

Self-Organized Node Placement for Area Coverage in Pervasive Computing Networks

Dibakar Saha and Nabanita Das

Abstract In pervasive computing environments, it is often required to cover a certain service area by a given deployment of nodes or access points. In case of large inaccessible areas, often the node deployment is random. In this paper, given a random uniform node distribution over a 2-D region, we propose a simple distributed solution for self-organized node placement to satisfy coverage of the given region of interest using least number of active nodes. We assume that the nodes are identical and each of them covers a circular area. To ensure coverage we tessellate the area with regular hexagons, and attempt to place a node at each vertex and the center of each hexagon termed as *target points*. By the proposed distributed algorithm, unique nodes are selected to fill up the target points mutually exclusively with limited displacement. Analysis and simulation studies show that proposed algorithm with less neighborhood information and simpler computation solves the coverage problem using minimum number of active nodes, and with minimum displacement in 95 % cases. Also, the process terminates in constant number of rounds only.

Keywords Area coverage · Node deployment · Pervasive computing · Wireless sensor networks · Hexagonal tessellation

1 Introduction

In many applications of pervasive computing from home and health care to environment monitoring and intelligent transport systems, it is often required to place the sensors or computing nodes or access points to offer services over a predefined

D. Saha (✉) · N. Das

Advanced Computing and Microelectronics Unit, Indian Statistical Institute,
Kolkata, India

e-mail: dibakar.saha10@gmail.com

N. Das

e-mail: ndas@isical.ac.in

© Springer India 2016

A. Nagar et al. (eds.), *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*, Smart Innovation, Systems and Technologies 43, DOI 10.1007/978-81-322-2538-6_38

365

area. In some cases, like mobile surveillance, vehicular networks, mobile ad hoc networks, wireless sensor networks etc., the nodes are mobile and have limited energy, limited storage and limited computation and communication capabilities. These networks are often self-organized, and can take decision based on their local information only. In wireless sensor networks (WSN), large number of sensor nodes are spatially distributed over an area to collect ground data for various purposes such as habitat and ecosystem monitoring, weather forecasting, smart health-care technologies, precision agriculture, homeland security and surveillance. For all these applications, the active nodes are required to cover the area to be monitored. Hence, for these networks, the coverage problem has emerged as an important issue to be investigated. So far, many authors have modeled the coverage problem in various ways but most of them considered static networks. For deterministic node deployment, centralized algorithms can be applied to maximize the area coverage assuming the area covered by each node to be circular or square [9, 11, 12]. Many authors solved the coverage problem by the deterministic node placement techniques to maximize the network lifetime and to minimize the application-specific total cost. In paper [3], authors investigated the node placement problem and formulated a constrained multi-variable nonlinear programming problem to determine both the locations of the nodes and data transmission pattern in order to optimize the network lifetime and the total power consumption. Authors in [10, 15] proposed random and coordinated coverage algorithms for large-scale WSNs. But unfortunately, in many potential working areas, such as remote harsh environments, disaster affected regions, toxic regions etc., sensor deployments cannot be done deterministically. For random node deployment, virtual partitioning is often used to decompose the query region into square grid blocks and the coverage problem of each block by sensor nodes is investigated [6, 13, 14].

Whether the node deployment be deterministic or random, there is little scope of improving the coverage once the nodes are spatially distributed and they are static. Hence, mobility-assisted node deployment for efficient coverage has emerged as a more challenging problem. Many approaches have been proposed so far, based on virtual force [5, 18, 20], swarm intelligence [7, 8], and computational geometry [16], or some combination of the above approaches [4, 17]. In [19], a movement-assisted node placement method is proposed based on Van Der Waal's force where the relationship of adjacency of nodes was established by Delaunay Triangulation and force is calculated to produce acceleration for nodes to move. However, the computation involved is complex and it takes large number of iterations to converge. Authors in [1] proposed a distributed algorithm for the autonomous deployment of mobile sensors called *Push and Pull*, where sensors autonomously coordinate their movements in order to achieve a complete and uniform coverage. In [16], based on Voronoi diagram, authors designed and evaluated three distributed self-deployment algorithms for controlling the movement of sensors to achieve coverage. In these protocols, sensors move iteratively,

eventually reaching the final destination. This procedure is also computation intensive and may take longer time to converge. Moreover each node requires the location information of its every neighbor to execute the algorithm.

In this paper, given a random node deployment over a 2-D region, a simple self-organized distributed algorithm is proposed to satisfy coverage of a given region of interest using minimum number of nodes with limited mobility. To avoid an iterative procedure, here some target points are specified deterministically by tessellating the area with regular hexagons. After deployment, each node computes the nearest target point. Next, a unique node closest to a target point is selected in a distributed fashion based on local position information only, to move towards the target point. In this way, nodes attempt to fill up the target points mutually exclusively with minimum displacement. The set of nodes selected is made active to cover the area. It is evident that compared to the works in [16, 19], the computation involved in this algorithm is significantly simple, it requires no location information of its neighbors and it converges faster in two rounds only. Analysis and simulation results show that proposed algorithm with less neighborhood information and simpler computation solves the coverage problem using minimum number of active nodes, and with minimum displacement in 95 % cases. Also, the process terminates in constant number of rounds only.

The rest of the paper is organized as follows. Section 2 defines the problem and introduces the basic scheme of area coverage. Section 3 presents the distributed algorithm for self-deployment. Section 4 evaluates the performance of the proposed protocol by simulation. Finally, Sect. 5 concludes the paper.

2 Movement Assisted Area Coverage

2.1 Problem Formulation

Let a set of n nodes $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ be deployed randomly over a 2-D region A . It is assumed that each node is homogeneous, and covers a circular area with fixed sensing radius r . The goal of this paper is that given the random uniform distribution of a set of n nodes over a 2-D plane, to select a subset $P \subseteq \mathcal{S}$ and to place them at nearest target points such that the cardinality of P is minimum and it covers the area. Our objective is to develop a light weight self-organized distributed algorithm for node rearrangement to reduce the amount of computation and rounds of communication, and the average distance traversed by a node, to maximize coverage utilizing minimum number of nodes. This in turn helps us to conserve energy better and hence to enhance the network lifetime.

2.2 Area Tessellation

Given a rectilinear area to be covered by nodes, Fig. 1 shows a typical regular placement of nodes such that the overlapped region is minimum and the area is fully covered using minimum number of nodes. The positions of all the nodes basically defines a set of regular hexagons that tessellates the area as shown in Fig. 2. The sensor nodes are to be placed exactly on the vertices and the centers of the hexagons, termed here as the *target points*. In [2], it is proved that such node placement technique maximizes the area coverage using minimum number of nodes. In this case, the minimum number of nodes corresponds to the total number of *target points*, and can be computed easily as a function of the sensing radius r as shown below.

Let A be a 2-D axis-parallel rectangle $L \times W$ with $(0, 0)$ as the bottom-left corner point, termed here as the origin. For any arbitrary bottom-left corner point (x_0, y_0) , the co-ordinate system is to be translated appropriately. The tessellation of A with

Fig. 1 Target points for node placement

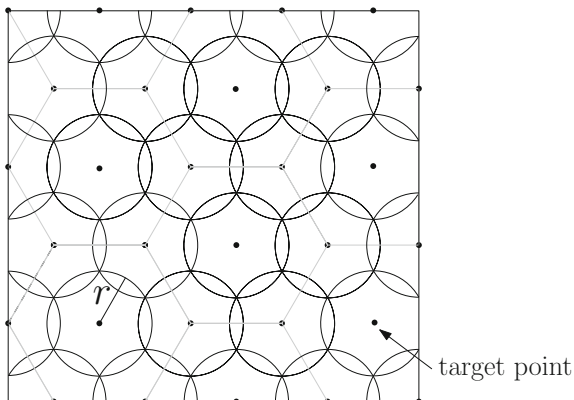
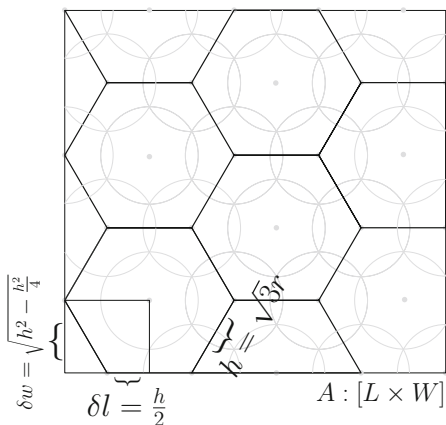


Fig. 2 Hexagons to tessellate the given area



regular hexagons of side h is shown in Fig. 2, where $h = \sqrt{3}r$ [1]. It is to be noted that the target points lie along some rows and columns parallel to the x -axis and y -axis respectively. Rows are separated by a distance:

$$\delta w = \sqrt{h^2 - \frac{h^2}{4}} = \frac{3}{2}r.$$

Similarly, columns are separated by a distance:

$$\delta l = \frac{h}{2} = \frac{\sqrt{3}}{2}r.$$

From Fig. 2, it is clear that, each even row- i starts with a target point $(0, i.\delta w)$, whereas each odd row- j starts with a target point $(\delta l, j.\delta w)$. Hence given the area A , the total number of rows is given by: $N_{row} = \left\lceil \frac{W}{\delta w} \right\rceil + 1$. The total number of target points is,

$$N = N_{row} \cdot \left\lceil \frac{L}{2\delta l} \right\rceil + \left\lceil \frac{N_{row}}{2} \right\rceil.$$

Therefore, to cover the area A , the number of nodes to be deployed is $n \geq N$, to fill up the target points exclusively. However, in practice, with random distribution of nodes, the area to be monitored is over-deployed, and $n \gg N$, providing sufficient redundant nodes to ensure coverage.

2.3 Nearest Target Point Computation

Let a set of n nodes $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ be deployed randomly over a 2-D region A . Each node- i only have the information of its physical location (x_i, y_i) and the sensing range r . Now to estimate the location of its nearest target point, it should have the knowledge of the origin, i.e. the bottom-left point of the area. The sink may directly broadcast it to all nodes for a static area. In case, the area of interest is dynamic, or depends on the deployment, the nodes may determine the origin as the point with minimum abscissa and ordinate of all the nodes deployed as described below. Here, during initialization, each node- i broadcasts its own location (x_i, y_i) , and maintains two variables initiated as $x_{min} = x_i$, and $y_{min} = y_i$ to keep the minimum abscissa and ordinate of all of the deployed nodes. It receives the messages with locations (x_j, y_j) from other nodes- j , and if $x_j \leq x_{min}$ and/or $y_j \leq y_{min}$ the values of x_{min} (y_{min}) are changed appropriately, and if there is any update, the new message is broadcasted again, otherwise it is ignored. In this way, after sufficient time, say T , all the nodes will acquire the same value of x_{min} and y_{min} and consider it as the

origin of the area under consideration. In case of an event, the affected nodes may define the event area in terms of this origin dynamically. Now, the initialization phase is completed and the next phase starts. In the worst case, each node may have to transmit n messages to complete the procedure.

After initialization phase, the nearest target point is to be computed by each node. We assume that each node knows the origin (x_{min}, y_{min}) of A . Next, each node i at location (x_i, y_i) , attempts to find out its nearest target point. It computes

$$t_x(i) = NI\left(\frac{|x_i - x_{min}|}{\delta h}\right)$$

and

$$t_y(i) = NI\left(\frac{|y_i - y_{min}|}{\delta w}\right).$$

Here $NI(x)$ denotes the nearest integer value of x . Next, it finds the location of its nearest target point $T_i(x_{Ti}, y_{Ti})$ as:

$$\begin{aligned} y_{Ti} &= t_y(i) \cdot \delta w \\ x_{Ti} &= t_x(i) \cdot h, \quad \text{when } t_x(i) \text{ is even,} \\ &= t_x(i) \cdot h + \frac{h}{2} \quad \text{otherwise.} \end{aligned}$$

Thus, each node finds its nearest target point and broadcasts it to its neighbors which lie within its communication range to select a unique node in a distributed fashion to be moved to a given target point exclusively. So, it is important to define the communication range of a node to define its set of neighbors.

2.4 Role of Communication Range

So far, we have mentioned the sensing range of the sensor node- i that defines the circular area with radius r , centered at node- i to be the area covered by node- i . When a node executes a distributed algorithm, it is very important to identify its neighborhood with which it can communicate directly. For that we should specify the communication range r_c of a node- i which indicates that when a node- i transmits, a node- j can receive the packet if and only if, the distance between the nodes $d(i, j) \leq r_c$. It is important to note that for all practical purposes, r_c is independent of r , since the transceiver hardware of the sensor node determines r_c and r is the property of the sensing hardware. For the proposed algorithm, it is required that

each node should cooperate with its neighboring nodes which are within a distance of $2(h+r) = 2(1+\sqrt{3})r$. So, here we assume that $r_c \geq 2(1+\sqrt{3})r$, where r is the sensing range.

3 Algorithms for Area Coverage

In wireless sensor networks, since nodes have limited computing and communication capabilities, it is always better to adopt distributed algorithms where nodes may take decisions with simple computation based on their local information only to produce a global solution.

Algorithm 1: Target Point Computation

```

Input: node:  $i(x, y)$ 
Output: Target point:  $T_i(x_{T_i}, y_{T_i})$ 
 $t_y \leftarrow NI(\frac{y}{l});$ 
if  $t_y$  is even number then
  |  $t_x \leftarrow NI(\frac{x}{h});$ 
  |  $x_{T_i} \leftarrow t_x \times h;$ 
else
  |  $t_x \leftarrow \lfloor x - \frac{h}{2} \rfloor;$ 
  |  $t_x \leftarrow NI(\frac{t_x}{h});$ 
  |  $x_{T_i} \leftarrow (t_x \times h) + \frac{h}{2};$ 
 $y_{T_i} \leftarrow t_y \times l;$ 

```

Here initially, each node is in *Active* mode and knows its location and the origin (x_{min}, y_{min}) of the area to be covered. In the first round, each node- i assumes a virtual tessellation of the area with hexagon tiles and compute its nearest target point by Algorithm 1. It broadcasts a *Target* message with data $(t_i(x, y), d_i)$, where $t_i(x, y)$ is its nearest target point and d_i is its distance from $t_i(x, y)$. Each node- i waits till it receives *Target* messages from all of its neighbors. Then it checks if it is at minimum distance from the target $t_i(x, y)$. Then node- i is selected to fill up the target $t_i(x, y)$. The case of tie may be resolved by node-id. It broadcasts *Selected* $(i, t_i(x, y))$ message and moves to $t_i(x, y)$ point with displacement d_i and goes to *Active* state. Otherwise, it goes into the *Idle* state. It is clear that within the circle C_i of radius r around a target point $t_i(x, y)$, if there exists at least one node, it will be filled up by it. The problem arises if any circle C_i is originally empty due to initial random node deployment. In that case, the *Idle* nodes will execute a second round of computation. Each idle node- i finds if there is any unfilled target node around it, i.e. within the six adjacent circles overlapping with C_i . Next it finds the unfilled target points and sort them according to the distance from it. Next it follows the same procedure described above for each target unless it becomes *selected* or all the targets are filled up.

The distributed **Algorithm 3**, describes the sequence of steps of the procedure.

Algorithm 2: Node Selection for Target Point

Input: node: i , $STATUS$ (active =1 or idle=0)
Output: movement: true/false
movement=true;
for each neighbor node do
 if receives target($t^j(x, y), d^j$) **message then**
 if $t^i(x, y) == t^j(x, y)$ // same target point **then**
 if $d^i > d^j$ **then**
 movement=false;
 $STATUS(i) \leftarrow 0$;
 if $d^i == d^j$ **then**
 if $i > j$ **then**
 movement=false;
 $STATUS(i) \leftarrow 0$;
 if movement==true **then**
 Move towards target point $t^i(x, y)$;
 $STATUS(i) \leftarrow 1$;
 broadcasts $Selected(t^i(x, y))$ message;

Algorithm 3: Distributed Algorithm for Adjacent Target Points

Input: free node n_i
Output: Active or idle
for each node i do
 Compute nearest target point $t^i(x, y)$ (call **Algorithm 1**);
 Compute all six neighbor target points of $t^i(x, y)$ and distances from its position.
 Include all the target points in \mathcal{L}_i sorted by distance \mathcal{D}_i ;
 for $\mathcal{L}_i \neq \{\phi\}$ **do**
 Take first point $t^i(x, y)$ and d_i from \mathcal{L}_i and \mathcal{D}_i respectively;
 broadcasts $target(t^i(x, y), d_i)$ message;
 Wait and listen until receives all *Target* message from the neighbors;
 Call **Algorithm 2**;
 if $STATUS(i) == 1$ **then**
 Goto Active Mode;
 Free \mathcal{L}_i and \mathcal{D}_i ;
 Terminate;
 else
 if receives $Selected(t^j(x, y))$ **message then**
 Remove target point $t^j(x, y)$ from \mathcal{L}_i ;
 Terminate and goto idle Mode;

3.1 Complexity Analysis

It is evident that to find the target points **Algorithm 1** is computed in constant time. To take decision for selecting a unique node for each target point, each node waits until it receives all the *target* messages from its d neighbors. Each node takes $O(d)$ time to get the minimum distance from its d neighbors. Therefore, **Algorithm 2**

computes in $O(d)$ time, where d is the maximum number of neighbors of a node. Finally, each node attempts to fill up at most seven target points. Therefore, the total time complexity of the distributed algorithm (**Algorithm 3**) is $O(d)$. In the distributed algorithm nodes broadcast at most seven *Target* messages and only one *Selected* message. Therefore, per node at most eight messages are needed to complete the procedure. Hence, the message complexity per node is $O(1)$ only.

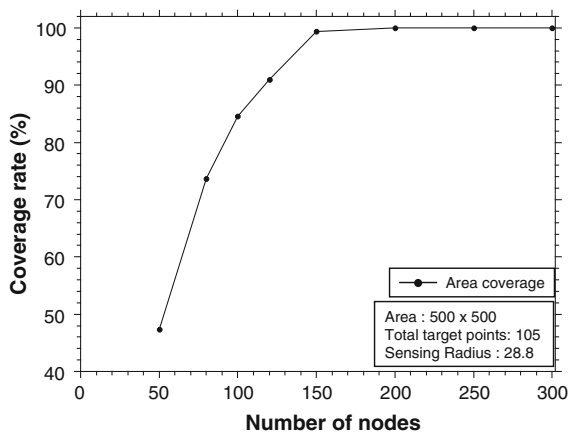
4 Simulation Results

In our simulation study, we assume that n nodes, $250 \leq n \leq 300$, are distributed randomly over a 500×500 area with radius $r = 28.86$ and side of the hexagon $h = 50$. All the target points associated with the circles are computed by nodes. After executing the distributed algorithm, each node moves to its target point. The performance of the proposed algorithm is evaluated in terms of coverage, rounds of computation needed and displacement of nodes. The graphs show the average value of 20 runs for 20 independent random deployments of nodes.

Figure 3 shows the variation of coverage percentage with n , the total number of nodes deployed. If the total number of target points is 105, for $n = 50, 100, 150$, the coverage percentage is found to be 47, 84.57 and 99.36 % respectively. It gives an idea that how much an area should be over deployed to achieve 100 % coverage with random node deployment.

In Fig. 4, the variation in the number of computation rounds with n is presented. It is to remember that with random deployment, if there is no empty circle C_i with center at a target t_i and radius r , the procedure completes in a single round only. This fact is exactly revealed in Fig. 4. For $n = 100, 150, 200$, target points are filled

Fig. 3 Coverage rate with number of nodes



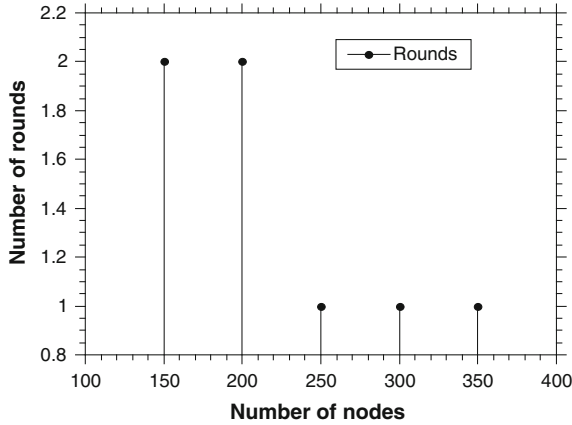


Fig. 4 Number of rounds for filling the target points

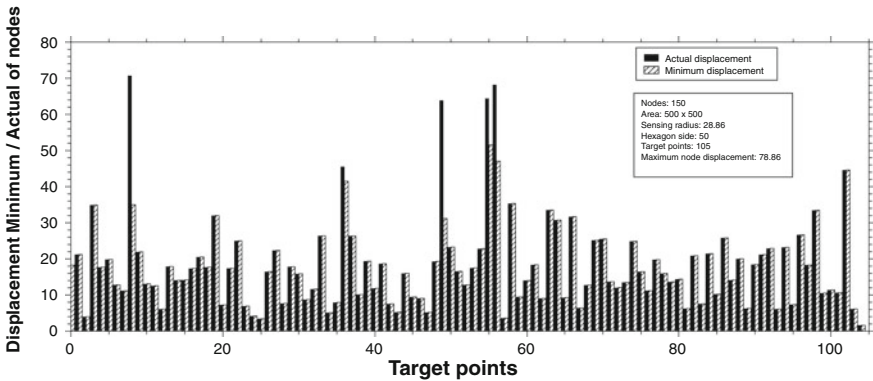


Fig. 5 Node displacement versus target points

up in two rounds whereas if $n > 200$, the proposed technique takes a single round to complete.

For a given random deployment of $n = 150$ nodes, Fig. 5 shows the distances traversed by each node to fill up target point with $r = 28.86$. It shows that in more than 95% cases, the target is filled up by a node with minimum possible displacement. It is obvious that with greater values of n , this percentage can be improved further.

5 Conclusion

In this paper, we propose a self-organized node placement algorithm to satisfy the coverage of a given region of interest in Wireless Sensor Networks. The area is logically tessellated by regular hexagonal tiles starting from an origin. To get full coverage with random deployment of n nodes over a $2D$ region, we need to place unique nodes on every target point, which are essentially the vertices and the centers of the hexagons. With just the knowledge of its own location and the origin, each node executes a simple self-organized distributed algorithm $O(d)$ time complexity (d is the maximum number of neighbors of a node) and constant message complexity, to fill up all the target points mutually exclusively with minimum possible displacement. In case of failure of nodes, existing free nodes may take necessary action to fill up the empty target points to make the system fault-tolerant. We evaluate the performance of our proposed model by simulation. It shows that with sufficient node density the algorithm attains full coverage using minimum number of nodes and terminates in one round only. Also, in more than 95 % cases the displacement of an individual node is minimum.

References

1. Bartolini, N., Calamoneri, T., Fusco, E., Massini, A., Silvestri, S.: Push and pull: autonomous deployment of mobile sensors for a complete coverage. *Wireless Netw.* **16**(3), 607–625 (2010)
2. Brass, P.: Bounds on coverage and target detection capabilities for models of networks of mobile sensors. *ACM Trans. Sens. Netw. (TOSN)* **3**(2), 9 (2007)
3. Cheng, P., Chuah, C.N., Liu, X.: Energy-aware node placement in wireless sensor networks. In: *Global Telecommunications Conference, GLOBECOM*, vol. 5, pp. 3210–3214. IEEE (2004)
4. Han, Y.H., Kim, Y.H., Kim, W., Jeong, Y.S.: An energy-efficient self-deployment with the centroid-directed virtual force in mobile sensor networks. *Simulation* **88**(10), 1152–1165 (2012)
5. Heo, N., Varshney, P.K.: An intelligent deployment and clustering algorithm for a distributed mobile sensor network. *IEEE Int. Conf. Syst. Man Cybernet.* **5**, 4576–4581 (2003)
6. Ke, W.C., Liu, B.H., Tsai, M.J.: The critical-square-grid coverage problem in wireless sensor networks is np-complete. *Comput. Netw.* **55**(9), 2209–2220 (2011)
7. Kukunuru, N., Thella, B.R., Davuluri, R.L.: Sensor deployment using particle swarm optimization. *Int. J. Eng. Sci. Technol.* **2**(10), 5395–5401 (2010)
8. Liao, W.H., Kao, Y., Li, Y.S.: A sensor deployment approach using glowworm swarm optimization algorithm in wireless sensor networks. *Expert Syst. Appl.* **38**(10), 12180–12188 (2011)
9. Luo, C.J., Tang, B., Zhou, M.T., Cao, Z.: Analysis of the wireless sensor networks efficient coverage. In: *International Conference on Apperceiving Computing and Intelligence Analysis (ICACIA)*, pp. 194–197 (2010)
10. Poe, W.Y., Schmitt, J.B.: Node deployment in large wireless sensor networks: coverage, energy consumption, and worst-case delay. In: *Conference on Asian Internet Engineering*, pp. 77–84. AINTEC, ACM, USA (2009)

11. Saha, D., Das, N., Bhattacharya, B.B.: Fast estimation of coverage area in a pervasive computing environment. In: *Advanced Computing, Networking and Informatics*, vols. 2, 28, pp. 19–27. Springer (2014)
12. Saha, D., Das, N., Pal, S.: A digital-geometric approach for computing area coverage in wireless sensor networks. In: *10th International Conference on Distributed Computing and Internet Technologies (ICDCIT)*, pp. 134–145. Springer (2014)
13. Saha, D., Das, N.: Distributed area coverage by connected set cover partitioning in wireless sensor networks. In: *First International Workshop on Sustainable Monitoring through Cyber-Physical Systems (SuMo-CPS)*, ICDCN. India (2013)
14. Saha, D., Das, N.: A fast fault tolerant partitioning algorithm for wireless sensor networks. In: *Third International Conference on Advances in Computing and Information Technology (ACITY)*. pp. 227–237. CSIT, India (2013)
15. Sheu, J.P., Yu, C.H., Tu, S.C.: A distributed protocol for query execution in sensor networks. In: *IEEE Wireless Communications and Networking Conference*, vol. 3, pp. 1824–1829 (2005)
16. Wang, G., Cao, G., Porta, T.L.: Movement-assisted sensor deployment. *IEEE Trans. Mob. Comput.* **5**(6), 640–652 (2006)
17. Wang, X., Wang, S., Ma, J.J.: An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment. *Sensors* **7**(3), 354–370 (2007)
18. Yu, X., Huang, W., Lan, J., Qian, X.: A van der Waals force-like node deployment algorithm for wireless sensor network. In: *8th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 191–194. IEEE (2012)
19. Yu, X., Liu, N., Huang, W., Qian, X., Zhang, T.: A node deployment algorithm based on van der Waals force in wireless sensor networks. *Distrib. Sens. Netw.* **3**, 1–8 (2013)
20. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. In: *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM)*. vol. 2, pp. 1293–1303 (2003)