Deploying Sensors for Maximum Coverage in Sensor Networks
Ruay-Shiung Chang
Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien, Taiwan
rschang@mail.ndhu.edu.tw
Shuo-Hung Wang
Department of Information Management
National Dong Hwa University, Hualien, Taiwan
m9221501@em92.ndhu.edu.tw

Abstract
Sensing coverage is an important issue in wireless mobile sensor networks. The strategy of how to deploy sensor nodes in an environment, especially in unknown large environment, will affect the utility of the network just like the quality of communication. We present an efficient method for sensor deployment assuming that global information is not available. Our algorithm (Self-Deployment by Density Control, SDDC), uses density control by each node to deploy sensor nodes concurrently. We also make nodes to form clusters to achieve area density balance. The characteristics in SDDC are concurrent multi-sensors moving, distributed operation, localized calculation, and self-deployment. Simulations show the good performances compared to the incremental self-deployment algorithm.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Designs]: Wireless communication

General Terms
Algorithms, Measurement, Performance, Design.

Keywords
Deployment, Coverage, Sensor networks

1. Introduction
A mobile sensor network is composed of a distributed collection of nodes, each of which has sensing, computation, communication and locomotion capabilities. Sensor nodes are often deployed in the inaccessible terrains or used for disaster relief operations. How to achieve the goal of search and rescue as soon as possible is important for this kind of applications in a sensor network. We envision sending a lot of sensor nodes to the spot of a catastrophe and hope these nodes can cover the scene and find out all the survivors as soon as possible. Usually, we have no a priori knowledge of the disaster environment. These sensor nodes need to configure by themselves and have a good coverage for saving all survivors.

The coverage algorithms proposed are either distributed or centralized. In centralized algorithms [1, 2], where the node will be deployed is decided by some nodes. These few nodes have the prior or part of knowledge of the environment and use that to estimate the better position of all nodes. In distributed and localized algorithms, each node makes use of only neighborhood information including the number of nodes and obstacles. These nodes may receive the information by exchanging information with each other to enlarge the coverage area [3]. Or, they define some formulations to decide where to move [4]. In [5], an incremental and greedy algorithm is used to enlarge sensor nodes coverage.

In the situation like urban search and rescue, if the deployment is concurrent, the mobile sensor networks will be more efficient. The basic idea of this research is based on density of nodes. When the perfume is spouted out, the room is filled with the odor. If sensors can deploy themselves like perfume molecules, it will fill the room uniformly and the network will be more efficient. The idea is to let every sensor node calculate its local density and adjust its location from high density to low density. The rest of this paper is organized as follows. Section 2 is the related work. In Section 3, we provide a solution for deployment of mobile sensor networks. We call it “Self-Deployment by Density Control” (SDDC). Section 4 is the simulation experiments of SDDC. Section 5 is the conclusions.

2. Related Work
The famous one is the “Art Gallery Problem” [6] that requires determining how to deploy observers such that every point in the room is seen by at least one observer. Dynamic coverage, on the other hand, is addressed by algorithms which cover the environment and do not settle to a particular configuration or to a particular pattern of traversal.

In [3], Batalin and Sukhatme show some local approaches to multi-robot coverage. The approach, called
Informatic, is based on the idea of exchanging the coordination information such as local identities and relative locations between the interacting robots. When a robot received the message from others, it decides where to go for enlarging the total coverage.

Heo and Varshney proposed a “Distributed Self-Spreading Algorithm” [4]. This distributed algorithm for uniform deployment and energy saving is executed at each node. They define a formula called partial force calculation to decide the movement of nodes.

Virtual Force Algorithm was suggested in [2]. This approach is executed by the cluster head for cluster-based sensor network architecture. The cluster head calculates its local area coverage and sends instructions to its member to move to the position requested. The major differences between this algorithm with ours are in the distributed operation and the knowledge of the environment. The algorithms in [2] need the knowledge of obstacles that implies a certain degree of a priori knowledge of the terrain. All cluster-heads use the information of obstacle and the initial deployment of sensor nodes to calculate the positions for redeployment. These cluster-heads need to have more powerful computation than other nodes for the centralized decision processes. A paper [5] describes an incremental deployment algorithm for a mobile sensor network. This deployment algorithm is both incremental and greedy. Nodes are deployed one at a time, with each node making use of data gathered from previously deployed nodes. The advantage of this approach is simplicity and clarity, with each node moves one at a time. Due to the sequential step, it usually takes more time than concurrent work.

3. Self-deployment by Density Control
In this paper, we focus on coverage protocol and skip the discussion of the routing protocol just as doing in [2-5]. Our proposed method to rearrange sensor nodes in the search and rescue scene uses the density calculated for each node to decide where it moves. The goal of this method is to deploy nodes as soon as possible to obtain maximized coverage while keeping network connected. The characteristics are concurrent multi-sensors moving, distributed operation, localized calculation, and self-deployment.

3.1 Assumptions
There are some assumptions.
(a) Every sensor node can detect if there are obstacles in its sensing range and knows these obstacles’s position and the area measure of the free area.
(b) Every sensor node can receive another node’s message within its communication range and knows the distance and angles θ between it with another. That means every sensor node has the ability of global localization using the way as in [8].
(c) Sensor node’s communication range is larger than the sensing range.
(d) Every sensor node has one unique machine ID.

3.2 Definitions
3.2.1 Area
Considering the sensing range of a sensor node as Fig. 1, we can divide the circular shape to k equal small areas. K is a system parameter. Each area measure is 1/k circle area measure and can be represented as $K_s$. In the example of Fig.1, there are 8(k=8) equal areas in node 3’s sensing area. The reason to divide one node’s sensing area to k parts is that it is convenient to calculate the ratio of occupied area measure mentioned in section 3.2.2. The higher k is, the more accurate the ratio of occupied area measure is.

3.2.2 Density
The density of actual sensor nodes in a communication area can be represented by

$$D = \frac{N_a}{K_m}$$  

where $N_a$ is the actual number of nodes within $K_m$. Similar to $K_s$, $K_m$ is the measure of the division of the node’s communication area. On the other hand, the density of the theoretical minimal sensor nodes in a communication area can be represented by

$$D_t = \frac{N_t}{K_m}$$  

where $N_t$ is the theoretical minimal number of nodes within the node’s sensing range needed to cover $K_m$. $N_t$ is a parameter that is adjusted by the sensor node’s efficiency. If the detected area measure and the distance of sensing range are known, $N_t$ can be calculated by the equation [9]:

$$N_t = \frac{\sqrt{27K_m}}{2\pi r^2}$$  

In this equation, $r$ is the sensing range of a sensor node. Having the density of actual and theoretical sensor nodes, we can obtain the ratio of actual nodes to theoretical nodes in a communication area $Km$.

$$R = \frac{N_a}{K_m} = \frac{N_t}{K_m}$$  

The ratio of occupied area measure is defined to be

$$R_o = \frac{K_s - O_m}{K_s}$$  

where $O_m$ is the area measure of the free area which a sensor node can detect in the division area with some obstacles.

3.2.3 Cluster status of node
The cluster status of each node can be in one of the following status: undecided, cluster-head or member.
(a) Undecided: A node which initializes to attend the network has undecided status.
(b) Cluster-head: A sensor node will be a cluster-head when it satisfies the conditions that it has the smallest machine ID in its neighborhood [10]. However, no two cluster-heads are connected.
(c) Member: A sensor node will be a member node if it has a larger machine ID than one of its neighbors in its communication neighborhood.

The way to form a cluster is as follows. Initially, each node has the “Undecided” cluster status. After exchanging machine ID with each other in a neighborhood, the one with smallest ID will be a cluster-head. Then, these cluster-heads change their status and send a “cluster-head” message to their neighbor nodes. Every node receiving the “cluster-head” message will change its status to “member” and transmits its status to its neighbor.

![An example of sensor node 3’s sensing area division.](image)

3.2.4 Next position
Every node will receive the message from the neighbors about angle \( \theta \) and the distance \( D \) between them. In Fig. 1, Node 4 and Node 5 are within Node 3’s communication range. The nodes surrounding Node 3 are the stress to 3. These nodes can be imagined to be the forces imposing on Node 3. That means the more nodes in one side the stronger force pushing Node 3 to another side. The force \( F \) imposing on one sensor node can be calculated by

\[
F = \frac{R}{D} \quad (6)
\]

where \( R \) is the ratio of actual nodes to theoretical nodes in the small area (1/k communication area) and \( D \) is the distances between one sensor node and another within the small area. If there are several \( n \) nodes in the same small area, the force \( F \) can be calculated by

\[
F = \sum_{i=1}^{n} \frac{R_i}{D_i} \quad (7)
\]

where \( R_i \) is the distances between one sensor node and the sensor node \( i \) within the same small area. The force \( F_o \) of an obstacle is calculated by

\[
F_o = \frac{R_o}{D_o} \quad (8)
\]

where the \( R_o \) is the density of an obstacle and the \( D_o \) is the distances between the sensor node and the obstacle within its range. After calculating the values of \( F \) and \( F_o \) of every small area, we can obtain the value of the total force in each small area. \( F_k \) can be represented as the total force of \( F \) and \( F_o \) in small area \( k \). \( F_k \) means the force on x axis of \( F_k \) and \( F_{yk} \) is the force on y axis of \( F_k \). We can obtain the concurrence force \( F_{con} \) (the total force of all small area) and the angle \( \delta \) to determine the sensor node’s movement.

\[
F_{con} = \sqrt{\left( \sum_{k=1}^{m} F_{yk} \right)^2 + \left( \sum_{k=1}^{m} F_{xk} \right)^2} \quad (9)
\]

The constraint of the new position is that the node must be able to communicate with at least one other node after moving. For the sake of maintaining the sensor networks connection, before moving, the node sends the new position to its neighbors to ask if it will leave their communication range. If one of its neighbors sends “ok” or “suggested” message, then it can move to the new position or with a suggested reduced distance in the same direction.

“Ok” message means that one sensor node can communicate with the node sending “ask” message in its new position. “Suggested” message means that one sensor node can communicate with the node sending “ask” message in a restricted range. When a node receiving the “ask” message, it checks the new position of that node. If the position is in its communication range, it sends “ok” message. If it cannot cover as far as the new position requires, it sends a message with a “suggested” distance. If a node has sent the “ask” message and is still waiting for its neighbor’s reply message, it skips the “ask” message from others. If a node does not receive any reply message after asking, it gives up moving temporarily. After a random time interval, it sends the “ask” message again. The random time interval avoids deadlock of waiting for replies.

3.3 The Phases in Density Control
3.3.1 Initialization
Every sensor is active. When the sensor detects the surrounding, it can find obstacles and other sensors. These sensors detect how many numbers of neighbor nodes they have and send messages to their neighbor nodes and form the cluster.

Every node has two kinds of status. One is cluster status and another is node status. The cluster status which includes “undecided”, “cluster-head” and “member” is used for forming the clusters. The node status which includes “not deployed” and “deployed” is used for indicating whether it should try to move again.

In initialization phase, a sensor node becomes active. Its node state is “not deployed” and cluster state is “undecided”. It will detect the surrounding in its
communication range. A node broadcasts “Hello” messages to its neighbor nodes. The “Hello” messages include its machine ID and cluster state. Sensor nodes use the “Hello” messages to construct cluster and fill its neighbor table with the cluster status.

3.3.2 Goal selection
In this phase, every sensor node calculates its local density and estimates the next position. After having the data of angle and distance, it can calculate the density in every smaller communication area.

3.3.3 Goal resolution
To avoid forming a fragmented sensor network, the node’s future position should be checked before moving. The node broadcasts its possible next position to its neighbors and waits for one “ok” or “suggested” message. If this node doesn’t receive any message in a time interval $\lambda$, its state is still “not deployed”. After the time interval $\lambda$, it will send the “ask” messages to its neighborhood in every time interval $\tau$ until it receive the message. The value $\tau$ is assigned randomly within one time interval.

If the sensor can not go anywhere because of the obstacle or it will leave another node’s communication range or the distance between the present and next position is smaller than a threshold $\omega$, the sensor’s state also becomes “deployed”. After all sensors are deployed, the cluster-head calculates its average density in its cluster and sends this message to its neighbor cluster-heads. A neighbor cluster-head should be at most three-hop away. After receiving the neighbor cluster’s density, the cluster-head will compare it with its local average density. If one group’s density is larger than another by a threshold $\epsilon$, the higher density cluster-head will order a member sensor node that is the closest to the lower density cluster-head to move toward it. To determine which node to move, there are two cases. In case 1, the cluster-head receives the density message from its member node. Then the member node is ordered to move. In case 2, the cluster-head receives the message from another cluster-head’s member node. Then the cluster-head itself should move.

3.3.4 Execution
Sensors adjust their location until they have deployed status and any two cluster head’s density difference is smaller than a threshold $\epsilon$. This algorithm iterates through the selection, resolution and execution phases. It will terminate when all nodes have “deployed” status and any two cluster head’s density difference is smaller than a threshold $\epsilon$. In our protocol, the movements of sensor nodes like the Brown Movement of small molecules are ceaseless. The convergence time to be deployed is related with the threshold $\epsilon$. The smaller $\epsilon$ is, the more evenly the sensors spread but the longer time to move.

4. Simulation Results
Our simulation experiments are conducted using the Stage multi-agent simulator [12, 13]. Stage simulates the behavior of real sensors and actuators. We have conducted a series of simulation experiments aimed at testing the performance of the proposed approach. The results are compared with those of the incremental deployment algorithm (ISD) [5] and the local approach to multi-robot coverage (LAMRC) [3]. The simulated sensor network consists of 10 to 100 sensors, each of which is equipped with a scanning sonar range finder and an omni-directional mobile robot base. The sonar has a 360 degree field-of-view and can determine the range of obstacles within 5m.

The initial moving speed of the robot is 0.6 m/s. For better results of both the elapsed deployment time and the coverage, the system parameter $\epsilon$ (Density threshold) is 0.125 and $\omega$ (Distance threshold) is 1.25m in all our experiments. For each test, we have at least 10 runs. A run means the moving process of sensor nodes from initial to final deployment.

Figs. 2 and 3 show the initial and final network deployments for a sensor network with 50 nodes. The polygons indicate the obstacles and the total measure of the area is 50m × 50m. (The experiments with more than 50 nodes are using 100 m × 100m area.) The area coverage in the initial deployment is 185 $m^2$. The final area coverage is 1565 $m^2$ that is about 8 to 9-fold improvement over the initial scene.

We compared our algorithm with ISD and LAMRC by the elapsed time for deployment and coverage area. In Fig. 4, the coverage areas are closely in ISD and SDDC. The major difference between these three algorithms is showed in Fig. 5. Because the ISD’s deployment tactic is sequential, the more the number of nodes is, the more time it needs for deployment. The time to exchange information among nodes in LAMRC make the whole time of coverage longer than SDDC.

5. Conclusion
The idea of this paper is that sensor nodes can deploy themselves like quickly spreading small molecules. The experiments clearly establish the usefulness of our SDDC algorithm.

6. References


