Article

Optimizing Movement for Maximizing Lifetime of Mobile Sensors for Covering Targets on a Line

Peihuang Huang [1], Wenxing Zhu [2] and Longkun Guo [2,*]

[1] College of Physics and Information Engineering, Fuzhou University, Fuzhou 350116, China; peihuang.huang@foxmail.com

[2] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China; wxzhu@fzu.edu.cn

* Correspondence: lkguo@fzu.edu.cn

Received: 13 December 2018; Accepted: 8 January 2019; Published: 11 January 2019

Abstract: Given a set of sensors distributed on the plane and a set of Point of Interests (POIs) on a line segment, a primary task of the mobile wireless sensor network is to schedule covering the POIs by the sensors, such that each POI is monitored by at least one sensor. For balancing the energy consumption, we study the min-max line barrier target coverage (LBTC) problem which aims to minimize the maximum movement of the sensors from their original positions to their final positions at which the coverage is composed. We first proved that when the radius of the sensors are non-uniform integers, even 1-dimensional LBTC (1D-LBTC), a special case of LBTC in which the sensors are distributed on the line segment instead of the plane, is $\mathcal{NP}$-hard. The hardness result is interesting, since the continuous version of LBTC to cover a given line segment instead of the POIs is known polynomial solvable. Then we present an exact algorithm for LBTC with uniform radius and sensors distributed on the plane, via solving the decision version of LBTC. We argue that our algorithm runs in time $O(n^2 \log n)$ and produces an optimal solution to LBTC. The time complexity compares favorably to the state-of-art runtime $O(n^3 \log n)$ of the continuous version which aims to cover a line barrier instead of the targets. Last but not the least, we carry out numerical experiments to evaluate the practical performance of the algorithms, which demonstrates a practical runtime gain comparing with an optimal algorithm based on integer linear programming.

Keywords: mobile sensor; $\mathcal{NP}$-hard; target coverage; line boundary; optimal solution

1. Introduction

In the past decades, wireless sensor networks have brought tremendous changes to human society and proposed many technique challenges. Among them, the coverage topics including area coverage [1] and barrier coverage [2] are among the hop spots that attract lots of research interest. In area coverage, the task is to schedule the new positions of the sensors, such that each point in the given target region is covered by at least one sensor. Differently, in barrier cover the task is to monitor only the boundary of a given region, and the aim is to guarantee that intruders can be detected when they are crossing the barrier. Comparing to area coverage, barrier coverage has an advantage of using significantly less sensors and hence is scalable for large scale wireless sensor networks (WSN). Furthermore, some applications only require a set of Points Of Interest (POIs) along the boundary to be monitored. In the context, a problem arises how to guarantee every POI on the barrier to be covered. The current-state-of-art method is to first cover POIs using the stationary sensors, and then use mobile sensors to cover every not-yet covered POI along the barrier. For the second phase, we traditionally have the following assumptions for the modeling: (1) Sensors are acquired with mobile ability; (2) The initial positions of the sensors are distributed on the plane, and the POIs are distributed along a line segment (Although the shape of the boundary can be various, most researches nonetheless focus on line boundary because curves in other shapes can be considered as a variable of line segments); (3) The aim of sensor networks is to prolong the lifetime. This arises the min-max 2D Line Boundary Target Coverage problem (min-max 2D-LBTC) as follows:

Definition 1. *Let P and Γ be respectively a set of POIs distributed in a line segment $[0, L]$ and a set of mobile sensors distributed on the plane, where $j \in P$ has a position $(p_j, 0)$ while $i \in \Gamma$ has a position (x_i, y_i) and a positive sensing radius r_i. The min-max* 2D-LBTC *problem aims to compute a new position $(x'_i, 0)$ for each sensor $i \in \Gamma$, such that each POI $j \in P$ is* covered *by at least one sensor, and the maximum movement of the sensors from their original positions to the new positions is minimized that $\max_{i\in\Gamma}\left\{ \sqrt{(x_i - x'_i)^2 + {y_i}^2} \,\middle|\, i \in \Gamma \right\}$ is attained, where $j \in P$ is* covered *means there exists a sensor $i \in \Gamma$ with position $(x'_i, 0)$ that $x'_i - r_i \le p_j \le x'_i + r_i$.*

When no confusion arises, we shall use LBTC short for the min-max 2D-LBTC problem for briefness. In particular, we use one dimensional min-max Line Boundary Target Coverage problem (1D-LBTC) to denote the special case of LBTC when the initial positions of all the sensors are also distributed on the line boundary. Moreover, the decision version of LBTC (decision LBTC for short) is, for a given movement bound D, to determine whether there exists a feasible coverage with each sensor's movement bounded by D. Besides, when the aim is to cover the line boundary itself instead of the POIs thereon, we respectively have the min-max Line Boundary Coverage (LBC) problem and one-dimensional-LBC (1D-LBC) problem, which have already been well studied and a number of algorithms have been developed. All the notations of this paper are summarized as in Table 1.

1.1. Related Works

To the best of our knowledge, Kumar et al. [2] were the first to consider the boundary (barrier) coverage problem using sensors against a closed curve (i.e., a moat), via transforming the coverage problem to the path problem of determining whether there exists a path between two specified nodes, although the research of barrier coverage started from early 90s due to Gage [3]. The algorithm from Kumar et al. is scalable and can also be extended to solve the k-coverage problem by transforming to the k-disjoint path problem. However, the disadvantage is that it can only be used to determine whether a coverage exists using the deployed *stationary* sensors. A problem for stationary sensors is that, after deployment there might exist no coverage over all POIs. For the case, a state-of-art solution is to employ mobile sensors to fill the gaps between the stationary sensors. In the scenario, the WSN applications would require to maximize the minimum lifetime of the mobile sensors or to minimize the total energy consumption. For the former, the aim is to schedule new positions for the mobile sensors such that the barrier is completely covered, and that the maximum movement of the sensors is minimized as to prolong the lifetime of the WSN. When the sensors are on the line of the barrier, the 1D-LBC problem is shown optimally solvable in $O(n^2)$ time for uniform radii in Paper [4]. The same paper has also proposed an algorithm with $O(n)$ time for uniform radii and $\sum_i r_i \le L$, and with $x_1 \le \cdots \le x_n$ for the sensor $\Gamma = \{s_1, \cdots, s_n\}$, where L is the length of the barrier, n is the number of the sensors. Later, Chen et al have improved the time complexity to $O(n \log n)$ for uniform sensor radii and proposed an $O(n^2 \log n)$ time algorithm for non-uniform radii in paper [5]. Besides straight line barrier, circle/simple polygon barriers has been studied and two algorithms have been given developed by Bhattacharya et al. in [6], which have an $O(n^{3.5} \log n)$ time relative to cycle barriers and an $O(mn^{3.5} \log n)$ time relative to polygon barriers, in which m is the number of the edges on the polygon. The later time complexity was then decreased to $O(n^{2.5}\log n)$ in [7]. For the more generalized case in which the sensors are distributed on the plane, the LBC problem is known to be strongly $\mathcal{NP}$-hard for sensors with general integer sensing radius [8], while LBC using uniform radius sensors is shown solvable in $O(n^3\log n)$ time [9]. Although these elegant algorithms have been developed for LBC for both 1D and 2D setting, none of them is applicable to LBTC since as we shall show in the paper, LBTC is $\mathcal{NP}$-complete for non-uniform radius.

Other than the Min-Max case, there are also applications require min-sum coverage that is to minimize the total energy consumption, which is to minimize the total movement of the mobile sensors. For this objective, Min-Sum LBC, which aims to minimize the sum of the movements of all the sensors, were studied in literature. Min-Sum LBC was shown $\mathcal{NP}$-complete for arbitrary radii while solvable in time $O(n^2)$ for uniform radii by Czyzowicz et al. [10]. The Min-Num relocation problem of minimizing the number of sensors moved, is also proven $\mathcal{NP}$-complete for arbitrary radii and polynomial solvable for uniform radii by Mehrandish et al. [11]. A PTAS has been developed for the Min-Sum relocation problem against circle/simple polygon barriers by Bhattacharya et al. [6], which was later improved to an $O(n^4)$ time exact algorithm by Tan and Wu [7]. For covering a barrier with Min-Sum movement, the most recent result is a factor-$\sqrt{2}$ approximation algorithm for covering POIs along a barrier using uniform-radius sensors, aiming to minimize the sum of the movement [12]. However, it remains open whether the min-sum LBC problem is $\mathcal{NP}$-hard. For target coverage task in the plane, Liao et al. have develop algorithms for Min-Sum movement to minimize the total consumed energy [13].

Table 1. Notations used in the paper.

Notations	**Description**
LBTC	Brief for the min-max 2D Line Boundary Target Coverage problem
Decision LBTC	The problem that only determines whether LBTC is feasible under given D
LBC	Brief for the min-max 2D Line Boundary Coverage Problem
1D-LBTC	LBTC but with the original positions of the sensors on the line
1D-LBTC	LBC but with the original positions of the sensors on the line
POI	Points Of Interest which are the targets to be covered
3-partition	A combinatorial optimization problem that is know strongly $\mathcal{NP}$-complete
L	The length of the line segment where the POIs are placed
D	The bound of the maximum movement of the sensors
D^*	The minimum one among all feasible D
P	The set of POIs
p_j	The position of $j \in P$ on the line segment
Γ	The set of sensors
d_{max}	The maximum distance between the POIs and the sensors
d_{ij}	The distance between POI j and sensor i
l_i	The leftmost point sensor i can cover under the given movement bound D
g_i	The rightmost point sensor i can cover under the given movement bound D
$l(C)$	The leftmost POI of the POI set C
$g(C)$	The rightmost POI of the POI set C
Φ	The possible maximal sets of POIs (called combinations) that can be covered by a single sensor
Ψ	$\Psi = \{d_{ij} \mid i \in \Gamma, c_j \in \Phi\}$ is the set of distances between the combinations and the sensors

1.2. Our Results

In this paper, we first show that 1D-LBTC is $\mathcal{NP}$-hard when the sensors are with non-uniform integer radii by proposing a reduction from the 3-partition problem that is known strongly $\mathcal{NP}$-complete [14]. This hardness result is interesting, because 1D-LBC, the continuous version of 1D-LBTC, is shown solvable in polynomial time $O(n^2 \log n)$ [5]. This means LBTC and LBC belong to different classes of computational complexity.

Then, we propose a sufficient and necessary condition to determine whether there exists a feasible cover for the barrier under the relocation distance bound D. Based on the condition, we propose a simple greedy approach that outputs "infeasible" if $D < D^*$, and otherwise computes a feasible solution under the movement bound D, such that the sensors cover all the POIs in their new positions. We show that the decision algorithm is with a runtime $O(n \log n)$. By employing the binary search technique, we propose an algorithm using the decision algorithm as a routine to actually find a minimum integer movement bound $D = D^*$, when D^* is integral. The algorithm takes $O(n \log n \log d_{max})$ time, where d_{max} is the maximum distance between the sensors and the POIs.

For instances with D^* being a real number or with large d_{max}, we propose another algorithm that employs the binary search method against $O(n^2)$ possible values of D^* instead of the continuous value range. The trick is to construct the set of all possible values of D^* and show its size is $O(n^2)$. This improves the runtime of the algorithm to $O(n^2 \log n)$, which is the time needed to sort the $O(n^2)$ possible values of D^*. The later algorithm remains correct even when D is allowed to be any real number. In contrast, the former algorithm can only work for integer D^*. To the best of our knowledge, our algorithms are the first polynomial algorithms for LBTC.

The following paragraphs will be organized as follows: we shall first give the $\mathcal{NP}$-completeness proof in Section 2; Then present the algorithm for Decision LBTC with uniform sensor radii together with the correctness proof in Section 3; Next, actually solve the LBTC problem by employing the binary search method first against a continuous range, and then over our proposed discrete set in Section 4; After that, evaluate the proposed algorithms via experiments in Section 5; At last, conclude the paper in Section 6.

2. $\mathcal{NP}$-Completeness of Decision 1D-LBTC

In this section, we shall show the decision LBTC problem is $\mathcal{NP}$-complete when the sensors are with non-uniform integer radii, by giving a reduction from the 3-partition problem that is known strongly $\mathcal{NP}$-complete [14]. In 3-partition, we are given a set of $3n$ integers $\mathcal{U} = \{a_1, \ldots, a_{3n}\}$ with $\sum_{i=1}^{3n} a_i = Bn$ for an integer $B > 0$. The aim is to determine whether $\mathcal{U}$ can be divided into n subsets, such that each subset is with an equal sum B.

Theorem 1. *Decision 1D-LBTC is NP-complete when the sensors are with non-uniform integer radii.*

The key idea of the construction of a reduction from 3-Partition to the decision LBTC problem is to model $a_i \in \mathcal{U}$ as the diameter of the sensors, and place POIs on the line segment in a way that a coverage of the POIs is actually a partition of the numbers in $\mathcal{U}$. More detailed, for a given instance of 3-Partition, the construction of the corresponding instance of decision LBTC is simply as below:

1. Construct a line segment with length $(2n-1)B$;
2. Place $4nB$ POIs on the line barrier as below:
    (a) Decompose the line segment into $2n-1$ subsegments with equal length;
    (b) Select n sections from the subsegments, where the ith section is the $(2i-1)$th subsegment;
    (c) **For** the ith *section*, $i = 1, \ldots, n$, **do**

```
        For $j = 0, \ldots, B-1$ do
            Put two POIs respectively to the two positions $(2(i-1)B + j + j\epsilon, 0)$ and
            $(2(i-1)B + j + 1 - (B-j)\epsilon, 0)$, where $\epsilon$ is a small positive number;
        Endfor
    Endfor
```

3. Place $3n$ sensors on position $(0, 0)$, where sensor i is with radii $\frac{a_i}{2}$;
4. The maximum movement is set as $D := (2n-1)B$.

An example of the above construction is as depicted in Figure 1. Note that, the instance of decision 1D-LBTC constructed above contains $2nB$ POIs and $3n$ sensors. Anyhow, 3-Partition is known strongly $\mathcal{NP}$-complete, which means, 3-Partition remains $\mathcal{NP}$-complete even when B is polynomial to n. Therefore, the construction can be done in polynomial time for B being polynomial to n.

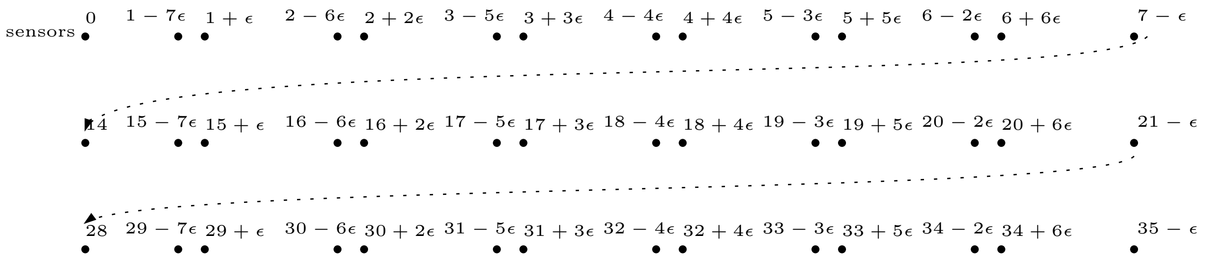


Figure 1. An example of the construction of 1D-LBTC against a 3-Partition instance $\mathcal{U} = \{1, 1, 2, 2, 2, 3, 3, 3, 4, \}$, $n = 3$ and $B = 7$. There are two sensors with diameter 1, one with diameter 4, three with diameter 2 and the other three with diameter 3, whose original positions are all on $(0, 0)$. The movement bound is set $D = 35$.

The main idea behind the construction is to construct a relationship between the number of covered POIs and the diameters of the sensors that are actually the integers in $\mathcal{U}$. More precisely, the property on the relationship is as in the following:

Proposition 1. *Against a 1D-LBTC instance produced by the above construction, a sensor with diameter $2r$ can cover at most $4r$ POIs.*

Proof. When a sensor is with a diameter 2, apparently it can cover at most 4 POIs. Suppose the proposition is true for sensors with diameter smaller than $2r$. Then, let $r_1 + r_2 = r$ be two positive integers smaller than r. By induction, we have that sensors with diameters $2r_1$ and $2r_2$ can cover upto $4r_1$ and $4r_2$ POIs, respectively. In addition, the two sensors with radii r_1 and r_2 can cover as many POIs as a sensor with a radii $r = r_1 + r_2$ does. Therefore, the sensor with diameter $2r$ can cover no more than $4r_1 + 4r_2 = 4r$ POIs. This completes the proof. □

Lemma 1. *An instance of 3-Partition is feasible if and only if the corresponding 1D-LBTC instance is feasible.*

Proof. Suppose the instance of 3-Partition is feasible. Without loss of generality, we assume that $\{U_i | i = 0, \ldots, n-1\}$ is a solution to the 3-Partition instance which divides $\mathcal{U}$ to a collection of n sets, among which $U_i = \{a_{l_i+1}, \ldots, a_{l_{i+1}}\}$ and $l_0 = 0$. Since $D = (2n-1)B$ equals the length of the barrier and the original position of each sensor is $(0, 0)$, each sensor can be moved any point of the barrier. Then we need only to use the sensors in U_i, which are with radius $a_{i_j}, \ldots, a_{i_{j+1}}$ and with a sum exactly B, to cover the segment from $2iB$ to $(2i+1)B$. That apparently results in a coverage for all the POIs in the ith section.

Conversely, suppose the corresponding LBTC instance is feasible. Then since sensor j with radii $\frac{a_j}{2}$ can at most cover $2a_j$ continuous POIs, and each section contains exactly $2B$ POIs, so the diameter sum of the sensors for each section is at least B. Then because the diameter sum of all the sensors is Bn, and there are n sections, the diameter sum of the sensors for each section is exactly B. Therefore, the diameters for the sensors for the sections is a solution to the corresponding instance of 3-Partition. □

From the fact that 3-Partition is strongly $\mathcal{NP}$-complete, and following a similar idea of the above proof for Theorem 1, we immediately have the following hardness for LBTC:

Corollary 1. *Decision 1D-LBTC is strongly NP-complete.*

3. A Greedy Algorithm for 2D-LBTC with Uniform Sensors

The basic idea of the algorithm is to cover the POI from left to right, preferably using sensors that are likely less useful for later coverage. More precisely, let $[l_i, g_i]$ be the possible coverage range of sensor i, where l_i and g_i are respectively the positions of the leftmost and the rightmost POIs, with respect to the given distance D. That is, l_i and g_i are the leftmost and the rightmost positions of POIs sensor i can cover within movement D. Then the key idea of our algorithm is to cover the POIs from left to right, using the sensor that can cover the leftmost uncovered POI within movement D and is with minimum g_i.

The algorithm is first to compute its possible coverage range $[l_i, g_i]$ for each sensor i with respect to the movement constraint D. Apparently, $(x_i, 0)$ is the projective point of sensor i on the line, so we have $l_i = x_i - \sqrt{D^2 - y_i^2}$ and $g_i = x_i + \sqrt{D^2 - y_i^2}$ for each sensor i. Then, the algorithm starts from point $s = (0, 0)$, to cover the line from left to right. The algorithm prefers using the sensor with a small g_i, since a sensor with a large g_i would has a better potential to cover the POIs on the right part of the line.

Let s be the position the uncovered leftmost POI on the line barrier. Then among the set of sensors $\{i | l_i \leq s \leq g_i\}$, the algorithm repeats selecting the sensor with minimum g_i to cover the uncovered POIs of the line barrier starting at s. Note that $\{i | l_i \leq s \leq g_i\}$ is exactly the set of sensors that can monitor a set of uncovered POIs starting at s by relocating at most D distance. The algorithm terminates either the set of POIs are completely covered, or the instance is found infeasible (i.e., there exists no unused sensor i with $l_i \leq s \leq g_i$ while the coverage is not yet done). The algorithm is formally as in Algorithm 1.

Note that Algorithm 1 takes $O(n)$ time to compute l_i and g_i for all the sensors in Steps 2–3, and takes $O(n \log n)$ time to assign the sensors to cover the targets on the line barrier in Steps 4–15. Therefore, we have the time complexity of the algorithm:

Lemma 2. *Algorithm 1 runs in time $O(n \log n)$.*

Before proving the correctness of Algorithm 1, we need the following lemma stating the existence of a special coverage for a feasible LBTC instance.

Proposition 2. *Let (x_j, y_j) be the position of sensor j in the plane. Assume $p_1(s, 0)$, $p_2(x'_j, 0)$ and $p_3(x''_j, 0)$ are three points on a line segment. If $s \leq x''_j \leq x'_j$, then $d(j, p_3) \leq \max\{d(j, p_1), d(j, p_2)\}$ holds. That is, the distance between the sensor and the middle point is not larger than the larger distance between the sensor and the other two points.*

Lemma 3. *If an instance of LBTC is feasible, then there must exist a coverage in which the sensors are s-ordered.*

Algorithm 1 A greedy algorithm for decision LBTC

```
Input: A bound $D \in \mathbb{Z}^+$ on maximum movement, a set of sensors $\Gamma = \{1, \dots, n\}$ with original
    positions $\{(x_i, y_i)|i \in [n]^+\}$ and $r$ being the sensing radii, a set of POIs $P = \{1, \dots, m\}$ with
    positions $p_1 \preceq p_2 \preceq \cdots \preceq p_m$, where $p_j$ is the position for $j \in P$ ;
Output: New positions $\{x'_i|i \in [n]^+\}$ for the sensors or return "infeasible".
 1: Set $\mathcal{I} := \Gamma$ and $s := p_1$, where $s$ is the leftmost point of the uncovered part of the barrier.
 2: For each sensor $i$ do
 3:    Compute the leftmost position $l_i$ and the rightmost position $g_i$ that sensor $i$ can monitor;
 4: EndFor
 5: While $\mathcal{I} \neq \emptyset$ do
 6:    If there exists $i' \in \mathcal{I}$, such that $l_{i'} \leq s \leq g_{i'}$ then
 7:       Select the sensor with minimum $g_i$ among all the sensors $\{i'|l_{i'} \leq s \leq g_{i'}\}$, which is to find
            sensor $i \in \mathcal{I}$ that $g_i = \min_{i':\, l_{i'} \leq s \leq g_{i'}} \{g_{i'}\}$;
 8:       Set $t := \min\{s + 2r, g_i\}$ and $x'_i := t - r$;
 9:       Set $s := \min\{p_j|p_j > t\}$ and $\mathcal{I} := \mathcal{I} \setminus \{i\}$;
10:    Else
11:       Return "infeasible";
12:    Endif
13:    If $t \geq p_m$ then /*All POIs have been covered. */
14:       Return "feasible" together with the new positions $\{x'_i|i \in \Gamma\}$;
15:    Endif
16: Endwhile
```

Proof. The key idea of the proof is that, any coverage of LBTC that is not s-ordered, can be converted to an s-ordered coverage by re-scheduling the sensors of covering the POIs.

Suppose there exist two sensors i and j, such that $g_i > g_j$ but $x'_i < x'_j$. Then we need only to swap the final positions of i and j, i.e., to simply set the new final positions x''_i and x''_j of sensor i and j as below:

```
If $x'_i - r \geq s$ then
   Set $x''_i := x'_j$ and then $x''_j := x'_i$;
Else
   Set $x''_i := x'_j$ and $x''_j = s + r$.
EndIf
```

Apparently, the POIs exclusively covered by i are now covered by sensor j, and *vice versa*. So after the swap the sensors will remains a coverage for the POIs on the line. It remains to show the swap will not increase the maximum movement. Recall that the leftmost and the rightmost points sensor j can cover are respectively l_j and g_j. Because sensor j can move to x'_j under the movement bound D, we have

$$l_j \leq x'_j - r \leq x'_j + r \leq g_j \leq g_i. \tag{1}$$

On the other hand, in either case of the swap, we have $x''_i = x'_j \geq x'_i$. So combining Inequality (1), we have $l_i \leq x''_i - r \leq x''_i + r \leq g_i$. That means

$$l_i + r \leq x''_i \leq g_i - r.$$

Then following Proposition 2, the distance between sensor i and its new position x''_i is bounded by $D = \max\{d(i,\ (l_i + r,\ 0)),\ d(i,\ (g_i - r,\ 0))\}$. The case for the new position of sensor j is similar except that the distance between sensor j and its new position x''_i is bounded by $D = \max\{d(j, (\max\{s, l_j + r\}, 0)), d(i, (g_i - r, 0))\}$. This completes the proof. □

Based on Lemma 3, given a feasible instance of LBTC, we can assume there exists an s-ordered coverage, say $\Gamma' = \{s_1, \ldots, s_k\}$ which is the set of sensors used to compose the coverage with j_i being the rightmost POI covered by s_i. Then we have the following lemma, which leads to the correctness of the algorithm:

Lemma 4. *When running against a feasible LBTC instance, Algorithm 1 covers the POIs $\{1, \ldots, j_i\}$ without using any sensor in $\{s_{i+1}, \ldots, s_k\}$.*

Proof. We shall prove this claim by induction. When $i = 1$, the lemma is obviously true, as we need only s_1 to cover the POIs $\{1, \ldots, j_1\}$. Suppose the lemma holds for $i = h$, then it remains only to show the case for $i = h + 1$. By induction, Algorithm 1 covers the POIs $\{1, \ldots, j_h\}$ without using any sensor in $\{s_{h+1}, \ldots, s_k\}$. Then Algorithm 1 can simply cover POIs $\{j_h + 1, \ldots, j_{h+1}\}$ by using sensor s_{h+1}. Combining with the induction, we covers $\{1, \ldots, j_{h+1}\}$ without using any sensor in $\{s_{h+2}, \ldots, s_k\}$. This completes the proof. □

We can now prove the following theorem and have the correctness of Algorithm 1:

Theorem 2. *Algorithm 1 returns "feasible" iff the POIs can be completely covered by the sensors within relocation distance D.*

Proof. Suppose Algorithm 1 returns "feasible", then obviously the produced solution $\{x'_i | i \in \Gamma\}$ is truly a coverage, because in the solution the movement of each sensor is bounded by D and all the POIs are covered by at least one sensor.

Conversely, suppose there is a coverage for the instance. Then by Lemma 3, there must exist an s-ordered coverage, say $\Gamma' = \{s_1, \ldots, s_k\}$ which is the set of sensors used to compose the coverage. Following Lemma 4, Algorithm 1 covers POIs $\{1, \ldots, j_i\}$ without using any sensor in $\{s_{i+1}, \ldots, s_k\}$ for every $i \in [1, k]$. So the algorithm can always find sensors for further coverage, and in the worst case use s_{i+1} to cover the POIs $\{j_i + 1, \ldots, j_{i+1}\}$. Therefore, the algorithm will eventually find a feasible coverage. This completes the proof. □

4. The Complete Algorithms

In this section, we will show how to employ Algorithm 1 to really compute D^* the minimum movement bound for LBTC. Firstly, when only considering integer D^*, we can find it simply by employing the binary search method against a large range that contains D^*; Secondly, for real number D^*, we construct a set of size $O(n^2)$ which arguably contains D^*, and then eventually finds D^* in the set again by the binary search method.

4.1. A Simple Binary Search Based Algorithm

The algorithm is simply applying the binary search method to find D^* within the range of $[1, d_{max}]$, where d_{max} is the maximum distance between the POIs and the sensors. The main observation is as the following proposition whose correctness is easy to prove:

Proposition 3. *If LBTC is feasible, then $D^* \leq d_{max}$ holds.*

The detailed algorithm is as in Algorithm 2.

Algorithm 2 The whole algorithm for optimal LBTC.

```
Input: A movement bound $D \in \mathbb{Z}^+$, a set of sensors $\Gamma = \{1, \ldots, n\}$ with positions $\{(x_i, y_i)|i \in [n]^+\}$,
    the sensing radii $r$, a set of POIs $P = \{1, \ldots, m\}$ with positions $p_1 \preceq p_2 \preceq \cdots \preceq p_m$, where $p_j$ is
    the position for $j \in P$;
Output: The minimized maximum movement of the sensors together with new positions $\{x'_i|i \in [n]^+\}$.
 1: Set $ub := d_{max}$ and $lb := 1$, where $d_{max}$ is the maximum distance between the sensors and the POIs;
 2: If there exists no coverage returned from calling Algorithm 1 wrt $D = d_{max}$ then
 3:     Return "infeasible";
 4: EndIf
 5: Set $tmp := \left\lceil \frac{lb+ub}{2} \right\rceil$;
 6: While $ub - lb > 1$ do
 7:     If there exists no coverage returned from calling Algorithm 1 wrt $D = tmp$ then
 8:         Set $lb := tmp$ and $tmp := \left\lceil \frac{lb+ub}{2} \right\rceil$;
 9:     Else
10:         Set $ub := tmp$ and then $tmp := \left\lceil \frac{lb+ub}{2} \right\rceil$;
11:     EndIf
12: EndWhile
13: Return the result of call Algorithm 1 wrt $D = tmp$.
```

For the correctness and time complexity of Algorithm 2, we immediately have the following lemma:

Lemma 5. *Using binary search and employing Algorithm 1 for $O(\log d_{max})$ times, Algorithm 2 will compute the optimum movement D^* within time complexity $O(n \log n \log d_{max})$.*

4.2. An Improved Algorithm via Discrete Binary Search

In this subsection, we shall show the time complexity of our algorithm can be further improved via a more sophisticated implementation over the binary search. The key observation is that, we need only to apply a binary search over a set of discrete values which arguably contain the optimum min-max movement D^*. Let $\Phi = \{c_1, \ldots, c_t\}$ be the set of possible combinations, that is, all possible different combinations covered by a sensor (An example of combinations is as depicted in Figure 2). Let d_{ij} be the minimum movement using sensor i to cover combination c_j. Then we have the following lemma:

Lemma 6. *Let d_{opt} be an optimal solution to the uniform 2D-LBTC problem. Then $d_{opt} \in \Psi = \{d_{ij}|i \in \Gamma, c_j \in \Phi\}$.*

Proof. Suppose the lemma is not true. Then let $d_{max} = \max_d\{d \mid d \in \Psi, d < d_{opt}\}$. First we show that under maximum distance d_{max} and d_{opt}, every sensor i covers an identical collection of combinations. That is because every POI, which sensor i can cover under movement bound d_{opt}, can also be covered by sensor i under movement bound d_{max} (as $d_{ij} \leq d_{max}$ iff $d_{ij} < d_{opt}$), and conversely every POI, which cannot be covered by sensor i under d_{max}, can not be covered by the same sensor within the movement bound d_{opt} ($d_{ij} > d_{max}$ iff $d_{ij} > d_{opt}$). Therefore, a feasible coverage solution under maximum movement d_{opt} would also remain feasible under d_{max}. This together with $d_{max} < d_{opt}$ contradicts with the fact that d_{opt} is an optimal solution to the problem. □

Following the above lemma, our algorithm will first compute $\oplus = \{c_1, \ldots, c_t\}$ the set of possible combinations; Then compute all the distances between each sensor in Γ and each combination in $\mathcal{C}$, which is $\Psi = \{d_{ij}|i \in \Gamma, c_j \in \Phi\}$; Later sort the distance in Ψ in non-decreasing order and apply the binary search method to Ψ by using Algorithm 1 as a subroutine. Eventually, the algorithm finds a minimum

$d^*_{ij} \in \Psi$, such that by setting the bound the maximum movement $D = d^*_{ij}$ there exists a relocation of the sensors to cover all the POIs.

According to the main structure of the complete algorithm above, we shall first compute $\oplus$. The key idea of the computation is to find all the pairs of POIs which can be the first and the final POIs covered by a sensor. The most important part of the computation is to exclude those pairs of POIs, say i and j, for which $i-1$ and $j+1$ are too close, i.e., $d(i-1, j+1) \leq 2r$. In the case, to cover i and j, either $i-1$ or $j+1$ will be also covered, and hence i and j can not respectively be the first and the final POIs covered by a sensor. The detailed algorithm for computing Φ and the complete algorithm are formally stated in Algorithms 3 and 4.

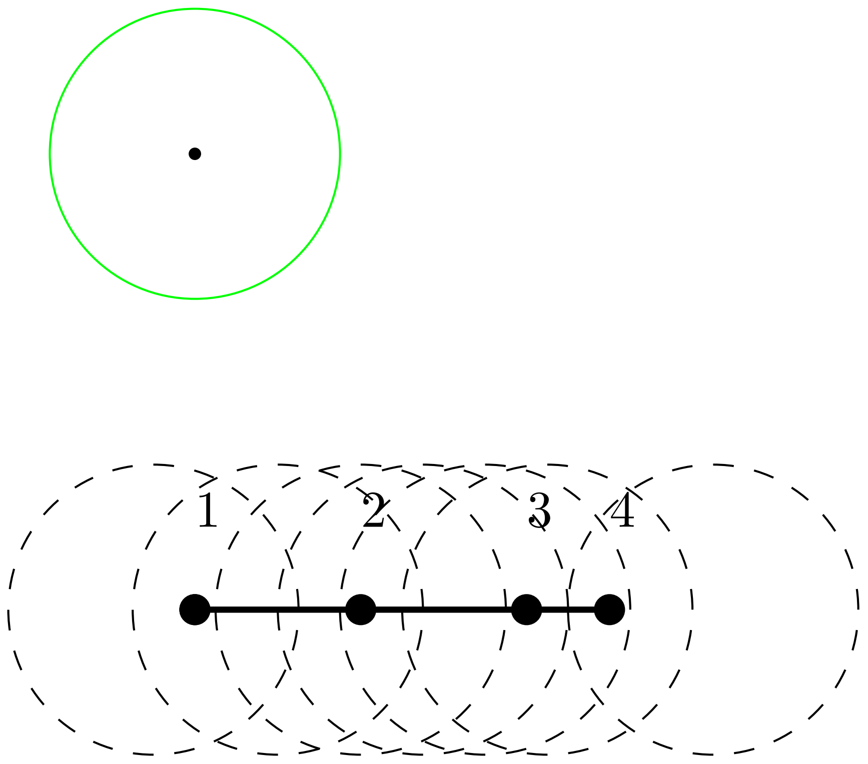


Figure 2. An example of combinations: For POIs on the line segment and the radius of sensors (in green) as depicted, the collection of combinations is $\Phi = \{\{1\}, \{1, 2\}, \{2\}, \{2, 3\}, \{2, 3, 4\}, \{3, 4\}, \{4\}\}$.

Algorithm 3 A fast algorithm for constructing combinations.

```
Input: A set of POIs $P = \{1, \ldots, m\}$ on the line segment with positions $p_1 \preceq p_2 \preceq \cdots \preceq p_m$,
    where $(p_j, 0)$ is the position for $j \in P$ , and a real number $r$ that is the radii of the sensors;
Output: The set of combinations $\Phi := \{c_1, \ldots, c_t\}$.
 1: Set $\Phi := \emptyset$, $p_0 := -2r$ and $p_{m+1} = p_m + 2r$;
 2: For POI $j = 1$ to $|P|$ do
 3:     For $i := j$ to $|P|$ do
 4:         If $p_i - p_j \leq 2r$ and $p_{i+1} - p_{j-1} \geq 2r$ then
 5:             $\Phi := \Phi \cup \{\{j, \ldots, i\}\}$;
 6:         EndIf
 7:     EndFor
 8: EndFor
 9: Return $\Phi$ and terminate.
```

Algorithm 4 A fast algorithm for LBTC.

```
Input: A set of sensors $\Gamma = \{1, \ldots, n\}$ with original positions $\{(x_i, y_i)|i \in [n]^+\}$ and an identical
    sensing radii $r$, a set of POIs $P = \{1, \ldots, m\}$ with positions $p_1 \preceq p_2 \preceq \cdots \preceq p_m$ on the line
    segment, where $p_j$ is the position for $j \in P$;
Output: A minimum movement $D$ under which the sensors can be relocated to covered all POIs in $P$.
 1: Set $\Psi := \emptyset$ and compute $\Phi := \{c_1, \ldots, c_t\}$ the collection of combinations by Algorithm 3;
 2: For each sensor $i$ do
 3:    For each combination $c_j \in \Phi$ do
 4:       Compute $d_{ij}$, the minimum movement needed when using sensor $i$ to cover $c_j$;
 5:       Set $\Psi := \Psi \cup \{d_{ij}\}$;
 6:    EndFor
 7: EndFor
 8: Sort $\Psi$ in a non-decreasing order and set $lb := 1$ and $ub := |\Psi|$, to prepare for binary search.
 9: Use $\Psi[1]$ as the movement bound (i.e., $D$) and call Algorithm 1;
    /*Note that $\Psi[1]$ is the smallest element in $\Psi$. */
10: If there exists a feasible coverage under movement bound $\Psi[1]$ then
11:    Return $\Psi[1]$ as the optimum movement bound;
12: Endif
13: While $ub - lb > 1$ do
14:    Set $idx := \left\lceil \frac{lb+ub}{2} \right\rceil$;
15:    Use $\Psi[idx]$ as the movement bound (i.e., $D$) and call Algorithm 1;
       /*$\Psi[idx]$ is the $idx$th smallest element in $\Psi$. */
16:    If there exists a feasible coverage under movement bound $\Psi[idx]$ then
17:       Set $ub := idx$;
18:    Else
19:       Set $lb := idx$;
20:    Endif
21: Endwhile
22: Return $\Psi[idx]$ as the optimum movement bound.
```

Lemma 7. *Algorithm 3 returns* Φ with $|\Phi| = O(m)$*.*

Proof. W.l.o.g. assume the combinations in Φ are sorted, such that for each c_i and $c_{i+1} \in \Phi$, $l(c_i) \leq l(c_{i+1})$ and $g(c_i) \leq g(c_{i+1})$, where $l(c_i)$ and $g(c_i)$ are respectively the leftmost and rightmost POIs in c_i. Then, following the construction of the combinations as in Algorithm 3, if a POI j appears in c_i and $c_{i+\Delta}$, then for any $k \in [i, i+\Delta]$, $j \in c_k$. That is, a POI can result in at most two different combinations, one when the POI is added and the other when the POI is removed. Therefore, there are at most $2m$ combinations in Φ. □

Lemma 8. *The time complexity of Algorithm 4 is $O(mn(\log m + \log n))$.*

Proof. From Lemma 7, $|\Phi| = O(m)$ holds, so we apparently have $|\Psi| = O(mn)$. Then, it takes $O(|\Psi| \log |\Psi|) = O(mn \log mn) = O(mn(\log m + \log n))$ time to sort the elements in Ψ, provided merge sort is used [15]. Besides, the while-loop from Step 12 to Step 20 will be repeated for at most $O(\log m + \log n)$ times, each of which takes $O(n \log n)$ time to run Algorithm 1. Therefore, the total time complexity of the algorithm is $O(mn(\log m + \log n))$. □

Following Lemma 6, we immediately have the correctness of Algorithm 4 as below:

Theorem 3. *Algorithm 4 produces an optimum solution to the LBTC problem.*

5. Experiments

In this section, we shall numerically evaluate the practical performance of both Algorithm 2 (the simple Algorithm based on Binary Search, denoted as ABS) and Algorithm 4 (the Algorithm employing Binary Search over a specially constructed Discrete set, denoted as ABSD) by comparing its runtime and practical solution quality with an optimal algorithm baseline—an exact algorithm by solving an Integer Linear Programming (ILP) formulation which produces optimal solution for LBTC. The experiments are carried out on a platform with Intel I5 CPU and 8G DDR3 Memory, using Windows 10 Operating system, and the algorithms are implemented in C++ (The code is available upon request) adopting the *CPLEX* library (https://www.ibm.com/analytics/cplex-optimizer) to solve the ILP.

5.1. An ILP Formulation for LBTC

Let variable x_{ij} represents whether combination j is covered by sensor i and D be the maximum movement. Then we have the ILP formulation for LBTC as below (Denoted as ILP (1)):

$$\begin{array}{lll} \min & D & \\ s.t. & \sum_{i\in\Gamma} x_{ij} \geq 1 & \forall j \in \Phi \qquad (2) \\ & \sum_{j\in\Phi} x_{ij} \leq 1 & \forall i \in \Gamma \qquad (3) \\ & \sum_{j\in\Phi} x_{ij} d_{ij} \leq D & \forall i \in \Gamma \qquad (4) \\ & x_{ij} \in \{0, 1\} & \forall i \in \Gamma, \forall j \in \Phi \qquad (5) \end{array}$$

In the above ILP formula, Inequality (2) guarantees that each POI is covered; Inequality (3) ensures that each sensor is used for at most once; Inequality (4) means D is the maximum movement. It is then easy to obtain the following property:

Lemma 9. *An optimal solution to ILP (1) is an optimum solution to LBTC, and vice versa.*

5.2. Numerical Experiments

In the numerical experiments, we shall first evaluate the runtime of the algorithms with n sensors distributed in the plane, where the radius of the sensors are set 10, the number of sensors n is set in a range $[100, 900]$, and the positions of the sensors are uniformly and randomly distributed in a rectangle along the line segment of a length L that is set proportional to the number of the sensors. The POIs are randomly and uniformly distributed on the line segments, and are with a number proportional to the number of the sensors.

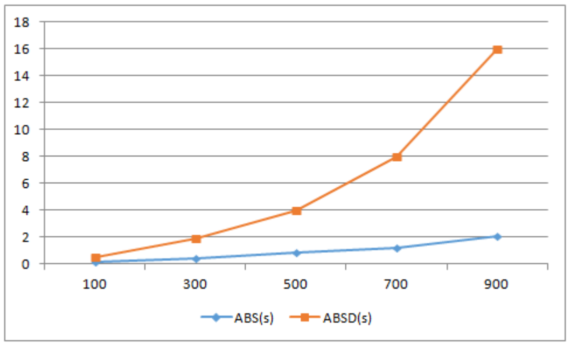
In both Figures 3 and 4, we compare the runtime of ABS, ABSD and ILP in seconds. For each $n \in \{100, 300, 500, 700, 900\}$, to obtain the runtime, we produced 1000 different instances of LBTC, against which we respectively ran the three algorithms and used the average runtime of the 1000 runs as the final runtime. Note that as we shall choose $L = n * r/4$, the generated LBTC instances are all feasible, i.e., in every instance, all the POIs can be completely covered.

In Figure 3, the experiments compare the runtimes of ABS and ABSD for different d_{max}. When d_{max} is small, i.e., when the rectangle where the sensors distribute is $60 * L$ along the line segment, the runtime of ABS is much better than ABSD (i.e., about 8 times faster as depicted in Figure 3a). However, when d_{max} is large (i.e., when the rectangle is $10{,}000 \times 5L$), the runtime of ABSD then is very close to ABS that about 1.9 times of that of ABS as depicted in Figure 3b. This result fits the time complexity $O(n \log n \log d_{max})$ and $O(mn(\log m + \log n))$, where m, n and d_{max} are respectively the numbers of the POIs, sensors and the maximal distance between the POIs and the sensors. In addition, the runtime of ABS depends on d_{max}

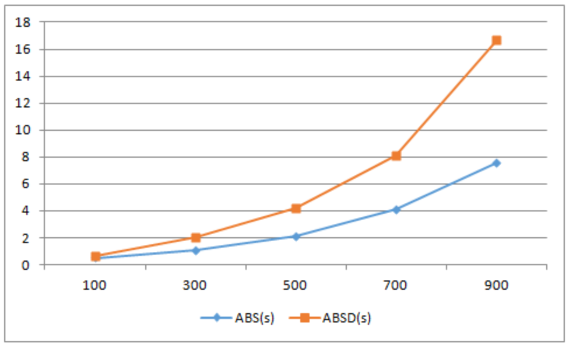
while ABSD is insensitive to the distance between the sensors and the POIs, which again coincides with the theoretical runtime analysis of ABSD. When d_{max} grows, the runtime gap between ABS and ABSD shrinks.

As depicted in Figure 4, when the problem size grows, the runtime of ABS and ABSD both grows slow comparing to ILP whose runtime grows fast that it becomes nearly not applicable when there are more than 500 sensors (its runtime is more than thousands of seconds (The runtime here is the average of only several runs instead of 1000 runs, because it is too large.) as depicted both in Figure 4a,b). Notably, the runtime of ABS and ABSD are both under 5 s at 500. So both ABS, ABSD have a significant runtime advantage comparing to the ILP baseline. Also, the runtime of ILP is insensitive to the distance between sensors and POIs, as the runtime are almost the same for both instances of small and large d_{max}.

Note that, although the ABS has the runtime among all the three algorithms, it can only produce optimal solutions when the movement is an integer. However, in the experiments, the distance are real numbers, so ABS only produce approximation solutions. On the other hand, according to our theoretical analysis, ABSD always produces optimal solutions. As depicted in Figure 5a,b, throughout all the thousands of LBTC instances, every solution produced by ABSD has a maximum movement coincides with the optimal solution produced by ILP. In contrast, ABS produces only approximation solutions. When for the instance of small d_{max}, the solutions produced by ABS are with an average maximum movement about 1.04 times of the optimal solutions, as illustrated in Figure 5a; while d_{max} grows large, the approximation ratio becomes even better that it is 1.02, as illustrated in Figure 5b.

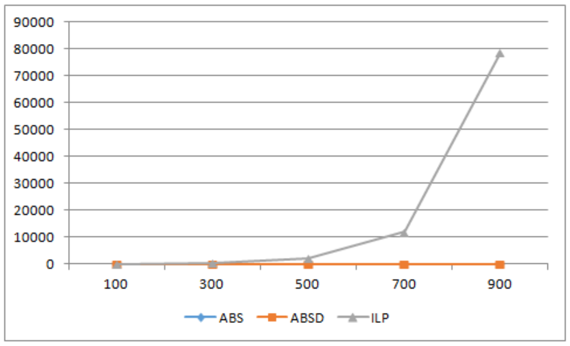


(**a**) Instances of small d_{max}.

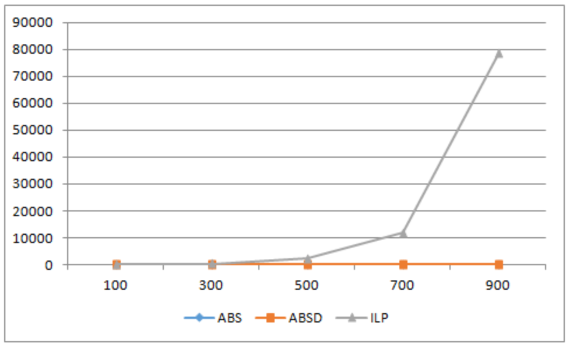


(**b**) Instances of large d_{max}.

Figure 3. Runtime comparison between ABS and ABSD.

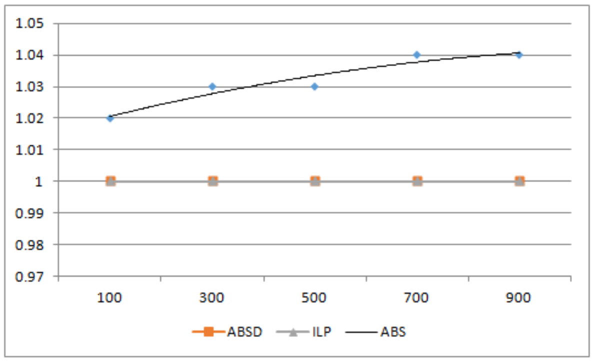


(**a**) Instances of small d_{max}.

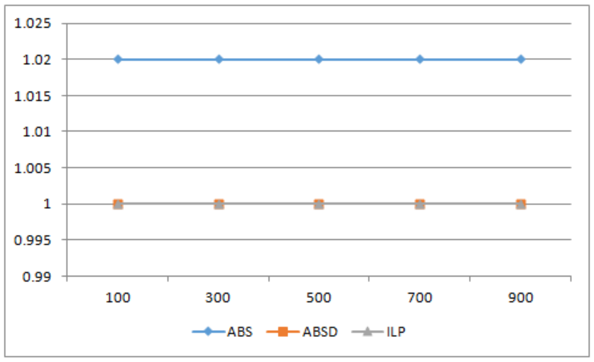


(**b**) Instances of large d_{max}.

Figure 4. Runtime comparison of ABS and ABSD against ILP, where ABS and ABSD overlap as one line because they are very close to each other comparing to ILP.



(**a**) Instances of small d_{max}.



(**b**) Instances of large d_{max}.

Figure 5. Comparison of practical solution quality of the algorithms.

6. Conclusions

In this paper, we first proved that 1D-LBTC is $\mathcal{NP}$-hard when the radius of the sensors are not identical. It is worth to note that this result is interesting because 1D-LBC problem can be efficiently solved in a polynomial time. Then, we designed an algorithm for decision LBTC with uniform radius, and consequently proposed an algorithm for really solving LBTC based on the binary search method. Moreover, we improved the binary search method to a runtime $O(n^2 \log n)$ by observing that the optimum movement bound is within the set of distances between the POIs and the sensors. We also evaluated the practical performance of our algorithms via numerical experiments. We are currently investigating how to further improve the runtime of the algorithm.

Author Contributions: Conceptualization, P.H.; methodology, P.H., L.G. and W.Z.; analysis, P.H., L.G. and W.Z.; writing—original draft preparation, P.H. and L.G.; writing—review and editing, P.H., L.G. and W.Z.; supervision, W.Z.; project administration, W.Z.; funding acquisition, L.G. and W.Z.

Funding: This research was funded by Natural Science Foundation of China under its grant numbers 61772005 and 61672005, and by Natural Science Foundation of Fujian Province under its grant number 2017J01753.

Acknowledgments: An extended abstract of part of this paper as "On the Complexity of and Algorithms for Min-Max Target Coverage on a Line Boundary: appeared in proceedings of the 15th Theory and Applications of Models of Computation.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Li, X.; Frey, H.; Santoro, N.; Stojmenovic, I. Localized sensor self-deployment with coverage guarantee. In *ACM SIGMOBILE Mobile Computing and Communications Review*; ACM: New York, NY, USA, 2008; Volume 12, pp. 50–52.
2. Kumar, S.; Lai, T.H.; Arora, A. Barrier coverage with wireless sensors. In Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, Cologne, Germany, 28 August–2 September 2005; ACM: Chicago, IL, USA; 2005; pp. 284–298.
3. Gage, D.W. *Command Control for Many-Robot Systems*; Technical Report; The Research, Development, Test & Evaluation (RDT&E) Infrastructure Division of Naval Command Control and Ocean Surveillance Center: San Diego, CA, USA, 1992.

4. Czyzowicz, J.; Kranakis, E.; Krizanc, D.; Lambadaris, I.; Narayanan, L.; Opatrny, J.; Stacho, L.; Urrutia, J.; Yazdani, M. On minimizing the maximum sensor movement for barrier coverage of a line segment. In Proceedings of the International Conference on Ad-Hoc Networks and Wireless, Murcia, Spain, 22–25 September 2009; Springer: Berlin, Germany, 2009; pp. 194–212.
5. Chen, D.Z.; Gu, Y.; Li, J.; Wang, H. Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain. *Discret. Comput. Geom.* **2013**, *50*, 374–408. [CrossRef]
6. Bhattacharya, B.; Burmester, M.; Hu, Y.; Kranakis, E.; Shi, Q.; Wiese, A. Optimal movement of mobile sensors for barrier coverage of a planar region. *Theor. Comput. Sci.* **2009**, *410*, 5515–5528. [CrossRef]
7. Tan, X.; Wu, G. New algorithms for barrier coverage with mobile sensors. In Proceedings of the International Workshop on Frontiers in Algorithmics, Wuhan, China, 11–13 August 2010; Springer: Berlin, Germany, 2010; pp. 327–338.
8. Dobrev, S.; Durocher, S.; Eftekhari, M.; Georgiou, K.; Kranakis, E.; Krizanc, D.; Narayanan, L.; Opatrny, J.; Shende, S.; Urrutia, J. Complexity of barrier coverage with relocatable sensors in the plane. *Theor. Comput. Sci.* **2015**, *579*, 64–73. [CrossRef]
9. Li, S.; Shen, H. Minimizing the maximum sensor movement for barrier coverage in the plane. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, China, 26 April–1 May 2015; IEEE: Hoes Lane Piscataway, NJ, USA, 2015; pp. 244–252.
10. 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 Proceedings of the International Conference on Ad-Hoc Networks and Wireless, Edmonton, AB, Canada, 20–22 August 2010; Springer: Berlin, Germany, 2010; pp. 29–42.
11. Mehrandish, M.; Narayanan, L.; Opatrny, J. Minimizing the number of sensors moved on line barriers. In Proceedings of the 2011 IEEE Wireless Communications and Networking Conference (WCNC), Cancun, Quintana Roo, Mexico, 28–31 March 2011; IEEE: Piscataway, NJ, USA : 2011; pp. 653–658.
12. Cherry, A.; Gudmundsson, J.; Mestre, J. Barrier Coverage with Uniform Radii in 2D. In Proceedings of the International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, Vienna, Austria, 4–8 September 2017; Springer: Berlin, Germany, 2017; pp. 57–69.
13. Liao, Z.; Wang, J.; Zhang, S.; Cao, J.; Min, G. Minimizing Movement for Target Coverage and Network Connectivity in Mobile Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 1971–1983. [CrossRef]
14. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to NP-Completeness*; W. H. Freeman and Company: San Francisco, CA, USA, 1979.
15. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2009.