New Algorithms for Barrier Coverage with Mobile Sensors

Xuehou Tan[1] and Gangshan Wu[2]

[1] Tokai University, 4-1-1 Kitakaname, Hiratsuka 259-1292, Japan
[2] State Key Lab. for Novel Software Technology, Nanjing University, China
tan@wing.ncc.u-tokai.ac.jp

Abstract. Monitoring and surveillance are important aspects in modern wireless sensor networks. In applications of wireless sensor networks, it often asks for the sensors to quickly move from the interior of a specified region to the region's perimeter, so as to form a barrier coverage of the region. The region is usually given as a simple polygon or even a circle. In comparison with the traditional concept of full area coverage, barrier coverage requires fewer sensors for detecting intruders, and can thus be considered as a good approximation of full area coverage.

In this paper, we present an $O(n^{2.5} \log n)$ time algorithm for moving n sensors to the perimeter of the given circle such that the new positions of sensors form a regular n-gon and the maximum of the distances travelled by mobile sensors is minimized. This greatly improves upon the previous time bound $O(n^{3.5} \log n)$. Also, we describe an $O(n^4)$ time algorithm for moving n sensors, whose initial positions are on the perimeter of the circle, to form a regular n-gon such that the sum of the travelled distances is minimized. This solves an open problem posed in [2]. Moreover, our algorithms are simpler and have more explicit geometric flavor.

1 Introduction

Wireless Sensor Networks (WSN) are envisioned to be developed for a wide range applications. A WSN is composed of a large number of sensor nodes, which are densely deployed either inside the phenomenon or very close to it [1]. Each sensor node is equipped with a sensing device, a low computational capacity processor, a short-range wireless transmitter-receiver and a limited battery-supplied energy. Sensor nodes monitor some surrounding environmental phenomenon, process the data obtained and forward these data towards a base station. These characteristics of a WSN require that sensor network protocols and algorithms possess self-organizing capabilities, i.e., sensors are able to cooperate in order to organize and perform networking tasks efficiently.

A typical application of WSN is to monitor a specified region either for measuring purposes or for reporting various types of activities (e.g., fire alarms, calamities, etc). Another application concerns security and safety systems, such as, detecting intruders (or movement thereof) around infrastructure facilities and regions. Particularly, it often asks to monitor an area so as to detect intruders as they penetrate the protected area or as they cross the area border.

For example, research efforts are currently under way to extend the scalability of wireless sensor networks so that they can be used to monitor international border as well [8,11].

The study of *barrier coverage* with mobile sensors was originated in [3,11], and later in [2]. Differing from the traditional concept of *full coverage*, it asks to cover the entire deployment region by guaranteeing that there is no path through this region that can be traversed undetectedly by an intruder, i.e., all crossing paths through the region are covered by sensors [2,3,11]. Since mobile sensors are allowed to move inside the deployment region, a crossing path may occur occasionally. So, an interesting problem is to reposition the sensors quickly so as to repair the existing security hole and thereby detect intruders [2]. Since barrier coverage requires fewer sensors for detecting intruders, it thus gives a good approximation of full area coverage. The planar region on which sensors move is usually represented by a simple polygon or even a circle.

In this paper, we study the problem of moving n sensors to the perimeter of a circular region to form a regualr n-gon such that either the maximum of the distances travelled by mobile sensors or the sum of the travelled distances is minimized. An efficient solution to the *min-max* or *min-sum* problem is important, as the energy required by a mobile sensor is an increasing function of the distance it travels. First, we present an $O(n^{2.5} \log n)$ time algorithm for moving n sensors to the perimeter of the given circle such that the new positions of sensors form a regular n-gon and the maximum of the distances travelled by mobile sensors is minimized. This improves upon the previous time bound $O(n^{3.5} \log n)$ [2]. Also, we describe an $O(n^4)$ time algorithm for moving n sensors, whose initial positions are on the perimeter of the given circle, to form a regular n-gon such that the sum of the travelled distances is minimized. This solves an open problem posed in Section 5.5 of [2]. Moreover, since our algorithms are based on some properties from elementary geometry, they are simple and easy to implement.

2 Problem Definition and Previous Work

Suppose that mobile sensors are working inside a planar region. As discussed in [2,3], individual sensors are able to locally determine the existence of barrier coverage, even when the region is arbitrarily shaped. For simplicity, the region is usually delimited by a simple polygon or even by a circle. In this paper, we mainly focus on the problem of moving n sensors to the perimeter of a unit-radius circular region such that the new positions of sensors form a regular n-gon and thereby give barrier coverage, assuming that the mobile sensors have detected the existence of a crossing path. We assume that the sensors are location aware (i.e., they know their geometric coordinates) and know the center of the given circle. Assume also that the range of the sensor's transmitter-receiver is always longer than an edge of the regular n-gon; otherwise, barrier coverage is impossible.

Denote by C the given circular region. For n sensors, denote by A_1, A_2, …, A_n their initial positions in the interior or on the perimeter of C, and A'_1, A'_2, …, A'_n their goal positions on the perimeter of C, which form a regular n-gon.

Moreover, denote by AA' the line segment with two endpoints A and A', and $|AA'|$ the length of AA' (i.e., the Euclidean distance between A and A'). Then, one can easily define the following two problems:

1. **The min-max problem**: Minimizing the maximum of the distances travelled by the sensors, i.e., $min\ \{max_{i=1}^{n}|A_iA'_i|\}$.
2. **The min-sum problem**: Minimizing the sum of the distances travelled by the sensors, i.e., $min\ \sum_{i=1}^{n}|A_iA'_i|$.

An $O(n^{3.5}\log n)$ time algorithm has been proposed by Bhattacharya et al. to solve the min-max problem [2]. For a simple polygon with m vertices, an $O(mn^{3.5}\log n)$ time solution is also given. Their algorithms are based on the parametric searching technique, which requires many sophisticated procedures (e.g., a sorting algorithm with an unknown optimal value [4], and a parallel sorting network [12]). As many other geometric applications of parametric searching, the necessary condition for giving an optimal solution is not explictly described [2]. These drawbacks of parametric searching have previously been pointed out in the literature (see Section 1 of [9]).

Two approximation algorithms for the min-sum problem for circular regions are further provided in [2]. A simple $O(n^2)$ time solution to a special version of the min-sum problem, in which both the initial positions and the movements of all sensors are limited on the perimeter of the given circle, is also given. Again, the similar result was extended to polygonal regions [2]. Whether a polynomial-time solution to the min-sum problem exists is left open, even when the initial positions of all sensors are on the perimeter of the given circle (see Section 5.5 of [2]).

3 Min-max Problem

Suppose that C is a unit-radius circle, and o is the center of C. Denote by ∂C the perimeter of the circle C. Moreover, denote by λ_C the optimal solution to the min-max problem for the circle C, i.e., $\lambda_C = min\ \{max_{i=1}^{n}|A_iA'_i|\}$. Clearly, $\lambda_C \leq 2$.

In the following, we study the geometric properties of the points on ∂C that may contribute to the optimum λ_C, and then present our new algorithm for computing λ_C.

3.1 Geometric Properties of the Boundary Points Related to $\boldsymbol{\lambda_C}$

Denote by X_i the point of ∂C, which is closest to A_i. Clearly, X_i is an intersection point of ∂C with the line passing through A_i and o. Denote by Y_i the other intersection point, which is the point of ∂C furthest from A_i. See Figure 1(a).

The following properties are important to our algorithm for the min-max probem.

Lemma 1. *Suppose that an optimal solution to the min-max problem is obtained with $\lambda_C = |A_iA'_i|$, for some i ($1 \leq i \leq n$). Then, either A'_i is the point X_i, or there exists another sensor A_j ($j \neq i$) such that $\lambda_C = |A_jA'_j|$ also holds.*

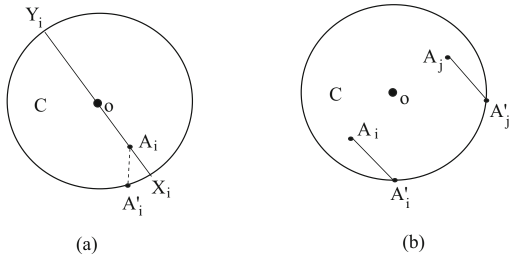

Fig. 1. (a) The points X_i and Y_i on ∂C; (b) $|A_iA'_i| = |A_jA'_j|$

In the latter case, a slight rotation of the regular n-gon giving λ_C in either direction monotonically increases one of the two distances $|A_iA'_i|$ and $|A_jA'_j|$, and decreases the other.

Proof. Assume first that in the obtained optimal solution, the sensor A_i is the only one satisfying $\lambda_C = |A_iA'_i|$, but A'_i is not the point X_i. So $|A_iX_i| < |A_iA'_i|$ holds (Figure 1(a)). Let us rotate the regular n-gon giving λ_C by moving the vertex A'_i towards X_i, with a very small distance ϵ. Clearly, the distance function between A_i and A'_i decreases monotonically during the rotation of the n-gon. Denote by A''_1, A''_2, ..., A''_n the new positions of the sensors after the rotation stops. Since ϵ is arbitrarily small and A_i is the only one satisfying $\lambda_C = |A_iA'_i|$, we have $|A_iA''_i| \geq |A_kA''_k|$ for all $k \neq i$, and moreover, $|A_iA''_i| < |A_iA'_i|$ holds; it contradicts that λ_C $(= |A_iA'_i|)$ gives the optimal solution to the min-max problem.

Suppose now that there exists another sensor A_j such that $\lambda_C = |A_jA'_j|$ $(j \neq i)$ also holds (Figure 1(b)). A slight rotation of the regular n-gon giving λ_C in either direction cannot make both $|A_iA''_i| < |A_iA'_i|$ and $|A_jA''_j| < |A_jA'_j|$ hold, where A''_i and A''_j denote the new positions of A'_i and A'_j after the rotation stops; otherwise, it contradicts the equations $\lambda_C = |A_iA'_i|$ and $\lambda_C = |A_jA'_j|$. This implies that the rotation of the regular n-gon giving λ_C increases one of the two distances $|A_iA'_i|$ and $|A_jA'_j|$, but decreases the other. The proof is complete. □

The points of ∂C satisfying the conditions described in Lemma 1 clearly contribute to the candidate values for λ_C. The points X_h of all sensors A_h $(1 \leq h \leq n)$ can simply be found. So an important task is to find all the pairs (A_i, A_j) $(i \neq j)$ such that the distance from A_i to a vertex of a regular n-gon equals to the distance from A_j to another vertex of the n-gon, and a slight rotation of the n-gon in either direction monotonically increases one of the two distances but decreases the other. For simplicity, we call such distances the *equal distances.*

Let P be an arbitrary regular n-gon with the vertices P_1, P_2, ..., P_n on ∂C. In order to find all equal distances, we simulate below a clockwise rotation of the polygon P on ∂C with an arc distance $2\pi/n$. First, compute the distances between all pairs of a sensor and a vertex of P, and sort these n^2 distances into the sequence, say, $d_1 < d_2 < \ldots < d_{n^2}$. Denote by D the resulting sequence of these distances. For any two adjacent distances in D, we then check whether they can produce an equal distance during the procedure of rotating P. Note that the two sensors related to that equal distance as well as the two intervals of ∂C (of arc length $2\pi/n$) on which the two vertices of P move are known. Moreover, since one of the two distances increases but the other decreases to the equal distance, the exact distance between the goal positions of two sensors is also known. (The distance between two goal positions can be represented as $2sin(w\pi/n)$, where $w(\leq n)$ is a known, positive integer.) Thus, we can determine in constant time whether the equal distance for a pair of adjacent distances in D can be attained. If *yes*, report the found equal distance. From the initial sequence $d_1, d_2, \ldots, d_{n^2}$, we can thus obtain some equal distances. Denote by E the set of the found equal distances.

To achieve an equal distance e, the polygon P is rotated on ∂C with an arc distance from its *initial* position. Denote by $r(e)$ $(\leq 2\pi/n)$ this arc distance (from the initial position of P) required to obtain the equal distance e. Let R be the set of these arc distances $r(e)$, and r_{min} the smallest arc distance in R.

The clockwise rotation of P with the arc distance $2\pi/n$ can then be performed as follows. First, take out (i.e., delete) r_{min} from the set R. Since an equal distance happens only between two adjacent distances in D, the first equal distance occurs when the polygon P is rotated on ∂C with the arc distance r_{min}. Afterwards, the two adjacent distances, which contributed to r_{min}, have to be exchanged in the sequence D. The change of these two distances in D may introduce at most two new equal distances, and thus, we further check whether these two equal distances can be attained. If a new equal distance is ever found, we compute the corresponding arc distance from the initial position of P, and then insert it into R. Next, take out the current arc distance r_{min} from R and continue to rotate P on ∂C according to (the remaining value of) r_{min}. In this way, the polygon P is gradually rotated, and whenever the arc distance represented by r_{min} is reached, we maintain the sequence D of n^2 distances between sensors and the current vertices of P, and the set R of the arc distances, which are computed from the found equal distances. All equal distances are clearly reported after the rotation of P is complete.

Lemma 2. *Let m be the number of the equal distances. Then, $m = O(n^3)$.*

Proof. Let $A_i(P_x)$ denote the distance function from a sensor A_i to a vertex P_x of the n-gon P. Clearly, the function $A_i(P_x)$ increases or decrease monotonically in the procedure of rotating P, except that the interval of ∂C on which P_x moves contains the point X_i or Y_i; in this case, we divide that interval of ∂C into two sub-intervals such that $A_i(P_x)$ is monotone in either sub-interval. All functions $A_i(P_x)$, $P_x \in P$, can thus be grouped into two sets S_{i1} and S_{i2} such that the functions in the set S_{i1} monotonically increase and the functions in S_{i2} monotonically decrease. The union of the intervals of ∂C, on which the vertices

P_x of all functions of S_{i1} (resp. S_{i2}) move, is the clockwise chain of ∂C from X_i to Y_i, or from Y_i to X_i. Hence, all functions in S_{i1}, or in S_{i2} can be considered as a single distance function from the sensor A_i, which is also monotone. Similarly, we can define the monotone functions $A_j(P_y)$, for all other sensors A_j and all polygon vertices P_y.

Let us now give an upper bound on the number of the equal distances. In the procedure of rotating P with the arc distance $2\pi/n$, any decreasing (resp. increasing) function $A_j(P_y)$ can produce at most one equal distance with a function of S_{i1} (resp. S_{i2}), $i \neq j$. This is because the function $A_j(P_y)$ is monotone, and all functions in S_{i1} (resp. S_{i2}) can be considered as a monotone distance function from A_i. Since both the number of the sensors A_j ($j \neq i$) and the number of the vertices P_y are no more than n, the sensor A_i can contribute to $O(n^2)$ equal distances. Therefore, we have $m = O(n^3)$. □

3.2 Algorithm

Let p be a point on ∂C, and d the distance between p and an arbitrary sensor. We can then give an algorithm for determining whether $\lambda_C \leq d$ [2].

Algorithm Distance-Test

1. Compute the regular n-gon by fixing one of its vertices at p, and denote by B_1, B_2, ..., B_n the vertices of the obtained n-gon.
2. Construct a bipartite graph H_d between the sensors A_1, A_2, ..., A_n and the vertices B_1, B_2, ..., B_n; sensor A_i is linked to vertex B_j ($1 \leq i, j \leq n$) if and only if $|A_i B_j| \leq d$.
3. Check whether there exists a perfect matching in H_d. If *yes*, report $\lambda_C \leq d$; otherwise, report $\lambda_C > d$.

The time complexity of the algorithm **Distance-Test** is $O(n^{2.5})$. This is because the first two steps of **Distance-Test** clearly take $O(n^2)$ time, and the last step requires $O(n^{2.5})$ time to check whether there exists a perfect matching in the bipartite graph H_d [6].

To give a simple solution to the min-max problem, one can run a binary search over all the distances $|A_i X_i|$ ($1 \leq i \leq n$) and the equal distances e_j ($1 \leq j \leq m$) using the fixed-size decision algorithm **Distance-Test** to determine whether the optimum λ_C is larger than, smaller than or equal to the selected distance d. Clearly, the smallest of the values d satisfying $\lambda_C \leq d$ gives the answer to the min-max problem.

Let us now describe a more efficient algorithm for the min-max problem. Again, denote by P a regular n-gon with the vertices P_1, P_2, ..., P_n on ∂C. First, we perform a binary search over all the distances between sensors and the initial positions of vertices of P, i.e., d_1, d_2, ..., d_{n^2}, to find two distances d_k, d_{k+1} such that $d_k < \lambda_C \leq d_{k+1}$. Without loss of generality, we assume that

$d_0 = 0$, $d_{n^2+1} = 2$, and $0 \leq k \leq n^2$. In the following, we show that there are at most $O(n^2)$ equal distances e, with $d_k < e \leq d_{k+1}$.

Let P^i $(1 \leq i \leq n)$ denote the regular n-gon obtained by fixing one of its vertices at the point X_i. Suppose that the vertices of P^i are indexed clockwise, starting from the vertex $P_1^i = X_i$. Clearly, there are $\lceil (n-2)/2 \rceil$ pairs of the vertices in P^i, whose distances to the sensor A_i are the same. Let $d_1^i < d_2^i < \ldots < d_{\lceil n/2 \rceil +1}^i$ denote the sequence of the distances from A_i to all vertices of P^i (including P_1^i) and the point Y_i as well when n is odd. When n is odd, the point Y_i is not the vertex of P^i. But, as discussed above, some distance function needs to be divided into two monotone sub-functions using the point Y_i.

Lemma 3. *Let P be an arbitrary regular n-gon on ∂C, and d_1, d_2, ..., d_{n^2} the increasing order of the distances between all sensors and the vertices of P. Assume that $d_0 = 0$ and $d_{n^2+1} = 2$, and that we know $d_k < \lambda_C \leq d_{k+1}$ for some k, $0 \leq k \leq n^2$. Let $m(k)$ be the number of the equal distances e, with $d_k < e \leq d_{k+1}$. Then, $m(k) = O(n^2)$, and all of these equal distances can be computed in $O(n^2 \log n)$ time.*

Proof. Suppose that we have known $d_k < \lambda_C \leq d_{k+1}$. For any polygon P^i $(1 \leq i \leq n)$, the range $(d_k, d_{k+1}]$ is clearly contained in $[d_j^i, d_{j+2}^i]$ for some $j < \lceil n/2 \rceil$, or in $[d_j^i, d_{j+1}^i]$ when $j = \lceil n/2 \rceil$. Also, denote by $A_i(P_x^i)$ the distance function from the sensor A_i to a vertex P_x^i of the n-gon P^i. Since the distance functions from A_i to all vertices of P^i can be grouped into two monotone functions in the rotation of P^i with the arc distance $2\pi/n$, at most four distance functions $A_i(P_x^i)$ may vary in the range $(d_k, d_{k+1}]$. These distance functions from A_i can be found in $O(\log n)$ time, provided that the increasing order of the distances d_1^i, d_2^i, ..., $d_{\lceil n/2 \rceil+1}^i$ is known.

Consider now the procedure of rotating the polygon P on ∂C clockwise with the arc distance $2\pi/n$. Suppose that all the polygons P^1, P^2, ..., P^n are also rotated with the arc distance $2\pi/n$ simultaneously. Assume that all the $O(n)$ distance functions, which vary in the range $(d_k, d_{k+1}]$, have been sorted according to their initial values (i.e., using the corresponding distances d_j^i, $1 \leq i, j \leq n$). Denote by $D(k)$ the increasing order of these distance functions. For any two adjacent distances in $D(k)$, we then check whether they can be equal in the procedure of rotating the polygon P. If *yes*, report that equal distance. Also, we can simply check whether the found equal distance is between d_k and d_{k+1}. From the initial sequence $D(k)$, we can compute a set of equal distances. Denote by $R(k)$ the set of the arc distances, which are required to attain the found equal distances. As in the proof of Lemma 2, the polygon P can then be rotated according to the arc set $R(k)$. When the set $R(k)$ becomes empty in the procedure of rotating P, we obtain all the equal distances e, with $d_k < e \leq d_{k+1}$. The number $m(k)$ of the equal distances e, with $d_k < e \leq d_{k+1}$, is $O(n^2)$. This is because $D(k)$ contains $O(n)$ monotone functions in the whole procedure of rotating P and a pair of adjacent distances in $D(k)$ at any instant time of the rotation can contribute to at most one equal distance.

Finally, consider the time required to compute these equal distances. First, the sequences of the distances d_1^i, d_2^i, …, $d_{\lceil n/2\rceil+1}^i$, for all i ($1 \le i \le n$), can totally be computed in time $O(n^2 \log n)$. The initial sequence $D(k)$ of the distance functions, which vary in the range $(d_k, d_{k+1}]$, can then be found in $O(n \log n)$ time. When an equal distance occurs between two adjacent distances of $D(k)$ in the procedure of rotating P, we insert its corresponding arc distance into the set $R(k)$. For efficiency, $R(k)$ is maintained in a priority queue. Hence, an insertion of an arc distance to $R(k)$ as well as a deletion of the smallest arc distance from $R(k)$ takes $O(\log n)$ time. So the total time required is $O(n^2 \log n)$. □

By now, we can give our algorithm for the min-max problem. First, perform a binary search over the distances d_0, d_1, …, d_{n^2+1} using the fixed-size decision algorithm **Distance-Test** to determine whether the optimum λ_C is larger than, smaller than or equal to the selected distance. After this binary search is done, we can assume that $d_k < \lambda_C \le d_{k+1}$ for some k, $0 \le k \le n^2$. Next, rotate the polygon P with the arc length $2\pi/n$ to find all the equal distances e, with $d_k < e \le d_{k+1}$. Furthermore, we sort these $m(k)$ equal distances and the other n distances $|A_jX_j|$ ($1 \le j \le n$). Denote the resulting sequence by $ed_1 < ed_2 < \ldots < ed_{n+m(k)}$. Since the optimum λ_C is one of these $n + m(k)$ values (Lemmas 1 and 3), it suffices to run another binary search using the algorithm **Distance-Test**. We conclude the algorithm in the following:

Algorithm Min-Max

1. Let P be an arbitrary regular n-gon on ∂C, and $d_1, d_2, \ldots, d_{n^2}$ the increasing order of the distances between all sensors and the vertices of P. Assume also that $d_0 = 0$ and $d_{n^2+1} = 2$.
2. Run a binary search over the distances d_0, d_1, …, d_{n^2+1} using the fixed-size decision algorithm **Distance-Test** to find two values d_k, d_{k+1} ($0 \le k \le n^2$) such that $d_k < \lambda_C \le d_{k+1}$. (The regular n-gon used in **Distance-Test** is constructed according to the selected distance.)
3. For every i ($1 \le i \le n$), place the regular n-gon P^i on ∂C by fixing one of its vertices at the point X_i, and then sort the distances from A_i to all vertices of P^i and the point Y_i as well when n is odd, into the sequence $d_1^i < d_2^i < \ldots < d_{\lceil n/2\rceil+1}^i$. Next, find at most $4n$ vertices of P^i such that their distance functions (from A_i) vary in the range $(d_k, d_{k+1}]$ when P^i is rotated with the arc distance $2\pi/n$.
4. Rotate the polygon P on ∂C with the arc distance $2\pi/n$ to compute all the equal distances e, with $d_k < e \le d_{k+1}$. Then, sort these $m(k)$ equal distances and the other n distances $|A_jX_j|$ ($1 \le j \le n$) into the sequence , say, $ed_1 < ed_2 < \ldots < ed_{n+m(k)}$.
5. Run a binary search over the distances ed_1, ed_2, …, $ed_{n+m(k)}$ using the fixed-size decision algorithm **Distance-Test** to determine whether λ_C is larger than, smaller than or equal to the selected distance.
6. Report the smallest of the values ed_j satisfying $\lambda_C \le ed_j$.

It follows from the discussion made above that Steps 2 and 5 of **Min-Max** take $O(n^{2.5} \log n)$ time, which clearly dominates the time complexity of the algorithm **Min-Max**. Hence, we obtain the main result of this paper:

Theorem 1. *The min-max problem for a given circle can be solved in* $O(n^{2.5} \log n)$ *time.*

4 Min-sum Problem: When Sensors Are Initially on the Perimeter of the Circle

This section presents an $O(n^4)$ time algorithm for a special version of the min-sum problem, in which all sensors are initially on the perimeter of the unit-radius circle C. Our result solves an open problem posed in [2].

Let A_1, A_2, …, A_n denote the initial positions of n sensors on ∂C, and A'_1, A'_2, …, A'_n their goal positions on ∂C. Denote by Δ_C the optimal solution to the min-sum problem for C, i.e., $\Delta_C = min \sum_{i=1}^{n} |A_i A'_i|$. The following property is important to the solution of our min-sum problem.

Lemma 4. *Suppose that all sensors are initially on the perimeter of the unit-radius circle C. In any assignment between the initial and goal positions of n sensors, which gives Δ_C, there exists some sensor A_x ($1 \leq x \leq n$) such that $A_x = A'_x$.*

Proof. The proof is by contradiction. Assume that in any assignment between the initial and goal positions of n sensors, which gives Δ_C, all sensors move, i.e., $A_i \neq A'_i$ for all i, $1 \leq i \leq n$. For our purpose, we construct below a line segment L such that the length L is equal to Δ_C. To avoid confusion, we use the small letters 'a_k' ('a'_k') to represent the points A_k (A'_k) on L. See Figure 2(b). First, fix a segment $A_s A'_s$ for an aribitrary index s, in an assignment giving Δ_C. Then, draw all other segments contributed to Δ_C, in an arbitrary order, on the extension of the segment $A_s A'_s$ by connecting the point A'_y of the segment $A_y A'_y$ to the previously existed point A_x. Assume that a_t is the last endpoint obtained in constructing the segment L. Since a'_s is the other endpoint of L, we have $|a_t a'_s| = \Delta_C$. See Figure 2(b) for an example, where $a'_s = a'_3$ and $a_t = a_1$.

Let us now move all points A'_j on ∂C clockwise, by a very small arc distance α, say, $n\alpha < \pi/4$. Assume that no point A_j is passed over during this clockwise rotation of the n-gon (it is always possible since α can be arbitrarily small). Denote by B_j the new position of A'_j ($1 \leq j \leq n$), and Δ_1 the solution of the min-sum problem in which every sensor A_j moves to B_j. See Figure 2(a). Since three points A_j, A'_j and B_j ($1 \leq j \leq n$) are on ∂C, the angle $\angle A'_j A_j B_j$ is α. Next, we translate the segment $A_t B_t$ such that the translated segment, denoted by $a_t b_t$, and the segment $a_t a'_t$ (on L) form the angle α at the point a_t. See Figure 2(b). Draw all other segments in the assigment giving Δ_1 on the extension of the segment $a_t b_t$, in the reversed order of the segments added to construct L. Denote by L_1 the resulting segment. See Figure 2(b). Clearly, b_s is the other endpoint of L_1. Also, we have $|a_t b_s| = \Delta_1$.

Analogously, we move all points A'_j on ∂C counterclockwise, by the same arc distance α. Assume also that no point A_j is passed over during this counterclockwise rotation of the n-gon. Denote by E_j the new position of A'_j $(1 \leq j \leq n)$, and Δ_2 the solution of the min-sum problem in which every sensor A_j moves to E_j. Similarly, we can obtain a line segment L_2, with two endpoints a_t and e_s, of length Δ_2. See Figure 2(b). Without loss of generality, assume that L_1 and L_2 are to the different sides of L. Thus, L_1 and L_2 form the angle 2α at the point a_t.

In the above construction of L and L_1 (resp. L_2), only the translation and the rotation of segments are empolyed. Hence, two points a'_s and b_s (resp. a'_s and e_s) can be connected by n arcs of length α, which are scanned by all sensors in transforming the assignment for Δ_C into that for Δ_1 (resp. Δ_2). These n arcs can thus be replaced by a long arc of length $n\alpha$, i.e., a'_s and b_s (resp. a'_s and e_s) are connected by an arc of length $n\alpha$ $(< \pi/4)$. It immediately implies that three points a'_s, b_s and e_s are passed by another unit-radius circle, say, C'. See Figure 2(b). Since the angle formed by L_1 and L_2 (i.e., two segments $a_t b_s$ and $a_t e_s$) at a_t is 2α, the point a_t cannot be contained in the interior of the circle C' (otherwise, since the arc distance between b_s and e_s is $2n\alpha$, we would have $2\alpha > 2n\alpha$ $(n \geq 2)$, a contradiction). The segment L then has two common points with $\partial C'$. If the center of C' happens to be on L, we have $\Delta_1 < \Delta_C$ and $\Delta_2 < \Delta_C$, contradicting that Δ_C gives an optimal solution to the min-sum problem. In the case that the center of C' is not on L, either $\Delta_1 < \Delta_C$ or $\Delta_2 < \Delta_C$ holds, a contradiction again. This completes the proof. □

Based on Lemma 4, the min-sum problem for the circle C can be solved by fixing a vertex of the regular n-gon at the initial position of every sensor once, then solving the weighted matching problem (n times) in a complete bipartite graph such that one subset of its nodes represents the initial positions of all sensors and the other represents the goal positions of sensors, and finally reporting the smallest of the obtained solutions to all weighted matching instances.

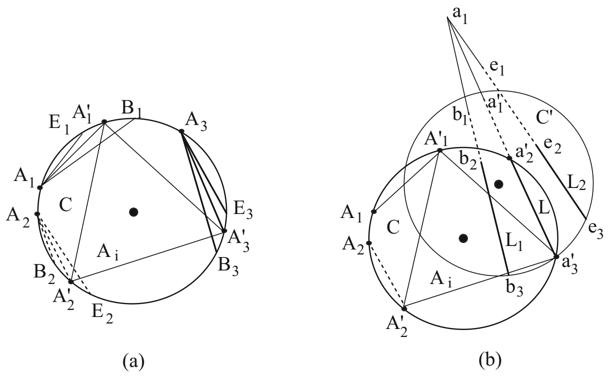

Fig. 2. Illustration of the proof of Lemma 4

Algorithm Min-Sum

1. For each $i = 1, 2, \ldots, n$, do the following:
   (a) Place a regular n-gon P on ∂C by fixing one of its vertices at the sensor A_i. Then, construct a complete bipartite graph H_i between the sensors $A_1, A_2, \ldots, A_n$ and the vertices $P_1, P_2, \ldots, P_n$ of the n-gon (i.e., any of n sensors is linked to all polygon vertices). Moreover, define the Euclidean distance between a sensor and a vertex of P as the weight of the edge of H_i connecting the corresponding nodes.
   (b) Compute an optimal solution $\varDelta(H_i)$ to the weighted bipartite matching problem for the graph H_i (i.e., finding the optimal assignment of the n sensors to the vertices of P in the weighted bipartite graph H_i).
2. Report the smallest of all found solutions $\varDelta(H_i)$.

Since the number of nodes of the graph H_i is $2n$, the Hungarian method can solve in $O(n^3)$ time the weighted matching problem in the complete bipartite graph H_i [10]. Hence, we have the following result.

Theorem 2. *Suppose that all sensors are initially located on the perimeter of the given circle. Then, the min-sum problem can be solved in $O(n^4)$ time.*

5 Concluding Remarks

In this paper, we have presented an $O(n^{2.5} \log n)$ time algorithm for moving n sensors to the perimeter of the given circle such that the new positions of sensors form a regular n-gon and the maximum of the distances travelled by mobile sensors is minimized. This greatly improves upon the previous time bound $O(n^{3.5} \log n)$. Also, we have described an $O(n^4)$ time algorithm for moving n sensors, given on the perimeter of the circle, to form a regular n-gon such that the sum of the travelled distances is minimized. This solves an open problem posed in [2]. Moreover, our algorithms are simple and easy to implement.

There are several open questions which are interesting for further research. First, whether the min-sum problem is NP-hard is not known. The answer might be negative, since it seems quite difficult to specify a finite number of candidate points on the perimeter of the given circle such that an optimal solution can be computed from these candidate points. Also, it is interesting to extend our methods to polygonal or convex regions. For a polygonal or convex polygon, the sensors are required to move to the perimeter of the polygon such that the polygonal distance along the perimeter between any two consecutive sensors are the same [2]. Finally, although our algorithms as well as the previous algorithms given in [2] were developed for a wireless sensor network, we have assumed a centralized control server, where nodes are connected using a gateway. The distributed self-deployment algorithms for full/barrier coverage, which consider various designing strategies, such as oblivious robots, uniformity and system lifetime, have been studied in [5,7]. Whether the distributed version of our algorithms can be developed is an interesting question for further work.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: A Survey. Computer Networks (Elsevier) Journal, 393–422 (March 2002)
2. Bhattacharya, B., Burmester, M., Hu, Y., Kranakis, E., Shi, Q., Wiese, A.: Optimal movement of mobile sensors for barrier coverage of a planar region. Theoretical Computer Science 410, 5515–5528 (2009)
3. Chen, A., Kumar, S., Lai, T.H.: Designing localized algorithms for barrier coverage. In: Proc. 13th ACM Int. Conf. on Mobile Computing and Networking, pp. 63–73 (2007)
4. Cole, R.: Slowing down sorting networks to obtain faster algorithms. Journal of the ACM 34(1), 200–208 (1987)
5. Défago, X., Souissi, S.: Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theoretic Computer Science 396, 97–112 (2008)
6. Hopcroft, J.E., Karp, R.M.: An $n^{2.5}$ algorithm for maximum matching in bipartite graphs. SIAM J. Comput. 2(4), 225–231 (1973)
7. Heo, N., Varshney, P.K.: Energy-efficient deployment of intelligent mobile sensor networks. IEEE Trans. Systems, Man and Cybernetics, Part A 35(1), 78–92 (2005)
8. Hu, S.S.: 'Virtual Fence' along border to be delayed, Washington Post (February 28, 2008)
9. Katz, M.J., Sharir, M.: An expander-based approach to geometric optimization. In: Proc. 10th ACM Symp. on Computational Geometry, pp. 198–207 (1993)
10. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97 (1955)
11. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. Wireless Networks 13(6), 817–834 (2007)
12. Megiddo, N.: Applying parallel computation algorithms in design of serial algorithms. Journal of the ACM 30(4), 852–865 (1983)