Peer-to-Peer Netw. Appl. (2017) 10:1063–1078
DOI 10.1007/s12083-016-0469-9

Minimizing mobile sensor movements to form a line K-coverage

Yang Wang[1] · Shuang Wu[1] · Xiaofeng Gao[1] · Fan Wu[1] · Guihai Chen[1]

Received: 30 September 2015 / Accepted: 16 May 2016 / Published online: 10 June 2016

Abstract In wireless sensor networks and social networks, distributed nodes usually form a network with coverage ability for a lot of applications, such as the intrusion detection. In this paper, a new kind of coverage problem with mobile sensors is addressed, named Line K-Coverage. It guarantees that any intruder trajectory line cutting across a region of interest will be detected by at least K sensors. For energy efficiency, we aim to schedule an efficient sensor movement to satisfy the line K-coverage while minimizing the total sensor movements, which is named as LK-MinMovs problem. We firstly construct two time-efficient heuristics named LK-KM and LK-KM+ based on the famous Hungarian algorithm. By sacrificing optimality a little bit, these two algorithms have better time efficiency. Then we propose a pioneering layer-based algorithm LLK-MinMovs to solve LK-MinMovs in polynomial time. Here, we assume that all sensors are initially located in a closed region. We validate its correctness by theoretical analysis. Later, the more general situation are considered that all sensors are allowed to locate outside of the region. We improve LLK-MinMovs algorithm to the general version: GenLLK-MinMovs. More importantly, our GenLLK-MinMovs fixes a critical flaw for MinSum algorithm which was proposed by previous literature to solve line 1-coverage problem. We show the flaw using a counter example. Finally, we validate the efficiency of all our designs by numerical experiments and compare them under different experiment settings.

Keywords Line K-coverage · Efficient movement schemes · Optimization algorithm · Mobile sensor network

1 Introduction

Wireless Sensor Network (MWSN) and Social Network (SN) nowadays attract special attentions from scientific and technological community. Coverage is a fundamental problem among all challenges of MWSN [5, 6] and SN [7, 8]. Broadly speaking, coverage is a measure that determines how well a network monitors objectives. Many variations of coverage problem have been proposed for different applications [9]. As an example, the basic area K-coverage problem [10] requires that each point in the area should be covered by at least K sensors.

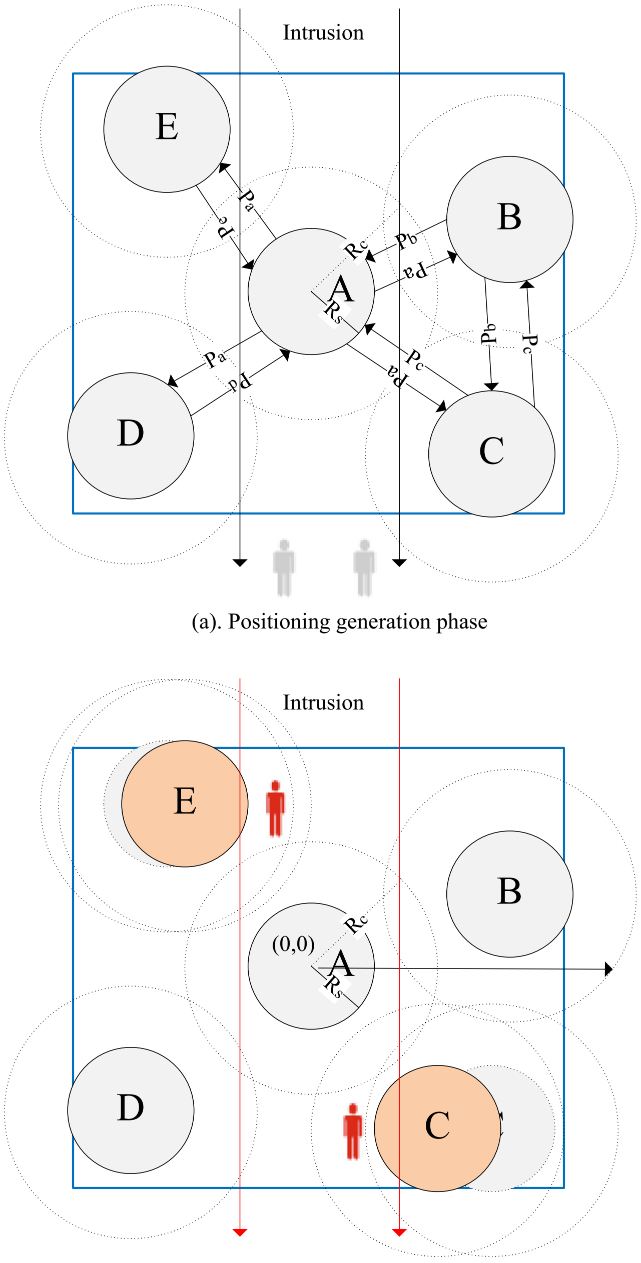
In MWSNs and SNs, distributed nodes usually form a network with coverage ability for a lot of applications, such

This work was supported in part by the State Key Development Program for Basic Research of China (973 project 2012CB316201), in part by the Opening Project of Key Lab of Information Network Security of Ministry of Public Security (The Third Research Institute of Ministry of Public Security) Grant number C15602, the Opening Project of Baidu (Grant number 181515P005267), China NSF grant 61422208, 61472252, 61272443 and 61133006, Shanghai Science and Technology fund 15220721300, and in part by CCF-Tencent Open Fund. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

✉ Xiaofeng Gao
gao-xf@cs.sjtu.edu.cn

[1] Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

as the intrusion detection. How does the network knows that an intruder or a new node enters its region? First, we should locate all nodes to structure the entire network. It is common to borrow Global Position System (GPS) or Local Position Algorithm (LPA) to finish the positioning duty (GPS-free), which is studied in many literatures [4]. Especially, the local position algorithm requires that distributed nodes communicate with neighbors to calculate the local positions for their neighbors. Then nodes communicate the relative positions information to implement the global coordinate generation. Second, according to the global coordinates of nodes, some nodes are scheduled to move for repairing all loopholes and forming a barrier in the region which can detect new intruders. Figure 1 shows the scenario for the two procedures. In Fig. 1a, all sensors have the same communication range radius R_r and the same sensing range radius R_s. Every sensor sends neighbors its position system which takes it as the original point (0,0) and has relative positions of neighbors. Actually, some sensors cannot produce position due to the lack of neighbors, such as sensor D and sensor E. Then, a head sensor is selected (sensor A) as the global original point and horizontal direction is X-axis. Some intrusions along with vertical straight line can penetrate the region. In Fig. 1b, the sensors near the loopholes move a shortest distance to form a barrier according to the global position system, which guarantees that every vertical intrusion will be detected by at least one sensor. The example shows that two intruders are detected by E and C after they move to repair loopholes. If some sensors fail to work due to damage or exhausted power, the barrier is broken. Then, the two-step procedure should be done again locally near the sensors.



(a). Positioning generation phase

(b). Forming barrier phase

Fig. 1 Illustration for two phase barrier formation

Barrier coverage is more applicable to monitor borders because it exploits less sensors than area coverage. In front of or surrounding an area, a barrier is a belt-like region in which the sensors are scattered. The barrier is said to be K-covered [11, 12] if every path that passes through the barrier touches the sensing range of at least K sensors. Many researchers considered line track rather than arbitrary paths for barrier coverage [13–16], since in reality intruders usually go through a region with a line track. Moreover, intruders usually do not know any knowledge on the distribution of sensors so they cannot figure out a "smart" path to follow. In [17], the authors proposed an algorithm named MinSum to build a barrier by scheduling mobile sensors, so that any line intrusion will be detected by at least one sensor. We refer this problem as Line 1-Coverage problem.

In this paper, we consider an advance version of line 1-coverage problem: Line K-Coverage. A line is said to be K-covered if it is detected by at least K sensors. A region is called line K-covered if any line intrusion is K-covered. Usually, sensors at their initial positions may not form a line K-cover for the target region. Thus mobile sensors could move according to some strategy to form a line K-cover. For energy efficiency purpose, we hope that sensors will move with a shortest distance. In all, our optimization object is to minimize the sum of sensor movements to achieve the line K-cover for target region. We refer it as LK-MinMovs problem. For solving the problem, we design several solutions: Kuhn-Munkres (KM) based algorithms and layer-based algorithms.

Firstly, we design two algorithms for LK-MinMovs problem based on the famous Hungarian algorithm. The basic idea is to place the sensors to several fixed points (positions) evenly distributed. We construct a group of sensors and a group of fixed points as the two parties of a bipartite graph. We are aiming to match the two parties to satisfy the line K-coverage requirement. As a kind of Hungarian algorithm, KM algorithm can solve maximum matching problem in bipartite graph. However, we cannot use it directly in our minimization problem. We should transform it to LK-KM algorithm to calculate the minimization solution. Because LK-KM produces bad results in some cases, we improve it and get a new algorithm LK-KM+. Its idea will be explained later. Because both LK-KM and LK-KM+ are not optimal algorithms we take them as baselines. However, they have better time efficiency in spite of sacrificing the optimality a little bit.

Later, we consider a layer-based and optimal algorithm. If the initial sensors deployment does not form a line K-cover, the target region will have “gaps” (K-uncovered intervals) against the intrusion. Due to the structural complexity, it is not easy to form line K-cover in one shot. A natural idea is to build line K-cover layer by layer because gaps have different degrees. We first fill up 1-level gaps by twofold overlaps, and then fill up 2-level gaps by threefold overlaps, until K-level gaps are filled. With this idea we propose a layer-based algorithm named LLK-MinMovs. To the best of our knowledge, we firstly solve the line K-coverage problem in mobile sensor networks, which has both theoretical and practical significance.

About sensor initial deployment there can be two assumptions: closed and open assumptions. In closed assumption, we limit sensor initial deployment into the target region strictly. It is suitable to describe applications like gate or channel barriers. However, for more general application scenarios such as important area protection, the open initial deployment is more practical because it is assumed that all sensors can locate outside the target region. LLK-MinMovs algorithm is proposed under closed assumption while we improve it into the general version: GenLLK-MinMovs under open assumption.

For one layer repairing, although the authors in [17] claimed that MinSum outputs the optimal solution, it is not always correct. We illustrate the critical flaw of MinSum by a counter example, and fix the problem in our LLK-MinMovs algorithm. We also construct another two time-efficient heuristics named LK-KM and LK-KM+ based on the famous Hungarian algorithm. By sacrificing optimality a little bit, these two algorithms runs extremely fast with suboptimal results. We analyze their time complexity, and then validate their efficiency in numerical experiments under different experiment settings.

The rest of the paper is organized as follows. Section 2 introduces some related works. Section 3 presents the problem statement. In Section 4, we design LK-KM and its enhanced version LK-KM+ as the baselines. Section 5 describes our layer-based algorithm (LLK-MinMovs) and gives its optimality proof and time complexity analysis. Section 6 generalizes the LK-MinMovs problem in terms of sensor open initial deployment and gives a corresponding algorithm GenLLK-MinMovs. A counter example in general situation against MinSum is also introduced and corrected. Numerical experiments are presented in Section 7. Finally, Section 8 gives a conclusion.

2 Related works

Coverage in WSNs can be calculated by a central sink, which knows the location of the sensors in the field. This is called global coverage measurement. In contrast, problem of local coverage measurement is to assess the coverage around one sensor, based on its geometrical relation to its neighbors. This problem has been addressed in many articles on wireless sensor networks. References [1, 3] proposed GPS-free scheduling when the only available information is the number of neighbors, and the sensor uses a probabilistic algorithm for its scheduling decisions. In some scenarios, a sensor measures the distance to each neighbor using the received signal strength (RSS), the time of arrival (ToA), the time difference of arrival (TDoA), and even measures the direction of the neighbor based on the angle of arrival (AoA) capability. Reference [2] used ToA and AoA to discover the location of neighbors according to each sensor and used the Voronoi diagram to detect sensors which could be powered off without an impact on the coverage value. In this paper, we consider the global coverage measurement which can also be viewed as relative global measurement in local measurement by clustering with several head nodes as sinks.

In the research of mobile sensor networks, several recent papers considered the strategy of mobile sensor movement to cover a region of interest, for example [18–20]. Unlike the problem considered in this paper, they aimed to form an area coverage rather than barrier coverage for the region of interest. The problem studied in this paper addresses the problem of ensuring efficient intruder detection without covering the entire region of interest. The difference between area coverage and barrier can be illustrated in Fig. 1b. It forms barrier to detect vertical intrusion while it cannot detect all events occurring in the region everywhere. Besides, people-centric sensing with smart devices (such as smart phones) is a new trend for social network. Authors in [21] considered a problem which is to schedule a minimal number of moving persons to cover area of interest based on an artificial map.

Some of the existing works focus on line coverage [22] in a region. Baumgartner et al. [14] proposed the track coverage problem. Their objective is to place a set of sensors in the region such that the chance of detecting the given tracks by at least some given number of sensors is maximized. Path coverage is a measure used for tracking moving objects in a straight line path. Other path coverage metrics are defined in [15, 16] by analytical expressions for any random deployment in a region. Balister et al. [13] defined a coverage metric called trap coverage. It measures the longest distance an intruder can achieve within the region before touching the sensing range of any sensor. They are essentially more like a partial area coverage with restrictions. From objective angle, obviously researchers want to cover line-track intrusion from any direction [9]. Authors in [9] proposed measures for line coverage problem. Therefore, we also name the problem as Line Coverage from objective angle.

In terms of mobile sensor for barrier, distributed algorithms are proposed in [23] to find the new positions for sensors to form a barrier, when sensors are initially located at arbitrary positions and can move along the barrier. Bar-Noy et al. [24] studied the problem of maximizing the coverage lifetime of a barrier by mobile sensors with limited battery powers. How to exploit sensor movement to improve the quality of barrier coverage are studied in [25]. All of them consider barrier coverage for path, which is not the objective of this paper. Thus, we focus on line coverage.

For K-fold barrier coverage, in [26], authors proposed a similar problem: weak barrier coverage which is close to our work. However, they just considered random sensor deployment and devised a relationship between number of sensors and probability of weak K barrier coverage. Authors in [27] also considered how to maximize the number of separate barriers while they discussed a strong barrier problem with sensors with directional fan-shape range. The similar objective is discussed in [28], however, authors mainly studied the relationship between the maximum number of barriers and the number of sensors as well as movement range of sensors.

Czyzowicz's work [17] is the most related to our objective. The authors proposed MinSum algorithm for line 1-coverage problem which inspired us to design the LLK-MinMovs algorithm. However, their algorithm cannot always output an optimal solution for any instance. We provide a counter example for MinSum and correct it in our design.

3 Problem statement

As related works section mentioned, we exploit global coverage measurement which assumes there is a sink takes charge of communicating with all other sensors and scheduling mobile sensors to form line K-coverage. The energy consumption for communication is much less than the energy consumption for moving. Therefore, we ignore communication energy and focus on minimizing the movements. In the following, we give notations for the problem statement.

Assume the region of interest is a rectangle with horizontal length of L (otherwise we can use the minimum bounding rectangle of this region to abstract). n sensors $s_1, s_2, \ldots, s_n$ are randomly deployed in this region. Each sensor s_i has coordinator (x_i, y_i), regarding to left-bottom point $(0, 0)$, with the same sensing range R. Commonly, R is much less than L, thus, $R << L$. Sensors can move freely in this region, but they are supported by nonrenewable battery powers.

Figure 2 is an example scenario (Assume $K = 3$), where the intrusion direction is vertical against the rectangle. Hence, we could project s_i into a horizontal line segment represented by interval $[x_i - R, x_i + R]$. Then, we just need to consider our problem on line segment $[0, L]$ after the whole region is projected too. To better describe our problem, we label sensors with their x-coordinate, and assume that each sensor has distinct x_i value in increasing order. Note that we should have "enough" sensors to satisfy a line K-coverage. Thus, initially we have at least n sensors. The number of sensors n lets $2Rn \geq KL$ be satisfied.

Easy to see, if we want to form a line K-coverage, every point along the x axis in $[0, L]$ should belong to as least K intervals transformed from sensors. However, as shown in

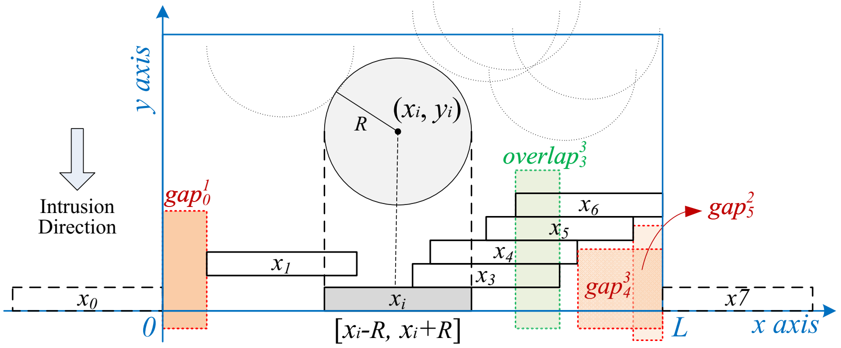


Fig. 2 An example of line coverage transformation ($K = 3$)

Fig. 2, there are many intervals on x axis that are covered by less than K sensors, i.e. the interval $[x_4 + R, L]$. We consider such intervals as gaps. Thus, to form a line K-cover is equivalent as to fill up the gaps along x axis after sensor projection. Note that gaps may have different degrees. Some gaps are already covered by several sensors (but less than K), while some other gaps are even bare. To describe a gap rigorously, we have the following definition.

Definition 1 (Line k-Covered Gap) A line k-covered gap, denoted as gap_i^k, is an interval which starts from the ending point of sensor x_i and ends up to the starting point of sensor x_{i+k}, say the interval $[x_i + R, x_{i+k} - R]$, where $x_{i+k} - x_i > 2R$.

Easy to see, gap_i^k is covered by less than k sensors. Three example gaps are shown in Fig. 2, which are line 1-covered gap_0^1, line 2-covered gap_5^2 and line 3-covered gap_4^3. We add two virtual sensors adhesively to the starting point and ending point of the target interval. In this example, left virtual sensor s_0 locates at $x_0 = -R$ and right virtual sensor s_7 locates at $x_7 = L + R$. To illustrate our design, we have another definition for overlap intervals as follow.

Definition 2 (Line k-Covered Overlap) A line k-covered overlap, denoted as $overlap_i^k$, is an interval which starts from the starting point of sensor x_{i+k} and ends up to the ending point of sensor x_i, say the interval $[x_{i+k} - R, x_i + R]$, where $x_{i+k} - x_i < 2R$.

Similarly, $overlap_i^k$ is covered by $k + 1$ sensors at least. Thus, some sensors could move to cover other gaps. An example line 3-covered $overlap_3^3$ is shown in Fig. 2, which is covered by sensors x_3, x_4, x_5, and x_6 respectively.

We will move some sensors to fill up all gaps in $[0, L]$. Define the final position of sensor s_i as x_i^f. Then the moving distance d_i of s_i is $|x_i^f - x_i|$. The LK-MinMovs problem is to find the final position for n sensors $s_1, s_2, \cdots, s_n$, so that these sensors will form a line K-coverage while the total sensor movements $\sum_{i=1}^{n} |x_i^f - x_i|$ is minimized. We require sensor movement schedule to obey order preservation restriction given by a Lemma in [17] and sensors cannot move out of $[0, L]$. In all, we formalize the LK-MinMovs problem as the following linear programming:

$$\min \quad \sum_{i=1}^{n} |x_i^f - x_i| \tag{1}$$

$$s.t. \quad 0 \le x_i \le x_j \le L, \quad \forall\, 1 \le i \le j \le N \tag{2}$$

$$0 \le x_i^f \le x_j^f \le L, \quad \forall\, 1 \le i \le j \le N \tag{3}$$

$$x_{i+k}^f - x_i^f \le 2R, \quad \begin{array}{l} \forall\, 1 \le k \le K, \\ \forall\, 0 \le i \le n - k + 1 \end{array} \tag{4}$$

Here the expression (1) describes the optimizing objective, Inequation (2) restricts the initial positions of sensor deployment, which is closed in interval [0, L]. Meanwhile, Inequations (2) and (3) together guarantee an important property: Order Preservation, which is described in Lemma 1. Obviously, Inequation (4) expresses the Line K-Coverage requirement for the whole region interval.

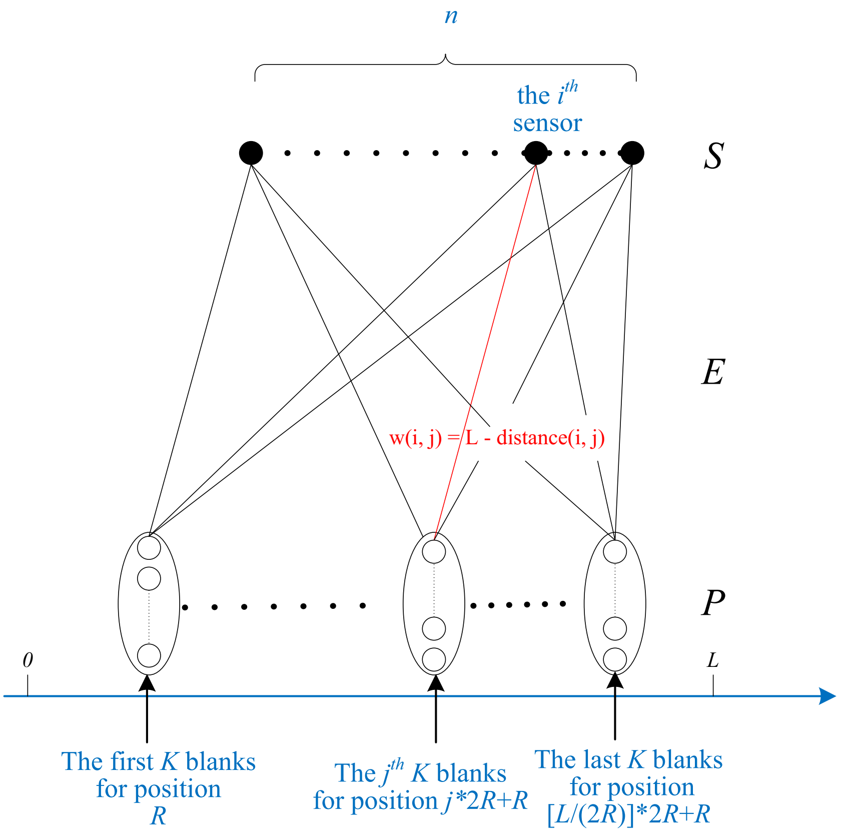
In the following sections, we will introduce three algorithms to solve the LK-MinMovs problem.

4 Two baselines: LK-KM and LK-KM+ algorithms

We designed two algorithms for LK-MinMovs problem based on a famous Hungarian Algorithm: Kuhn-Munkres (KM) algorithm. The two baselines have better time efficiency in spite of sacrificing the optimality a little bit. Therefore, their results are not optimal. Several researchers have designed similar algorithm based on Hungarian algorithm for coverage problem. Authors in [29] designed efficient scheduling algorithms to maximize the lifetime of a given whole wireless sensor network by considering adjusting sensing range, locations of target and sensors, the residue battery power of sensor nodes, and assignment between sensors and targets simultaneously. In comparison with target coverage [29], authors in [27] also used Hungarian Algorithm to solve its Minimum Total Cost Sensor Movement problem. In comparison with us, they considered how to maximize the number of separate barriers while they discussed a strong barrier problem by sensors with directional fan-shape sensing range.

Our basic idea is to place the sensors to several fixed points (positions) evenly distributed on the line segment $[0, L]$. Obviously, it is a perfect Line K-coverage to put K sensors at each virtual fixed points (positions) with $2R$ distance between two neighbors. Then we construct a complete sensor-position bipartite graph $G(S, P, E)$, see Fig. 3. S represents the sensor set and P represents the virtual fixed point set. Each virtual fixed point has K copies for K-cover. Then, we should transform our minimization problem to the Maximal Matching problem for P. We let the edge weight $w(i, j)$ be set as $L - distance(i, j)$, where $i \in S$ and $j \in P$, then the maximum matching for P is the solution of original LK-MinMovs problem. We can use the KM algorithm (also known as the famous *Hungarian* algorithm) [30, 31] to compute this matching. Our algorithm is referred as LK-KM algorithm shown in Algorithm 1. The inputs are an array $X[1 \ldots n]$ representing the initial positions of n sensors, the length of the region, L, the coverage degree, K, and the sensor radius, R. It returns an array $X^f[1 \ldots n]$ of n elements representing the final positions of sensors.

Fig. 3 Illustration for sensor-position bipartite graph $G(S, P, E)$


n
the i^{th} sensor
S
E
w(i, j) = L - distance(i, j)
P
0
L
The first K blanks for position R
The j^{th} K blanks for position j*2R+R
The last K blanks for position [L/(2R)]*2R+R

```
Algorithm 1 LK-KM
  Input: $X[1 \ldots n]$, $L$, $K$, $R$
  Output: The final positions $X^f[1 \ldots n]$ of $n$ sensors
  Let $S = \{s_1, s_2, \ldots, s_n\}$ and
    $P = \{d_1, d_2, \ldots, d_{K \cdot \lceil L/(2R) \rceil}\}$ where $d_j$ represents the
    point in $R + (i \mod K) \cdot R$
  Let $w_{i,j} = L - dis(s_i, v_i)$;
  Using the KM algorithm to compute a matching for
    $(S, P, E)$;
  for each $e_{i,j}$ in the matching do
      $X^f[i] \leftarrow R + (\lceil j/K \rceil - 1) \cdot 2R$;
  return $X^f[1 \ldots n]$
```

However, the LK-KM algorithm produces bad solutions for many cases. Let us see an extreme example in Fig. 4a. The sensor initial deployment already forms line K-coverage. If we run LK-KM algorithm on this case, LK-KM will force all sensors to move to the preset positions like result Fig. 4b shows. Obviously, these movements are not necessary. We can find the surplus movements pushed by LK-KM for almost all inputs if there exists a lot of sensor overlap redundancy. Intuitively, we try to improve the LK-KM algorithm with an idea to pull back the sensors which do not need to go so far away from their original positions because the sensor redundancy provides the chance. The line K-coverage enhanced KM algorithm, denoted as LK-KM+, firstly sorts the terminal positions to keep the Order Preservation property after calling the original LK-KM algorithm. Then, LK-KM+ uses $PullToLeft(\cdot)$ and $PullToRight(\cdot)$ to shorten the distance between terminal and original positions of sensors. The two subroutines both use function $move(\cdot)$ for movement of each sensor back to its original position. We give an example for calling $move(\cdot)$ in $PullToLeft$ on sensor x_i which has moved to right and need to be pulled left back, thus is $move(i, i, \min\{X[i] - X^f[i], X^f[i-1] - X^f[i], X^f[i+K] - X^f[i] - 2R\})$. It move x_i to left in a distance which is shortest one among its shift distance, distance between it and its previous neighbor, and distance between it and its K-hops later neighbor (they should form line K-cover together). The corresponding procedure is implemented in $PullToRight$. For space limited, the pseudo-code of $PullToLeft(\cdot)$ and $PullToRight(\cdot)$ is not extended.

```
Algorithm 2 LK-KM+
  Input: $X[1 \ldots n]$, $L$, $K$, $R$
  Output: The final positions $X^f[1 \ldots n]$ of $n$ sensors
  $X^f \leftarrow LK - KM(X, L, R, K)$; /*get a feasible
    solution using previous LK-KM Algorithm*/
  $sort(X^f)$;
  $PullToLeft(X^f, X)$;
  $PullToRight(X^f, X)$;
  return $X^f[1 \ldots n]$
```

In the original KM algorithm, the time complexity is $O(M^2N)$ where M is the number of target nodes for matching, and N is the number of source nodes. For LK-KM algorithm, the time complexity is $O((\frac{KL}{2R})^2n)$. It is deducted by letting $M = \frac{KL}{2R}$ and $N = n$. And for LK-KM+ algorithm, in spite of adding $move(\cdot)$ steps using only $O(n)$ time, the total complexity of LK-KM+ is also $O((\frac{KL}{2R})^2n)$. They are both better than the later LLK-MinMovs in time complexity. However, they are not optimal.

5 A layer-based algorithm for LK-MinMovs problem

In this section, we propose an algorithm for LK-MinMovs. The algorithm uses two concepts: Line k-covered gap and line k-covered overlap defined in Section 3. Intuitively, we want to reduce the number of uncovered gap with the help of the "oversubscribed" overlaps, i.e. the $overlap_3^3$ in Fig. 2, which is covered by sensors x_3, x_4, x_5, and x_6 respectively. We can schedule one redundant sensor in the overlap to cover other unsatisfied gaps. In our design, we plan to fill up the gaps in an ascending order of their coverage degree. Correspondingly, we fill up line 1-covered gaps first, then line 2-covered gaps, and so forth until filling the line K-covered gaps. It is a reasonable operation consequence because we want to heal the worst conditions then the second worst ones in real application scenarios. This is why we call it layer-based algorithm.

5.1 LLK-MinMovs algorithm

In this layer-based algorithm, named as LLK-MinMovs, we try to find the two closest overlaps and select a cheaper one to fill up a target gap. Note that such movement process should maintain current coverage level. That means the algorithm will not bring in new gaps or degrade the current coverage quality. LLK-MinMovs solves the problem layer by layer.

Algorithm 3 is the pseudo-code of LLK-MinMovs. Input and output are similar to that of Algorithm 1. In Algorithm 3, Line 1 sets two virtual stable sensors to bound the region. Line 2 depicts the layer-based procedure. At each level $k \in [1, K]$, Algorithm 3 finds all line k-covered gaps and fills up them in a left-right order. The function $isCovered(i, k)$ in Line 4 is a binary function to determine whether the interval $[x_i + R, x_{i+k} - R]$ is line k-covered at the current stage. If there exists a gap_i^k, we find its left and right closest overlaps as potential candidates, see line 5–6, then pick up cheaper one according to cost functions $Lcost(\cdot)$ and $Rcost(\cdot)$ and move corresponding sensors to fill up gap_i^k according a distance constraint function $Ldist(\cdot)$ and $Rdist(\cdot)$ to keep the current coverage level, see line 8–9. The "while" loop from Line 4 to 9 guarantees that we will fill up all k-covered gaps generated by each x_i.

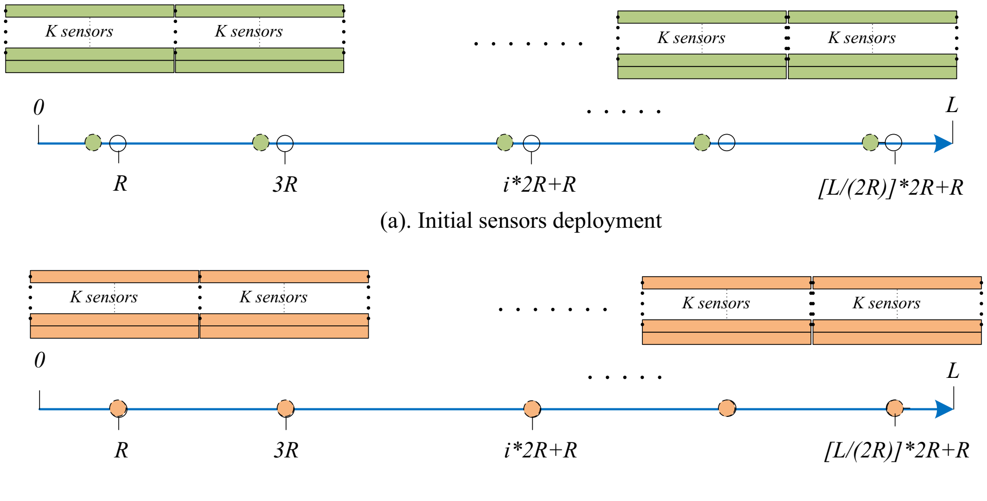
Algorithm 3 LLK-MinMovs

```
Input: $X[1 \dots n]$, $L$, $K$, $R$
Output: The final positions $X^f[1 \dots n]$ of $n$ sensors
1  $X[0] = -R$, $X[n+1] = L + R$;
2  for $k \leftarrow 1$ to $K$ do
3      for $i \leftarrow 0$ to $n$ do
4          while $isCovered(i, k) = 0$ do
5              $l = find(gap_i^k, \text{left})$ ;    // find the left closest $overlap_l^k$
6              $r = find(gap_i^k, \text{right})$ ;    // find the right closest $overlap_r^k$
7              if $Lcost(i, l, k) \leq Rcost(i, r, k)$ then
8                  $move(l, i, Ldist(l, i, k))$ ;  // fill gap by its left overlap
9              else $move(i, r, -Rdist(i, r, k))$ ; // fill gap by its right overlap
10 return $X^f[1 \dots n]$;
```

At the initialization stage, LLK-MinMovs sets two virtual static sensors by setting $P[0] = -R$ and $P[n + 1] = L + R$. They cannot move in later steps which guarantees the closure of algorithm. Then algorithm repeats for different level Line k-coverage, k from 1 to K, which shows our layer-based idea. If there is a gap_i^k at the end of sensor x_i, $isCovered(i, k) = 0$, otherwise $isCovered(i, k) = 1$. By scanning all the sensors, we can find every k-level gap and fill them completely

In the most inner part of the algorithm, we find available overlaps in an in-out order for each k-level gap. We can find two nearest k-level overlaps, separatively to the left of the i^{th} sensor (include itself) and to the right of $i + k^{th}$ sensor (include itself). Let the left one formed by the l^{th} and $(l + k)^{th}$ sensors and right one formed by the r^{th} and $(r + k)^{th}$ sensors. The function $find()$ will calculate three values for left (right) overlap: l (r), its left (right) overlap cost l_{cost} (r_{cost}), and left maximum shift l_{max} (right maximum shift r_{max}). It should be noticed that the value l (r) and k satisfies $mod(i-(l+k), k) = 0$ ($mod(r-(i+k), k) = 0$) for each i. Because it guarantees the overlap sensor is at the same level of gap bound sensor for the k-level coverage. And if there is no left (right) overlap, set $l = 0, l_{cost} = \infty, l_{max} = 0$ ($r = n + 1, r_{cost} = \infty, r_{max} = 0$).

Then we will choose the cheaper one from them according to l_{cost} and r_{cost}. Here, we define the effect shift window as the smallest left (right) shift of ones of $(l + k)^{th}, (l + 2k)^{th}, ..., (l + mk)^{th}, ..., i^{th}$ ($(r)^{th}, (r - k)^{th}, ..., (r - mk)^{th}, ..., (i + k)^{th}$) sensors for left (right) overlap. The value l_{max} is equal to the minimal one among effect shift window, size of gap, and size of left overlap. The value r_{max} is equal to the minimal one among effect shift window, size of gap and size of right overlap. The



(a). Initial sensors deployment

(b). Sensors deployment after LK-KM calculation

Fig. 4 A case shows disadvantage of LK-KM

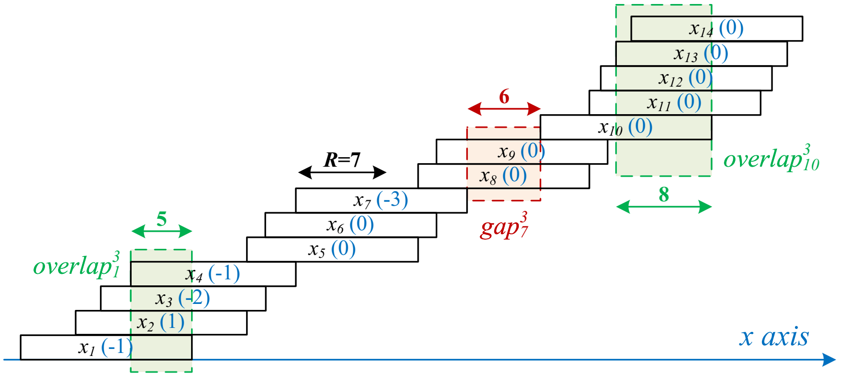
effect shift window is the smallest left (right) shift of ones of $(l+k)^{th}$, $(l+2k)^{th}$, ..., $(l+mk)^{th}$, ..., i^{th} ($(r)^{th}$, $(r-k)^{th}$, ..., $(r-mk)^{th}$, ..., $(i+k)^{th}$) sensors for left (right) overlap. If there is no left (right) shift among these sensors for left (right) overlap, effect shift window is equal to ∞. Later, we will give an example to illustrate it. Then the function $move()$ will takes the corresponding sensors to move to right (to left) a l_{max} (r_{max}) distance. The $move()$ operation also shows our layer-based idea. Because the overlap cannot "fly" to the gap, sensors "relay-like" move to fill gap from the $(l+k)^{th}$ to i^{th} sensor (resp. from r^{th} to $(i+k)^{th}$ if the right overlap is chosen). Additionally, for k-level coverage, just right (left) movements of sensors at the positions separated a multiply of $k-1$ from i^{th} ($(i+k)^{th}$) are helpful for filling the current gap. These sensors are called "involved" and we denote left (right) ones as $InvLeftSet_i^k$ ($InvRightSet_i^k$).

Now, let us define the cost functions and distance functions respectively. At the beginning of every iteration, we say one sensor has *negative shift* if it has moved to left and has *positive shift* if it has moved to right or stays still compared to its initial position. Define $shift_i$ as the shift distance of x_i. Before the definition of overlap cost, we have a definition for Left/Right Overlap Active Sensor Set as following.

Definition 3 (Left/Right Overlap Active Sensor Set) For gap_i^k, the l^{th} to $(l+k)^{th}$ sensors left to x_i form $overlap_l^k$ and the r^{th} to $(r+k)^{th}$ sensors right to x_{i+k} form $overlap_r^k$. Left Overlap Active Sensors Set are sensors at $x_{l+k}, x_{l+2k}, \cdots, x_{l+mk}, \cdots, x_i$, where $l+mk \leq i$. We denote it as $LeftActSet_{i,l}^k$. Correspondingly, Right Overlap Active Sensors Set are sensors at $x_r, x_{r-k}, x_{r-2k}, \cdots, x_{r-mk}, \cdots, x_{i+k}$, where $r-mk \geq i+k$. We denote it as $RightActSet_{i,r}^k$.

Definition 4 (Left/Right Overlap Cost) For gap_i^k, the l^{th} to $(l+k)^{th}$ sensors left to x_i form $overlap_l^k$ and the r^{th} to $(r+k)^{th}$ sensors right to x_{i+k} form $overlap_r^k$. Let NS_l^k (PS_l^k) be the set of sensors which have negative (positive include 0) shift among $LeftActSet_{i,l}^k$. Let NS_r^k (PS_r^k)be the set of the set of sensors which have negative include 0 (positive) shift

Fig. 5 Illustration for left overlap cost l_{cost} and right overlap cost r_{cost} (K = 3)



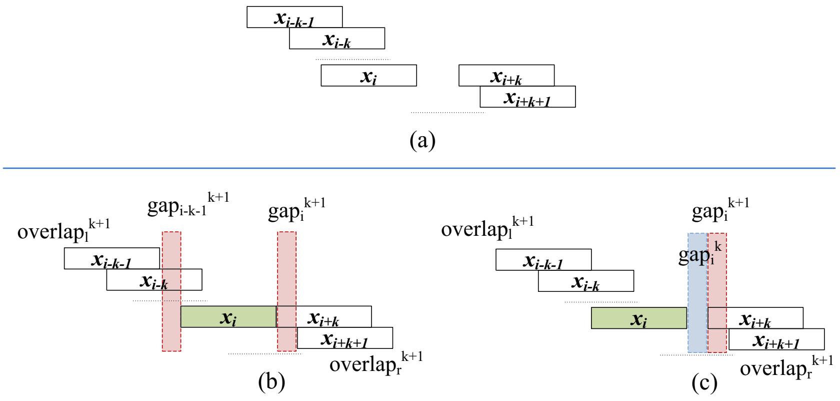


Fig. 6 Deal with $k + 1$ level gaps created by different strategies

among $RightActSet_{i,r}^k$. Then Eq. (5) computes the left/right overlap costs.

$$\begin{cases} Lcost(l,i,k) = |PS_l^k| - |NS_l^k|, \\ Rcost(i,r,k) = |NS_r^k| - |PS_r^k| \end{cases} \tag{5}$$

Definition 5 (Left/Right Overlap Shift Distance) Easy to know, the size of gap_i^k is $x_{i+k} - x_i - 2R$, denoted as GapSize, the size of $overlap_l^k$ and $overlap_r^k$ are $x_l - x_{l+k} + 2R$ and $x_r - x_{r+k} + 2R$ respectively, denoted as LOverlapSize and ROverlapSize. If $NS_l^k \neq \emptyset$, let $MinLeftShift = \min\{|shift_i| \mid x_i \in NS_l^k\}$, which is the effect shift window of left overlap; else $MinLeftShift = \infty$. Similarly, if $PS_r^k \neq \emptyset$, let $Min - RightShift = \min\{|shift_i| \mid x_i \in PS_r^k\}$, which is the effect shift window of right overlap; else $MinRightShift = \infty$. Then the left/right shift distance are

$$\begin{cases} Ldist(l,i,k) = \\ \min\{GapSize, LOverlapSize, MinLeftShift\}, \\ Rdist(i,r,k) = \\ \min\{GapSize, ROverlapSize, MinRightShift\} \end{cases} \tag{6}$$

Figure 5 is an example to illustrate the shift window. We label sensor segments by their id and the shift distance. Here gap_7^3 has size 5. Its left closest overlap is $overlap_1^3$. $InvLeftSet_i^k = \{x_4, x_7\}$. Both two sensors have negative shift ($shift_4 = -1$ and $shift_7 = -3$) so $NS_1^3 = \{x_4, x_7\}$. $PS_1^3 = \emptyset$ because no sensor in $InvLeftSet_i^k$ has right or zero shift. Similarly, $NS_{10}^3 = \{x_{10}\}$ $PS_{10}^3 = \emptyset$. Thus, $Lcost(1,7,3) = 0{-}2 = -2$ and $Rcost(7,10,3) = 1{-}0 = 1$ by Eq. (5). The effect shift window $MinLeftShift = \min\{|shift_4|, |shift_7|\} = 1$, size of gap_7^3 is 6 and size of $overlap_1^3$ is 5 so $Ldist(1,7,3) = \min\{1,5,6\} = 1$. Similarly, the right closest overlap $overlap_{10}^3$ has size 8, $Rdist(7,10,3) = \min\{6,8,\infty\} = 6$. Thus we will choose the left overlap to fill gap_7^3 since it has cheaper cost.

After calculating the costs of nearest left and right overlaps, we can choose the cheaper one. So the left overlap should be chosen in example in spite of seeming far from the gap. The difference is for right overlap zero is contained by negative shift set, while for left overlap zero is contained by position shift set. That means right (left) shift will benefit those sensors who have moved left (right) to minimize the total shift distance. Authors in [17] also considered the compensation by just counting the negative moving sensors and minus the number of negative moving from the cost function. However, due to their case of $K = 1$, the left shifts by right overlap involve moves of sensors whose shift values are all zero or negative, while right shifts by left overlap involve moves of sensors whose shift values are zero, positive, or negative. It is because of the left-to-right processing of the gaps and one round processing. Our algorithm is a multi-round processing scheme, thus left shifts by right overlap involve moves of sensors whose shift values are all zero, negative or negative as well. This is complexity brought by the K-coverage.

5.2 Optimality and complexity of LLK-MinMovs

Now let us prove the correctness of LLK-MinMovs and analyze its complexity. Firstly, we will give a lemma below named *Order Preservation.*

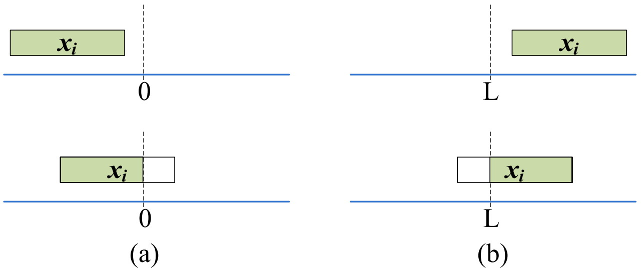


Fig. 7 Overlaps outside of target interval

Lemma 1 (Order Preservation) *For any instance of the LK-MinMovs problem, if initially sensors are sorted increasingly ($x_1 \le ... \le x_n$) then there exists an optimal solution where the final positions of sensors satisfy $x_1^f \le ... \le x_n^f$.*

This lemma is easy to prove by an inequality $|x_i - x_i^f| + |x_j - x_j^f| \le |x_i - x_j^f| + |x_j - x_i^f|$, if $x_i \le x_j$ and $x_i^f \le x_j^f$.

Theorem 1 *LLK-MinMovs outputs an optimal solution.*

Proof Firstly, we state that LLK-MinMovs can solve LK-MinMovs when $K = 1$. If we replace the moving strategy by the new one considering the effect shift window, Theorem 3 in [17] can prove the statement. We then use induction on K. Assume when $K = k \ge 1$, this theorem holds. Let us consider $K = k + 1$.

When LLK-MinMovs deals with the k^{th} level gaps, it does not produce new gaps which are k^{th} level or lower levels. However, the procedure may create $k + 1^{th}$ level gaps or lengthen $k + 1^{th}$ level gaps. If there is no such new produced gap, the proof is trivial. So we assume there is a new $k+1^{th}$ level gap gap_{i-k-1}^{k+1}, see Fig. 6, now we should make efforts to recover it. There are two possibilities why the gap is created. The one is caused by sensor $x_{i-k-1's}$ left-shift while another one is caused by sensor $x_{i's}$ right-shift. They are symmetrical so in Fig. 6b we take the later one as an example. Before giving further proof we give two claims first.

Claim 1 A new $k + 1^{th}$ level gap which is created during covering k^{th} level while a corresponding k^{th} level gap is filled up. Reversely, if the $k + 1^{th}$ level gap is not created while the corresponding k^{th} level gap is left. They have the same size.

Claim 2 A k^{th} level gap is a part of a $k+1^{th}$ level gap. Cost for filling $k+1^{th}$ level gap which contains any k^{th} level gap is more expensive than pure $k + 1$ level gap with the same size.

Claim 1 is obvious. We emphasize on explaining **Claim 2**, because we need to pay more cost for the k^{th} level gap contained in the $k + 1^{th}$ gap. Moreover, when filling the k^{th} level gap, algorithm may use overlaps which are also the nearest overlaps for the $k + 1^{th}$ gap. That means we should find more expensive overlaps for the $k + 1^{th}$ gap. **Claim 2** is always right.

See Fig. 6c, we assume there is another solution which completes the k^{th} level coverage and does not create any new $k + 1^{th}$ overlap. Thus, x_i and x_{i-k-1} are attached. According to **Claim 1**, a k^{th} level gap is left. In the subgraph Fig. 6b, it is the result of LLK-MinMovs. Currently, let movements for Fig. 6b be $CurMovs_{(b)}$ and for Fig. 6c be $CurMovs_{(c)}$. We have $CurMovs_{(b)} \le Cur - Movs_{(c)} + Cost_{(c)}(gap_i^k)$ because of the optimality assumption for $K = k$. Let us compare their total movements, denoted as $Movs_{(b)}$ and $Movs_{(c)}$. For Fig. 6b, we should fill up gap_{i-k-1}^{k+1} and gap_i^{k+1}. For Fig. 6c, we should fill up the gap_i^k and gap_i^{k+1}. We can also have $Movs_{(b)} = CurMovs_{(b)} + Cost_{(b)}(gap_{i-k-1}^{k+1}) + Cost_{(b)}(gap_i^{k+1})$ and $Movs_{(c)} = CurMovs_{(c)} + Cost_{(c)}(gap_i^k) + Cost_{(c)}(gap_i^{k+1})$. Because left and right overlaps for $k + 1$ level gaps are the same, we show them as $overlap_l^{k+1}$ and $overlap_r^{k+1}$ in Fig. 6b, Fig. 6c, filling $k + 1$ level overlap(s) with the same size takes same cost. Then according to **Claim 2**, $Cost_{(b)}(gap_{i-k-1}^{k+1}) + Cost_{(b)}(gap_i^{k+1}) \le Cost_{(c)}(gap_i^{k+1})$. Combining it with previous equations we get $Movs_{(b)} \le Movs_{(c)}$. So for $K = k + 1$, our algorithm is better than any other solution which does not produce $k + 1$ level gaps. Thus, LLK-MinMovs is optimal for $K = k + 1$. Proof is finished. □

Next, we discuss the time complexity for the LLK-MinMovs algorithm. There are K loops to cover each level of gaps. For each loop, we consider the total times of movements. There are two types of movements, left-movement and right-movement. In each left-movement, either a gap or an overlap will be removed so the total times of left-movements $T_l \le T_{ol} + T_{gl}$, where T_{ol} is the times of movement when an overlap removes, and T_{gl} is the times of movement when an gap removes. In each right-movement, a gap or an overlap will remove or a sensor is moved back to its initial location. Similarly, we have $T_r \le T_{or} + T_{gr} + T_b$ where T_b is the times of movement when a sensor is moved back. Since in our algorithm the number of gaps as well as the number of overlaps will not change, we have $T_{gl} + T_{gr} = |gaps|$, $T_{ol} + T_{or} \le |overlaps|$ and $|gaps| + |overlaps| < n$. On the other hand, for each level k, at most n sensors are moved back so $T_b \le Kn$. Thus we get the total time of movement $T = T_l + T_r < n + Kn$. For each movement, we will move at most n sensor. Thus the time complexity is $O(K^2n^2)$.

6 Generalized sensor initial deployment

In the above discussions, we limit sensor initial deployment into the target interval $[0,L]$. It is a closed deployment assumption. However, for more application scenarios open initial deployment is more practical. In this section we will address this topic. First of all, we will redefine overlap in comparison with the previous definition. Then, new left and right overlap costs are given. At the algorithm part, how to identify the first sensor to fill the first gap is the most important and need be elaborated. The formal problem definition should be changed. In open distribution scenario, we find a critical flaw against MinSum algorithm. We will propose a counter example in last subsection. Corresponding simulation results will be shown in the section of numerical experiments.

6.1 Problem redefinition for general initial deployment

We view any intersection between sensor range interval and outside interval relative to target interval as an overlap. So Line k-Covered Overlap is redefined as following.

Definition 6 (General Line k-Covered Overlap) A general line k-covered overlap, denoted as $GenOverlap_i^k$, is an interval which starts from the starting point of sensor x_i and ends up to point $min(x_i + R, 0)$ if x_i locates on the left of point R, say the interval $[x_i - R, min(x_i + R, 0)]$. Or it is an interval which starts from the point $max(x_i - R, L)$ and ends up to the ending point of sensor x_i if x_i locates on the right of point $L - R$, say the interval $[max(x_i - R, L), x_i + R]$. We denote overlap of this type as TYPE ONE. $GenOverlap_i^k$ can also be an interval which starts from the starting point of sensor x_{i+k} and ends up to the ending point of sensor x_i, say, the interval $[x_{i+k} - R, x_i + R]$. We denote overlap of this type as TYPE TWO.

From Fig. 7a, we can see the two cases for overlap on the left outside of interval $[0, L]$ and Fig. 7b shows us the right case. We can see that if we use the overlap for filling up the gap inside of $[0, L]$, extra distance for the blank between sensor and ending point of segment $[0, L]$ will arise the cost for the upper cases of Fig. 7a and Fig. 7b. So we need to update the cost calculation functions for this reason.

Because we remove the constraint that sensors cannot be deployed outside of $[0, L]$, we refine the LK-MinMovs problem as the following linear programming:

$$\min \quad \sum_{i=1}^{n} |x_i^f - x_i| \tag{7}$$

$$s.t. \quad s \le t - \lceil \tfrac{K*L}{2R} \rceil, \qquad \exists 1 \le s \le t \le n. \tag{8}$$

$$x_{i+k}^f - x_i^f \le 2R, \qquad \begin{array}{l} \forall\, 1 \le k \le K, \\ \forall\, s \le i \le t - k \end{array} \tag{9}$$

$$x_s^f \le R \text{ and } x_t^f \ge L - R. \tag{10}$$

From the linear programming, we can see that the changes are not just removing the initial deployment condition, the additional condition (8) conveys that there is a sensor sequence that K-covers the interval $[0, L]$. How to find the sequence is the most important and most difficult for solving the problem. For our algorithm, we always find a current gap to fill by moving the cheapest overlap. In close deployment assumption, we directly use the first sensor (virtual sensor) x_0 to locate the first gap we should fill. After filling up it, we continue to find the next gap, and this procedure can be pushed forward until all gaps are filled. However, for the open deployment assumption, we do not know which sensor are the breakout one. For optimality reason, we always choose the nearest to the starting point 0 to start.

6.2 General cost function and our algorithm

Firstly, we redefine the overlap cost as follows:

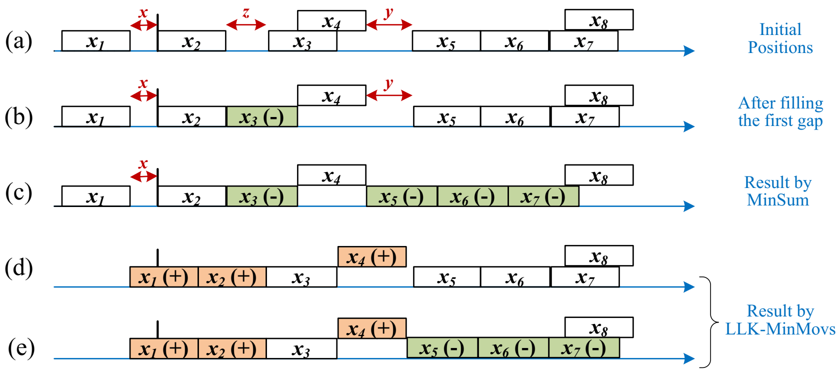
Definition 7 (General Left/Right Overlap Cost) For gap_i^k, $GenOverlap_l^k$ is the left nearest overlap to x_i and $GenOverlap_r^k$ is the right nearest overlap to x_{i+k}. Let NS_l^k (PS_l^k) be the set of sensors which have negative (positive include 0) shift among $x_{l+k}, x_{l+2k}, \cdots, x_{l+mk}, \cdots, x_i$ sensors, where $l + mk \le i$. Let NS_r^k (PS_r^k) be sensors which have negative include 0 (positive) shift among $x_r, x_{r-k}, x_{r-2k}, \cdots, x_{r-mk}, \cdots, x_{i+k}$ sensors, where $r - mk \ge i + k$. Let $LBlank$ (resp. $RBlank$) be the distance between $x_l + R$ (resp. $x_r - R$) and 0 (resp. L) if $GenOverlap_l^k$ is outside left (right) of $[0, L]$. Let $MinShift = min(MinLeftShift, MinRightShift)$. Then Eq. (11) computes the general left/right overlap costs.

$$\begin{cases} GenLcost(l, i, k) = \\ (|PS_l^k| - |NS_l^k|) * MinShift + LBlank. \\ GenRcost(i, r, k) = \\ (|NS_r^k| - |PS_r^k|) * MinShift + RBlank. \end{cases} \tag{11}$$

Algorithm 4 General LLK-MinMovs

```
Input: $X[1 \ldots n]$, $L$, $K$, $R$
Output: The final position $X^f[1 \ldots n]$ of $n$ sensors
for $k \leftarrow 1$ to $K$ do
    Collect a set of sensors $INSensors$ which are
    located in interval $[-R, R]$, Let $s$ be the first sensor
    index, $t$ be the last one index.
    while true do
        $i = findFirstGap(k)$;
        if $i \neq -1$ then
            $l = find(gap_i^k, \text{left})$;
            $r = find(gap_i^k, \text{right})$;
            if $GenLcost(i, l, k) \le GenRcost(i, r, k)$
            then
                if $l$ is the left outside overlap then
                    $move(l, l, LBlank)$
                $move(l, i, GenLdist(l, i, k))$
            else if $r$ is the right outside overlap then
                $move(r, r, -RBlank)$
            $move(i, r, -GenRdist(i, r, k))$
        else Break
        Update($s$,$t$)
return $X^f[1 \ldots n]$;
```

Fig. 8 A counter example to algorithm in [17] ($K = 1$)



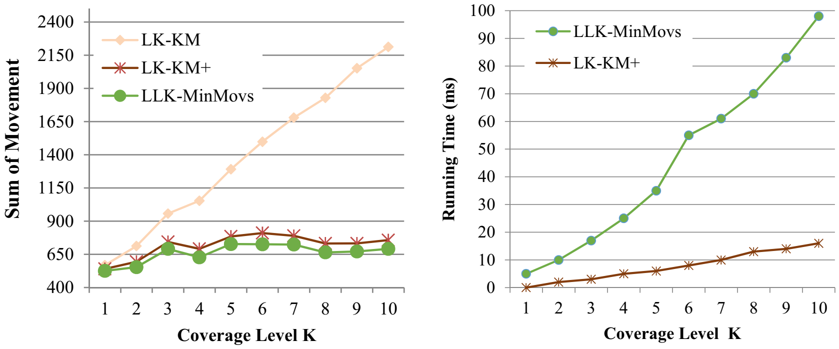
In Algorithm 4, we should maintain a sensors sequence $X[s, t]$ which is located in interval $[-R, R]$, named as $INSensors$. So we define a $findFirstGap(k)$ which performs to find the first gap and returns its index. The range of the function is $0, s, ..., t$. It returns 0 when the gap is between 0 and the starting point of sensor x_s. It returns i ($s \leq i \leq t - k$) when the gap is between x_s and x_{s+k}. It returns $t - k$ when the gap is between x_{t-k} and L. It returns -1 when there is no gap of level k. For filling up one gap we may use sensors outside of interval $[0, L]$ (They may penetrate interval $[-R, R]$), and new sensors become the member of $INSensors$. We use $Update(s, t)$ function to expand the bound of the set. Notice that sensors once become the member of $INSensors$ will not quit in the later procedure. For the definition of $GenLdist$ and $GenRdist$, we just replace the $MinLeftShift$ and $MinRightShift$ by $MinShift$ in Eq. (6). We still use the same functions such as $find(\cdot)$, $move(\cdot)$ in Algorithm 3.

6.3 A counter example for MinSum algorithm

Note that the movement cost for the inside overlaps and outside overlaps are different. The authors in [17] also considered the shift compensation in spite of using similar cost functions for $K = 1$. However, when dealing with outside overlaps, they did not consider the influence of the distance between the closest overlap sensor to the endpoints of the target interval. A counter example is shown in Fig. 8.

In Fig. 8a we give the original positions of 8 sensors. Figure 8b shows the situation when the first gap_2^1 with size z are filled, and there is still a gap_4^1 with size y. The results are the same for our algorithm and MinSum. Figure 8c shows the result using MinSum which exploits $overlap_7^1$ to fill up gap_4^1. Here, $Lcost(l, 1, 1) = 2 + x$ and $Rcost(7, r, 1) = 3$, we assume $x > 1$ then $Lcost(l, 1, 1) > Rcost(7, r, 1)$. Figure 8d shows that our algorithm uses outside overlap $overlap_1^1$. Now, $GenLcost(l, 1, 1) = 2 * z + x$ and $GenRcost = 3 * z$, we assume $GenLcost(l, 1, 1) < GenRcost(7, r, 1)$ then $x < z$. Figure 8e shows the result after using $overlap_7^1$ in our algorithm. Now, $GenLcost(l, 1, 1) = 4$ and $GenRcost(7, r, 1) = 3$, then $GenLcost(l, 1, 1) > GenRcost(7, r, 1)$. We can calculate the total distance for MinSum in this procedure as $3 * z + 3 * y$ while our algorithm total distance is $3 * z + x + 3 * (y - z)$. We let $x = 2, z = 10, y = 12$. Total movements distance for MinSum is 66 and the one for our algorithm is 38. Obviously, MinSum does not produce an optimal solution.

Fig. 9 Efficiency comparison under closed deployment assumption



(a) Experiments on K
$L = 1000,\ R = 50$
Redundancy Rate is 1.25

(b) Running Time
$L = 1000,\ R = 50,$
Redundancy Rate is 1.25

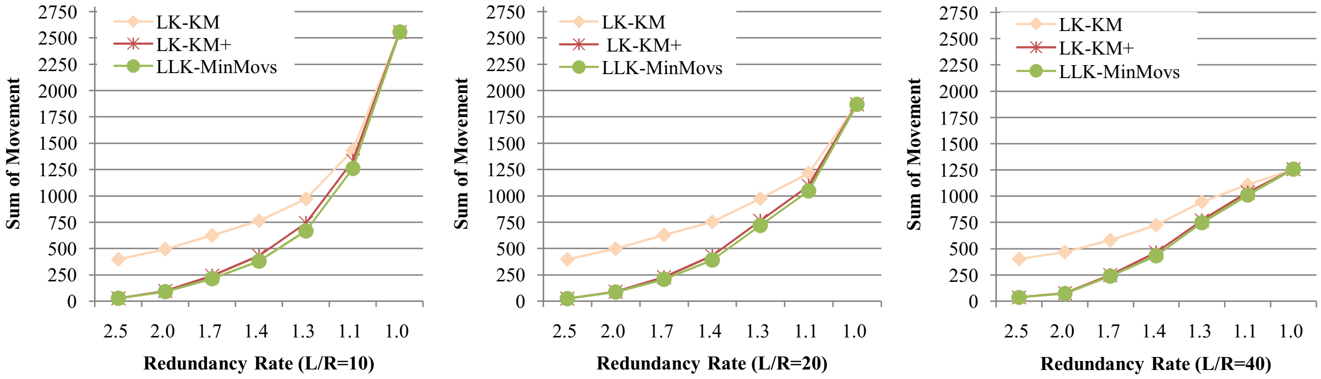


Fig. 10 Experiments on redundancy rate and L/R under closed deployment assumption ($L = 1000$, $K = 3$)

7 Numerical experiments

In experiments, we respectively compare our algorithms under closed and open deployment assumption. Under open deployment assumption, we also compare the general version of LLK-MinMovs with MinSum [17] to show our optimality for solving LK-MinMovs problem. Algorithms are all implemented in C++. For each case, we run each algorithm 100 times at random inputs and calculate the average sum of movements. We define the redundancy rate as $\frac{2Rn}{LK}$.

First of all, we conduct a group of experiments under closed deployment assumption. We study the coverage level K. Here we set $K = 1$ to 10, $L/R = 20$ and redundancy rate be 1.25. We can see that LLK-MinMovs and LK-KM+ get more superiority than LK-KM when the coverage level K is increasing. Figure 9a gives the results. In term of running time, obviously, LK-KM uses less time than LK-KM+ without pulling back operations so we just compare LK-KM+ with LLK-Minmovs. Figure 9b shows that LK-KM+ outperforms LLK-MinMovs much more when coverage level K is increasing.

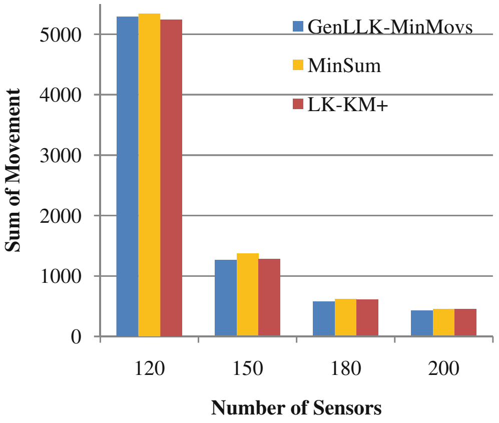


Fig. 11 Experiments under open deployment assumption ($L = 1000$, $R = 5$, $K = 1$, Divergence is 200)

Then, we study the influence on the result with different sensor redundancy rate. Besides, we conduct 3 groups of experiments on different $L/R = 10, 20$ *and* 40. And we set $L = 1000, K = 3$. Figure 10 shows that in all cases LLK-MinMovs has the best performance. It worths to notice that three algorithms have the same result at redundancy rate 1 because each sensor should move to a definite position for the perfect K-coverage. Sensors need to move shorter distance when L/R becomes bigger because that more sensors involved in makes K-coverage more easy to achieve. The LK-KM is influenced more by L/R. The LK-KM+ improves LK-KM and is very close to LLK-MinMovs and is not influenced by L/R.

Under the open deployment assumption, we verify that general algorithm GenLLK-MinMovs have the best performance for line 1-coverage, which proves that the MinSum is not optimal. Let $L = 1000$, $R = 5$ and the results are shown in Fig. 11. LLK-MinMovs is the best and even sub-optimal LK-KM+ is better than MinSum. We denote the maximal distance with which sensors can be deployed away from the endpoints of the target interval as *divergence*. Here, the divergence is set to 200.

For showing the influence of open sensor initial deployment, we conduct a group of experiments which consider different divergences. We record the sum of movements in Table 1. We form line 4-coverage for a region with length 1000 while the divergences are 100, 300, 500, 1000. That means sensors can be deployed in intervals [−100,1100], [−300,1300], [−500,1500] and [−1000,2000] respectively while the target interval is [0,1000]. The results show that

Table 1 Comparison on different divergences ($L = 1000$, $R = 50$, $K = 4$)

Divergence	GenLLK-MinMovs	LK-KM+	LK-KM
100	1023.22	1163.74	1464.53
300	3223.26	3229.61	3306.45
500	6119.75	6131.21	6234.12
1000	14322.3	14643.8	14698.9

sensors should move much more distance when divergence increases, which is reasonable. Additionally, the general version of LLK-MinMovs always has the best performance in all cases.

8 Conclusion and future work

In this paper, we mainly addressed the Line K-Coverage problem (LK-MinMovs) in mobile wireless sensor network. The problem can be viewed as an important objective in social or distributed networks when the position system is settled down. We firstly proposed two sub-optimal but faster algorithms, LK-KM and LK-KM+, which are based on the famous Hungarian algorithm. Then we proposed an optimal layer-based algorithm LLK-MinMovs with a polynomial time complexity. We gave its optimality proof. Further we considered the general open initial deployment assumption for this algorithm. It fixes a critical flaw of the MinSum algorithm designed in [17] for line 1-Coverage problem. LK-KM and LK-KM+ have good time complexity $O(\frac{K^2L^2n}{R^2})$ while LLK-MinMovs has $O(K^2n^2)$ as its time complexity.

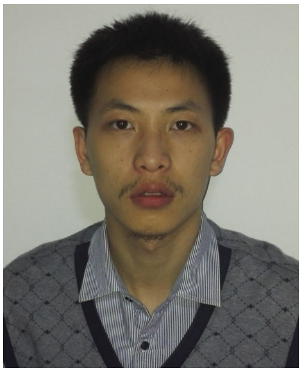
In the future, we will consider to implement algorithms we proposed above in a parallel and distributed way. Because distributed algorithm fits the social network applications better. Additionally, in order to make the sensor network keep a K-coverage for the target region for a longer time, it is necessary to keep the remaining power of all sensors be similar after the movement to guarantee a valid network lifetime. In [32] and [33], authors considered another optimization objective, that is, minimize the maximum movement. We can also adapt the line K-coverage problem for this objective. Currently we consider to minimize the overall movement of sensors to extend the network lifetime, while it might be more practical to consider to maximize the makespan of individual sensor movement. Authors in [28] considered the strong barrier coverage problem with sensor movement in top-down directions. However, they did not consider for real 2D scenario in which intruders can be detected in arbitrary direction. We could extend our work to real 2D, even 3D applications.

References

1. Choi W, Das SK (2009) Cross: a probabilistic constrained random sensor selection scheme in wireless sensor networks. Perform Eval 66(12):754–772
2. Boukerche A, Xin F (2007) A voronoi approach for coverage protocols in wireless sensor networks. In: IEEE global telecommunications conference (GLOBECOM), pp 5190–5194
3. Bai H, Chen X, Li B, Han D (2007) A location-free algorithm of energy-efficient connected coverage for high density wireless sensor networks. Discrete Event Dynamic Systems 17(1):1–21
4. Chizari H, Poston T, Razak SA, Abdullah AH, Salleh S (2014) Local coverage measurement algorithm in GPS-free wireless sensor networks. Ad Hoc Networks 23:1–17
5. Wang B (2010) Coverage control in sensor networks. Springer, London
6. Liang JB, Liu M, Kui XY (2014) A survey of coverage problems in wireless sensor networks. Sensors & Transducers 163(1):240–246
7. Fan X, Li VOK (2011) The probabilistic maximum coverage problem in social networks. In: IEEE global telecommunications conference (GLOBECOM), 6613(1): 1–5
8. Zhang P, Chen W, Sun X, Wang Y, Zhang J (2014) Minimizing seed set selection with probabilistic coverage guarantee in a social network. In: ACM international conference on knowledge discovery & data mining (SIGKDD), pp 1306–1315
9. Dash D, Gupta A, Bishnu A, Nandy SC (2014) Line coverage measures in wireless sensor networks. J Parallel Distrib Comput 74(7):2596–2614
10. Huang C, Tseng Y (2005) The coverage problem in wireless sensor network. Mobile Network Application 10(4):519–528
11. Kumar S, Lai TH, Arora A (2005) Barrier coverage with wireless sensors. In: The annual international conference on mobile computing & networking (ICMCN), pp 284–298
12. Shen C, Cheng W, Liao X, Peng S (2008) Barrier coverage with mobile sensors. In: IEEE international symposium on parallel architectures, algorithms & networks (ISPAN), pp 99–104
13. Balister P, Zheng Z, Kumar S, Sinha P (2009) Trap coverage: allowing coverage holes of bounded dimeter in wireless sensor network. In: IEEE international conference on computer communications (INFOCOM), pp 136–144
14. Baumgartner K, Ferrari S (2008) A geometric transversal approach to analyzing track coverage in sensor networks. IEEE Transactions on Computers (TC) 57(8):1113–1128
15. Harada J, Shioda S, Saito H (2009) Path coverage properties of randomly deployed sensors with finite data-transmission ranges. Comput Netw 53(7):1014–1026
16. Sundhar Ram S, Manjunath D, Iyer SK, Yogeshwaran D (2007) On the path coverage properties of random sensor networks. IEEE Trans Mob Comput 6(5):446–458
17. Czyzowicz J, Kranakis E, Krizanc D, Lambadaris I, Narayanan L, Opatrny J, Stacho L, Urrutia J, Yazdani M (2010) On minimizing the sum of sensor movements for barrier coverage of a line segment. In: International conference on ad hoc networks & wireless (ADHOC-NOW), 6288: 29–42
18. Li X, Frey H, Santoro N, Stojmenovic I (2008) Localized sensor self-deployment with coverage guarantee. ACM Mobile Computing & Communications Review 12(2):50–52
19. Yang SH, Li ML, Wu J (2007) Scan-based movement-assisted sensor deployment methods in wireless sensor networks. IEEE Trans Parallel Distrib Syst 18(8):1108–1121
20. Zou Y, Chakrabarty K (2005) A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. IEEE Transactions on Computers (TC) 54(8):978–991
21. Ahmed A, Yasumoto K, Yamauchi Y, Ito M (2011) Distance and time based node selection for probabilistic coverage in people-centric sensing. IEEE Communications Society Conference on Sensor, Mesh & Ad Hoc Communications and Networks 2011:134–142
22. Mondal D, Kumar A, Bishnu A, Mukhopadhyaya K, Nandy SC (2011) Measuring the quality of surveillance in a wireless sensor network. Int J Found Comput Sci 22(4):983–998
23. Hesari ME, Kranakis E, Krizanc D, Ponce OM, Narayanan L, Opatrny J, Shende SM (2013) Distributed algorithms for barrier coverage using relocatable sensors. In: ACM symposium on principles of distributed computing (SPDC), pp 383–392

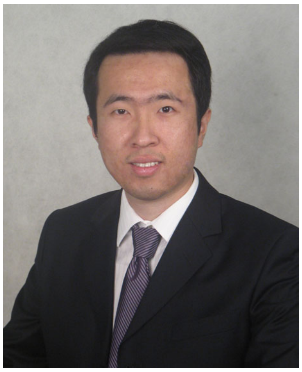
24. Bar-Noy A, Rawitz D, Terlecky P (2013) Maximizing barrier coverage lifetime with mobile sensors. In: The Annual European Symposium Algorithm (ESA), pp 97–108
25. Saipulla A, Westphal C, Liu B, Wang J (2013) Barrier coverage with line-based deployed mobile sensors. Ad Hoc Netw 11(4):1381–1391
26. Li L, Zhang B, Shen X, Zheng J, Yao Z (2011) A study on the weak barrier coverage problem in wireless sensor networks. Computer Networks (CN) 55(3):711–721
27. Wang Z, Liao J, Cao Q, Qi H, Wang Z (2013) Achieving k-barrier coverage in hybrid directional sensor networks. IEEE Trans Mob Comput 13(7):1443–1455
28. Saipulla A, Liu B, Xing G, Fu X, Wang J (2010) Barrier coverage with sensors of limited mobility. In: ACM international symposium on mobile ad hoc networking & computing (MOBIHOC), pp 201–210
29. Chang WY, Yu KM, So WT, Lin C-T, Lin CY (2011) Target coverage in wireless sensor networks. In: IEEE international conference on mobile ad-hoc & sensor networks (MSN), pp 408–412
30. Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2:83–97
31. Munkres J (1957) Algorithms for the assignment and transportation problems. J Soc Ind Appl Math 5(1):32–38
32. Czyzowicz J, Kranakis E, Krizanc D, Lambadaris I, Narayanan L, Opatrny J, Stacho L, Urrutia J, Yazdani M (2009) On minimizing the maximum sensor movement for barrier coverage of a line segment. In: International conference on ad hoc networks & wireless (ADHOC-NOW), 5793: 194–212
33. Chen DZ, Gu Y, Li J, Wang H (2013) Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain. Discrete Comput Geom 50(2):374–408

Yang Wang is a Ph.D. student from the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He received the B.S. degree from in computer science and technology from Gui Zhou University, China, in 2005; the M.S. degree in computer applied technology from Shanghai Jiao Tong University, China, in 2008. His current research interests include network optimization problems, approximation algorithm, data mining and analysis.



Shuang Wu is a graduate student from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include approximation algorithm, sensor coverage problems in wireless networks.

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. Her research interests include wireless communications, data engineering, and combinatorial optimizations. She has published more than 90 peer-reviewed papers and 7 book chapters in the related area, including well-archived international journals such as IEEE TC, IEEE TKDE, IEEE TPDS, TCS, and also in well-known conference proceedings such as INFOCOM, SIGKDD, ICDCS. She has served on the editorial board of Discrete Mathematics, Algorithms and Applications, and as the PCs and peer reviewers for a number of international conferences and journals.



Fan Wu is an associate professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He received his B.S. in Computer Science from Nanjing University in 2004, and Ph.D. 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 (UIUC) as a Post Doc Research Associate. His research interests include wireless networking and mobile computing, algorithmic game theory and its applications, and privacy preservation. He has published more than 100 peer-reviewed papers in technical journals and conference proceedings. He is a recipient of the first class prize for Natural Science Award of China Ministry of Education, NSFC Excellent Young Scholars Program, ACM China Rising Star Award, CCF-Tencent “Rhinoceros bird” Outstanding Award, CCF-Intel Young Faculty Researcher Program Award, and Pujiang Scholar. He has served as the chair of CCF YOCSEF Shanghai, on the editorial board of Elsevier Computer Communications, and as the member of technical program committees of more than 60 academic conferences.

Guihai Chen is a distinguished professor of Shanghai Jiao Tong University. He earned B.S. degree in computer software from Nanjing University in 1984, M.E. degree in computer applications from Southeast University in 1987, and Ph.D. degree in computer science from the University of Hong Kong in 1997. He had been invited as a visiting professor by Kyushu Institute of Technology in Japan, University of Queensland in Australia and Wayne State University in USA. He has a wide range of research interests with focus on parallel computing, wireless networks, data centers, peer-to-peer computing, high-performance computer architecture and data engineering. He has published more than 350 peer-reviewed papers, and more than 200 of them are in well-archived international journals such as IEEE TPDS, IEEE TC, IEEE TKDE, ACM/IEEE TON and ACM TOSN, and also in well-known conference proceedings such as HPCA, MOBIHOC, INFOCOM, ICNP, ICDCS, CoNext and AAAI. He has won several best paper awards including ICNP 2015 best paper award. His papers have been cited for more than 10000 times according to Google Scholar.