Efficient Line K-Coverage Algorithms in Mobile Sensor Network

Yang Wang, Shuang Wu, Xiaofeng Gao(✉), Fan Wu, and Guihai Chen

Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{y_wang,steinsgate}@sjtu.edu.cn,
{gao-xf,fwu,gchen}@cs.sjtu.edu.cn

Abstract. In this paper, we address a new type of coverage problem in mobile sensor network, named Line K-Coverage. It guarantees that any line cutting across a region of interest will be detected by at least K sensors. We aim to schedule an efficient sensor movement to satisfy the line K-coverage while minimize the total sensor movements for energy efficiency, which is named as LK-MinMovs problem. We propose a pioneering layer-based algorithm LLK-MinMovs to solve it in polynomial time. Compared with a MinSum algorithm from previous literature to solve line 1-coverage problem, LLK-MinMovs fixes a critical flaw after finding a counter example for MinSum. We further 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 runs extremely faster than algorithm LLK-MinMovs. We validate the efficiency of our designs in numerical experiments and compare them under different experiment settings.

1 Introduction

Wireless Sensor Network (WSN) nowadays attracts special attentions from scientific and technological community. Coverage is a fundamental problem among all challenges of WSN. Broadly speaking, coverage is a measure that determines how well a sensor network monitors objectives. Many variations of coverage problem have been proposed for different applications. As an example, the area K-coverage problem requires that each point in the area be covered by at least K sensors.

This work was supported in part by the State Key Development Program for Basic Research of China (973 project 2012CB316201), in part by China NSF grant 61422208, 61202024, 61472252, 61272443 and 61133006, CCF-Intel Young Faculty Researcher Program and CCF-Tencent Open Fund, the Shanghai NSF grant 12ZR1445000, Shanghai Chenguang Grant 12CG09, Shanghai Pujiang Grant 13PJ1403900, and in part by Jiangsu Future Network Research Project No. BY2013095-1-10. 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.

K. Xu and J. Zhu (Eds.): WASA 2015, LNCS 9204, pp. 581–591, 2015.
DOI: 10.1007/978-3-319-21837-3_57

Barrier coverage is more applicable to monitor borders due to exploiting less sensors than area coverage. In front of or surrounding an area, a barrier is a belt-like region in which the sensors are spread. The barrier is said to be K-covered [2,3] 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 pathes for barrier coverage [4–7], since in reality intruders usually go through a region with a line track. Moreover, intruders do not know any knowledge on distribution of sensors, they cannot figure out a "smart" path to follow. In [16], the authors proposed a MinSum algorithm 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.

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. For one layer repairing, a MinSum was proposed in a previous literature [16]. However, although authors in [16] 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 the time complexity of them, and then validate their efficiency in numerical experiments under different experiment settings. 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.

The rest of the paper is organized as follows. Section 2 introduces some related works. Section 3 presents the problem statement. Section 4 describes our layer-based algorithm (LLK-MinMovs) and gives its time complexity analysis. A counter example for MinSum is also introduced and corrected. In Sect. 5, we design LK-KM and its enhanced version LK-KM+. Numerical experiments are presented in Sect. 6. Finally, Sect. 7 gives conclusion.

2 Related Works

In the research area of mobile sensor networks, several recent literatures considered the strategy of mobile sensor movement to cover a region of interest, for example [9,10]. Unlike the problem considered in this paper, they aimed to form an area coverage rather than barrier coverage for the region of interest.

Some of the existing works focus on line coverage [8] in a region. Baumgartner et al. [5] proposed the track coverage problem. Their objective is to place a set of sensors in the region such that the chance of detecting the path tracks by at least some given number of sensors is maximized. Other path coverage metrics are defined in [6,7] by analytical expressions for any random deployment in a region. Balister et al. [4] defined a coverage metric called trap coverage which measures the longest distance an intruder can achieve within the region before touching the sensing range of any sensor.

In terms of mobile sensor for barrier, distributed algorithms are proposed in [11] to schedule mobile sensors for forming a barrier. Bar-Noy et al. [12] studied the problem of maximizing the coverage lifetime of a barrier composed by mobile sensors with limited battery powers. How to guide sensor moving to improve the quality of barrier coverage are studied in [13]. All of them consider barrier coverage for path, this is not the objective of this paper. We focus on line K-coverage.

Czyzowicz's work [16] is 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, MinSum cannot always output an optimal solution. We provide a counter example and correct it in our design, so that LLK-MinMovs could work optimally for arbitrary instance. We also provide numerical experiments to compare LLK-MinMovs with MinSum when $K = 1$.

3 Problem Statement

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

Figure 1 is an example scenario (Assume $K = 3$), where the intrusion direction is vertical against the rectangle. Hence, we could project s_i horizontally as a line segment represented by interval $[x_i - R, x_i + R]$. Then, we just need to consider our problem on line segment $[0, L]$. To better describe our problem, we label sensors with their x-coordinate, and assume 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 $2Rn \geq KL$ to get a feasible solution.

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.

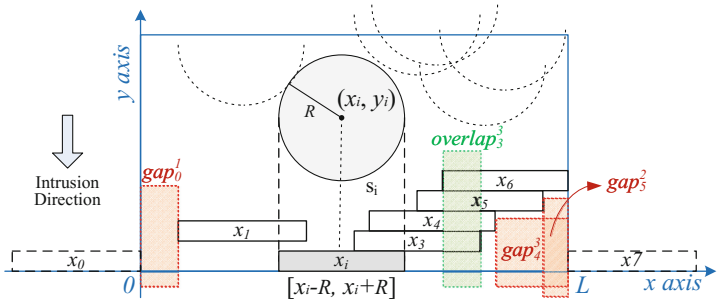


Fig. 1. An example of line coverage transformation (K=3).

However, as shown in Fig. 1, 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 $[x_i + R, x_{i+k} - R]$.*

Easy to see, gap_i^k is covered by $k-1$ sensors, and $x_{i+k} - x_i > 2R$ (otherwise they will overlap to each other). Three example gaps are shown in Fig. 1, 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 region. In this example, s_0 locates at $x_0 = -R$, and 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 $[x_{i+k} - R, x_i + R]$.*

Similarly, $overlap_i^k$ is covered by $k+1$ sensors, and $x_{i+k} - x_i < 2R$ (otherwise we cannot find an overlap interval). Thus, some sensors could move to cover other gaps. An example line 3-covered $overlap_3^3$ is shown in Fig. 1, 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-cover while the total sensor movements $\sum_{i=1}^{n} |x_i^f - x_i|$ is minimized. We require sensor movement schedule obeying an order preservation restriction given by a Lemma in [16] and sensors cannot move out of $[0, L]$.

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

4 A Layer-Based Algorithm for LK-MinMovs Problem

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. During the procedure of filling the line k-covered gaps, we can use the line k-covered overlaps. This strategy is proved to be efficient and better than MinSum by experiments in Sect. 6.

4.1 LLK-MinMovs Algorithm

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

Algorithm 1 is the pseudo-code of LLK-MinMovs. The inputs are an array $X[1 \dots n]$ 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 $X^f[1 \dots n]$ of n elements representing the final positions of sensors.

In Algorithm 1, 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 1 finds every line k-covered gap and fills 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 current stage. If there exists a gap_i^k, we find its left and right closest overlaps as potential candidates (Line 5-6), compare the cost to use them filling the gap according to cost functions $Lcost(\cdot)$ and $Rcost(\cdot)$, pick up the cheaper one, 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 (Line 7-9). The "while" loop from Line 4 to 9 guarantees that we will fill up all k-covered gaps generated by each x_i.

Algorithm 1. 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 ← 1 to K do
3      for i ← 0 to n do
4          while isCovered(i, k) = 0 do
5              l = find(gap_i^k, left) ;          // find the left closest overlap_l^k
6              r = find(gap_i^k, right) ;         // find the right closest overlap_r^k
7              if Lcost(i, l, k) ≤ 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             ;
11 return X^f[1 . . . n];
```

Now, let us define the cost functions and distance constraint 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.

Definition 3 (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) shift among $x_{l+k}, x_{l+2k}, \cdots, x_{l+mk}, \cdots, x_i$ sensors, where $l+mk \leq i$. Let S_r^k be the set of $x_r, x_{r-k}, x_{r-2k}, \cdots, x_{r-mk}, \cdots, x_{i+k}$ sensors, where $r-mk \geq i+k$. Then Eq. (1) computes the left/right overlap costs.*

$$Lcost(l,i,k) = |PS_l^k| - |NS_l^k|, \qquad Rcost(i,r,k) = |S_r^k|. \tag{1}$$

Definition 4 (Left/Right Overlap Shift Distance). *Easy to know, the size of gap_i^k is $x_{i+k} - x_i - 2R$, 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. Let $MinShift = \min\{|shift_i| \mid x_i \in NS_l^k\}$, which is the effect shift window. Then the left/right shift distance are*

$$\begin{cases} Ldist(l,i,k) = \min\{x_{i+k} - x_i - 2R, x_l - x_{l+k} + 2R, MinShift\}, \\ Rdist(i,r,k) = \min\{x_{i+k} - x_i - 2R, x_r - x_{r+k} + 2R\} \end{cases} \tag{2}$$

4.2 A Counter Example for MinSum Algorithm

Note that the shifting cost for the left and right overlaps are different. The left shifts for the right overlap only involve sensors whose shift values are zero or negative, while the right shifts for the left overlap involve sensors will all possible shift values. It is due to the left-to-right processing of the gaps. That means right shift will benefit those sensors which have moved left. The authors in [16] also considered the compensation by using similar cost functions for $K = 1$. However, they did not consider the influence of right shift distance to the calculation of $Lcost(\cdot)$. We find that only if the movement strategy considers the shift window effect of the left overlap, the algorithm could work optimally. A counter example is shown in Fig. 2.

In Fig. 2 (a) we give the original positions of 11 sensors. Figure 2 (b) shows the situation when the first 3 gaps are filled, and there is still a gap_7^1 with size g which is greater than *unit* distance a lot. The results are same for our algorithm and MinSum. We can see the left shifts of x_4 and x_6 are very small (Assume x_6 left shift 1 *unit* distance, x_4 left shift 2 *unit*s distance). Now the cost function $Lcost(1,7,1)$ of $overlap_1^1$ equals to $4-2=2$ because 4 positive shifts ($PS_1^1 = \{x_2, x_3, x_5, x_7\}$) and 2 negative shifts ($PS_1^1 = \{x_4, x_6\}$). The cost function $Rcost(7,10,1)$ of $overlap_{10}^1$ equals to 3 ($S_{10}^k = \{x_8, x_9, x_{10}\}$). So $overlap_1^1$ is the cheaper one. Figure 2 (c) shows the result using MinSum which exploits $overlap_1^1$ to fill gap_7^1. Figure 2 (d) shows our algorithm uses the effect moving window. Then $Lcost(1,7,1)$ of $overlap_1^1$ is evaluated again since the negative shift of x_6 changes to positive, thus $Lcost(1,7,1) = 5-1=4$.

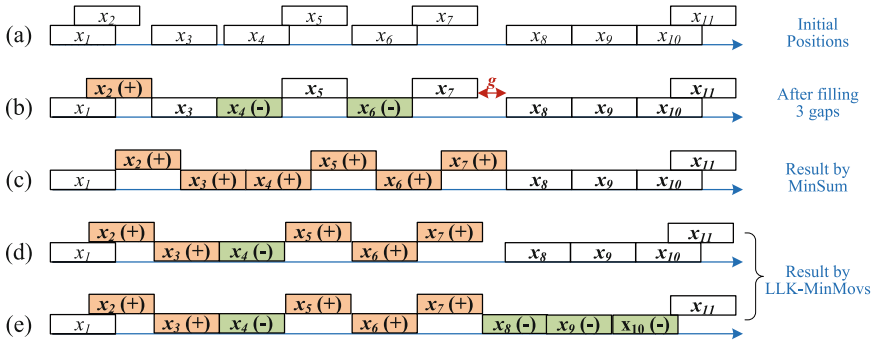


Fig. 2. A counter example to algorithm in [16] (K=1).

So $overlap_{10}^{1}$ is the cheaper one. Figure 2 (e) shows the result after using $overlap_{10}^{1}$ in our algorithm. MinSum fills gap_{7}^{1} with additional $6 \cdot g - 2 \cdot (2+1) \cdot unit = 6 \cdot g - 5 \cdot unit$ movements while LLK-MinMovs just uses additional $3 \cdot g$ movements. Obviously, Our algorithm is better when $g >> unit$. This is because MinSum does not consider the effect shift window which influences the cost calculation.

4.3 Time Complexity of LLK-MinMovs

For the LLK-MinMovs algorithm, there are K loops to cover each level 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 a overlap will disappear, thus the total times of left-movements $T_l \leq T_{ol} + T_{gl}$, where T_{ol} is the times of movement where an overlap disappears and T_{gl} is the times of movement where an gap disappears. In each right-movement, a gap or a overlap will disappear or a sensor is moved back to its initial location. Similarly, we have $T_r \leq T_{or} + T_{gr} + T_b$ where T_b is the times of movement where a sensor is moved back. Since in our algorithm, neither a new gap or a new overlap will occur, we have $T_{gl} + T_{gr} = |gaps|$, $T_{ol} + T_{or} \leq |overlaps|$ and $|gaps| + |overlaps| < n$. On the other hand, since any sensor will not move to the left after moving to the right, $T_b \leq n$. Thus we get the total times of movement $T = T_l + T_r < 2n$. For each movement, we will at most move n sensors, thus the time complexity is $O(Kn^2)$.

5 The Design of LK-KM and LK-KM+ Algorithms

We new design another two algorithms for LK-MinMovs problem based on the famious Hungarian Algorithm. They have better time efficiency in spite of sacrificing the optimality a little bit. But their results are still sub-optimal.

The basic idea is to place the sensors to several fixed points 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 with $2R$ distance between two neighbors.

Then we construct a complete bipartite graph (S, P, E). S represents the sensor set and P represents the virtual fixed point set. Each virtual fixed point has K copies for K-coverage. The edge weight $w(s, p)$ is $L - distance(s, p)$, then the maximum matching for P is the solution of original LK-MinMovs problem. We can use the Kuhn-Munkres algorithm (also known as the famous *Hungarian* algorithm) [14,15] to compute this matching. This algorithm is referred as LK-KM algorithm shown in Algorithm 2.

Algorithm 2. LK-KM

Input: $X[1 \dots n]$, L, K, R
Output: The final positions $X^f[1 \dots n]$ of n sensors

1 Let $S = \{s_1, s_2, \dots, s_n\}$ and $P = \{d_1, d_2, \dots, d_{KL/(2R)}\}$ where d_j represents the point in $R + (i(mod)K) \cdot R$
2 Let $w_{i,j} = L - dis(s_i, v_i)$;
3 Using the Hungarian algorithm to compute a matching for (S, P, E);
4 **for** *each* $w_{i,j}$ *that has been added up* **do**
5 $\quad X[i] \leftarrow R + (i(mod)K) \cdot R$;
6 **return** $X^f[1 \dots n]$

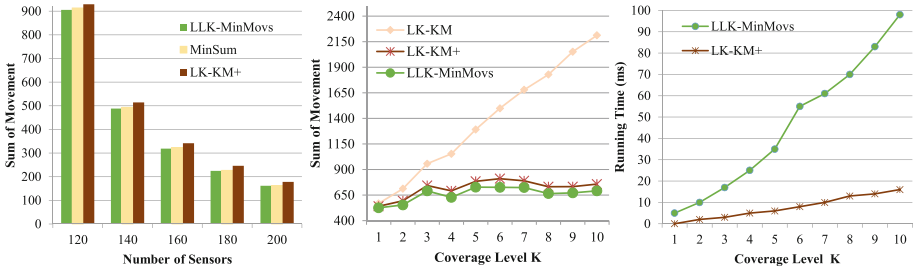
Besides, we enhance the LK-KM algorithm with an idea to pull back the sensors which do not need to go so far away from their original positions because the sensor redundance provides the chance. The line K-coverage enhanced KM algorithm, denoted as LK-KM+ uses $PullToLeft(\cdot)$ and $PullToRight(\cdot)$ are added using a function $move(\cdot)$ for movement back of each sensor. We give an example for function call on $move(\cdot)$ in $PullToLeft$, thus is $move(iK+j, iK+j, min(X[iK+j] - X^f[iK+j], X^f[iK+j+1] - X^f[iK+j], X^f[iK+j] - X^f[iK+j-K])$. It move x_{iK+j} to left in a distance which is shortest one among its shift distance, distance between it and its later neighbor, and distance between it and its K-hops previous neighbor (they should form line K-cover together). The similar procedure in $PullToRight$.

For LK-KM algorithm, the complexity is $O((KL/2R)^2 n)$. 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 $O((KL/2R)^2 n)$, which is still better than LLK-MinMovs.

6 Numerical Experiments

In experiments, we mainly compare our LLK-MinMovs with MinSum, LK-KM, and LK-KM+. They are all implemented in C++. For each case, we ran each algorithm 100 times at random inputs and calculate the average sum of movement. We define the redundance rate as $\frac{LK}{2Rn}$. First of all, we verify that LLK-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. 3 (a). LLK-MinMovs is the best and MinSum is better than LK-KM+.

Next, we study the coverage level K. Here we set $K = 1$ to $10, L/R = 20$ and redundance rate be 0.8. We can see LLK-MinMovs and LK-KM+ get more superiority than LK-KM when coverage level K is increasing. Figure 3(b) gives the results. In term of running time, Fig. 3(c) shows LK-KM+ outperforms LLK-MinMovs much more when coverage level K is increasing.



(a) Line 1-coverage $L = 1000, R = 5$

(b) Experiments on K $L = 1000$, $R = 50$ redundance rate is 0.8

(c) Running Time $L = 1000$, $R = 50$, redundance rate is 0.8

Fig. 3. Comparison On LLK-MinMovs, MinSum and LK-KM+(LK-KM)

Then, we study influence on the result with different sensor redundance rate. Besides, we conduct 3 groups of experiments on different $L/R = 10, 20$ *and* 40. And we set $L = 1000, K = 3$. Figure 4 shows that in all cases LLK-MinMovs has best performance. Three algorithms have the same result at redundance rate 1. Sensors need move less distance when L/R becomes bigger. The LK-KM+ also improves LK-KM very much and is very close to LLK-MinMovs.

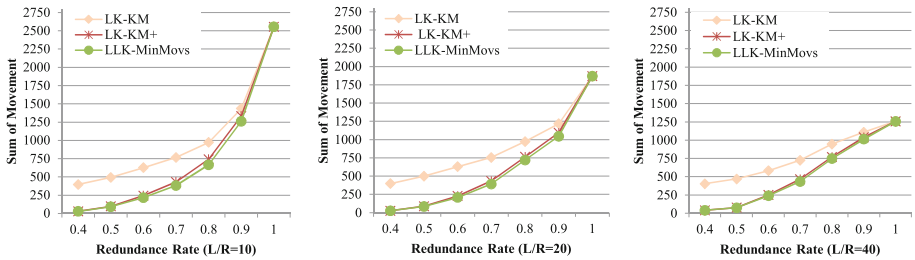


Fig. 4. Experiments on redundance rate and L/R ($L = 1000, K = 3$).

7 Conclusion

In this paper, we address the Line K-coverage problem (LK-MinMovs) in mobile sensor network. We first propose a polynomial time optimal layer-based algorithm LLK-MinMovs. It fixes a critical flaw for the MinSum algorithm designed

in [16] for line 1-Coverage problem. We also designed two sub-optimal but faster algorithms, LK-KM and LK-KM+, based on Hungarian algorithm, which have good time complexity $O(\frac{K^2L^2n}{R^2})$ in comparison with LLK-MinMovs of $O(Kn^2)$ considering n is significantly bigger than K usually.

References

1. Liang, J.B., Liu, M., Kui, X.Y.: A survey of coverage problems in wireless sensor networks. Sens. Transducers **163**(1), 240–246 (2014)
2. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. In: The Annual International Conference on Mobile Computing and Networking (ICMCN), pp. 284–298 (2005)
3. Shen, C., Cheng, W., Liao, X., Peng, S.: Barrier coverage with mobile sensors. In: The IEEE International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), pp. 99–104 (2008)
4. Balister, P., Zheng, Z., Kumar, S., Sinha, P.: Trap coverage: allowing coverage holes of bounded dimeter in wireless sensor network. In: The IEEE International Conference on Computer Communications (INFOCOM), pp. 136–144 (2009)
5. Baumgartner, K., Ferrari, S.: A geometric transversal approach to analyzing track coverage in sensor networks. IEEE Trans. Comput. **57**(8), 1113–1128 (2008)
6. Harada, J., Shioda, S., Saito, H.: Path coverage properties of randomly deployed sensors with finite data-transmission ranges. Comput. Netw. **53**(7), 1014–1026 (2009)
7. Ram, S.S., Manjunath, D., Iyer, S.K., Yogeshwaran, D.: On the path coverage properties of random sensor networks. IEEE Trans. Mob. Comput. **6**(5), 446–458 (2007)
8. Mondal, D., Kumar, A., Bishnu, A., Mukhopadhyaya, K., Nandy, S.C.: Measuring the quality of surveillance in a wireless sensor network. Int. J. Found. Comput. Sci. **22**(4), 983–998 (2011)
9. Li, X., Frey, H., Santoro, N., Stojmenovic, I.: Localized sensor self- deployment with coverage guarantee. ACM SIGMOBILE Mob. Comput. Commun. Rev. **12**(2), 50–52 (2008)
10. Yang, S.H., Li, M.L., Wu, J.: Scan-based movement-assisted sensor deployment methods in wireless sensor networks. IEEE Trans. Parallel Distrib.Syst. **18**(8), 1108–1121 (2007)
11. Hesari, M.E., Kranakis, E., Krizanc, D., Ponce, O.M., Narayanan, L., Opatrny, J., Shende, S.M.: Distributed algorithms for barrier coverage using relocatable sensors. In: The ACM Symposium on Principles of Distributed Computing (SPDC), pp. 383–392 (2013)
12. Bar-Noy, A., Rawitz, D., Terlecky, P.: Maximizing barrier coverage lifetime with mobile sensors. CoRR, vol. abs/1304.6358 (2013)
13. Saipulla, A., Westphal, C., Liu, B., Wang, J.: Barrier coverage with line-based deployed mobile sensors. Ad Hoc Netw. **11**(4), 1381–1391 (2013)
14. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**, 83–97 (1955)
15. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)

16. Czyzowicz, J., Kranakis, E., Krizanc, D., Lambadaris, I., Narayanan, L., Opatrny, J., Stacho, L., Urrutia, J., Yazdani, M.: On minimizing the sum of sensor movements for barrier coverage of a line segment. In: International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW), 6288: 29–42 (2010)
17. Chen, D.Z., Gu, Y., Li, J., Wang, H.: Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain. Discrete Comput. Geom. **50**(2), 374–408 (2013)