On Perimeter Coverage in Wireless Sensor Networks

Ka-Shun Hung and King-Shan Lui, *Senior Member, IEEE*

***Abstract*—Many sensor network applications require the tracking and the surveillance of target objects. However, in current research, many studies have assumed that a target object can be sufficiently monitored by a single sensor. This assumption is invalid in some situations, especially, when the target object is so large that a single sensor can only monitor a certain portion of it. In this case, several sensors are required to ensure a 360^o coverage of the target. To minimize the amount of energy required to cover the target, the minimum set of sensors should be identified. Centralized algorithms are not suitable for sensor applications. In this paper, we describe our novel distributed algorithm for finding the minimum cover. Our algorithm requires fewer messages than earlier mechanisms and we provide a formal proof of correctness and time of convergence. We further demonstrate our performance improvement through extensive simulations.**

***Index Terms*—Sensor network, perimeter coverage, minimum cover.**

I. Introduction

WIRELESS sensor networks have caught lots of attentions in recent years. People expect many applications which may be too dangerous or too costly to be done by human to be performed by wireless sensor nodes easily. Examples of such applications include environmental monitoring, industrial control, battlefield surveillance, home automation and security, health monitoring, and asset tracking [1], [2]. In monitoring applications, sensor nodes usually cooperate to achieve a certain monitoring objective. The monitoring objective is usually transformed to a coverage problem, which can be regarded as a measurement on quality of service (QoS) of how well the sensor network functions in the physical world. There are two common monitoring objectives suggested and widely studied [3]. They are area coverage and target coverage.

Area coverage refers to the cover of a certain target area, so that any changes within the target area can be discovered immediately and an appropriate action can then be made on time. On the other hand, target coverage refers to the cover of one or more target objects within the area considered. For instance, in an art gallery, several invaluable arts are monitored instead of the whole gallery.

In this paper, however, we are specifically interested in a scenario in which the perimeter of a large target object is our main concern. One typical application scenario is to monitor the coastline of a large lake so as to ensure that no people can go through its perimeter intentionally or accidentally. Another application scenario is to monitor the wall of the prison so as to ensure no criminal can escape easily by digging hole through the wall. Therefore, perimeter monitoring is also an important problem.

Due to the limited battery power, it is desirable to activate as few sensors as possible to cover the whole perimeter of the target object. By activating the minimum set of sensors, the total amount of energy required to monitor the target object is minimal. Moreover, it is possible to identify multiple sets of sensors so that they can be activated one after the other in each round to further extend the network lifetime [4]. Other than extending the lifetime, the additional sets of sensors found can also act as backups so that another set can be activated immediately in case any node fails suddenly.

The problem of perimeter coverage is very similar to the circle-cover problem in a circular-arc graph in which a number of centralized algorithms have been proposed [5]–[8]. Unfortunately, all these proposed algorithms cannot be directly applied in a wireless sensor network scenario which is distributed in nature. Previously, we [9] proposed a distributed algorithm to solve this problem. The approach requires all the nodes passing through 0^o to initiate the search and thus the overhead is not optimal. In this paper, we further enhance our distributed protocol so that it is possible to find a minimum set of sensors to cover the target object using O(size of the minimum set) number of messages. We also provide a formal proof of correctness and convergence time analysis of our proposed algorithm.

The paper is organized as follows. In Section II, we will briefly discuss the related work about the coverage problems in wireless sensor networks and the circle-cover problems in circular-arc graphs. Then, the perimeter coverage problem will be discussed in Section III. Afterwards, our proposed distributed protocol to solve the perimeter coverage problem in wireless sensor networks will be discussed in details in Section IV. Through extensive simulations, we show that our proposed algorithm outperforms some earlier developed distributed minimum cover algorithms in Section V. Finally, we conclude our paper in Section VI.

II. Related Work

Our problem is analogous to the problem of finding a minimum size open cover on a topological space S in the topology literature, where S wraps around on a certain value range in a real line [10]. We are interested in studying the problem from the algorithmic perspective, and the perimeter coverage problem has been studied as the circle cover problem

Manuscript received June 20, 2008; revised April 6, 2009 and November 20, 2009; accepted April 24, 2010. The associate editor coordinating the review of this paper and approving it for publication was D. Zeghlache.

This work was supported in part by the University of Hong Kong Seed Funding Programme for Basic Research, Project No. 200811159057.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong (e-mail:{kshung, kslui}@eee.hku.hk).

Digital Object Identifier 10.1109/TWC.2010.07.080817

in circular-arc graphs. Therefore, in this section, we will briefly discuss the related work on the coverage problems in wireless sensor networks and the related work on the centralized algorithms proposed for the circle-cover problems in circular-arc graphs.

A. Coverage in Wireless Sensor Networks

As we have discussed before, there are two main kinds of coverage problems suggested and studied in wireless sensor networks. They are area coverage and target coverage.

1) Area Coverage: Area coverage problem refers to the cover of the whole target area by the sensors. There are a number of variations, including single area coverage, multiple area coverage and fractional area coverage, etc. Single area coverage problem generally refers to the problem of finding a minimum set of sensors which can cover the whole area. A centralized algorithm to find a small-sized connected sensor cover is presented in [11], while a localized algorithm for area coverage with no prior knowledge of neighbor existences and neighbor location is presented in [12]. Cao *et al.* [13] considered the movement of the sensor so as to cover the area which has not been covered by random distribution of the sensors.

The studies above assume every point within the target area has to be covered by at least one sensor. However, due to various reasons, a certain area, such as some sensitive military area, may be required to be covered by multiple sensors instead. This leads to the k-coverage problems suggested in the literature. Huang *et al.* [14], [15] transformed the coverage problem to a decision problem so as to determine whether the target area is covered by at least k sensors. On the other hand, fractional coverage problem has also been considered in [16], [17]. In this problem, it is generally not necessary to cover the whole target area. Instead, only a certain fraction of the target area has to be covered by the sensors.

2) Target Coverage: Target coverage problem refers to the cover of a certain target object or a number of target objects within a certain area. Kar and Banerjee [18] studied how to place sensors to ensure all the targets are covered. Their algorithm runs in a polynomial time. Cardei *et al.* [19], [20] studied the target coverage problem with the focus in energy efficiency. They assumed that the placement of the nodes are random and they aimed at selecting a maximum number of disjoint sets of sensors such that every set can cover all the target objects. By doing so, the network lifetime can be increased. They proved that the disjoint set problem is NP-complete, and they proposed a centralized algorithm to solve this problem. Later, Thai *et al.* [21] proposed a $O(logN)$ distributed algorithm to solve the target coverage problem, where N is the number of sensors in the network.

B. Circle-Cover in Circular Arc Graphs

In this paper, we are specifically interested in the angle/perimeter coverage problem. Unlike the area coverage problem in which a certain target object area is of particular interest, we are interested in whether the perimeter of the target object is 360° covered by enough sensors. Unlike the target/point coverage problem in which the target objects are small and can be covered by a single sensor near the object's vicinity, in our perimeter coverage problem, a sensor can only cover a certain portion of the perimeter. In fact, the angle/perimeter coverage problem is the same as finding a circle-cover in the circular-arc graph. Circular-arc graph problems have been studied for quite a long time. Generally speaking, there are two main types of algorithms — sequential and parallel algorithms.

1) Optimal Sequential Algorithms: Lee and Lee [5] proposed an optimal sequential algorithm for finding the minimum circle-cover. They formally proved that the time complexity for finding the minimum circle-cover is $O(NlogN)$ if the arcs are not sorted and $O(N)$ if the arcs are sorted with the use of one processor, where N is the number of arcs.

2) Optimal Parallel Algorithms: On the other hand, Bertossi [6] was known to be the first to propose a parallel algorithm to tackle this problem. The algorithm achieves a time complexity of $O(logN)$ with the use of $O((N^2/(logN)+qN)$ processors, where q is the minimum number of arcs overlap at a certain point in the graph. Ref. [7] and Ref. [8] proposed similar optimal parallel algorithm to tackle this problem. Both approaches achieve a time complexity of $O(logN)$. However, the algorithm in [7] requires $O(N/logN)$ processors, while the one in [8] requires $O(N)$ processors. All the algorithms discussed are centralized, so the processors are supposed to be able to access all the sorted arcs in the graph.

3) Distributed Algorithms: All the algorithms discussed above find the minimum circle-cover of the circular-arc graphs without any specific applications in mind. At the same time, all of them are centralized with one or more processors. On the other hand, Watfa and Commuri [22], [23] proposed a distributed algorithm to find a subset of nodes to cover the border of a rectangle. Unfortunately, they did not provide the proof of correctness of their algorithm.

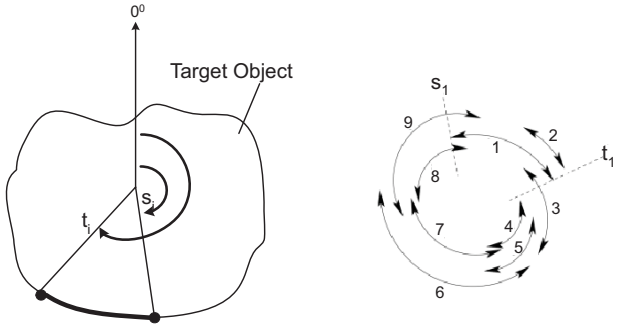
To the best of our knowledge, we are the first to propose an optimal distributed algorithm to find the minimum number of visual sensor nodes which are necessary to cover 360^o of the target object [9]. Unfortunately, the protocols in [9] require a large number of messages, which is expensive for wireless sensor networks. In this paper, we describe our new optimal protocol that requires only a few messages. We compare our algorithm with other existing algorithms both theoretically and through extensive simulation.

III. Problem Statement and Definitions

We are considering a system in which the perimeter of a big target object has to be monitored. The target object is surrounded by randomly distributed sensors. Each sensor can monitor only part of the perimeter, and each sensor can communicate to its neighbors only. We would like to identify a set of sensors that can monitor the whole perimeter. To save energy, the number of sensors needed should be minimized. Before we describe our distributed protocol, we define our problem in this section.

A. Cover Range, Cover, and Size of Cover

For the ease of discussion, we model the perimeter of an object as a circle and use $[0^\circ, 360^\circ)$ to denote the whole



(a) Cover Range (b) A Simple Sensor Network

Fig. 1. Illustrations of covers.

perimeter. The target is surrounded by a randomly distributed set of sensors S. Each sensor can cover only a portion of the perimeter and we call this portion the *cover range*. We denote the cover range of sensor $i \in S$ as $[s_i, t_i]$, which is a portion of the whole perimeter $[0°, 360°)$. The range spans in the clockwise manner as illustrated in Fig. 1(a). The term s_i is the starting angle of the range, while t_i is the ending angle of the range. If i does not cover $0°$, $s_i < t_i$; otherwise, $t_i < s_i$. The cover range of two or more sensors is the union of their ranges. Although we model the perimeter as a circle, it is worth noting that our protocol works for any arbitrary shape of perimeter as long as sensors can determine their cover ranges. How a sensor determines the range is application dependent and it is outside the scope of this paper. Interested readers are referred to [24]–[26].

A *cover* is a proper subset D of S such that for each angle $\gamma \in [0°, 360°)$, there exists a sensor $j \in D$ such that $\gamma \in [s_j, t_j]$, where $[s_j, t_j]$ denotes the cover range of the Sensor $j \in D$. In other words, $\bigcup_{i \in D}[s_i, t_i] = [0°, 360°)$. Fig. 1(b) illustrates a scenario of 9 sensors surrounding a target object. Each arrow represents the cover range of a node. In the figure, the sets $\{1, 3, 5, 7, 8\}$, $\{1, 2, 3, 5, 6, 9\}$, and $\{1, 3, 5, 7, 9\}$ are all covers. On the other hand, the *cover size a.k.a. size of cover* of D, denoted as $|D|$, is the number of sensors in the cover D. In the figures, the sizes of the covers $\{1, 3, 5, 7, 8\}$, $\{1, 2, 3, 5, 6, 9\}$, and $\{1, 3, 5, 7, 9\}$ are 5, 6, and 5, respectively.

B. Minimum Cover

The *minimum cover (MC)* is the cover with the minimum size. Formally, C is a minimum cover if C is a cover such that for every cover D, $|C| \leq |D|$. In Fig. 1(b), both {1, 3, 5, 7, 8} and {1, 3, 5, 7, 9} are a minimum cover with the cover size equals to 5. Minimum cover is not necessarily unique. In the next section, we will describe our protocol that allows a sensor to determine whether it is inside a selected minimum cover.

We define *the smallest cover that consists of Sensor* i, denoted as $MC(i)$, to be the smallest cover among those covers that consists of i. In other words, $|MC(i)| \leq |D|$ for every cover D where $i \in D$. For example, $\{1, 2, 3, 5, 6, 9\}$ is a cover in Fig. 1(b). It is also $MC(2)$ since 2 is a member and it is the smallest in size among all covers that consist of 2. However, it is not $MC(1)$ because there is another cover $\{1, 3, 5, 6, 9\}$ that consists of 1 and is smaller in size. The example also illustrates that not every $MC(i)$, such as $MC(2)$, is an MC. On the other hand, there must exist a sensor i such that $MC(i)$ is also an MC.

C. Backward and Forward Neighbors

Two nodes are *neighbors* if their cover ranges overlap. Formally, Sensor i and Sensor j are neighbors if $s_i < s_j < t_i$ or $s_i < t_j < t_i$ or $s_j < s_i < t_j$ or $s_j < t_i < t_j$ [1]. Each node can communicate directly with neighbors only. It is possible that $[s_j, t_j]$ completely contains $[s_i, t_i]$, that is $s_j < s_i < t_i < t_j$, like Sensors 9 and 8 in Fig. 1(b). In this situation, it is not necessary for i to participate in the selection process. It is because i can be replaced by j even if it appears in an MC. Since i and j can identify this situation after they discover each other when the protocol starts, for the ease of discussion, in the following, we assume that each sensor that participates in our algorithm has a cover range that is not completely inside the cover range of another participant. When two sensors have overlapping cover ranges, one of them is a *backward neighbor* and the other is a *forward neighbor*. Sensor i is a backward neighbor of Sensor j and Sensor j is a forward neighbor of Sensor i if $s_i < s_j < t_i$. Refer to Fig. 1(b), Sensors 2, 7, and 8 are pruned as their cover ranges are completely inside Sensors 1, 6, and 9, respectively. After Sensors 2, 7, and 8 are pruned, Sensor 3 is the forward neighbor of Sensor 1, while Sensor 9 is the backward neighbor of Sensor 1. It is possible that a portion of the perimeter of the target object is not covered by any sensor, i.e., a gap exists. We do not consider this situation in this paper but refer interested readers to [9], [27] for details.

D. Default Member

A sensor is a *default member* if it covers a portion of the perimeter that no other sensor is covering. Formally, Sensor i is a default member if there exists a certain angle $\gamma \in [s_i, t_i]$ such that $\gamma \notin [s_j, t_j]$ for any other sensor j. In this case, Sensor i must be in any cover and $MC(i)$ must be a minimum cover. Sensor i can identify whether it is a default member by checking whether there is any backward neighbor overlaps the sensing range with a forward neighbor. For example, in Fig. 1(b), Sensor 1 is a default member since there is no backward neighbor with a cover range overlapping with a forward neighbor.

IV. Distributed Minimum Cover (DMC)

In this section, we will describe our distributed protocol for identifying the minimum cover in details. The formal proof of the mechanism, the pseudocodes, and a complete example are in the Appendix.

A. Finding MC(i)

As mentioned before, if Sensor i is a default member, then $MC(i)$ is a minimum cover. Even if there is no default member, there must exist a sensor i such that $MC(i)$ is a minimum cover. If we could identify this sensor i, finding

[1]This applies when both i and j do not cover $0°$. The definition can be extended easily to ranges that cover $0°$ but we leave it out for the ease of discussion.

$MC(i)$ would solve our problem. Therefore, we first describe how we can find $MC(i)$ given i.

Sensor i must be in $MC(i)$. Therefore, to find $MC(i)$, we need to find the smallest set of sensors that cover the remaining portion of the perimeter that Sensor i cannot cover, which is $[t_i, s_i]$. For example, to find $MC(1)$ in Fig. 1(b), we need to identify as few sensors as possible to cover t_1 to s_1 in the clockwise manner. We adopt the greedy strategy to find these sensors. Without loss of generality, we assume Sensor i selects a forward neighbor, and sensors are selected in the clockwise direction. That is, Sensor i selects j such that $t_j \geq t_k$ for all forward neighbor k. We call the forward neighbor selected in this manner the *greedy forward neighbor*, and denote the greedy forward neighbor of node i as $GFN(i)$. Sensor i informs its greedy forward neighbor j that it is selected, and Sensor j selects its own greedy forward neighbor. The process ends when a selected node realizes that Sensor i is a forward neighbor. To facilitate this, the identity of Sensor i has to be carried around in the selection process. Refer to the example in Fig. 1(b), suppose that we want to find $MC(1)$. $GFN(1)$ is 3 and so Sensor 1 informs Sensor 3 that it is selected. Sensor 3 selects Sensor 5 as it is the greedy forward neighbor of 3. Sensor 5 selects Sensor 6 and Sensor 6 selects Sensor 9. As Sensor 1 is a forward neighbor of Sensor 9, Sensor 9 informs Sensor 1 and the searching process is concluded.

It is worth noting that each selected node only knows which neighbors, one backward and one forward, are also selected, but no node, even i, has the complete knowledge of $MC(i)$. Besides, only the selected nodes would send a message, and so the message complexity is $O(|MC(i)|)$, which is very efficient.

B. Greedy Forward Neighbor (GFN)

We now have a mechanism that finds the minimum cover containing a certain sensor. If we can identify a sensor i that is in an MC, we solve the problem. Before we describe how to find this sensor, in this section, we describe some properties related to GFN. The proofs can be found in the Appendix.

Property 1: Let i and j be two sensors.

- Property 1.1: If j is a forward neighbor of i, $GFN(i)$ is either j or a forward neighbor of j.
- Property 1.2: If j is a forward neighbor of i and $GFN(i) \neq j$ and $GFN(j) \neq GFN(i)$, $GFN(j)$ is a forward neighbor of $GFN(i)$.
- Property 1.3: If nodes i and j share the same greedy forward neighbor, i.e, $GFN(i) = GFN(j)$, and $s_i \leq s_j$, then $|MC(i)| \leq |MC(j)|$.

These properties allow us to develop our efficient distributed algorithm which will be described in the next section.

C. Finding MC

1) Main Algorithm: Let S_0 be the set of sensors that cover 0°. At least one of the sensors in S_0 must be in an MC. Intuitively, if we find out $MC(q)$ for all $q \in S_0$, the minimum size $MC(q)$ will be the minimum cover. However, if different $MC(q)$'s are found independently, it may be very expensive as there may be many nodes in S_0. Therefore, we "combine" the searches of different $MC(q)$'s and then "prune" some unnecessary searches to reduce the message overhead. We now describe how to prune and combine the searches, followed by how the search terminates.

2) Pruning and Combining mechanism: Let q_m be the sensor in S_0 with the largest ending angle. That is, $t_{q_m} > t_i$ for all $i \in S_0$ and q_m is a forward neighbor of all the nodes in S_0. By Property 1.1, for all $q \in S_0, q \neq q_m$, $GFN(q)$ is either q_m or a forward neighbor of q_m. Let $T \subseteq S_0$ such that $T = \{q | GFN(q) = q_m\}$. Suppose that $s_i \leq s_j$ where $i, j \in T$. Then, by Property 1.3, we know that $|MC(j)| \geq |MC(i)|$ where $i \neq j$. Therefore, we do not have to bother finding $MC(j)$ and we can *prune* the search of $MC(j)$. On the other hand, it is worth noting that every forward neighbor of every $q \in S_0$ is also a neighbor of q_m because both q_m and a forward neighbor of q cover t_q. In other words, based on the cover ranges of its neighbors, q_m can identify $GFN(q)$ for all $q \in S_0$. Suppose now q_m realizes that both i and j select the same GFN f and $s_i < s_j$. In this case, q_m can prune the search of $MC(j)$ because $\{i, f\}$ also covers the range that $\{j, f\}$ can cover. Generally speaking, when two or more nodes select the same greedy forward neighbor, we can prune some of the searches.

To further reduce the message overhead, we "combine" the unpruned searches by one message. In other words, in our algorithm, q_m initiates the algorithm and sends the information of the unpruned searches to $GFN(q_m)$. That is, q_m sends $GFN(q_m)$ a list of $< q, s_q, GFN(q) >$ where $q \in S_0$, s_q is the start angle of q and q is not pruned. Note that each $GFN(q)$ in the list is different and is not $GFN(q_m)$. For each $q \neq q_m$ in the list, according to Property 1.2, $GFN(q_m)$ must be a forward neighbor of $GFN(q)$ and so $GFN(q_m)$ can identify $GFN(GFN(q))$ for all q.

Specifically, the list being passed around the nodes consists of entries in the form of $< q, s_q, GFN^k(q) >$ where $q \in S_0$ and q is not pruned. We define $GFN^2(i)$ to be $GFN(GFN(i))$, which is the greedy forward neighbor of the greedy forward neighbor of i. Similarly, $GFN^k(i)$ means $GFN(GFN(...(GFN(i))))$ where GFN is found for k times. For convenience, we label $GFN^0(i)$ to be i. Therefore, k is related to how many hops that message has gone through. For example, q_m sends out $< q, s_q, GFN(q) >$ and $GFN(q_m)$ sends out $< q, s_q, GFN^2(q) >$.

3) Terminating Condition: In our algorithm, q_m starts the search by sending out a message to $GFN(q_m)$. Only nodes that are $GFN^k(q_m)$ for $0 \leq k < |MC(q_m)|$ would receive a message and send out a message. Besides, each entry $< q, s_q, f >$ in the message received by $GFN^k(q_m)$ satisfies $f = GFN^k(q)$. For a forward neighbor of $GFN^k(q_m)$ which overhears the search message from $GFN^k(q_m)$, it determines that it is not in any MC if it is neither q nor f in any entry $< q, s_q, f >$ in the search message.

The search can stop when the message goes around the perimeter and reaches a node $GFN^k(q_m)$ which receives the entry $< q, s_q, f >$ and realizes that q is a forward neighbor of f. In this case, $MC(q)$ is an MC. The node $GFN^k(q_m)$ informs q that it is in an MC. Then, q informs its GFN and the GFN further inform its own GFN and so on.

The message complexity of our protocol is $O(|MC|)$. Since

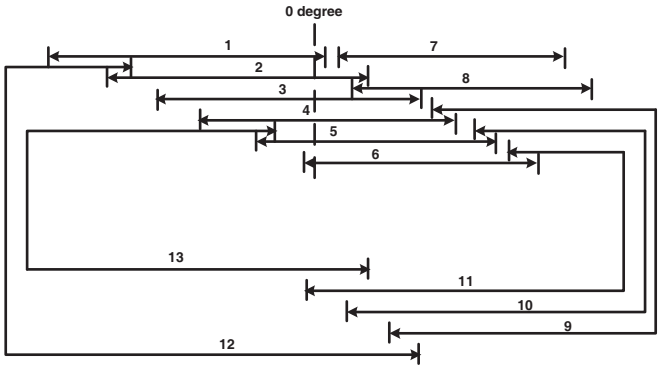


Fig. 2. An illustrative example.

the search messages are combined, only certain nodes need to process the search message. Algorithm 1 in the Appendix illustrates what $GFN^k(q_m)$ should do when it receives the search message. Lines $5-11$ describe how $GFN^k(q_m)$ determines whether it can terminate the search. Lines $12-28$ describe how a search can be pruned. Finally, a combined message containing the information of the unpruned searches is sent out as described in Lines $29-30$. The complete proof of the mechanism and an example can be found in the Appendix.

D. Finding Multiple MCs

We can observe two features of our protocol. Firstly, each message carries a list of $< q, s_q, f >$ where each tuple represents a search of an MC. Moreover, these MCs are unique and each sensor is inside at most one of them. It is because if they share a common sensor, the pruning mechanism would be applied on them and, only the search of one MC will remain. Previously, we simply choose one of the MCs as the final MC selected. However, multiple MCs are possible to be found with O(size of the minimum cover) number of messages by using this mechanism.

By finding multiple MCs, we can turn on different sets of sensors at different time to increase the network lifetime. In case a sensor of an MC fails, we can switch to another MC immediately without having interruption in monitoring. Let MC1 and MC2 be two MCs that do not share any common sensor. Suppose that MC1 is being used and Sensor i in MC1 fails. In MC2, there must be sensors that cover $[s_i, t_i]$. As they are neighbors of i, they can detect the node failure. These sensors should turn on immediately to cover the affected portion on the perimeter. After all the nodes in MC2 have turned on, the nodes in MC1 can safely go to the sleep state to complete the switching process.

V. Performance Analysis

A. Algorithms for Comparison

We compare our proposed algorithm with two other algorithms. The first one is the modification of the sequential algorithm proposed in [5]. We denote this algorithm as the *GMLL Algorithm*. The other algorithm is the distributed algorithm proposed in [9]. We denote this algorithm as the *Exhaustive Algorithm*.

1) GMLL Algorithm: In this algorithm, a randomly chosen node $i \in S$ looks for $GFN(i)$. Node $GFN(i)$ continues to look for $GFN^2(i)$ and so on. This process continues until a node has been visited twice, and this node is a node in MC. Refer to the example in Fig. 2, suppose Sensor 2 initiates the algorithm. It selects its GFN, i.e., Sensor 8. Sensor 8 will further select its GFN, i.e., Sensor 11. This process continues until the same sensor is reached. In other words, the algorithm involves Sensors 2, 8, 11, 13, 5, 10, and finally Sensor 13 again. At this moment, Sensor 13 knows that it is a node in MC. After a node in MC is determined, similar to our proposed algorithm, the node can then inform its GFN that it is in MC and so on. In this algorithm, visiting the same node twice indicating that a termination decision can be made. Therefore, this approach requires $O(|MC||S_0|)$ number of messages in the worst case, but it is not always that the search goes through all the nodes in S_0. For the details of the algorithm, the readers are referred to [5].

2) Exhaustive Algorithm: This algorithm is very similar to the parallel algorithm proposed in [6]. In this algorithm, every node $q \in S_0$ initiates the search for $MC(q)$ individually. All the searches can be carried out in parallel. After every node $q \in S_0$ determines $MC(q)$, it informs the other nodes in S_0. The one with the minimum size is the minimum cover. Then, the node in S_0 and also in the minimum cover informs its GFN and so on. This algorithm terminates with $O(|MC||S_0|)$ number of messages because this approach requires each node $q \in S_0$ to initiate the search for $MC(q)$ individually.

Our distributed algorithm terminates with $O(|MC|)$ number of messages. Therefore, we can conclude that our distributed algorithm performs better than the *Exhaustive Algorithm* and the *GMLL Algorithm* in terms of message complexity.

B. Simulation Results

We further study the performances of the algorithms through simulations. The simulation environment is similar to the one adopted in [9]. We consider a square area of $200\ units \times 200\ units$, which is divided into 200×200 grids, where each grid is of size $1\ unit^2$. The probability that there is a sensor in each grid is 0.5. We assume that each sensor can monitor an object that is within a certain distance from itself. We call this distance the *sensing range* of the sensor. In other words, the sensing area of a sensor forms a circle which is centered at the sensor with a radius equals to the sensing range of that sensor. The portion of the perimeter that falls in the sensing area is the cover range of the sensor. The target perimeter is a circle centered at (100, 100) with a radius of 62.5 units. In the experiment, we adjust the sensing range of the sensors from 18 units to 36 units with a step of 0.05 unit, and this contributes to 360 points on each line shown in Figs. 3 to 7. We generated 60 different topologies for each sensing range step, and so each point on the figure is an average results taken from these 60 different topologies.

Two performance metrics are studied in our simulations. The first one is the total number of messages generated by the protocol. The second one is the average time it takes for a node to determine whether it is in the selected minimum cover after the algorithm starts. This measures how early a

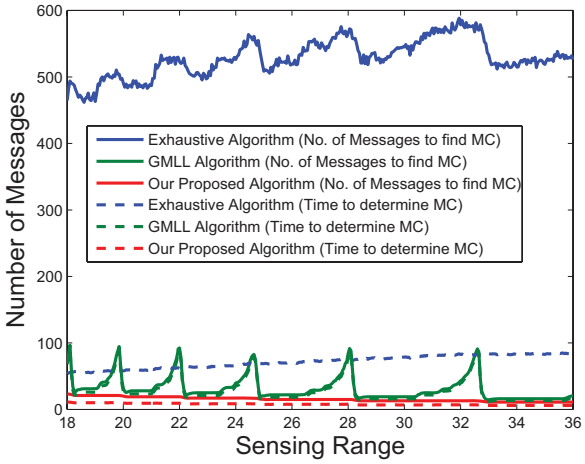


Fig. 3. Number of messages vs. sensing range.

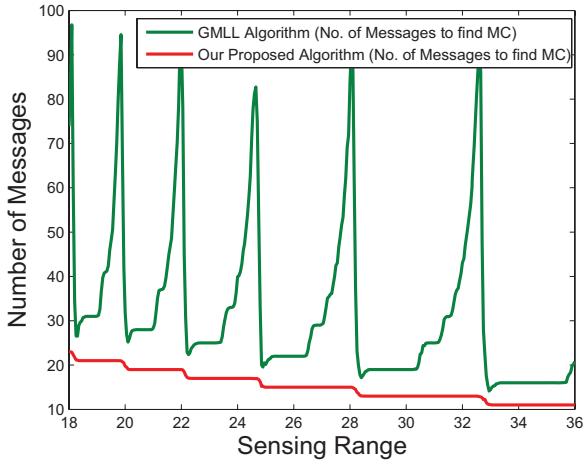


Fig. 4. Number of messages vs. sensing range in a zoom scale.

node which is not selected to be in the MC can go to the sleep mode to save energy. A node can determine that it is not in the MC when it overhears the message and finds out that it is not a GFN of any backward neighbor in our algorithm (Section IV-C). Other nodes that cannot determine its status by overhearing will know whether it is in the MC after the search is complete. As there is only one message at any time before our algorithm terminates, the time required to make a decision is directly related to the number of messages that have been generated before the decision can be made. Therefore, we measure the time by counting the number of messages needed. Similar approach can be used to measure the time of the *GMLL Algorithm*, except that a node can only determine whether it is in MC after the search is complete. On the other hand, since all the searches in the *Exhaustive Algorithm* can be carried out in parallel, the time can be approximated by measuring the number of messages needed to find $MC(q)$, where $q \in S_0$, together with the number of messages needed to exchange among nodes in S_0.

Figs. 5 to 7 are used to explain the performance of the algorithms shown in Figs. 3 and 4. Fig. 5 illustrates the change in the size of MC with increasing sensing range. On the other hand, Fig. 6 shows the change in the size of S_0. In Fig. 7, we consider the number of rounds, excluding the first round and the notification round, that *GMLL Algorithm* needs to go through in different sensing ranges. Fig. 4 presents the number of messages required where only the performances of our proposed algorithm and the *GMLL Algorithm* are shown. It can be observed that the number of messages of our proposed algorithm exhibits similar staircase behavior as $|MC|$ in Fig. 5. The simulation results support that the message complexity of our mechanism is directly related to the size of MC.

On the other hand, the number of messages required in the *Exhaustive Algorithm* shows a seesaw increasing trend with the sensing range in Fig. 3. The message complexity of this algorithm is $O(|S_0||MC|)$ because it requires all the nodes passing through 0° to initiate a search. When the sensing range becomes larger, more nodes will initiate the search due to the growth in the size of S_0 as shown in Fig. 6. But at the same time, the size of an MC decreases with an increase in the sensing range as shown in Fig. 5. Specifically, a drop in the number of messages can be observed in Fig. 3 at around the sensing range region where a drop in $|MC|$ is observed. Therefore, this results in a seesaw increasing trend in the number of messages as shown in Fig. 3. This demonstrates that the simulation results agree with our theoretical analysis.

For the *GMLL Algorithm*, we notice that the number of messages required to find MC may go up and down in Fig. 3. Recall that the *GMLL Algorithm* can terminate when a node receives the search message twice. Therefore, the message complexity depends on how many rounds the search message goes around the perimeter before the search terminates. When there are many MCs in a network, more rounds will be needed. Fig. 7 shows that the number of rounds needed for different sensing ranges. The number of rounds drops sharply at where $|MC|$ drops in Fig. 5. It is because at that sensing range, there are probably only one or two MCs in the network, and a node will be visited twice very soon. On the other hand, the number of MCs increases exponentially when the cover range increases before $|MC|$ drops again. Therefore, between two consecutive drops, number of rounds increases exponentially. This also explains why the message complexity of the *GMLL Algorithm* in Fig. 3 exhibits a sawtooth curve.

Fig. 3 also illustrates the average time required for a node to determine whether it is included in MC. In the *Exhaustive Algorithm*, all the nodes cover 0° can start at the same time to find MC as stated earlier, the average time needed is much less than the number of messages needed in finding MC. On the other hand, in the *GMLL Algorithm*, a node can only determine whether it is in MC after a node in MC is determined. As a result, the average time is similar to the number of messages needed in finding MC. In contrast, in our proposed algorithm, some of the nodes can conclude that they are not in MC before the search process ends, and so the average time is generally half of the total message overhead.

VI. Conclusion and Future Work

In this paper, we studied the angle/perimeter coverage problem in which multiple sensors are expected to collaborate to monitor the perimeter of a big target object. We proposed a distributed algorithm to solve the problem with O(size of the

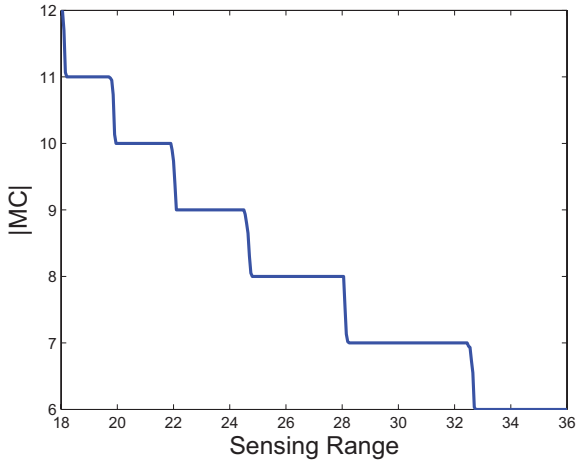


Fig. 5. Size of MC vs. sensing range.

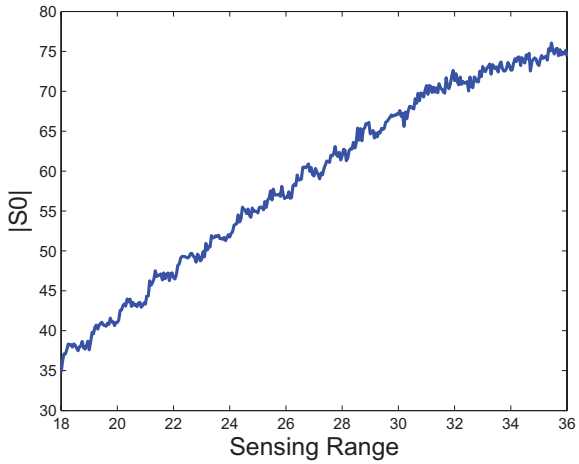


Fig. 6. Size of S_0 vs. sensing range.

minimum cover) number of messages and formally proved the correctness and message complexity of our protocol. Through extensive simulations, it is found that our proposed algorithm outperforms other existing methods in terms of message overhead and the average time required for a node to determine whether it is in the selected MC.

VII. Appendix

A. Proof of Correctness of our Proposed Algorithms

Lemma 1 *$\{GFN^j(i) \mid 0 \leq j \leq k$ and $GFN^k(i)$ is a backward neighbor of $i\}$ is an $MC(i)$.*

Proof:
Suppose that given a certain $MC(i)$, $C = MC(i)\backslash\{i\}$ and $|C| = k$. The cover ranges of the sensors in C arranged in clockwise order are $[\sigma_1, \tau_1]$, ... , $[\sigma_k, \tau_k]$, where $\tau_j < \tau_{j+1}$ for $1 \leq j < k$. We further assume that the set of greedy forward neighbors selected is C' with $|C'| = k'$. The cover ranges of the sensors in C' are $[s_1, t_1]$, ... ,$[s_{k'}, t_{k'}]$. To argue that $C' \cup \{i\}$ is also an $MC(i)$, we proof $k = k'$ by showing $\tau_j \leq t_j$ where $1 \leq j \leq k'$ using induction.

To simplify the notation, we label a sensor using the start angle of its cover range. That is, s_j denotes the sensor that covers $[s_j, t_j]$. Both σ_1 and s_1 are forward neighbors of i. As s_1 is the greedy forward neighbor, $\tau_1 \leq t_1$. Assume it is true for j that $\tau_j \leq t_j$. We now prove it is also true for $j+1$.

If $\tau_{j+1} \leq t_j$, as $t_j < t_{j+1}$, we have $\tau_{j+1} \leq t_{j+1}$. For $t_j < \tau_{j+1}$, by the assumption that $\tau_j \leq t_j$ and because $[\sigma_j, \tau_j]$ overlaps with $[\sigma_{j+1}, \tau_{j+1}]$ ($\sigma_{j+1} \leq \tau_j$), it leads to $\sigma_{j+1} \leq t_j \leq \tau_{j+1}$. This implies σ_{j+1} is a forward neighbor of s_j. As s_{j+1} is the greedy forward neighbor of s_j, $\tau_{j+1} \leq t_{j+1}$ and it completes the proof. ■

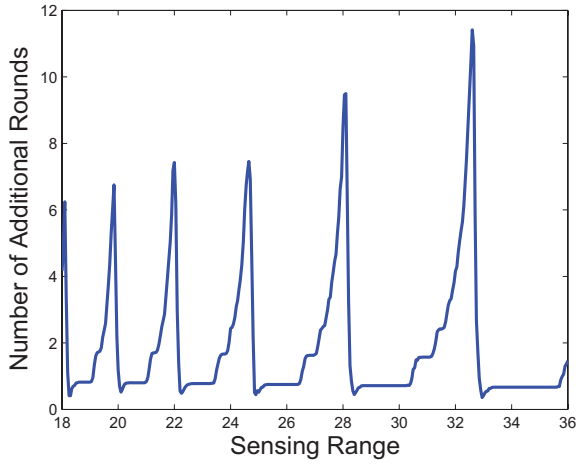


Fig. 7. Number of additional rounds vs. sensing range.

Property 1 *Let i and j be two sensors.*

Property 1.1: If j is a forward neighbor of i, $GFN(i)$ is either j or a forward neighbor of j.
Proof:
As j is a forward neighbor of i, it can be the greedy forward neighbor of i. If $x = GFN(i) \neq j$, $t_x \geq t_j$ and so x is a forward neighbor of j. ■

Property 1.2: If j is a forward neighbor of i and $GFN(i) \neq j$ and $GFN(j) \neq GFN(i)$, $GFN(j)$ is a forward neighbor of $GFN(i)$.
Proof:
Let $x = GFN(i)$ and so $s_x \leq t_i$ while $y = GFN(j)$ and $s_y \leq t_j$. As $GFN(i) \neq j$ and j is a forward neighbor of i, $t_j < t_x$. As y is $GFN(j)$, $t_x < t_y$. Therefore, y is a forward neighbor of x. ■

Property 1.3: If nodes i and j share the same greedy forward neighbor and $s_i \leq s_j$, then $|MC(i)| \leq |MC(j)|$.
Proof:
Let $GFN(i) = GFN(j) = x$, $|MC(i)| = k$ and $|MC(j)| = k'$. According to Lemma 1, $\{GFN^m(i) | 0 \leq m < k\}$ is an $MC(i)$. Note that $GFN^{k-1}(i) = GFN^{k-2}(x)$. That is, $GFN^{k-2}(x)$ is a backward neighbor of i and $GFN^{k'-2}(x)$ is a backward neighbor of j. Since $s_i \leq s_j$, $k-2 \leq k'-2$ and so $k \leq k'$ as stated in the property. ■

First of all, we use Lemma 2 and Lemma 3 to illustrate that MC exists in our proposed distributed algorithm. Afterwards, Lemma 4 illustrates that our distributed algorithm terminates

with an MC found. Finally, Theorem 1 and Theorem 2 prove the correctness and the complexity of our proposed algorithm discussed. Recall that q_m is the sensor in S_0 with the largest ending angle. That is, $t_{q_m} > t_i$ for all $i \in S_0$ and q_m is a forward neighbor of all the nodes in S_0.

Lemma 2 *In our algorithm, the search of $MC(j)$, $j \in S_0$ is pruned by $GFN^k(q_m)$ only if there exists another node $i \in S_0$ such that $MC(i)$ is not pruned and $|MC(i)| \leq |MC(j)|$.*

Proof:
As discussed in Section IV-C, only $GFN^k(q_m)$ will be responsible for sending out search message after receiving message M from $GFN^{k-1}(q_m)$ which contains $<< q_1, s_{q_1}, GFN^k(q_1) >,...,< q_L, s_{q_L}, GFN^k(q_L) >>$. According to Properties 1.1 and 1.2, $GFN^k(q_m)$ can find the $GFN^{k+1}(q_i)$, for each q_i in S_0.

From Algorithm 1 in Section VII-B, $GFN^k(q_m)$ will prune the search for $MC(j)$, where $j \in S_0$ under two situations.
Case 1: $GFN^{k+1}(i) = GFN^{k+1}(j)$ and $s_i < s_j$, for some i and j.

Let $GFN^{k+1}(i) = GFN^{k+1}(j) = x$, $|MC(i)| = l$ and $|MC(j)| = l'$. According to Lemma 1, $\{GFN^m(i)|0 \leq m < l\}$ is an $MC(i)$. Note that $GFN^{l-1}(i) = GFN^{l-k-2}(x)$. That is, $GFN^{l-k-2}(x)$ is a backward neighbor of i and $GFN^{l'-k-2}(x)$ is a backward neighbor of j. Since $s_i \leq s_j$, $l-k-2 \leq l'-k-2$ and so $l \leq l'$.

Case 2: $GFN^k(q_m) = GFN^{k+1}(j)$ and $s_j < s_{q_m}$, for some j.

Let $GFN^k(q_m) = GFN^{k+1}(j) = x$, $|MC(q_m)| = l$ and $|MC(j)| = l'$. According to Lemma 1, $\{GFN^m(q_m)|0 \leq m < l\}$ is an $MC(q_m)$. Note that $GFN^{l-1}(q_m) = GFN^{l-k-1}(x)$. That is, $GFN^{l-k-1}(x)$ is a backward neighbor of q_m, and $GFN^{l'-k-2}(x)$ is a backward neighbor of j. Since $s_j < s_{q_m}$, $l'-k-2 \leq l-k-1$. However, $GFN^{l'-k-2}(x)$ may also be a backward neighbor of q_m (i.e., $l'-k-2 = l-k-1$). Otherwise, $GFN^{l'-k-2}(x)$ can select $GFN^{l'-k-1}(x)$ and which must be a backward neighbor of q_m (The worst case is when $GFN^{l'-k-1}(x)$ is j). In that case, $l'-k-1 = l-k-1$. As a result, we know that $l-k-2 \leq l'-k-2$ and so $l \leq l'$. ■

Lemma 3 *In each message sent by $GFN^k(q_m)$, $0 \leq k < |MC(q_m)|$, there must exist an entry $< q, s_q, f >$ such that $MC(q)$ is an MC.*

Proof:
When q_m constructs the message in the beginning, at least one q where $MC(q)$ is an MC is in the message. By Lemma 2, $GFN^k(q_m)$ will prune the search of $MC(j)$. Then, $j \in S_0$ only if $\exists i \in S_0$, where $|MC(i)| \leq |MC(j)|$ and the search of $MC(i)$ is unpruned. This means that if $GFN^k(q_m)$ prunes the search of $MC(j)$ which is an MC, $MC(i)$ which is unpruned is also an MC. As a result, when $GFN^k(q_m)$ sends $< q, s_q, f >$ in message M for each unpruned $MC(q)$, $\exists q \in S_0$ and an entry $< q, s_q, f >$ in message M, such that $MC(q)$ is an MC. ■

Lemma 4 *If $GFN^k(q_m)$ receives an entry $< q, s_q, f >$ in the search message M and q is a forward neighbor of f, then $MC(q)$ is an MC.*

Proof:
If $GFN^k(q_m)$ receives an entry $< q, s_q, f >$ in which q is a forward neighbor of f, the search for $MC(q)$ is complete according to Lemma 1, and $|MC(q)| = k+1$. In this case, $GFN^k(q_m)$ can prune the search of $MC(p)$ of other entry $< p, s_p, f >$ in message M in which p is not a forward neighbor of f. It is because the searches of these $MC(p)$ still need one more node to complete. i.e., $|MC(p)| > k+1 = |MC(q)|$. By Lemma 3, $\exists q \in S_0$ and an entry $< q, s_q, f >$ in message M sent by $GFN^{k-1}(q_m)$, such that $MC(q)$ is an MC. As a result, we can conclude that if $GFN^k(q_m)$ receives an entry $< q, s_q, f >$ in M, and q is a forward neighbor of f. $MC(q)$ is an MC. ■

Theorem 1 *Our distributed algorithm is correct.*

Proof:
By Lemma 2 and Lemma 3, we prove that our algorithm proceeds with the search of $MC(i)$, where $i \in S_0$, in which at least one unpruned search of $MC(i)$ is an MC. Finally, our algorithm terminates when $GFN^k(q_m)$ receives an entry $< q, s_q, f >$ in which q is a forward neighbor of f. By Lemma 4, we prove that $MC(q)$ is an MC. As a result, our algorithm always terminates with an MC found and this proves that our distributed algorithm is correct. ■

Theorem 2 *Our distributed algorithm terminates with $O(|MC|)$ number of messages.*

Proof:
Our distributed algorithm starts with q_m and terminates when any q in $< q, s_q, f >$ is a forward neighbor of f. Only $GFN^k(q_m)$ will be responsible for sending out search message. In the worst case, our algorithm terminates with q_m being the forward neighbor of f in $< q_m, s_{q_m}, f >$. According to Lemma 1, $MC(q_m)$ is found in case q_m is a forward neighbor of f, so our distributed algorithm terminates with $O(|MC|)$ number of messages for the search of MC. After the search of MC, nodes in MC are informed. This also requires another $O(|MC|)$ number of messages. As a result, our distributed algorithm terminates with $O(|MC|)$ number of messages in the worst case. ■

B. Example and Pseudocodes

We use Fig. 4 as an example the illustrate the whole operation of our proposed distributed algorithm. Recall that q_m is the sensor in S_0 with the largest ending angle. In the figure, $q_m = 6$ and $MC(1)$ is pruned because $GFN(1) = 6$. As $GFN(2) = GFN(3) = 8$, $MC(3)$ is pruned. Here, $GFN(4) = 9$ and $GFN(5) = 10$ and hence both $MC(4)$ and $MC(5)$ are not pruned. After finding $GFN(6) = 11$, Sensor 6 sends the information of the unpruned searches to Sensor 11. To facilitate Sensor 11 to further identify the MC of the unpruned nodes, we tell Sensor 11 the greedy forward neighbors of 2, 4, and 5. Now, Sensor 11 receives

$<< 2, s_2, 8 >, < 4, s_4, 9 >, < 5, s_5, 10 >, < 6, s_6, 11 >>$. Since $GFN(8) = 11$, $MC(2)$ is pruned. On the other hand, $GFN(10) = GFN(11) = 13$ and so we should prune either $MC(5)$ or $MC(6)$. Since Sensor 5 starts earlier than Sensor 6 (i.e., $s_5 < s_6$), we should eliminate $MC(6)$. However, $MC(4)$ is not pruned, and Sensor 11 sends $<< 4, s_4, 12 >, < 5, s_5, 13 >>$ to Sensor 13. Finally, when Sensor 13 receives $<< 4, s_4, 12 >, < 5, s_5, 13 >>$, it finds that Sensor 5 is a forward neighbor of Sensor 13, it knows that $MC(5)$ is an MC. Sensor 13 can inform Sensor 5 that it is in an MC. Then, Sensor 5 can inform its GFN that it is in the MC and the GFN can further inform its own GFN and so on.

Algorithm 1 Node q receives message $<< q_1, s_1, f_1 >, ... < s_L, s_L, f_L >>$

Preconditions:
$q = GFN^k(q_m)$ for some k.
$f_i = GFN^k(q_i)$
$s_i = s_{q_i}$
/* Check whether an MC is identified. */
for i = 1 to L **do**
  if q_i is a forward neighbor of f_i **then**
    /* $MC(q_i)$ is an MC. */
    Inform q_i to start the real MC search and terminate.
  end if
end for
/* Determining whether search of $MC(q_i)$ can be pruned. Initially, assume all the searches cannot be pruned. */
for i = 1 to L **do**
  /* Prune $MC(q_i)$ if $GFN(f_i) = q$. */
  if $q = GFN(f_i)$ **then**
    Prune $MC(q_i)$.
    Continue
  end if
  for j = 1 to L **do**
    if $i \neq j$ and $MC(q_i)$ and $MC(q_j)$ have not been pruned **then**
      /* Check if they share the same greedy forward neighbor. */
      if $FN(f_i) = GFN(f_j)$ **then**
        prune $MC(q_j)$ if $s_i < s_j$;
        prune $MC(q_i)$, otherwise.
      end if
    end if
  end for
end for
/* Send the unpruned search of $MC(q_i)$ to $GFN(q)$. */
Send $<< q_i, s_i, GFN(f_i) >>$ to $GFN(q)$ for every $MC(q_i)$ that has not been pruned.

REFERENCES

[1] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the physical world with pervasive networks," *IEEE Pervasive Computing*, 2002.

[2] D. Chen and P. K. Varshney, "QoS Support in wireless sensor networks: a survey," in *Proc. Int. Conf. Wireless Networks*, 2004.

[3] M. Cardei and J. Wu, "Coverage in wireless sensor networks," *Handbook of Sensor Networks*, M. Ilyas and I. Magboub, editors, 2004.

[4] K.-Y. Chow, K.-S. Lui, and E. Y. Lam, "Wireless sensor networks scheduling for full angle coverage," *Multidimensional Systems and Signal Processing*, vol. 20, no. 2, pp. 101–119, June 2009.

[5] C. C. Lee and D. T. Lee, "On a circle-cover minimization problem," *Inf. Process. Lett.*, vol. 18, pp. 109–115, Feb. 1984.

[6] A. A. Bertossi, "Parallel circle-cover algorithms," *Info. Process. Lett.*, vol. 27, pp. 133–139, 1988.

[7] M. Atallah and D. Z. Chen, "An optimal algorithm for the minimum circle-cover problem," *Inf. Process. Lett.*, vol. 32, pp. 159–165, 1989.

[8] M. S. Yu, C. L. Chen, and R. C. T. Lee, "Optimal parallel circle-cover and independent set algorithms for circular ARC graphs," in *Proc. International Conf. Parallel Processing*, 1989, pp. 126–129.

[9] K.-Y. Chow, K.-S. Lui, and E. Lam, "Maximizing angle coverage in visual sensor networks," in *IEEE International Conf. Commun. (ICC)*, June 2007.

[10] G. F. Simmons, *Introduction to Topology and Modern Analysis*. McGraw Hill International Editions, 1963.

[11] H. Gupta, S. R. Das, and Q. Gu, "Connected sensor cover: self-organization of senor networks for efficient query execution," in *Proc. ACM MobiHoc*, 2003.

[12] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, Feb. 2004.

[13] G. Cao, G. Wang, T. L. Porta, S. Phoha, G. Wang, and W. Zhang, "Distributed algorithms for deploying mobile sensors," in *Proc. Theoretical Algorithmic Aspects Sensor, Ad Hoc Wireless Peer-to-Peer Networks*, 2004.

[14] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proc. ACM International Conf. Wireless Sensor Networks Applications (WSNA)*, 2003.

[15] ——, "The coverage problem in a wireless sensor network," *Mobile Networks Applications*, 2005.

[16] M. Ye, E. Chan, G. Chen, and J. Wu, "Energy efficient fractional coverage schemes for low cost wireless sensor networks," in *Proc. IEEE ICDCSW*, 2006.

[17] K.-S. Hung, K.-S. Lui, and Y.-K. Kwok, "A trust-based geographical routing scheme in sensor networks," in *Proc. IEEE WCNC*, 2007.

[18] K. Kar and S. Banerjee, "Node placement for connected coverage in sensor networks," in *Proc. ACM WiOpt*, 2003.

[19] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," in *Proc. ACM Wireless Networks*, 2005, pp. 333–340.

[20] M. Cardei, J. Wu, and M. Lu, "Improving network lifetime using sensors with adjustable sensing range," *International J. Sensor Networks*, pp. 41–49, 2006.

[21] M. T. Thai, Y. Li, F. Wang, and D.-Z. Du, "O(log n)-localized algorithms on the coverage problem in heterogeneous sensor networks," in *Proc. IEEE International Performance Computing Commun. Conf. (IPCCC)*, Apr. 2007.

[22] M. K. Watfa and S. Commuri, "Boundary coverage and coverage boundary problems in wireless sensor networks," *Int. J. Sensor Networks*, Apr. 2007.

[23] ——, "Power conservation approaches to the border coverage problem in wireless sensor networks," in *Proc. International Conf. Wireless Networks (ICWN)*, 2006.

[24] H. Lee and H. Aghajan, "Vision-enabled node localization in wireless sensor networks." in *Proc. Cognitive Systems Interactive Sensors (COGIS)*, Mar. 2006.

[25] C. McCormick, P.-Y. Laligand, H. Lee, and H. Aghajan, "Distributed agent control with self-localizing wireless image sensor networks." in *Proc. Cognitive Systems Interactive Sensors (COGIS)*, Mar. 2006.

[26] N. Tezcan and W. Wang, "Self-orienting wireless multimedia sensor networks for maximizing multimedia coverage," in *Proc. IEEE ICC*, May 2008.

[27] K.-Y. Chow, "Angle coverage in wireless sensor networks," M.Phil. thesis, The University of Hong Kong, Oct. 2007.

Ka-Shun Hung earned his B.Eng. (first class honors) in electrical and electronic engineering from the University of Hong Kong, Hong Kong. He continued his M.Sc. in electrical engineering at Columbia University, New York, and then received his Ph.D. in electrical and electronic engineering from the University of Hong Kong, Hong Kong. His research interests include perimeter coverage, trust, and security issues in wireless sensor networks.

King-Shan Lui (S'00-M'03-SM'09) received the B.Eng. and M.Phil. degrees in computer science from the Hong Kong University of Science and Technology. After receiving her Ph.D. degree from the University of Illinois at Urbana-Champaign, USA, she joined the Department of Electrical and Electronic Engineering at the University of Hong Kong. Her research interests include network protocol design and analysis, sensor networks, and quality-of-service issues.