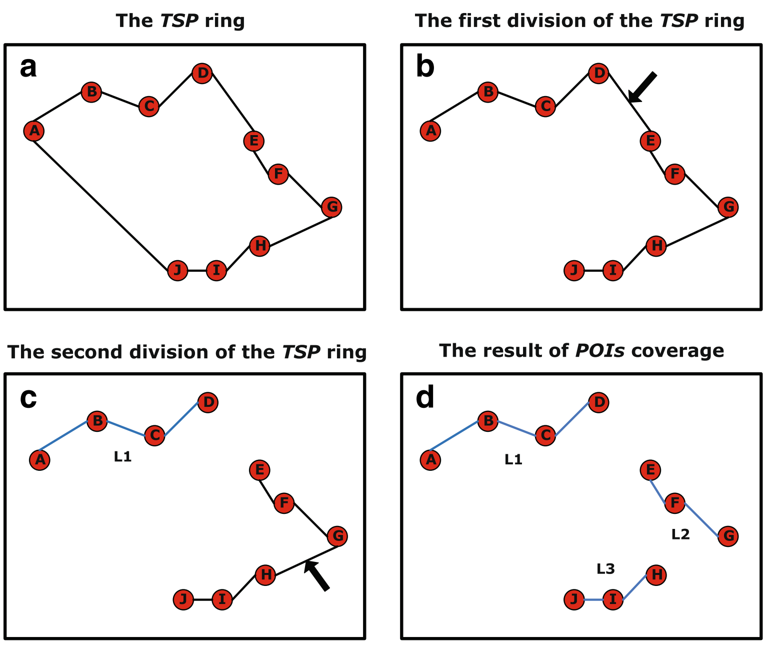
The *TSP* ring
a
A
B
C
D
E
F
G
H
I
J
The first division of the *TSP* ring
b
The second division of the *TSP* ring
c
L1
The result of *POIs* coverage
d
L1
L2
L3

Figure 3 An example of POIs coverage for TSP-S. (a) The TSP ring of the ten POIs. **(b)** Remove the longest edge in the TSP ring and construct the first trajectory $\overline{ABCD}$. **(c)** Remove edge $\overline{DE}$ and construct the second trajectory $\overline{EFG}$. **(d)** The three trajectories for three mobile sensors.

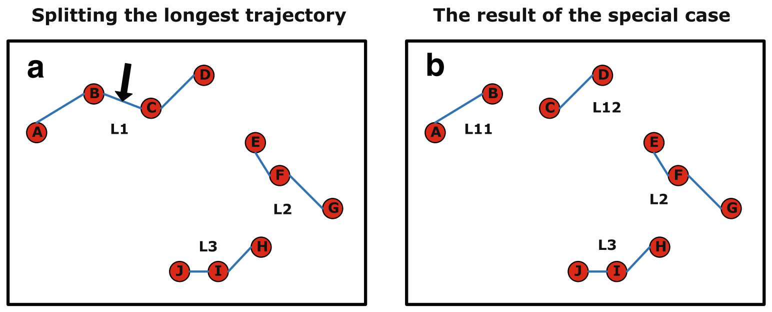
Splitting the longest trajectory
The result of the special case
a
b
A
B
C
D
E
F
G
H
I
J
L1
L11
L12
L2
L3

Figure 4 A special case of POIs coverage for TSP-S. (a) Remove the edge $\overline{BC}$ from the longest trajectory *L1*. **(b)** The longest trajectory *L1* is splitted into two trajectories *L11* and *L12*, so there are four trajectories now.

5.2 Modified stretched searching algorithm for the Minimum-Delay POIs Coverage problem without restriction

In this subsection, we modify the SSR algorithm to apply to the scenario that the start-positions of mobile sensors are not restricted. We propose an algorithm named *SSNOR* for this scenario.

In the following, we introduce the difference between the SSNOR and the SSR algorithms. In the SSR algorithm, the start locations of establishing trajectories are determined. In the SSNOR algorithm, there are only n POIs deployed in the region but no start positions of mobile sensors. We randomly select k POIs as the start locations of establishing k trajectories, we denote these k randomly selected POIs as $P_s = \{p_1^s, \ldots, p_k^s\}$ and set these POIs to covered.

After that, the rest operations are the same as that of in the SSR algorithm. We always choose the POI which minimizes the 2MDkT and add it to the trajectory. The procedure continues until all of the POIs have been added to the trajectories.

As shown in Figure 5, since the number of the mobile sensors is given, we generate two trajectories in the area. Apparently, the two mobile sensors can start sweeping at arbitrary positions in their trajectories which is more flexible in practical applications.

6 Performance evaluation

In this section, we will evaluate the performance of the proposed algorithms by extensive simulations. We test two important metrics: the delay and the max-min difference (2MDkT) of POIs Coverage schemes. The goal of the Minimum-Delay POIs Coverage problem is scheduling the given number of mobile sensors to monitor all POIs in the region and minimizing the delay. Since the length of each sensor's trajectory is various, the time of sweeping is different from each other. Although a mobile sensor finishes sweeping, it cannot start the next round of sweeping immediately until all sensors finish sweeping. The max-min difference affects the monitoring efficiency. In Figures 6 and 7, we discuss the performance of our algorithms for two scenarios, respectively.

6.1 Simulation setup

In our simulations, the POIs and mobile sensors are randomly deployed in a 3,000 m $\times$ 3,000 m square area. The moving velocity v of each mobile sensor is the same, which is 80 m/min. Let S denote the number of mobile sensors, P

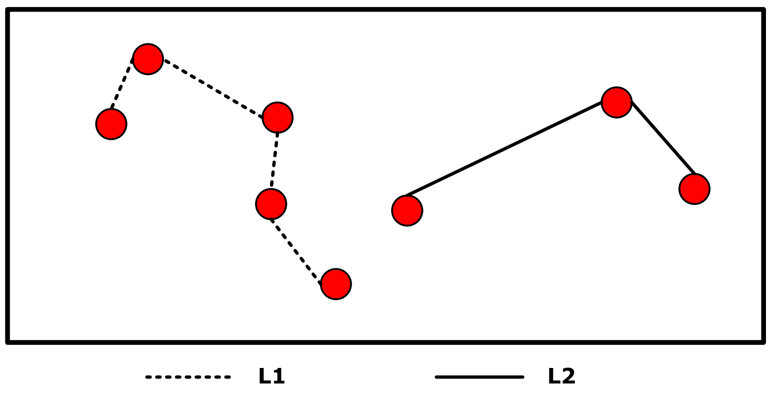

Figure 5 An example of SSNOR algorithm.

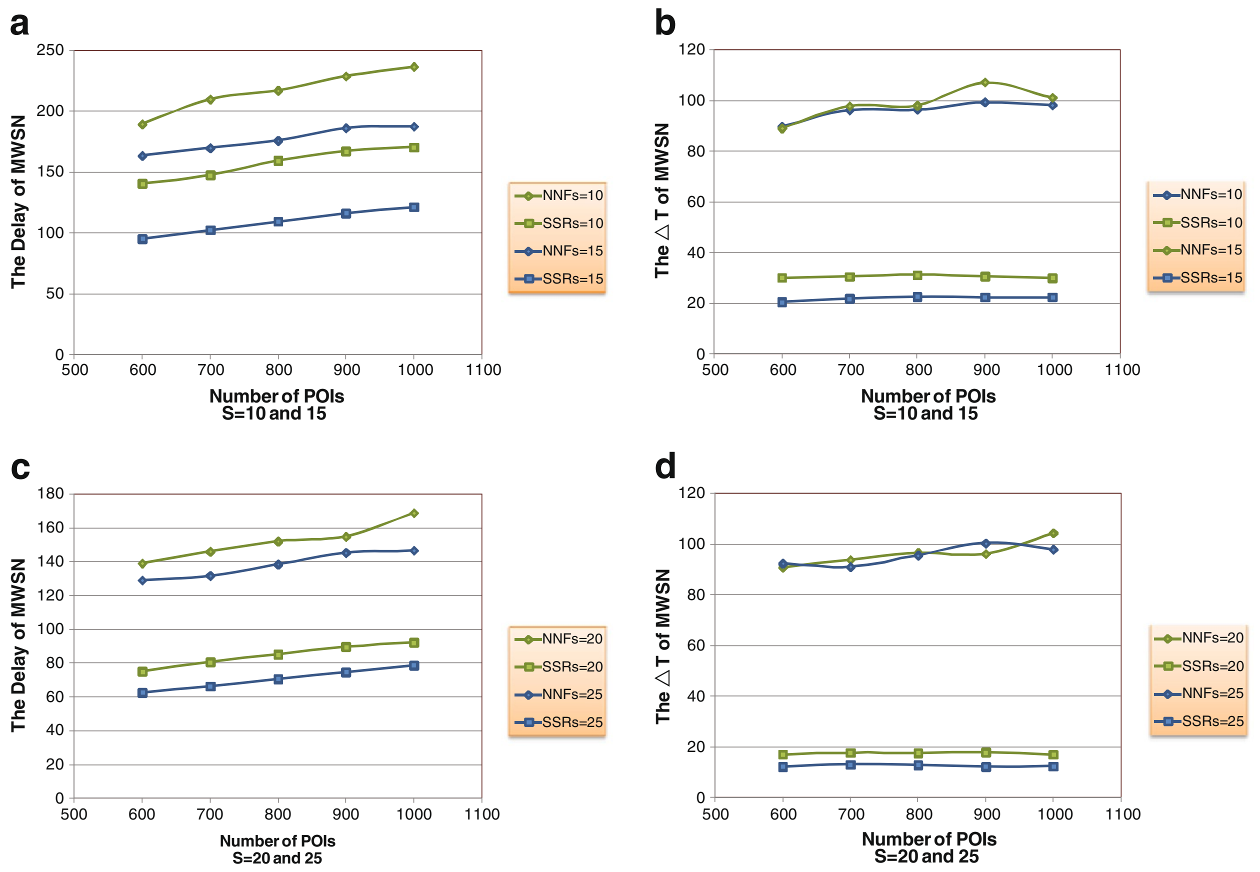

Figure 6 The performance for the scenario with determined start positions. (a) The delay of the NNF and SSR algorithms where $S = 10$ and 15. **(b)** The ΔT of the NNF and SSR algorithms where $S = 10$ and 15. **(c)** The delay of the NNF and SSR algorithms where $S = 20$ and 25. **(d)** The ΔT of the NNF and SSR algorithms where $S = 20$ and 25.

denote the number of POIs. The positions of all POIs are known in advance.

Since the moving velocity v of mobile sensors is the same, the 2MDkT of the minimum POIs coverage is equivalence to the ΔT, which is defined as the time interval from the first sensor accomplishing the sweeping to the last sensor finishing sweeping ($\Delta T = 2\text{MDkT}/v$). As far as we know, none of the previous works about POIs coverage considered the restriction on the number of mobile sensors and the start position restriction; therefore, we compare the SSR algorithm with a greedy algorithm called Nearest Neighbor First (*NNF*) algorithm. In the NNF algorithm, a nearest neighboring POI of each mobile sensor is always selected to add to the corresponding trajectory. In the second scenario, without the determined start positions, a Nearest Neighbor Prior algorithm (*UNNP*) is compared with TSP-S and SSNOR algorithms. Furthermore, we compare our TSP-S algorithm with the existing OSWEEP [2] algorithm.

For each simulation setting, we run 100 times and take the average value as the final results.

6.2 The performance of the SSR algorithm

To evaluate the performance of the SSR algorithm, we change the number of POIs and measure the delay and ΔT with the given number of mobile sensors. We set $S = 10$, 15, 20, and 25 in Figure 6a,b,c,d, respectively. Firstly, we can observe that increasing the number of mobile sensors can improve the performance of the SSR algorithm significantly. This is because the number of POIs assigned to each mobile sensor is decreased with more mobile sensors sweeping. However, the performance of NNF largely depends on the deployment of POIs, increasing the number of mobile sensors can not ensure the performance is improved.

Secondly, it is obvious that the performance of the SSR algorithm outperforms the NNF algorithm. It is because the SSR algorithm always guarantees that the difference between maximal and minimal trajectories is minimized when scheduling the sweep trajectory. In other words, the current delay of POIs coverage scheme in every step is also minimized. However, NNF only guarantees that the number of POIs swept by each mobile sensor is almost

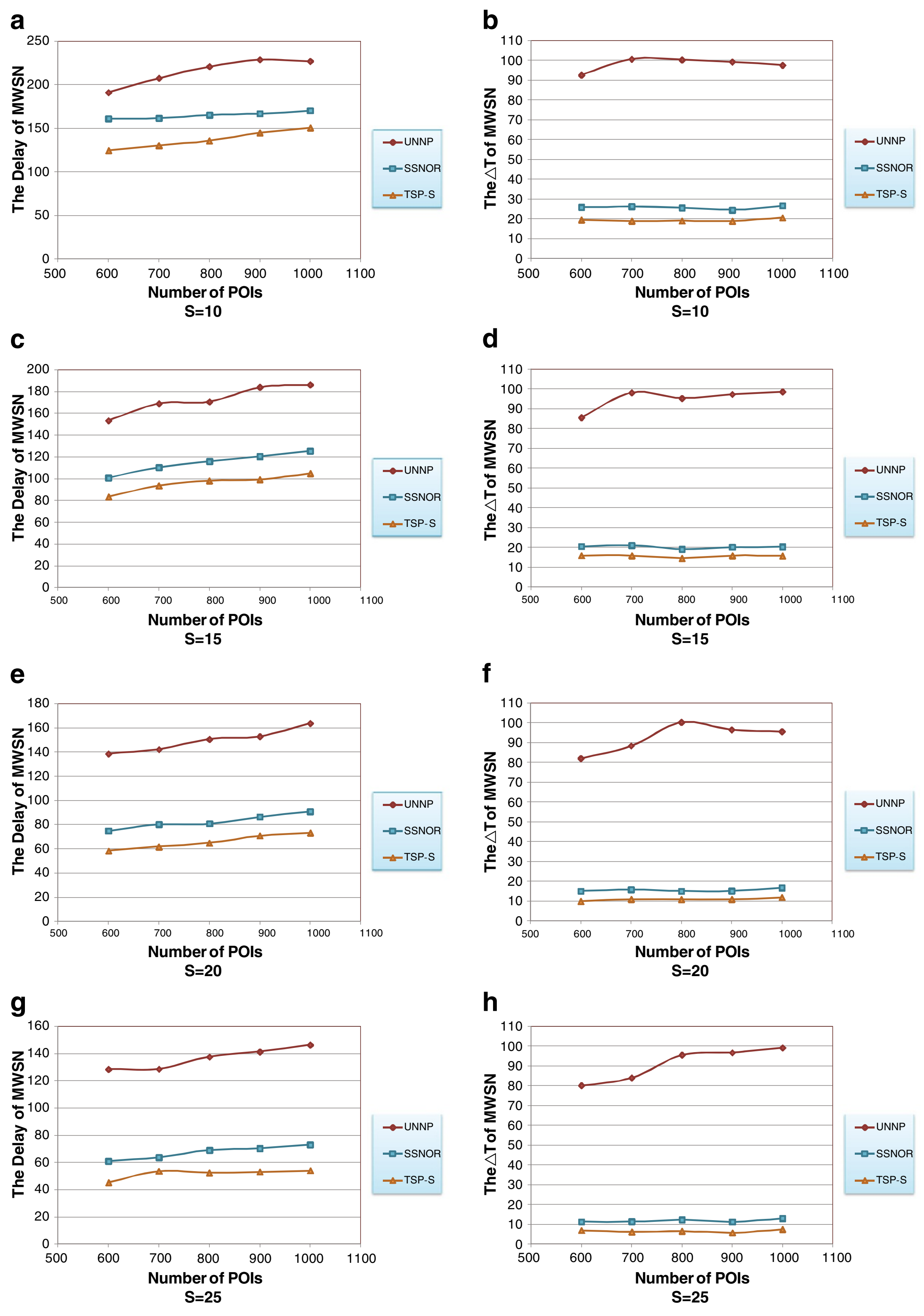

Figure 7 The performance for the scenario with undetermined start positions. (a) The delay of the UNNP, SSNOR, and TSP-S algorithms where $S = 10$. **(b)** The ΔT of the UNNP, SSNOR, and TSP-S algorithms where $S = 10$. **(c)** The delay of the UNNP, SSNOR, and TSP-S algorithms where $S = 15$. **(d)** The ΔT of the UNNP, SSNOR, and TSP-S algorithms where $S = 15$. **(e)** The delay of the UNNP, SSNOR, and TSP-S algorithms where $S = 20$. **(f)** The ΔT of the UNNP, SSNOR and TSP-S algorithm where $S = 20$. **(g)** The delay of the UNNP, SSNOR, and TSP-S algorithms where $S = 25$. **(h)** The ΔT of the UNNP, SSNOR, and TSP-S algorithms where $S = 25$.

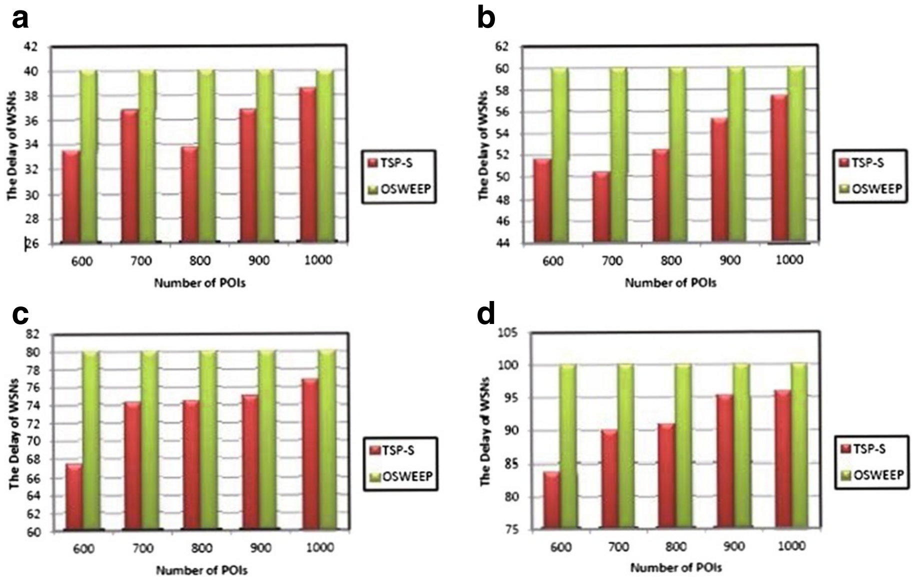

Figure 8 The comparison between the TSP-S algorithm and the OSWEEP algorithm. (a) The delay limitation of OSWEEP is set as 40 min. **(b)** The delay limitation of OSWEEP is set as 60 min. **(c)** The delay limitation of OSWEEP is set as 80 min. **(d)** The delay limitation of OSWEEP is set as 100 min.

equal, but it can not ensure that the length of sweeping trajectories is approximately equal.

Thirdly, since the size of region is fixed, when the number of POIs increasing, the length of each trajectory does not change too much. As shown in Figure 6, the fluctuation of the delay and ΔT is slight. Therefore, we can conclude that the delay and ΔT of the SSR algorithm change little when the number of POIs increasing.

6.3 The performance of the TSP-S algorithm and the SSNOR algorithm

We compare the delay and ΔT of TSP-S and SSNOR with UNNP. We employ 10, 15, 20, and 25 mobile sensors. As shown in Figure 7, it is obvious that the TSP-S and SSNOR outperform the UNNP. Furthermore, the TSP-S algorithm has better performance than the SSNOR algorithm. It is because SSNOR algorithm only considers local efficiency, while the TSP-S algorithm considers global efficiency.

We can also observe that with the increasing of *S*, the delay and ΔT of TSP-S and SSNOR decrease. When the number of POIs increases, the delay and ΔT of the TSP-S and SSNOR have slight change because the length of each trajectory changes a little.

6.4 The comparison between the TSP-S algorithm and the OSWEEP algorithm

We compare our TSP-S algorithm with the existing algorithm OSWEEP [2]. Since the OSWEEP addressed the Minimum Mobile sensor problem under the sweep delay limitation, which is different from ours, we first set the sweep delay as 40, 60, 80, and 100 min under the different number of POIs for OSWEEP, respectively, and obtain the corresponding minimum number of mobile nodes, named *S*, by the OSWEEP algorithm. Then we set the number of mobile nodes for our algorithm as *S* to com pare the sweep delay of the two algorithms. As shown in Figure 8, we can observe that the sweep delay of our TSP-S algorithm is always lower than the OSWEEP's. The delay difference of the two algorithms is 17% at most and 9% averagely. It is because the TSP-S algorithm remove the longer edge from the TSP ring when generating the trajectory of each mobile sensor which reduces the sweep distance and time.

7 Conclusion

In this paper, we study how to schedule the given number of mobile sensors to monitor the POIs such that the delay of the network is minimized. We discuss two scenarios. In the first scenario, with the determined start positions of mobile sensors, we propose the SSR algorithm. In the SSR, we generate trajectories by adding the POI which can make the current 2MDkT be minimized. In the second scenario, with the undetermined start positions, we propose TSP-S and SSNOR algorithms. In the TSP-S, we first input all POIs into the PTAS algorithm of TSP and obtain an approximate TSP ring. Then, we divide and adjust the TSP ring into a set of trajectories. The SSNOR algorithm is a modification of the SSR. We evaluate the performance of our algorithms by extensive simulations. The simulation results show that our proposed algorithms obviously outperform the NNF and UNNP algorithms. Meanwhile, the TSP-S algorithm performs better than the SSNOR algorithm.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
WPC and SC surveyed the literature and prepared the draft. DYL provided guidelines for the review. All authors read and approved the final manuscript.

Acknowledgements
This research was jointly supported in part by the National Natural Science Foundation of China under grants 61070191 and 91124001 and the Research Fund for the Doctoral Program of Higher Education of China under grant 20100004110001.

Received: 28 December 2012 Accepted: 22 October 2013
Published: 11 November 2013

References
1. M Li, W-F Cheng, K Liu, Y Liu, X-Y Li, X Liao, Sweep coverage with mobile sensors. IEEE Trans. Mob. Comput. **10**(11), 1534–1545 (2011)
2. J Du, Y Li, H Liu, K Sha, in *ICPADS*. On sweep coverage with minimum mobile sensors (Shanghai, 8-10 Dec 2010)
3. G Nicosia, Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, (1976)
4. M Cardei, D-Z Du, Improving wireless sensor network lifetime through power aware organization. Wireless Netw. **11**(3), 333–340 (2005)
5. M Cardei, MT Thai, Y Li, W Wu, in *INFOCOM*. Energy-efficient target coverage in wireless sensor networks (Miami, 13–17 March 2005)
6. H Liu, W Chen, H Ma, D Li, in *WASA*. Energy-efficient algorithm for the target q-coverage problem in wireless sensor networks (Beijing, 15–17 Aug 2010)
7. H Liu, P-J Wan, C-W Yi, X Jia, SAM Makki, N Pissinou, in *INFOCOM*. Maximal lifetime scheduling in sensor surveillance networks (Miami, 13–17 March 2005)
8. M Chaudhary, AK Pujari, in *ICDCN*. Q-coverage problem in wireless sensor networks (Hyderabad, 3–6 January 2009)
9. M Cardei, D MacCallum, MX Cheng, M Min, X Jia, D Li, D-Z Du, Wireless sensor networks with energy efficient organization. J. Interconnection Netw. **3**(3–4), 213–229 (2002)
10. S Slijepcevic, M Potkonjak, in *ICC*. Power efficient organization of wireless sensor networks (Helsinki, 11-14 June 2001)
11. X Bai, D Xuan, Z Yun, T-H Lai, W Jia, in *MobiHoc*. Complete optimal deployment patterns for full-coverage and k-connectivity ($k <= 6$) wireless sensor networks (Hong Kong, 27–30 May 2008)
12. S Kumar, T-H Lai, J Balogh, in *MOBICOM*. On k-coverage in a mostly sleeping sensor network (Philadelphia, 26 September - 1 October 2004)
13. X Wang, G Xing, Y Zhang, C Lu, R Pless, CD Gill, in *SenSys*. Integrated coverage and connectivity configuration in wireless sensor networks (Los Angeles, 5–7 November 2003)
14. B Liu, P Brass, O Dousse, P Nain, DF Towsley, in *MobiHoc*. Mobility improves coverage of sensor networks (Urbana-Champaign, 25–28 May 2005)
15. Z Zhou, SR Das, H Gupta, in *ICCCN*. Connected k-coverage problem in sensor networks (Chicago, 11–13 October 2004)
16. M Hefeeda, M Bagheri, in *INFOCOM*. Randomized k-coverage algorithms for dense sensor networks (Anchorage, 6–12 May 2007)
17. Y Wang, G Cao, in *INFOCOM*. On full-view coverage in camera sensor networks (Shanghai, 10–15 April 2011)
18. X Bai, S Kumar, D Xuan, Z Yun, T-H Lai, in *MobiHoc*. Deploying wireless sensors to achieve both coverage and connectivity (Florence, 22–25 May 2006)
19. S Kumar, T-H Lai, A Arora, in *MOBICOM*. Barrier coverage with wireless sensors (Cologne, 26 August - 2 September 2005)
20. A Chen, S Kumar, T-H Lai, in *MOBICOM*. Designing localized algorithms for barrier coverage (Montreal, 9–14 September 2007)
21. P Balister, B Bollobás, A Sarkar, S Kumar, in *MOBICOM*. Reliable density estimates for coverage and connectivity in thin strips of finite length (Montreal, 9–14 September 2007)
22. H Yang, D Li, Q Zhu, W Chen, Y Hong, in *WASA*. Minimum energy cost k-barrier coverage in wireless sensor networks (Beijing, 15–17 August 2010)
23. K-F Ssu, W-T Wang, F-K Wu, T-T Wu, K-barrier coverage with a directional sensing model. Smart Sensing Intell. Syst. **2**(1), 75–93 (2009)
24. A Saipulla, C Westphal, B Liu, J Wang, in *INFOCOM*. Barrier coverage of line-based deployed wireless sensor networks (Rio de Janeiro, 19–25 April 2009)
25. B Liu, O Dousse, J Wang, A Saipulla, in *MobiHoc*. Strong barrier coverage of wireless sensor networks (Hong Kong, 26–30 May 2008)
26. S He, J Chen, X Li, X Shen, Y Sun, in *INFOCOM*. Cost-effective barrier coverage by mobile sensor networks (Orlando, 25–30 March 2012)
27. W Wang, V Srinivasan, KC Chua, in *MOBICOM*. Trade-offs between mobility and density for coverage in wireless sensor networks (Montreal, 9–14 September 2007)
28. D Wang, J Liu, Q Zhang, in *IEEE IWQoS*. Probabilistic field coverage using a hybrid network of static and mobile sensors (Evanston, 21–22 June 2007)
29. E Ekici, Y Gu, D Bozdag, Mobility-based communication in wireless sensor networks. IEEE Commun. Mag. **44**(7), 56–62 (2006)
30. S Chellappan, W Gu, X Bai, D Xuan, B Ma, K Zhang, Deploying wireless sensor networks under limited mobility constraints. IEEE Trans. Mob. Comput. **6**(10), 1142–1157 (2007)
31. S Arora, in *FOCS*. Polynomial time approximation schemes for Euclidean TSP and other geometric problems (Burlington, 14–16 October 1996)

doi:10.1186/1687-1499-2013-262
Cite this article as: Chen *et al.*: **Minimum-Delay POIs Coverage in mobile wireless sensor networks.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:262.