Egyptian Informatics Journal (2011) **12**, 83–93

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

An Adaptive Relocation Strategy for heterogeneous sensor networks

Salah Abdel-Mageid *, Mohamed Zaki

Systems and Computers Dept., Faculty of Engineering, Al-Azhar University, Egypt

Received 17 December 2010; revised 7 April 2011; accepted 10 April 2011
Available online 17 May 2011

KEYWORDS

Heterogeneous sensor networks;
Adaptive Relocation;
Sweep radius;
Triangulation

Abstract Heterogeneous sensor networks (HSNs) have grown to be familiar in recent years due to their capabilities to increase network lifetime and reliability without a significant increase in the cost. Deploying sensor nodes in large-scale applications (i.e., battlefields and environmental monitoring) requires decentralized solutions. In this paper, we propose a novel decentralized approach enabling us to consider the heterogeneous characteristics of sensor nodes. In the Adaptive Relocation Strategy, new geometric approaches are designed to perfectly deal with the most heterogeneous sensor characteristics. The simulation results are presented to show that the proposed solution achieves the high coverage performance in few rounds with minimum energy consumption and minimum computations. The performance comparison is also introduced to study how the designed parameters affect the network performance in terms of the network cost, the coverage enhancement, and the total energy consumption measured by the computational complexity and the average moving distance.

1. Introduction

Wireless Sensor Networks (WSNs) consist of measuring, processing, and communication models, which enable us to mainly observe and react with events and phenomena in small-scale applications [1–3]. However, mobile sensor nodes (e.g., expensive nodes contain mobility and location finding parts) are required to facilitate working with large-scale applications such as environmental monitoring and battlefields [4,5]. Recently, *heterogeneous sensor nodes* have suggested increasing the network lifetime and reliability. In addition, the network cost of heterogeneous nodes is less expensive than the networks that are forced to choose each node mobile, especially for large-scale applications. However, the optimal deployment of the heterogeneous nodes is the objective of our work.

Many solutions have been proposed for mobile sensor networks. For example, the *potential fields*' concept has been suggested to deploy mobile sensor nodes such as [6–8]. The concept of *potential fields* cannot perfectly work with heterogeneous nodes. Most deployed nodes are mobile and they have the same sensing and communication ranges. In fact, there is

* Corresponding author.
E-mail address: smageid@azhar.edu.eg (S. Abdel-Mageid).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2011.04.002

Production and hosting by Elsevier

no arbitrary relation between the sensing and communication ranges in which the communication range is usually chosen to be the double of sensing range. The authors, in [9] and [10], exploited well-known geometrical solution, *Voronoi diagrams*, and introduced two solutions exploiting the heterogeneity in mobile support which the existence of mobile and static sensors is considered to save the network cost. However, that solution was restricted with *Voronoi diagrams*, which work with the same sensing and communication ranges. Accordingly, that solution is constraint to deploy heterogeneous nodes.

This paper introduces efficient strategy to deploy *heterogeneous sensor nodes* for large-scale applications. Appropriate geometrical solutions are designed to deal with an arbitrary relation between sensing and communication ranges to discover the coverage gaps and approximately draw their borders. Some static nodes are dynamically chosen to supervise those polygons representing the gaps' borders; they are so called "*Leaders*". Various announcements are used to connect among nodes. For example, *leader*/*leader announcement* is used among leaders to exchange the edges of incomplete polygons to construct complete polygons. In addition, the proposed solution enables the mobile sensors to receive from leaders and choose the best location to move which achieves the network requirements. In this work, a simulation tool is proposed to measure the effectiveness of the proposed strategy.

This paper is organized as follows: Section 2 briefly introduces the related works. Section 3 presents preliminaries for our work includes the assumptions and the proposed geometrical foundation. The proposed relocation strategy is discussed in Section 4. The basic operations of the proposed strategy are described in details in Section 5. The performance comparisons and simulation results are introduced in Section 6 to evaluate the designed parameters of the proposed strategy. Section 7 concludes this work and emphasizes the future work.

2. Related works

The last two decades have observed considerable research work in the area of deployment and configuration of mobile WSNs. Some solutions have been carried out based on different optimization techniques such as *Circle Packing*, *Genetic Algorithms*, and *Swarm Algorithms* for small-scale sensor networks [11–14]. Such solutions have polynomial complexity (i.e., quadratic or more); therefore, a centralized powerful node is required. Such solutions are appropriate for small-scale applications because their running time is clearly affected an increase of the number of deployed sensor nodes.

On the other hand, some solutions have been proposed for large-scale applications. For example, Howard et al. [6] developed a deployment framework in which sensor locations are determined based on the *potential fields* that are assumed to exist among sensors and between those sensors and obstacles in the field. Such framework improved the coverage assuming all sensors are mobile and have the same characteristics; however, it cannot perfectly work for different sensing and communication ranges. Furthermore, the network connectivity is unguaranteed all the time, and the possible movement oscillations loss more energy before each sensor reaches the static equilibrium state.

The authors in [7] and [8] suggested a clustering technique, for mobile sensors with the same characteristics, in which cluster heads gather information on the location of all devices in their clusters. A cluster head determines the new location for each sensor in its cluster through applying the potential field approach, i.e., virtual force, described in Howard et al. [6]. Such solutions avoided the oscillated movements, which computes the sensors' final destinations. All computations are assumed to be conducted by the cluster heads; which requires powerful sensors with additional computation capacity. In addition, such work has the same restrictions of sensor characteristics.

Wang et al. [15] introduced other approach to avoid the oscillated movements. The *VEC* algorithm was introduced that uses the virtual force concept and eliminates the oscillated movements using *Voronoi diagram*. The *Voronoi* cells' computations are able to prevent a sensor to move when no coverage improvement is discovered; however, the coverage performance is limited. Accordingly, the authors improve the coverage by suggesting two different ideas in the same work, *VOR* and *Minimax*, based on *Voronoi* layout. Such work assumed all sensors are mobile with the same characteristics.

In fact, the network cost increases when the mobile/static sensor cost ratio increases too. Accordingly, the authors in [9] suggested deploying a combination of static and mobile sensors to save the network cost. The mobile sensors move to hail the coverage gaps based on *Voronoi diagrams*. Such work may consume further energy, which a sensor moves to several positions until reach the final position. Therefore, such work was enhanced in [10], which a type of dynamic clustering is used to reduce the number of movement rounds and directly move sensors to the final destinations. However, such solution cannot be applied to *heterogeneous sensors*, especially for different sensing ranges because *Voronoi diagrams* as a geometric algorithm is designed to identical sites.

This paper introduces a new geometric framework for low power heterogeneous sensors to improve the coverage performance with minimum energy loss. The proposed framework designs lightweight geometric algorithms to be appropriate to work with low power nodes with limited resources. Furthermore, the communication cost is clearly acceptable in our strategy because the number of messages among sensor nodes is reduced. In addition, the average moving distance is eliminated because appropriate mobile sensors are chosen to heal the coverage gaps. The only work, to our knowledge, that addressed lightweight geometric algorithms for deploying heterogeneous sensors is our results in this paper.

3. Preliminaries

3.1. Assumptions

Given a target field in which thousands of *heterogeneous sensors* are randomly spread in a large-scale area. The sensing area of a sensor i is approximated by a circle with radius r_i indicating its sensing range. Each sensor can determine its current position by *GPS* services or determine it based on a localization algorithm, i.e., [16,17]. This information can be shared with all other sensors within the sensor's communication range. In addition, all other sensor characteristics can be shared when necessary.

3.2. Technical preliminary: circle's exposed arcs

In this work, the center of a circle is a sensor node in which its radius representing its sensing range. When some sensor nodes close to each other, their sensing areas may overlap. Since the sensing areas are represented by circles, those circles can be drawn overlapped. A circle may be overlapped by one or more circles resulting in intersection arcs. The *intersection arc* is the part of a circle perimeter that is covered by other circle area. The parts of a circle perimeter that are not covered by any other circles are called *exposed arcs*.

The idea in this technical preliminary is to propose a proper solution to compute the circle's exposed arcs information. An exposed arc is determined in terms of the circle center and its endpoints in polar coordinates. The input of the proposed algorithm is a set of the intersection arcs. Accordingly, the intersection arcs are initially determined. As shown in Fig. 1, two circles C_i and C_j are overlapped in which their radii are r_i and r_j, respectively. The intersection arc of C_i due to C_j is the arc drawn from p_j to q_j in anticlockwise. The intersection arc's endpoints (p_j and q_j) are determined as follows. First, the polar angle of point p_j, $\theta(p_j)$ is determined.

The angle φ_j can be geometrically determined from the law of cosines (Pythagorean Theorem) in terms of r_i, r_j and d_{ij} as follows:

$$\varphi_j = \cos^{-1}\left(\frac{r_i^2 + d_j^2 - r_j^2}{2r_i d_j}\right) \tag{1}$$

Also, the angle φ_j can be basically determined in terms of the centers of both circles.

$$\phi_j = \cos^{-1}\left(\frac{y_j - y_i}{x_j - x_i}\right) \tag{2}$$

Hence, the polar angle $\theta(p_j)$ is determined as follows:

$$\theta(p_j) = 180^{\circ} - (\phi_j + \phi_j) \in \quad [0, 360] \tag{3}$$

The polar angle of the second endpoint q_j is determined as follows:

$$\theta(q_j) = \theta(p_j) + 2\phi \quad \in [0, 360] \tag{4}$$

The Eqs. (1)–(4) are repeated for each overlapped circle C_j with the current circle C_i to determine the endpoints of the resultant intersection arc. The endpoint class is created in which its attributes are the endpoint id, polar angle, intersection arc id and status. The status attribute is used to show the endpoint is either lower endpoint (p_j) or upper endpoint (q_j). All endpoints are stored in a priority queue in which the priority of a point is determined by the smallest angle, which means that the highest priority point has the smallest angle limited with a range from 0 to 360.

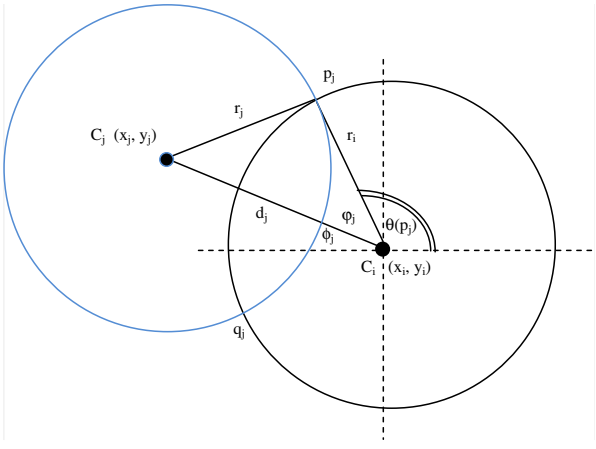

Figure 1 Sensor (Si) is intersected by sensor (Sj).

The *Sweep Radius Operation* is proposed to determine the exposed arcs. As shown in Fig. 2, four circles are assumed to overlap with the current circle in which the resultant intersection arcs are represented by dashed arcs. In the *Sweep Radius Operation*, the sweep radius scans the circle perimeter from the first point in the priority queue anticlockwise. When the first element p_1 is visited, which p_1 represents the lower endpoint of Arc_1, Arc_1 is stored into a list S. Afterwards, q_4 and q_1 are visited, respectively. At the point q_1, Arc_1 is removed from S, and then the list becomes empty. Empty list indicates that an exposed arc is discovered and then q_1 is the lower endpoint of such exposed arc. The *FindExposedArcs* Algorithm is described as follows:

ALGORITHM. FindExposedArcs (L)
INPUT. A set L of (x, y) points represent a sensor's location and its neighbors' locations
OUTPUT. A set E of exposed arcs belongs to one circle circumference and stored in a queue Q

```
1.  Determine a set I of intersection arcs
2.  Initialize an empty priority queue PQ. Next, insert the
intersection arc endpoints into PQ in which PQ starts with the
smallest lower point and ends with the start point; each endpoint
stores the arcs begin from and/or end at itself
3.  Initialize an empty list S
4.  Initialize an empty queue Q
5.  Visited = false
6.  WHILE PQ is not empty DO
7.     p = highest priority endpoint in PQ
8.     Delete the highest priority endpoint from PQ
9.     IF not visited THEN
10.       Temp = p
11.       Visited = true
12.    HandleEndpoint(p)
13. END Loop
END
PROCEDURE HandleEndpoint(p)
1.  IF p is an upper endpoint THEN
2.     IF list S is not empty THEN
3.        IF I' is not stored in the list S THEN
4.           clear Q
5.        ELSE remove I' from the list S
6.           IF list S is empty THEN
7.              Add E' into Q with lower endpoint p
8.           ELSE Swap (current Q, E' with p)
9.  IF p is a lower endpoint THEN
10.    IF there is E' in Q with no upper limit THEN
11.       Set p as an upper endpoint of E'
12.    Insert I' into the list S
END PROCEDURE
```

When p_2 is visited, which p_2 is the lower endpoint of Arc_2, p_2 is taken as the upper endpoint of the exposed arc and then Arc_2 is stored into S. Afterwards, the point q_2 is visited, and while q_2 is the upper endpoint of Arc_2, Arc_2 is removed from S and then the queue becomes empty again. Therefore, another exposed arc is discovered which q_2 is its lower endpoint. When p_3 is

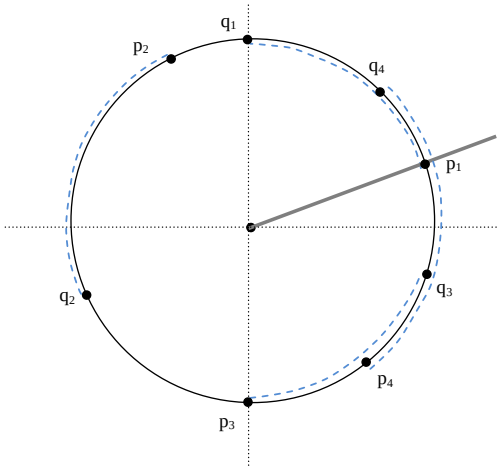

Figure 2 Sweep Radius Operation.

visited which p_3 is the lower endpoint of Arc_3, p_3 is taken as the upper endpoint of new exposed arc and then Arc_3 is stored into S. When the point p_4 is visited, Arc_4 is stored into S. When q_3 is visited, Arc_3 is removed from the list. At this point, q_3 cannot start new exposed arc because the list S is not empty. Finally, the sweep radius revisits p_1 and terminates the operation.

The time complexity of *Sweep Radius Operation* is illustrated as follows in details. The first step, as illustrated in line (1), determines the intersection arcs for the current circle. Assume the total number of intersection arcs is d; the time complexity of such step is $O(d)$. The intersection arcs' endpoints ($2d$) are pushed to a priority queue PQ according to the corresponding angle of each endpoint with complexity $O(2d)$. Each endpoint is popped from PQ with complexity $O(1)$ and handled by the *HandleEndpoint* procedure with complexity $O(1)$; therefore the complexity of the entire loop is $O(2d)$, as illustrated in lines from 6 to 13. Accordingly, the total complexity of *Sweep Radius Operation* is $O(n)$ which n is the number of overlapped circles.

4. The Adaptive Relocation Strategy

The Adaptive Relocation Strategy (*ARS*) is proposed to efficiently deploy heterogeneous sensors for large-scale applications. *Heterogeneous sensors* may differ in their characteristics such as the sensing and communication ranges, the remaining energy; furthermore, they may be static or mobile. The *ARS* strategy designs a geometric model, which in turn perfectly describes the coverage gaps through the target field. The proposed solution is mainly based on four types of announcements to plan the communication among sensing nodes.

(a) The *unidirectional sensor/sensor announcement* is used to enable each sensor to send a broadcast message to its communicated neighbors. This message carries the sensor id, current position, mobility status, sensing range, communication range and remaining energy level. When a sensor receives such announcement from its communicated neighbors, the neighbor information is registered in its neighbor list.

(b) The *bidirectional leader/sensor announcement* is used to enables leaders to send broadcast messages to its neighbors carrying sensor id and new status as a leader. The neighbor may be either other leader or ordinary sensor. When the neighbor leader j receives this broadcast message from the current leader i, the leader j inserts the leader i information into the leader group table. On the other hand, when ordinary sensor receives this type of announcement from one or more leaders; therefore, a sensor replies by one unicast massage to the nearest leader to avoid duplicated information among leaders.

(c) The *unidirectional leader/leader announcement* is used to enable a leader to send a multicast message to other leaders according to its leader group table. Such message carries the leader id and its incomplete polygons' information. When a leader receives from the incomplete polygons' information from other leader, such information is added to its incomplete polygons' information.

(d) The *unidirectional leader/mobile announcement* is used to enable a leader to send unicast messages to group of mobile sensors carrying leader id, target location and new score. When a mobile sensor receives from one or more leaders, the choice that achieves the highest score is selected to move. The Adaptive Relocation Strategy algorithm is describes as follows:

ALGORITHM ARS( )

BEGIN
1. Announcement_1(frame(sensor k))
2. Create List e(k) for sensor k
3. e(k) = SweepRadiusOperation (List LOC(k))
4. T(k) = Sum(e(k))
5. **IF** sensor(k).isMobile **THEN**
6. A(k) = BooleanPolygonOperation (List LOC)
7. Announcement_1(frame(static_sensor i), frame(mobile_sensor j))

At static sensor (i)
8. Create List T for collected T_i and List A for collected A_j and Loc_j
9. **IF** T_i is the maximum of {List T} **THEN**
10. static sensor(i).Leader = True
11. **ELSE** static sensor(i).Leader = False
12. Announcement_2(frame1(leader r), frame(sensor i));

At leader sensor (r)
13. Create List E for collected E_r, L for collected Leaders L_r and P for complete polygons
14. P_r = Create_Polygons(List E)
15. **Announcement_3(frame(leader r));** //Send incomplete polygons
16. Create List $\overline{P}$ for collected $\overline{P}$
17. **IF** $E(\overline{P}_r)$ is the maximum of $E(\overline{P})$ **THEN**
18. Append $\overline{P}_r$ to P_r
19. Create List M for midpoints of P_r and List FIT for mobile sensor-midpoint correspondence
20. M = Triangulation(P_r)
21. Fitting(selected(M), A_j, Loc_j)
22. **Announcement_4(frame2(leader r));** //Send selected M and new Score for mobile sensor

At mobile sensor (j)
23. Create List F for each new position and score from Leader r
24. IF F.Score(r) is the maximum F.Score and connected_after_move = true **THEN**
25. Move to F.Position(r)

END

Initially, as illustrated in lines (1–7), the *unidirectional sensor/sensor* announcement is performed to enable each sensor to know the neighbors' information such as neighbor's location and neighbor's sensing range. Accordingly, a sensor determines the intersection arcs, as shown above, to create a list (e) of the exposed arcs from *Sweep Radius Operation*. It is expected that such operation runs fast because the maximum number of overlapped sensing areas is about few tens or less when thousands of sensors are randomly deployed in the monitored field. The existence of exposed arcs indicates that a sensor is surrounded by undefined gap area(s). The total length of exposed arcs for sensor k, $T(k)$ is determined. When a sensor is static, an announcement is sent carrying sensor id and $T(k)$.

When a sensor is mobile, additional task is performed. A mobile sensor performs the *Boolean Polygon Operation*, discussed later in details, by considering its neighbors' locations to determine the area that is uniquely supervised by that mobile sensor which representing its current score. Afterwards, a mobile sensor sends an announcement carrying sensor id, $T(k)$ and its current score. While a static sensor, as illustrated in lines (8–12), creates two temporary lists in which the first list stores the total exposed arcs received from its neighbors (T) and the second list to store the current score received from mobile neighbors (S). A static sensor k compares its total exposed arcs length $T(k)$ with the list T. When a sensor k discovers that $T(k)$ is the greatest value and it has sufficient remaining energy, it is assigned as a leader.

The leader is a static sensor that performs additional computations. Before a leader starts its tasks, the second announcement is performed which a leader sends a *broadcast message* to its neighbors carrying sensor id and leader status. A sensor may receive a *broadcast messag*e from multiple leaders; therefore, a sensor chooses the nearest leader to send a *unicast message* to that leader carrying sensor id, sensor location and its exposed arcs' information. A sensor sends only to one leader to be supervised by one leader and save the communication and computation cost. As illustrated in lines (13–22), the leader performs its tasks, which initially creates three temporary lists. The first list, *Leader Group* (L), stores the other communicated leaders.

The second list, Exposed Arcs' information (E), stores the exposed arcs received from neighbors. Each exposed arc is transformed into edge (line segment as a first order approximation). Afterwards, the leader performs the *Polygon Construction Operation*, discussed later in details, to create complete polygons from the edges. When a complete polygon is constructed, it is stored in the third temporary list. On the other hand, such operation may construct incomplete polygon from known edges. The missed edges are known for the neighbor leaders; accordingly, the leaders are collaborated to transform incomplete polygons into complete polygons. Additional list is created to store the incomplete polygons' information to send it.

Such announcement is a multicast message where a leader sends a message to the *Leader Group* (L) carrying sensor id and its incomplete polygons' information. When a leader receives from other leaders, it attempts to construct complete polygons. However, before appending such polygon to the polygon list, a leader checks the polygon edges' owner. When a leader discovers it has a greatest number of edges, that polygon is appended to its polygon list. Afterwards, each leader performs the *Triangulation Operation*, discussed later in details, to divide each gap polygon into triangles in which additional list is created to store the midpoint of each triangle. Our solution is designed to move mobile sensors to proper midpoints.

In fact, the number of midpoints is usually larger than the number of mobile sensors. Consequently, the triangles whose large areas are selected to optimize the number of midpoints. Finally, the leader performs the *Fitting Operation*, discussed later in details, to fit the best mobile sensor to appropriate midpoint. The leader performs the last announcement in which *unicast messages* are sent to chosen mobile sensors carrying leader id, new score and new location. As illustrated in lines (23–25), the mobile sensor checks the network connectivity, i.e., the breadth search algorithm [18] can be used, through its zone area assuming it will leave the zone. When the connectivity is guaranteed, the mobile sensor checks the received requests from leaders.

The mobile sensor may receive from multiple leaders. The mobile sensor creates a list to store the received scores and examine the greatest value. Finally, the mobile sensor moves to the chosen position that achieves the highest score. After finishing this round, the coverage performance clearly increases. The *Adaptive Relocation Strategy* may perform one or two other rounds to slightly increase the coverage performance. In what follows, the basic operations of the proposed strategy are described.

5. The basic operations of the proposed strategy

5.1. The Boolean Polygon Operation

The *Boolean Polygon Operation* is applied at mobile sensors to determine their scores. For example, a sensing area of mobile sensor S_1 is marked by dashed lines and overlapped with three sensing areas for heterogeneous sensors S_2, S_3 and S_4, as shown in Fig. 3a. The current score of S_1 is represented by the dashed area that uniquely covered by S_1. Such area is determined as follows. All sensors' sensing area perimeters are initially approximated into hexagon shapes. Afterwards, the *Boolean Polygon Operation* is implemented, according to 2D Polygons algorithm [19], to determine vertices that represent the union area between two hexagons. The Boolean operations include union, intersection and difference between two polygons. The time complexity of a Boolean operation is $O((p + k) \log p)$ where p is the total number of the two polygon vertices and k is the number of vertices for the resultant polygon. The union operation is incrementally performed for all S_1 neighbors to obtain a polygon represent the union of all neighbors' sensing areas, as shown in Fig. 3b, which marked with B.

The sensing area of a mobile sensor S_1 is marked with A which the circular area is approximated into a hexagon shape. The Boolean difference operation is performed to determine the vertices that represent the difference area between A and B. Such difference area marked with C represents the S_1 current score. As shown above, the total number of two hexagons' vertices p is 12 and the number of resultant polygon's vertices k is less than p. Consequently, the total complexity of such operation for a mobile sensor is $O((n + 1)(p + k) \log p)$ where n represents the number of its overlapping neighbors. In fact, the number of the overlapping neighbors is around few tens at most. Therefore, the total complexity of such operation is

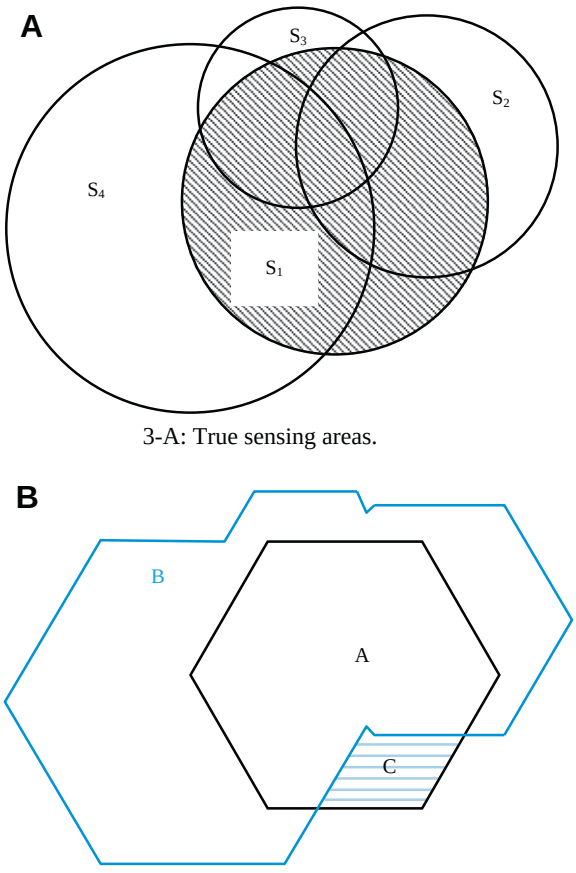

Figure 3 Boolean operations for 2D polygons.

considered $O(1)$. It is expected that a mobile sensor can determine its current score so fast with low computations.

The *Boolean Polygon Operation* is also applied at leaders in which a leader needs to estimate the expected score at the selected midpoint (p). A leader assumes a mobile sensor (i) at the selected midpoint (p) and discovers the sensors that are expected to overlap with its sensing area. A leader performs the *Boolean Polygon Operation*, as shown above, to determine the new score of a mobile sensor (i) when deciding to move to the selected midpoint (p). The leader repeats such operation for all selected midpoints. Assuming the maximum number of midpoints computed by a leader is m and the maximum number of mobile sensors discovered by that leader is n, then the total complexity of such operation at a leader is $O(nm)$. It is expected that the maximum value of the term (nm) is about few hundreds at most.

5.2. Polygon Construction Operation

The *Polygon Construction Operation* is performed at leaders to benefit from the received exposed arcs attempting to construct complete polygons. Initially, the exposed arcs are transformed into line segments (edges) because the first order approximation could facilitate the computations. The edges are stored in doubly linked list to provide with linear search. A node in a doubly linked list is provided with front and rear pointers. The front pointer is used to indicate to the next edge while the rear pointer is used to indicate to the previous edge. It is easy to determine the next edge for the current edge, which the upper endpoint of the current edge is the lower endpoint of the next edge. Similarly, the previous edge is determined. When no node is assigned as a next or previous node for the current node, either front or rear pointers is settled with *NULL* value.

Such operation searches the doubly linked to initially find incomplete polygons. When the first node having *NULL* rear is reached, such node is inserted into the first incomplete polygon data and deleted from the doubly linked list. According to the front pointer of the current node, the next node is assigned, then inserted into the first incomplete polygon data and deleted from the list. When a node having *NULL* front is reached, the first incomplete polygon structure is constructed. As long as there is a node having *NULL* rear in the list, an incomplete polygon can be discovered as shown above. When there is no node having *NULL* rear, the polygon construction operation starts to find the complete polygons. Starting with the first node, the next nodes are determined until reach the first node again constructing the first complete polygon and so on. The complete polygons are stored in P List and the incomplete polygons are stored in $\overline{P}$ List.

The *Polygon Construction Operation* is also performed at leaders when a leader receives incomplete polygons $\overline{P}$ from other leaders. The leader creates a doubly linked list with different structure in which the node data contains incomplete polygon information. The front and rear pointers are initially *NULL*. When an endpoint of incomplete polygon equals an endpoint of another incomplete polygon, the front or rear pointer is modified. Afterwards, the doubly linked list is searched, as discussed above, to find new complete polygons. Some polygons from the new complete polygons can be added to the leader information to avoid the duplicated polygons at other leaders. When a complete polygon is discovered, the leader determines how many edges it owns. When a leader has a greatest number of edges, the complete polygon is stored in P. The time complexity of *Polygon Construction Operation* is $O(n)$, where n represents the maximum number of edges because such operation depends on the doubly linked list structure.

5.3. Triangulation Operation

The *Triangulation Operation* is performed at leaders to manipulate the gap polygons attempting to assign the best location of mobile sensors, which in turn cover the gap. Such operation performs the *Monotone Polygon Triangulation algorithm* [19], which its time complexity is $O(v_i \log v_i)$ where v_i is the number of vertices for the gap polygon (i). Each polygon with v_i vertices is divided into $(v_i - 2)$ triangles. Assume the number of gap polygons is g and the total number of edges is E. Let the average number of vertices are V, then $E = gV$. The time complexity to triangulate a polygon is $O(V \log V)$, then the total complexity to triangulate g polygons is $O(gV \log V)$ or $O(E(\log E - \log g))$. The number of gap polygons is small number compared with the total number of edges. Accordingly, the time complexity of such operation is $O(E \log E)$. Compared with the above operations, the complexity of such operation generally increases from the order of $O(n)$ to the order of $O(n \log n)$.

After dividing the gap polygons into triangles, the midpoint of each triangle is determined and stored in Midpoints List (M) with size m. Generally, a gap polygon may contain short edges resulting in further midpoints. It is a bad choice to move

mobile sensors to all those points because the number of mobile sensors may be insufficient; furthermore, this leads to worst utilization of mobile sensor deployment. Accordingly, the total polygon area is determined to estimate the number of mobile sensors required (n) in which their sensing ranges are chosen as the average of sensing ranges of given heterogeneous sensors. Usually the number of mobile sensor required (n) is less than the number of midpoints (m). The *Triangulation Operation* attempts to reduce m to n by choosing the midpoints for triangles having the greatest areas. Accordingly, the Midpoints list size is reduced to n. It is clear that each element in the Midpoints list has several attributes, the midpoint position and different new scores depending on the number types of deployed heterogeneous sensors as discussed in Section 5.1.

5.4. The Fitting Operation

An important parameter is designed, Score to Distance Ratio (*SDR*), and formulated in Eq. (5) to enable the leaders to fit the best mobile sensor to an appropriate midpoint. The *SDR* is defined as the relative score (S_r) divided by the relative distance (D_r).

$$SDR(i,m) = \frac{S_r(i,m)}{D_r(i,m)}$$

where (5)

$$S_r(i,m) = \frac{S_{new}(i,m) - S_c(i)}{S_{new}(i,m)}, \text{ and } D_r(i,m) = \frac{D(i,m)}{r(i)}$$

The parameter $S_{new}(i,m)$ is the new score of mobile sensor (i) at midpoint location (m), $S_c(i)$ is the current score of mobile sensor (i), $D(i,m)$ is the distance between the current position of mobile sensor (i) and the midpoint location (m), and $r(i)$ is the mobile sensor (i) sensing range. When SDR ratio is negative, it means that the new score of mobile sensor (i) at (m) is less than its current score. Besides, the mobile sensor movement to (m) at such situation leads to decreasing the coverage performance. Otherwise, when the SDR ratio is positive, the midpoint location (m) that gives the highest SDR ratio is chosen as a new location for mobile sensor (i).

In fact, when the midpoint location (m) is chosen as the best location for mobile sensor (i), it is unnecessary that the mobile sensor (i) is the best sensor for the midpoint location (m). Consequently, the *Fitting Operation* is designed to solve this problem. A *FIT list* is created to establish the mobile sensors and the midpoints correspondence, which the *FIT list* size is ($i \times m$) where i represents the number of mobile sensors and m represents the number of midpoints. Each element of the *FIT list* consists of five attributes: {*SDR*, a mobile sensor id (i), a midpoint location (m), a connectivity state (c), a mobile sensor energy level (e)}.

Such elements are stored in a priority queue in which the priority is the attribute *SDR*; the highest priority element has the greatest score to distance ratio. The *Fitting Operation* linearly searches the priority queue to fit a best mobile sensor (i) to a midpoint location (m). At each element of the priority queue, three attributes are checked: (1) the *SDR* should be positive, (2) the connectivity is valid, and (3) the energy level of mobile sensor is appropriate to move. When those conditions are satisfied, both the current mobile sensor (i) and the current midpoint location (m) are marked as visited to avoid the duplication. Otherwise, when one of those conditions is not satisfied, such operation discovers that the current mobile sensor is improper for the current midpoint location and moves to the next element in the list and so on. Generally, some midpoints may be denied from coverage when the number of mobile sensors is less than the number of mobile sensors or there is no mobile sensor satisfies such midpoint requirements. Assuming the maximum number of midpoints computed by a leader is m and the maximum number of mobile sensors discovered by that leader is n, then the total complexity of such operation at a leader is O (nm). It is expected that the maximum value of the term (nm) is about few hundreds at most, as discussed in Section 5.1.

In the *Adaptive Relocation Strategy*, the deployed sensors perform different operations according to their characteristics. For example, the mobile sensor performs the *Sweep Radius* and the *Boolean Polygon* operations. On the other hand, the static sensor also performs the *Sweep Radius operation*, and when a static sensor is chosen as a leader, additional operations are performed such as *Polygon Construction*, *Triangulation* and *Fitting*. The time complexity is greater in the *Triangulation Operation* which reaches $O(E \log E)$ where E represents the maximum number of edges collected at a leader, and also greater at *Boolean Polygon* and *Fitting Operations* which reaches O (nm) where n represents the maximum number of mobile sensors discovered by a leader and m represents the maximum number of midpoints computed by that leader.

In fact, E is approximately in order of (nm); therefore, the total complexity of the *ARS* Strategy is $O(n \log n)$ where n represents the maximum number of edges collected at a leader. Such complexity is well appropriate for distributed computations. Furthermore, the proposed strategy is considered a distributed solution with an appropriate communication cost. As follows, the proposed solution performance is compared with the current distributed strategies for mobile sensors to show that the *ARS* strategy effectively achieves high coverage performance with less average moving distance and appropriate communication cost for heterogonous sensors.

6. Simulation results and performance comparisons

A simulation tool is developed by C# to implement the proposed geometric algorithms and examine the effectiveness of the proposed scheme (*ARS*) described above. Such simulation tool is used to perform several experiments to understand the behavior of heterogeneous sensor networks. In these experiments, the sensors deployment in a field of squared shape is considered for simplicity. The field dimensions are 60×60 m giving a total area of 3.6 km^2. Three different types of sensors with a sensing range of 4, 5, 6 m are used to match with other current sensor prototypes, such as Smart Dust (UC Berkeley), CTOS dust, and Wins (Rockwell) [20]. From [21], the current communication range equals 20 m.

Fig. 4 illustrates the different views of the tool's interface when thirty sensors are used from each sensor type giving a total of 90 sensors. Fig. 4a shows the sensing footprint for each sensor as initial positions which the dotted sensing footprint represents the mobile sensors and the gridded sensing footprint represents the static sensors chosen as leaders in the first round. While Fig. 4b illustrates the gap polygons constructed by leader sensors in which a polygon with multicolor edges indicates that it is constructed from collective information from multiple leaders.

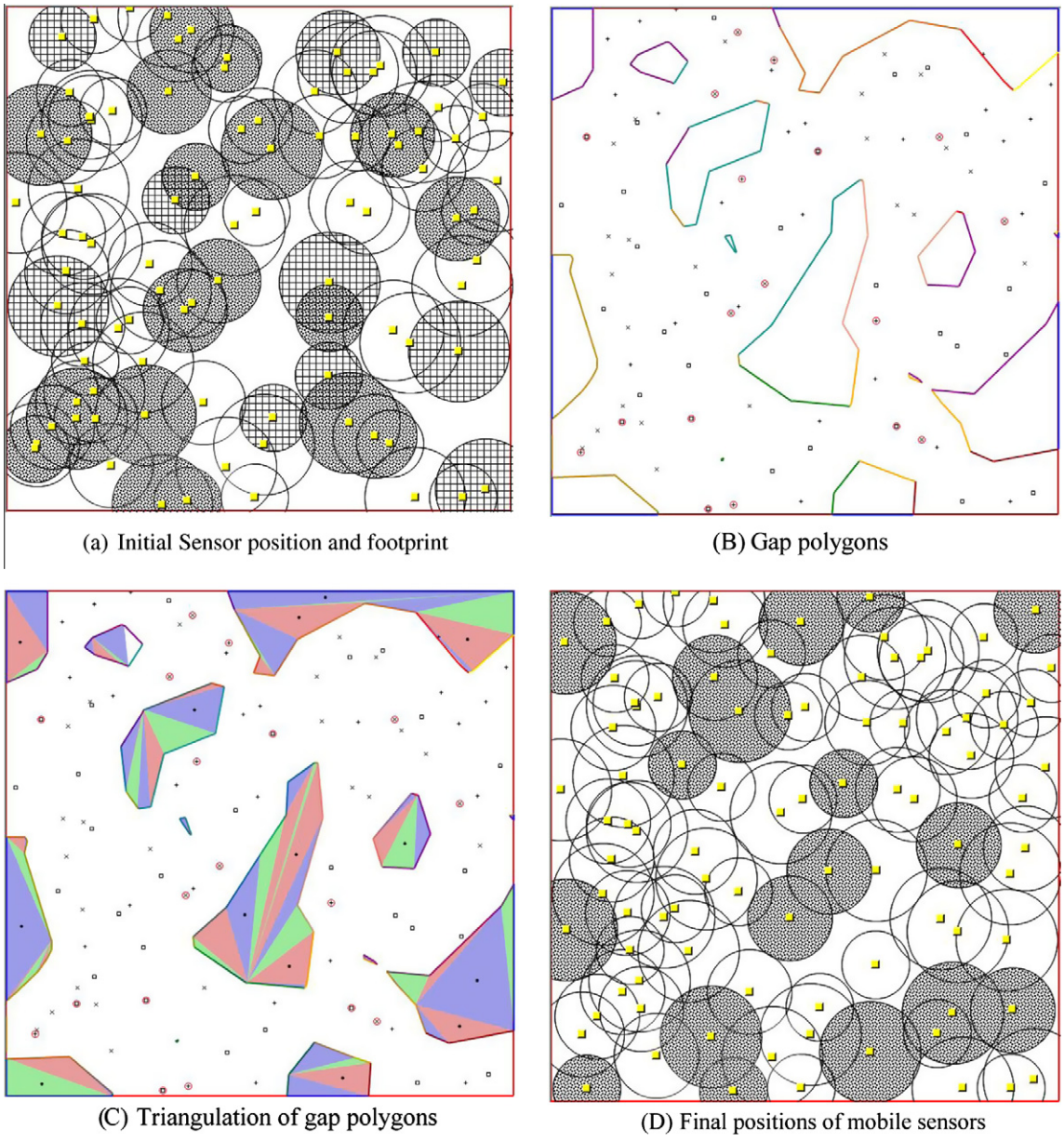

Figure 4 Different views of the simulation tool's interface.

Fig. 4c shows the output of the triangulation procedure and the selected midpoints. The final positions of mobile sensors are illustrated in Fig. 4d, which the coverage performance is clearly improved. The proposed simulator can compute the coverage performance by logically dividing the monitored field into small blocks. The number of blocks covered by deployed sensors represents the coverage area.

The minimum number of sensors required to achieve certain coverage is reduced when some sensors are chosen mobile. The first set of experiments is implemented to show the relation between the minimum number of sensors required to achieve certain coverage of the monitored field, and the number of mobile sensors. Initially, three types of sensors are chosen as mentioned above, and when all of them are static and randomly deployed, the minimum number of sensors required to achieve 90%, 95% and 97.5% of coverage, respectively, are 115, 150 and 180 sensors. Assuming there are 10% mobile sensors, the minimum number of heterogeneous sensors required reaches 100, 124 and 140 sensors as shown in Fig. 5.

Accordingly, the minimum number of sensors required to achieve certain coverage decreases when the percentage of mobile sensors included increases. The network cost depends on the cost ratio between mobile and static sensors. When such ratio approaches to one, the minimum network cost occurs at 100% mobile sensors. In fact, the mobile sensor cost is still higher than the static sensor cost; accordingly, when the cost ratio increases, the network cost increases with increasing the percentage of mobile sensors. In addition, choosing all sensors static cannot achieve minimum network cost because larger number of sensors is required to achieve the preferred coverage percentage. Accordingly, the choice of mobile sensors' percentage is depend on the planned coverage and the network cost in

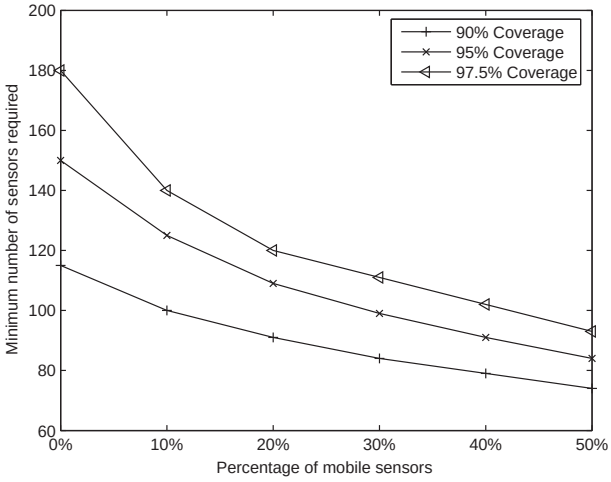

Figure 5 The effect of increasing mobile sensors.

terms of mobile/static sensor cost ratio. The proposed strategy performance is studied under several network metrics such as the percentage of coverage improvement, the average moving distance and the designed announcements cost. The percentage of mobile sensors is randomly chosen to vary from 10% to 50%, with an increment of 10%. Another set of experiments based on different initial distributions are applied, and then the average results are considered. Initially, we choose to randomly deploy thirty sensors of each sensor type giving a total of 90 sensors. The average of initial coverage for such set of experiments is approximately 83.84%.

In the *ASR strategy*, three fitting methods are suggested to evaluate the *Score to Distance Ratio* (*SDR*) parameter and its effect in improving the coverage performance. The first method, *FitDis*, is to fit based on the shortest distance. Such method can be applied at leaders when they decide which mobile neighbor is nearest to fit it to certain midpoint. In addition, the *FitDis* method can be applied at mobile sensors themselves that they receive from multiple leaders; therefore, they can choose the nearest midpoint to move. The second fitting method, *FitCov*, is to fit a mobile sensor with known score to a midpoint with higher score. The last fitting method, *FitSDR*, is to fit a mobile sensor to a midpoint based on the score to distance ratio. Both *FitCov* and *FitDSR* are applied at leaders and mobile sensors similar to the *FitDis* method.

As follows, the network performance is measured at different scenarios. Fig. 6 illustrates the coverage performance and the average moving distance when leaders perform the *FitDis* method. As shown in Fig. 6a, the coverage performance increases when the percentage of mobile sensors increases. When performing *FitSDR* or *FitCov* at mobile sensors to choose their targets, the coverage performance is better than the coverage at *FitDis*. At 50% mobile sensors, the coverage percentage reaches 95% at *FitSDR* and *FitCov* while it reaches 93.9% at *FitDis*.

However, *FitDis* saves the movement energy compared with other methods, which it performs less average moving distance, i.e., 1.76 m at 50% mobile sensors, as shown Fig. 6b. In addition, the *FitSDR* at mobile sensors saves energy more than the *FitCov* method, i.e., 2 and 2.2 m, respectively, at 50% mobile sensors. While the *FitCov* method performs larger moving distance, such increment is small because the move-

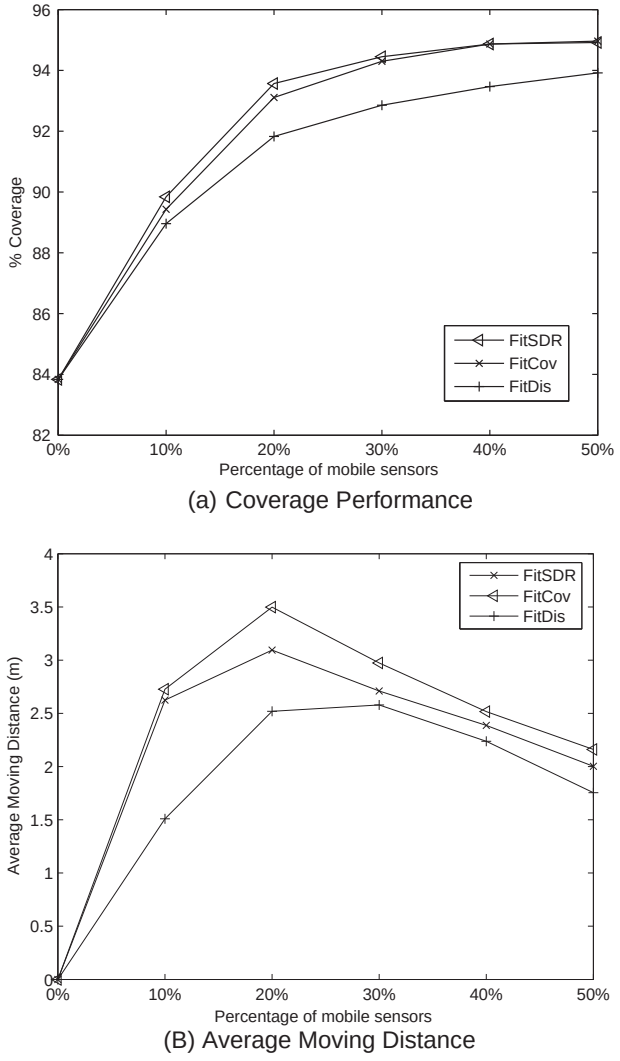

Figure 6 The network performance when the *FitDis* method is applied at leaders.

ment is originally controlled at leaders, which the *FitDis* method is applied.

Fig. 7 shows the coverage performance and the average moving distance when leaders perform the *FitCov* method. As shown in Fig. 7a, applying *FitCov* also at mobile sensors achieves the highest coverage percentage, i.e., 96.2% at 50% mobile sensors, while applying *FitSDR* or *FitDis* leads to less coverage performance, i.e., 95.3% and 95.4% at 50% mobile sensors, respectively. The average moving distance reaches 6.3, 5.3 and 5 m at 50% mobile sensors for *FitCov*, *FitSDR* and *FitDis*, respectively. While the results show the coverage performance of applying *FitCov* at leaders is rather improved comparing with the results of applying *FitDis*, the average moving distance increases by increasing the percentage of mobile sensors as shown in Fig. 7b. Therefore, while the number of mobile sensors increases, they move longer distances seeking for the best score regardless of their energy constraints.

The final set of experiments is performed to measure the network performance when applying the *FitSDR* at leaders as shown in Fig. 8. The results of such set show that applying the *FitSDR* method at leaders makes a balance between the network performance parameters.

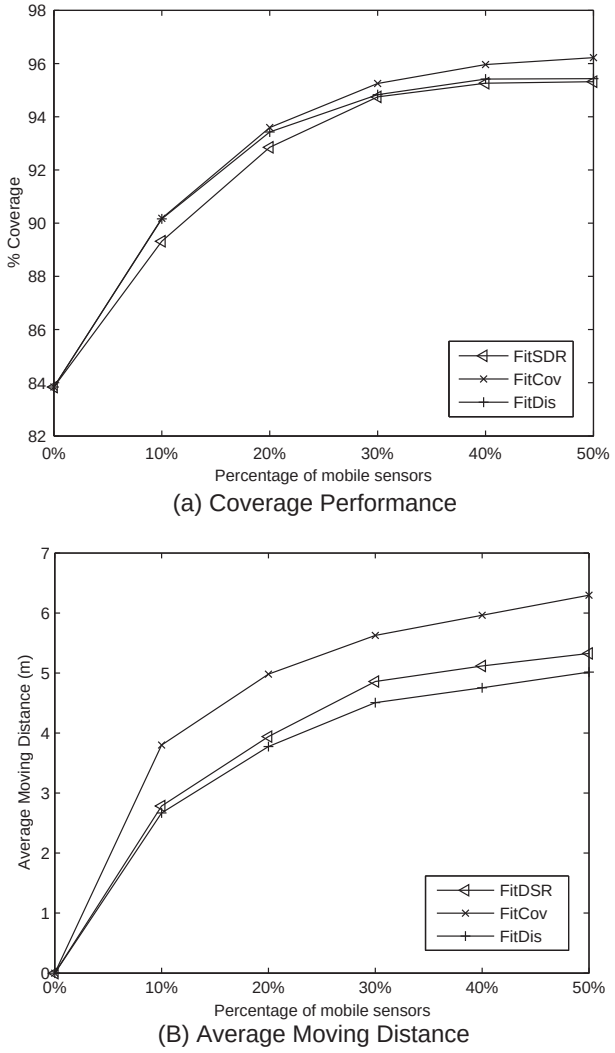

Figure 7 The network performance when the *FitCov* method is applied at leaders.

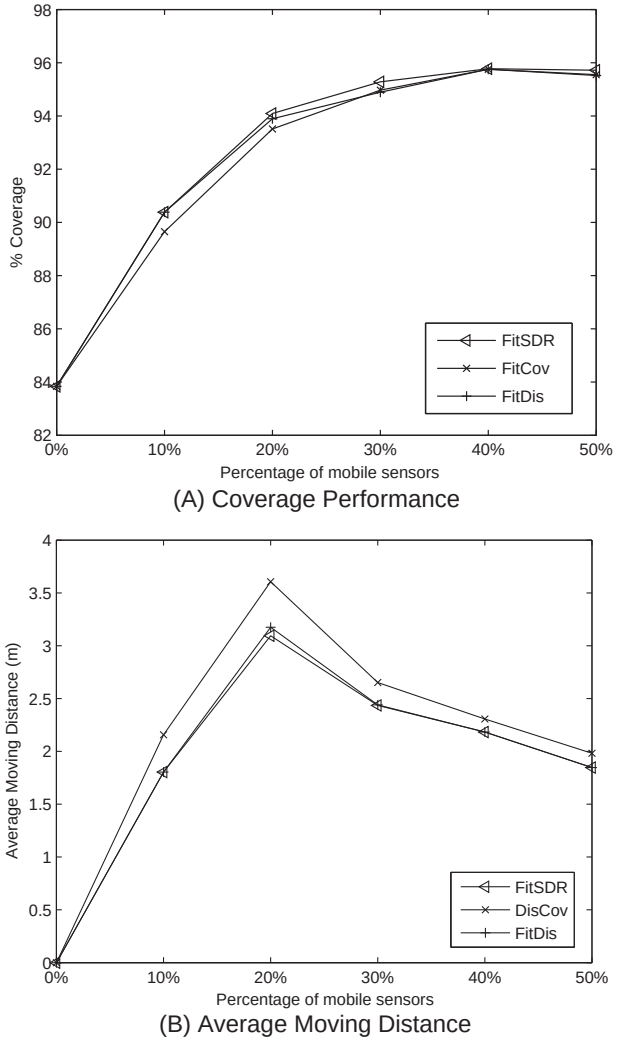

Figure 8 The network performance when the *FitSDR* method is applied at leaders.

The coverage performance, as shown in Fig. 8a, is approximately similar to the results of applying the *FitCov* method at leaders, i.e., 95.8%, 95.6% and 95.5% at 50% mobile sensors for *FitSDR*, *FitCov* and *FitDis* at mobile sensors, respectively. While the average moving distance, as shown in Fig. 8b, is approximately similar to the results of applying the *FitDis* method at leaders, i.e., 1.8, 1.9 and 1.8 m at 50% mobile sensors for *FitSDR*, *FitCov* and *FitDis* at mobile sensors, respectively. As mentioned above, at 50% mobile sensors, the greatest coverage percentage is 96.2% and the smallest average moving distance is 1.76 m. To summarize the results, we normalize the results at 50% mobile sensors with respect to the greatest coverage percentage and the smallest average moving distance as show in Tables 1 and 2. From such tables, we can easily discover that the *FitSDR* method is more appropriate to be applied at leaders and mobile sensors.

Finally, the monitored field dimensions are expanded to 300×300 m giving a total area of 90 km^2 and 1800 heterogeneous sensors are deployed to measure the network performance. The proposed strategy effectively runs, when a large-scale sensor network is considered, which gives the same rate of coverage improvement and average moving distance. The *FitSDR* method is still better than the other methods for large-scale sensor networks.

Table 1 Normalized coverage percentages.

		Leaders		
		FitDis	*FitCov*	*FitSDR*
Mobile sensors	*FitDis*	0.976	0.992	0.992
	FitCov	0.987	1	0.993
	FitSDR	0.986	0.990	0.996

Table 2 Normalized moving distances.

		Leaders		
		FitDis	*FitCov*	*FitSDR*
Mobile sensors	*FitDis*	1	2.85	1.02
	FitCov	1.23	3.58	1.08
	FitSDR	1.14	3.03	1.02

7. Conclusion and further work

An efficient strategy for autonomous sensor deployment is proposed to meet the heterogeneity requirements. As part of this work, lightweight geometric algorithms are designed to enable group of static sensors working as leaders to supervise the coverage gaps in the monitored field. Afterwards, the leaders exploit the designed algorithms to choose the best mobile sensors fitting them into the coverage gaps. The designed algorithms achieve an efficient relocation approach that maximizes the overall field coverage at proper average moving distance. Several simulation experiments are performed to examine the efficiency of the proposed strategy. Based on the results of such experiments, the proposed strategy autonomously enhances the heterogeneous sensors distribution in the field with minimum computations at sensor nodes, i.e., $O(n \log n)$ time complexity. Several extensions are considered for this research; for example, effort is underway to develop this work to enhance the designed algorithms. The further algorithms precisely construct the coverage gaps in the second order approximation to maximize the coverage.

References

[1] Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. IEEE Commun Mag. 2002:102–14.

[2] Sohraby K, Minoli D, Zntai T. Wireless sensor networks: technology, protocols, and applications. Hoboken, New Jersey: John Wiley and Sons, Inc.; 2007.

[3] Sibley GT, Rahimi MH, Sukhatme GS. Robomote: a tiny mobile robot platform for large-scale sensor networks. Proc IEEE Int Conf Robotics and Automation (ICRA'02). 2002.

[4] Lambrou T, Panayiotou T. Collaborative area monitoring using wireless sensor networks with stationary and mobile nodes, EURASIP J Adv Signal Process. Jan 2009.

[5] Alsharabi N, Ren Fa L, Zing F, Ghurab M. Wireless sensor networks of battlefields hotspot: challenges and solutions, 6th Int Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops (WiOPT). Apr 2008; 192–96.

[6] Howard A, Mataric MJ, Sukhatme GS. Mobile sensor network deployment using potential fields: a distributed scalable solution to the area coverage problem, Proceedings of the 6th International Conference on Distributed Autonomous Robotic Systems (DARS02), Fukuoka, Japan, 2002; 299–308.

[7] Zou Y, Chakrabarty K. Sensor deployment and target localizations based on virtual forces. Proc IEEE INFOCOM 2003.

[8] Abdel-Mageid S, Ramadan R. Efficient deployment algorithms for mobile sensor networks. IEEE Int Conf on Autonomous and Intelligent Systems (AIS). Jun 2010.

[9] Wang G, Cao G, LaPorta T. A bidding protocol for deploying mobile sensors. Proc IEEE Int Conf Network Protocols (ICNP'03). Nov 2003.

[10] Wang G, Cao G, LaPorta T. Bidding protocols for deploying mobile sensors. IEEE Trans Mobile Comput 2007;6(5).

[11] Ramadan R, Abdel-Mageid S. Efficient deployment of connected sensing devices using circle packing algorithms. IEEE Int Conf on Autonomous and Intelligent Systems (AIS). Jun 2010.

[12] Fan J, Parish D. Optimization of wireless sensor networks design using a genetic algorithm, parallel and distributed computing and systems proc. ACTA Press; 2008.

[13] Hamed A. A genetic algorithm for finding the k shortest paths in a network. Egypt Inform J 2010;11(2) (Production and hosing by Elsevier).

[14] Wang, X, Sun, X, Wang S, Wang Y. Distributed strategy for sensing deployment in wireless sensor networks. Int Symposium on Communications and Information Technologies (ISCIT). Dec 2010.

[15] Wang G, Cao G, LaPorta T. Movement-assisted sensor deployment. Proc IEEE INFOCOM 2004:469–2479.

[16] Niculescu D, Nath B. Ad Hoc Positioning Systems (APS) using AoA. Proc IEEE INFOCOM 2003.

[17] Savvides A, Han C, Strivastava MB. Dynamic fine-grained localization in Ad-Hoc networks of sensors. Proc ACM MobiCom 2001.

[18] Cormen T, Leiserson C, Rivest R, Stein C. Introduction to algorithms. MIT Press; 2009.

[19] de Berg M, van Kreveld M, Overmars M, Cheong O. Computational geometry algorithms and applications. Springer-Verlag; 2008.

[20] Wireless Sensing Networks, <http://wins.rsc.rockwell.com>, 2004.

[21] Berkeley Sensor and Actuator Center, <http://www.bsac.eecs.berkeley.edu>, 2004.