

# On Perimeter Coverage in Wireless Sensor Networks

Ka-Shun Hung and King-Shan Lui  
Department of Electrical and Electronics Engineering  
The University of Hong Kong  
Pokfulam Road, Hong Kong

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  $360^\circ$  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.

## I. INTRODUCTION

Wireless sensor networks have caught lots of attentions in recent years. The main reason is that 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]. Such applications require the sensor nodes to sense the environment and send some up-to-date information back to the sink node either periodically or when they are triggered by some events/when they have received requests from the sink node.

Among these applications, monitoring is the one which has caught the most attention. In monitoring applications, small-sized battery-powered sensor nodes are deployed in a large scale. Each sensor node is equipped with some sensing equipment, e.g., a video camera, which can be used to sense the environment up to a certain sensing range and therefore the sensing function of each sensor can only cover a limited

physical area. Due to the limited sensing range, 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. For instance, a sudden increase of temperature in a monitored forest area may represent a possible fire. Lastly, 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 of the typical application scenario is to monitor  $360^\circ$  of the white house so as to ensure its security. Due to the limited battery power and possible overlapping of the sensing functions of the sensors, we would only like to activate a minimum set of sensors which are possible to cover the whole perimeter of the target object. This problem is very similar to the circle-cover problem in a circular-arc graph in which a number of centralized algorithms have been proposed [4], [5], [6], [7]. Unfortunately, all these proposed algorithms can not be directly applied in a wireless sensor network scenario which is distributed in nature. Previously, we [8], [9] proposed a distributed algorithm to solve this problem. The approach requires all the nodes passing through  $0^\circ$  to initiate the search and thus the overhead can be several multiples of the size of the minimum cover. 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 within  $O(\text{size of the minimum cover})$  number of messages. We also provide a formal proof of correctness and convergence time 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 that we are focusing in will be discussed in Section III together with some definitions. Afterwards, our proposed distributed protocol to solve the perimeter coverage problem in wireless sensor networks will be discussed in details in Section IV together with a formal proof of

correctness and time of convergence. Through extensive simulations, we show that our proposed algorithm outperforms some earlier developed distributed minimum cover algorithms in Section V. Finally, we conclude our scheme with some possible future work in Section VI.

## II. RELATED WORK

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 graph.

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. 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 all the points in the area. A centralized algorithm to find a small-sized connected sensor cover is presented in [10], while a localized algorithm for area coverage with no prior knowledge of neighbor existences and neighbor location is presented in [11]. Cao *et al.* [12] considered the movement of the sensor so as to cover the area which has not been covered by random distribution of the sensors.

The work suggested above only considers that every point within the target area should be covered by at least one sensor. However, due to various reasons, a certain area may be required to be covered by multiple sensors instead, e.g., some sensitive military area. This leads to the  $k$ -coverage problems suggested in the literature. Huang *et al.* [13], [14] transformed the coverage problem to a decision problem so as to determine whether the target area is covered by at least  $k$  sensors. According to their approach, each sensor can determine whether the area is  $k$ -covered by considering whether its perimeter is  $k$ -covered. On the other hand, fractional coverage problem has also been considered in [15], [16]. 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.

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 [17] studied the target coverage problem. They focused on the

node placement in order to ensure all the targets are covered. They assumed that the targets' locations are known beforehand and the node placement is deterministic. Their algorithm runs in a polynomial time. Cardei *et al.* [18], [19] studied the target coverage problem with the focus in energy-efficiency in monitoring a set of target objects with known location. They assumed that the placement of the nodes are random and they aimed at selecting a maximum number of disjoint sets of sensors in which 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 tried to propose a centralized algorithm to solve this problem. Other than that, Thai *et al.* [20] have proposed a  $O(\log N)$  distributed algorithm to solve the target coverage problem.

In this paper, we are specifically interested in the angle/perimeter coverage problem. Unlike that of the area coverage problem in which a certain target object area is of particular interest, we are interested in whether the target object perimeter is  $360^\circ$  covered by enough sensors. Unlike that of 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, angle/perimeter coverage problem is focused on monitoring  $360^\circ$  of a large target object in which a sensor can only cover a certain angle range of the target object. In fact, the angle/perimeter coverage problem is the same as finding the minimum circle-cover for circular-arc graph. Circular-arc graph problems have been studied for quite a long time. Generally speaking, there are two main types of algorithms for finding the minimum circle-cover for circular-arc graph — sequential and parallel algorithms.

Lee and Lee [4] 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  $O(N \log N)$  if the arcs are not sorted and  $O(N)$  if the arcs are sorted with the use of one processor. Since their algorithm achieves a time complexity of  $O(N)$  in the worst case when the arcs are sorted based on the start angle, it is optimal in nature.

On the other hand, Bertossi [5] was known to be the first to propose a parallel algorithm to tackle this problem. Basically, their algorithm tries to find an arc  $i$  with end angle  $t_i$  which overlaps with the minimum number of arcs. The algorithm achieves a time complexity of  $O(\log N)$  with the use of  $O((N^2/(\log N) + qN))$  processors, where  $q$  is the minimum set of arcs overlap in the graph. Ref. [6]

and Ref. [7] proposed similar optimal parallel algorithm to tackle this problem. Generally speaking, their approaches built up  $q$  trees with the use of successor and inverse successor relationship (in which successor of an arc  $i$  is the greedy clockwise neighbor of  $i$ ). The set of nodes with the minimum depth from the root to leaf will be the MC. Both approaches achieve a time complexity of  $O(\log N)$ . However, the algorithm in [6] requires  $O(N/\log N)$  processors, while the one in [7] 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.

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 [21], [22] proposed a distributed algorithm to find a reduced set of nodes to cover the border of a rectangle. Instead of actively searching for the cover nodes, they proposed an algorithm in which the nodes determined that their cover ranges are completely covered by their neighbors will be declared to inactivate. Unfortunately, they did not provide the proof of correctness and convergence of their algorithm.

To the best of our knowledge, we are the first to apply minimum circle-cover problem of circular-arc graphs on the angle/perimeter coverage problem in wireless sensor networks. Previously, we have proposed an optimal centralized algorithm to find the minimum number of visual sensor nodes which are necessary to cover  $360^\circ$  of the target object [8], [9]. Also, a distributed algorithm is proposed to tackle this problem and this is also known to be the first distributed algorithm proposed. The proposed distributed algorithm is similar to that of [5]. In our algorithm, every node passing through  $0^\circ$  would find the smallest cover that contains itself using the greedy clockwise neighbor selection approach. The minimum set will then be selected as the minimum cover among the sensors. Unfortunately, a large message overheads in terms of the number of messages are required. In this paper, we adopt another approach and provide the proof of the correctness of our proposed algorithm. For comparison, we have adopted Lee and Lees' sequential algorithm [4] into a distributed environment which required the same number of messages as that of our previously proposed algorithm in the worst case. Also, we implemented our previously proposed distributed algorithm [8], [9] together with the algorithm described in this paper. These algorithms will then be compared through both theoretical and simulation analysis.

### 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 and 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 perimeter. A sensor  $i$  can cover only part of the perimeter and we denote that as  $[s_i, t_i]$ .  $[s_i, t_i]$  spans in the clockwise manner as illustrated in Figure 1. For a sensor  $i$  that does not cover  $0^\circ$ ,  $s_i < t_i$ .  $t_i < s_i$  if  $i$  covers  $0^\circ$ . The cover range of two or more sensors will be 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 [23], [24], [25].

Given a set of sensor  $S$ , a set of sensor  $D \subseteq S$  is a *cover* if for each angle  $\gamma \in [0^\circ, 360^\circ)$ , there exists a sensor  $j \in D$  such that  $\gamma \in [s_j, t_j]$ . In other words,  $\bigcup_{i \in D} [s_i, t_i] = [0^\circ, 360^\circ)$ . Figure 2 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 valid covers.

#### B. Minimum Cover

*Minimum Cover (MC)* is a set of sensors with the smallest size which preserves the entire cover range. Formally,  $C$  is a minimum cover if  $|C| \leq |D|$  for every cover  $D$ . In Figure 2, both  $\{1, 3, 5, 7, 8\}$  and  $\{1, 3, 5, 7, 9\}$  are a minimum cover. 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 minimum cover.

We define *minimum cover contains Sensor  $i$* , denoted as  $MC(i)$ , to be the smallest cover that contains  $i$ . In other words,  $|MC(i)| \leq |D|$  for every cover  $D$  where  $i \in D$ . Obviously, there exists a sensor  $i$  such

that  $MC(i)$  is also an MC. However, not every  $MC(i)$  is a MC. For example, in Figure 2,  $MC(2)$  is  $\{2, 3, 5, 6, 9, 1\}$  and this is not a MC.

### C. Backward and Forward Neighbors

Two nodes are *neighbors* if their cover ranges overlap. Formally,  $i$  and  $j$  are neighbors if  $s_i < s_j < t_i$  or  $s_i < t_j < t_i$ <sup>1</sup>. Each node can communicate directly with neighbors only. It is possible that  $[s_j, t_j]$  completely contains  $[s_i, t_i]$  like sensors 9 and 8 in Figure 2. When two sensors have overlapping cover ranges and none of them is contained in the other, one of them is a *backward neighbor* and the other is a *forward neighbor*.  $i$  is a backward neighbor of  $j$  and  $j$  is a forward neighbor of  $i$  if  $s_i < s_j < t_i$ . Refer to Figure 2, sensors 2 and 3 are forward neighbors of sensor 1, while sensors 9 and 8 are backward neighbors of sensor 1.

A node may not necessarily have backward neighbor and/or forward neighbor. An example is shown in Figure 3. Sensor 5 does not have a backward neighbor and Sensor 4 does not have a forward neighbor. When this occurs, it means that some portion on the perimeter cannot be covered by any sensor.

### D. Default Member

A sensor is a *default member* if it covers a portion of the perimeter that no other sensor is covering. Formally,  $i$  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,  $i$  must be in any cover and  $MC(i)$  must be a minimum cover.  $i$  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 Figure 2, Sensor 1 is a default member since none of its backward neighbor's cover range overlaps with the range of any forward neighbor. On the other hand, Sensor 2 is not a default neighbor because the sensing range of its backward neighbor, Sensor 1, overlaps with the one of its forward neighbor, Sensor 3.

<sup>1</sup>This applies when both  $i$  and  $j$  do not cover  $0^\circ$ . The definition can be extended easily to ranges that cover  $0^\circ$  but we leave it out for the ease of discussion.

#### IV. DISTRIBUTED MINIMUM COVER (DMC)

##### A. Finding $MC(i)$

As mentioned before, if  $i$  is a default member,  $MC(i)$  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  $MC(i)$  would solve our problem. Therefore, we first describe how we can find  $MC(i)$  given  $i$ .

To find  $MC(i)$ , we want to find the minimum set of sensors that cover the range  $[t_i, s_i]$ .  $i$  knows its neighbors and it adopts the greedy strategy that it selects one that covers as far as possible. Without loss of generality, we assume  $i$  selects a forward neighbor and sensors are selected in the clockwise direction. That is,  $i$  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  $GFN(i)$ .  $i$  informs its greedy forward neighbor  $j$  that it is selected and  $j$  selects its own greedy forward neighbor. The process ends when a selected node realizes that  $i$  is a forward neighbor. To facilitate this, the identity of  $i$  has to be carried around in the selection process. Refer to the example in Figure 2, suppose that we want to find  $MC(1)$ .  $GFN(1)$  is 3 and so 1 informs 3 that it is selected. Sensor 3 selects 5 as it is the greedy forward neighbor of 3. 5 selects 6 and 6 selects 9. As 1 is a forward neighbor of 9, 9 informs 1 and the searching process is finished.

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 selected node would send a message and so the message complexity is  $O(|MC(i)|)$ , which is very efficient. We now proof the correctness of the greedy strategy. 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  $GFN(GFN(\dots(GFN(i))))$  where  $GFN$  is found for  $k$  times. For convenience, we label  $GFN^0(i)$  to be  $i$ .

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

Proof: Suppose that given a certain  $MC(i)$ ,  $C = MC(i) \setminus \{i\}$  and  $|C| = k$ . The cover ranges of the sensors in  $C$  arranged in clockwise order are  $[\sigma_1, \tau_1], \dots, [\sigma_k, \tau_k]$ , 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  $[s_1, t_1], \dots, [s_{k'}, t_{k'}]$ . 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  $\sigma_1$  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  $\sigma_{j+1}$  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. ■

### 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 several properties related to GFN.

**Property 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 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 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. ■

### C. Finding MC

Let  $S_0$  be the set of sensors that cover  $0^\circ$ . 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 the 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 search of different  $MC(q)$ 's to reduce message overhead.

Let  $q_m$  be the sensor in  $S_0$  with the largest ending angle. That is,  $q_m$  is a forward neighbor of all the nodes in  $S_0$ . By Property 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  $T \subseteq S_0$  be the set of nodes such that  $GFN(q) = q_m$  and  $s_i \leq s_j$  where  $i, j \in T$ . By Property 3, we know that  $MC(j)$  cannot be better than  $MC(i)$  for all  $j \in T$  where  $i \neq j$ . Therefore, we do not have to bother finding  $MC(j)$  and we can *prune* the search of  $MC(j)$ . This is one way to reduce message overhead. 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  $\{i, f\}$  covers a larger range than  $\{j, f\}$ . 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. For example, refer to Figure 4,  $q_m = 6$  and  $MC(1)$  or node 1 is pruned because  $GFN(1) = 6$ . As  $GFN(2) = GFN(3) = 8$ , 3 is pruned.  $GFN(4) = 9$  and  $GFN(5) = 10$  and both are not pruned. After finding  $GFN(6) = 11$ , 6 sends the information of the unpruned searches to 11. To facilitate 11 to further identify the MC of the unpruned nodes, we tell 11 the greedy forward neighbors of 2, 4, and 5. That is,  $q_m$  sends  $GFN(q_m)$  a list of  $\langle q, s_q, GFN(q) \rangle$  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 2,  $GFN(q_m)$  must be a forward neighbor of  $GFN(q)$  and so  $GFN(q_m)$  can identify  $GFN(GFN(q))$  for all  $q$ .

Refer to Figure 4, 11 receives  $\langle \langle 2, s_2, 8 \rangle, \langle 4, s_4, 9 \rangle, \langle 5, s_5, 10 \rangle, \langle 6, s_6, 11 \rangle \rangle$ .  $GFN(8) = 11$  and so 2 is pruned.  $GFN(10) = GFN(11) = 13$  and we should prune one of them. Since 5 starts earlier than 6 (i.e.,  $s_5 < s_6$ ), we should eliminate 6. To allow 11 to determine this, we also put the starting angles of  $q$ 's in the message. In this case, 11 can prune 6. 9 is not pruned and 11 sends  $\langle \langle 4, s_4, 12 \rangle, \langle 5, s_5, 13 \rangle \rangle$  to Node 13. In other words, the list being passed around the nodes consists of entries in the form of  $\langle q, s_q, GFN^k(q) \rangle$  where  $q \in S_0$  and  $q$  is not pruned.  $k$  in the entry is related to how many hops that message has gone through. For example,  $q_m$  (6 in our example) sends out  $\langle q, s_q, GFN(q) \rangle$  and  $GFN(q_m)$  (11 in our example) sends out  $\langle q, s_q, GFN^2(q) \rangle$ .

The search can stop when the message goes around the perimeter and returns back to the starting node. That is, upon receiving the entry  $\langle q, s_q, f \rangle$ , the node realizes that  $q$  is a forward neighbor of  $f$ . For example, when 13 receives  $\langle \langle 4, s_4, 12 \rangle, \langle 5, s_5, 13 \rangle \rangle$ , it finds that 5 is a forward neighbor of 13, it knows that  $MC(5)$  is an MC. 13 can inform 5 that it is in an MC and 5 can launch the real search of minimum cover.

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  $\langle q, s_q, f \rangle$  in the message received by  $GFN^k(q_m)$  satisfies  $f = GFN^k(q)$ . Algorithm 1 illustrates what  $GFN^k(q_m)$  should do when it receives the search message. 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  $MC$  in case it does not present in  $q$  or  $f$  of any entry  $\langle q, s_q, f \rangle$  in the search message.

#### D. Proof of Correctness

**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  $\langle\langle q_1, s_{q_1}, GFN^k(q_1) \rangle, \dots, \langle q_L, s_{q_L}, GFN^k(q_L) \rangle\rangle$ . According to Property 1, and Property 2,  $GFN^k(q_m)$  can find the  $GFN^{k+1}(q_i)$ , for each  $q_i$  in  $M$ .

From Algorithm 1,  $GFN^k(q_m)$  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, as we know that  $GFN^{l'-k-2}(x)$  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  $GFN^{l'-k-1}(x)$  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  $\langle q, s_q, f \rangle$  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  $MC(j)$ ,  $j \in S_0$  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  $GFN^k(q_m)$  prune 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  $\langle q, s_q, f \rangle$  in message  $M$  for each unpruned  $MC(q)$ ,  $\exists q \in S_0$  and an entry  $\langle q, s_q, f \rangle$  in message  $M$ , such that  $MC(q)$  is an  $MC$ . ■

**Lemma 4** *If  $GFN^k(q_m)$  receives an entry  $\langle q, s_q, f \rangle$  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  $\langle q, s_q, f \rangle$  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  $\langle p, s_p, f \rangle$  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  $\langle q, s_q, f \rangle$  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  $\langle q, s_q, f \rangle$  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  $\langle q, s_q, f \rangle$  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  $O(|MC(q_m)|)$  number of messages.*

The proof of this is trivial. Our distributed algorithm starts with  $q_m$  and terminates when any  $q$  in  $\langle q, s_q, f \rangle$  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  $\langle q_m, s_{q_m}, f \rangle$ . 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  $O(|MC(q_m)|)$  number of messages for the search of  $MC$ . After the search of  $MC$ , nodes in  $MC$  are informed. This also requires another  $O(|MC(q_m)|)$  number of messages. As a result, our distributed algorithm terminates with  $O(|MC(q_m)|)$  number of messages in the worst case. ■

## V. PERFORMANCE ANALYSIS

We compare our proposed algorithm with two other algorithms. The first one is the modification of the sequential algorithm proposed in [4]. In order to cope with a distributed environment, we designed the message format to be passed from one node to another. In each message, there are three main fields, including,  $B1$  which is the first node chosen in the initial cover,  $k$  which is the size of the initial cover,  $GFN$  which is the greedy clockwise neighbor selected. In other words, the message format is  $\langle B1, k, GFN \rangle$  and this message will be initiated by  $B1 \in S$  and send to  $GFN(B1)$  until the message reaches the same node twice or the number of zone developed is equal to the size of initial cover. We denote this algorithm as *GMLL Algorithm*. The second algorithm considered is the one proposed in [8], [9] which is very similar to the parallel algorithm proposed in [5]. We denote this algorithm as *Exhaustive Algorithm*.

### A. Theoretical Analysis

As discussed in Theorem 2, our distributed algorithm terminates with  $O(|MC(q_m)|)$  number of messages. However, the *Exhaustive Algorithm* terminates with  $O(|MC(q_m)||S_0|)$  number of messages. It is because their approach requires each node  $q \in S_0$  to initiate the search for  $MC(q)$  individually. This results in high message overheads in terms of number of messages. On the other hand, *GMLL Algorithm* can be initiated by any node and adopted an approach in which visiting the same node twice indicating that a termination decision can be made. In that case, their approach, in fact, also requires  $O(|MC(q_m)||S_0|)$  number of messages in the worst case. As a result, we can conclude that our distributed algorithm performs better than that of *Exhaustive Algorithm* and *GMLL Algorithm* in terms of the number of messages theoretically.

### B. Simulation Results

In the simulation, a simulation environment similar to the one adopted in [8] was used. In our simulations, we considered a grid size of 30 units  $\times$  30 units, in which every grid has a probability of 0.8 to contain a sensor. Each sensor will have a certain sensing range. The target object is located at the center (15, 15) with the object radius, which can be regarded as the size of the object, defined in our simulation.

Two performance metrics will be used in our simulations. The first one is the total number of messages required to find the minimum cover. The second one is the average time (in terms of number of messages) required for a node to determine whether it is in the minimum cover. The rationale behind the first metric is trivial as the larger the number of messages, the larger will be the amount of energy required to transmit. On the other hand, the rationale behind the second metric is that in wireless sensor networks, most of the energy of a node is dissipated in idle mode instead of transmit or receive mode. As a result, the earlier the node can determine whether it should be included in the minimum cover, the sooner it can turn itself into the energy-saving sleep mode if it is not in MC.

1) *Effects of the Sensing Range:* Figure 5 illustrates the number of messages required for different algorithms to find MC in different sensing ranges. The target object is located at the center of the grid with a radius of 11. Generally speaking, the larger the sensing range, the smaller the MC will be to cover the target object. Since our algorithm is guaranteed to find MC in  $O(\text{size of the minimum cover})$  number of messages, the smaller the number of nodes required to cover the perimeter of the target object, the smaller will be the number of messages required. As a result, the number of messages required to find MC in our proposed algorithm decreases with the increase in sensing range. On the other hand, the number of messages required to find MC in *Exhaustive Algorithm* increases with the sensing range. The main reason is that the algorithm requires all the nodes passing through  $0^\circ$  to initiate and find MC(i), where  $i \in S_0$ . The larger the sensing range, the larger will be the amount of nodes passing through  $0^\circ$  and this results in the increase in the number of messages. For the *GMLL Algorithm*, we notice that the number of messages required to find MC increases with the sensing range at the beginning and then decrease afterwards. There are two termination conditions for *GMLL Algorithm*: the first one is that the same arc is considered twice and the second one is that the number of zones equal to the size of the initial cover developed. Generally speaking, when the sensing range is small, the choice of replaceable node in terms of cover range is limited and so it is easier to meet a node twice. At the same time, the increase in the sensing range also leads to the decrease in the number of nodes required to cover the target and this leads to the decrease of the number of messages required at the beginning. However, when the sensing range is increased up to a certain point, the number of replaceable nodes for the cover range becomes large and it becomes difficult to meet the same node twice. As a result, the number of messages required increases.

Figure 5 also illustrates the average time required for a node to determine whether it is included in MC. Generally speaking, the trends of all the curves are still the same as that of the number of messages to find MC. However, in *Exhaustive Algorithm*, all the nodes passing through  $0^\circ$  are assumed to start at the same time to find MC, so the nodes passing through  $0^\circ$  should find  $MC(i)$  nearly at the same time, and then compare the results among themselves to determine MC. As a result, the time in terms of number of messages is much less than that of finding MC. On the other hand, in *GMLL Algorithm*, a node can only determine whether it is in MC after a node in  $S_0$  starts the real search for MC, i.e., a node starts informing others after it concluded that it is in MC. As a result, the average time is generally a little bit less than that of finding MC. In contrast, in our proposed algorithm, some of the nodes can conclude that they are not in MC in the search message. As a result, the average time is generally half that of finding MC.

2) *Effects of the Object Radius:* Figure 6 illustrates the number of messages required for different algorithms to find MC in which the target object is of different size. Under this simulation, all the nodes are assumed to have a sensing range of 11 units. Generally, the larger the object radius, the larger the object size, and the larger amount of nodes is required to cover  $360^\circ$  of the object. As a result, a larger amount of messages is required to find MC due to the larger size of the target object. All the simulated algorithms exhibit increase in the number of messages when the object radius increases. Our proposed algorithm requires the least amount of messages as each message includes all the necessary information to find possible  $MC(i)$ , for  $i \in S_0$  and only  $O(|MC(q_m)|)$  number of messages are required to find MC. On the other hand, the *Exhaustive Algorithm* required to largest amount of messages to find MC as every nodes passing through  $0^\circ$  are initiating the search of  $MC(i)$ , where  $i \in S_0$ . Last but not least, *GMLL Algorithm* achieves larger amount of messages than our proposed algorithm but much less than that of *Exhaustive Algorithm* as *GMLL Algorithm* tries to look for  $MC(i)$ , where  $1 < i < N$  and it will terminate when it can determine MC is found.

Figure 6 also illustrates the average time for a node to determine whether it is in MC. The algorithms also exhibit similar trend as that of the number of messages required to find MC.

## VI. CONCLUSION AND FUTURE WORK

One of the typical applications of sensor networks is to solve coverage problems. Several common coverage problems are briefly discussed in this paper. In this paper, we focus on the angle/perimeter coverage problem in which a large target object is located in the center and multiple sensors are expected to collaborate to monitor  $360^\circ$  of this object. This coverage problem is very different from that of some common coverage problems in which a certain area or point are to be monitored instead.

The angle/perimeter coverage problem in fact is very similar to that of circle cover problem in circular-arc graph. To the best of our knowledge, we are the first to propose a distributed algorithm to solve the circle cover problem of the circular-arc graph with  $O(|\text{size of the minimum cover}|)$  number of messages together with the proof of correctness. With the help of broadcasting nature of the wireless medium, our proposed algorithm allows those sensors which overhear the search message to determine whether they are possible candidates for MC so as to turn into sleep mode as soon as possible once they determine they are not.

To investigate the performance of our proposed algorithm, we compared our solution with two other algorithms – *GMLL Algorithm* and *Exhaustive Algorithm*. In fact, the *GMLL Algorithm* is a centralized sequential algorithm and we have modified it to suit the distributed environment by adopting an appropriate message format. On the other hand, the *Exhaustive Algorithm* is known to be the first distributed algorithm proposed without a formal proof of correctness and it is very similar to that of a centralized parallel algorithm as stated in this paper.

Through extensive simulations, it is found that our proposed algorithm performs better than that of the *GMLL Algorithm* and *Exhaustive Algorithm* in terms of number of messages required to find MC and the average time required for a node to determine whether it is in MC. In the future, we would like to further develop a distributed algorithm for the minimum cost circle-cover problem in circular-arc graph and investigate the possibility of using broadcast nature of wireless medium to further improve the performance of the existing centralized and distributed algorithms proposed to solve this problem.

## 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. on Wireless Networks*, 2004.
- [3] M. Cardei and J. Wu, "Coverage in wireless sensor networks," *Handbook of Sensor Networks*, M. Ilyas and I. Magboub (eds.), 2004.
- [4] C. C. Lee and D. T. Lee, "On a circle-cover minimization problem," *Information Processing Letters*, vol. 18, pp. 109–115, February 1984.
- [5] A. A. Bertossi, "Parallel circle-cover algorithms," *Information Processing Letters*, vol. 27, pp. 133–139, 1988.
- [6] M. Atallah and D. Z. Chen, "An optimal algorithm for the minimum circle-cover problem," *Information Processing Letters*, vol. 32, pp. 159–165, 1989.
- [7] 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. 1989 International Conference on Parallel Processing*, 1989, pp. 126–129.
- [8] K.-Y. Chow, K.-S. Lui, and E. Lam, "Maximizing Angle Coverage in Visual Sensor Networks," in *IEEE International Conference on Communications (ICC 2007)*, June 2007.
- [9] K.-Y. Chow, "Angle coverage in wireless sensor networks," *MPhil Thesis, The University of Hong Kong*, October 2007.
- [10] H. Gupta, S. R. Das, and Q. Gu, "Connected sensor cover: Self-organization of sensor networks for efficient query execution," in *ACM MobiHoc*, 2003.
- [11] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, February 2004.
- [12] G. W. G. Cao, T. L. Porta, S. Phoha, G. Wang, and W. Zhang, "Distributed algorithms for deploying mobile sensors," in *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*, 2004.
- [13] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *ACM International Conference Wireless Sensor Networks and Applications (WSNA)*, 2003.
- [14] —, "The coverage problem in a wireless sensor network," *Mobile Networks and Applications*, 2005.
- [15] M. Ye, E. Chan, G. Chen, and J. Wu, "Energy Efficient Fractional Coverage Schemes for Low Cost Wireless Sensor Networks," in *IEEE ICDCSW*, 2006.
- [16] K.-S. Hung, K.-S. Lui, and Y.-K. Kwok, "A Trust-based Geographical Routing Scheme in Sensor Networks," in *IEEE WCNC*, 2007.
- [17] K. Kar and S. Banerjee, "Node placement for connected coverage in sensor networks," in *ACM WiOpt*, 2003.
- [18] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," in *ACM Wireless Networks*, 2005, pp. 333–340.
- [19] M. Cardei, J. Wu, and M. Lu, "Improving network lifetime using sensors with adjustable sensing range," in *International Journal of Sensor Networks*, 2006, pp. 41–49.
- [20] 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 *IEEE International Performance Computing and Communications Conference (IPCCC 2007)*, April 2007.
- [21] M. K. Watfa and S. Commuri, "Boundary coverage and coverage boundary problems in wireless sensor networks," *Int. J. Sensor Networks*, April 2007.
- [22] —, "Power Conservation Approaches to the Border Coverage Problem in Wireless Sensor Networks," in *International conference on Wireless Networks ICWN 06*, 2006.
- [23] H. Lee and H. Aghajan, "Vision-enabled node localization in wireless sensor networks," in *COGNITIVE systems with Interactive Sensors (COGIS)*, March 2006.
- [24] C. McCormick, P.-Y. Lalgand, H. Lee, and H. Aghajan, "Distributed agent control with self-localizing wireless image sensor networks," in *COGNITIVE systems with Interactive Sensors (COGIS)*, March 2006.
- [25] N. Tezcan and W. Wang, "Self-Orienting Wireless Multimedia Sensor Networks for Maximizing Multimedia Coverage," in *IEEE ICC*, May 2008.

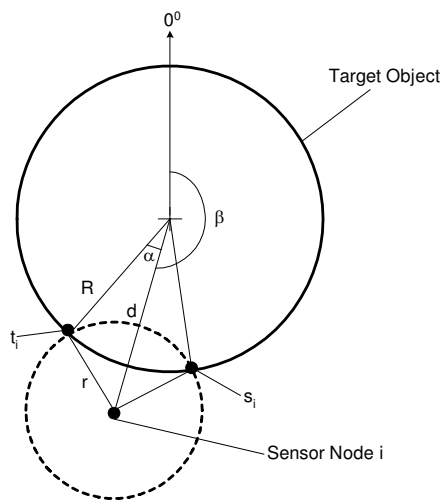


Fig. 1. A typical Sensor Network Scenario

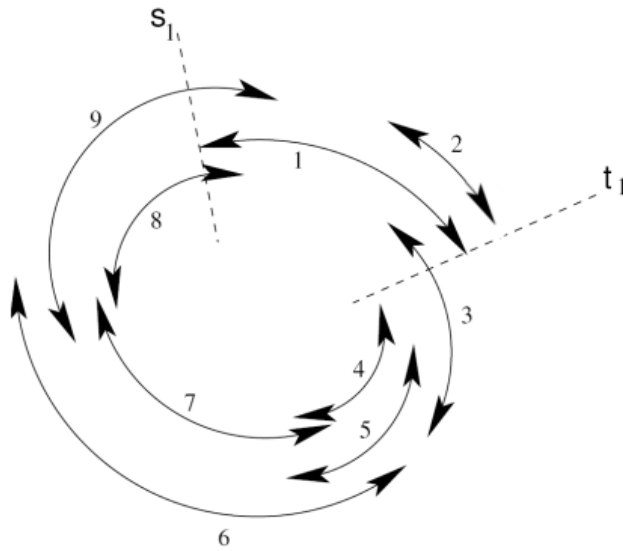


Fig. 2. Illustration of sensor covers

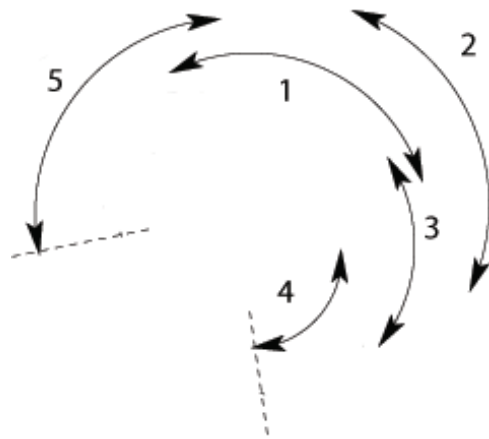


Fig. 3. Illustration of the existence of a gap

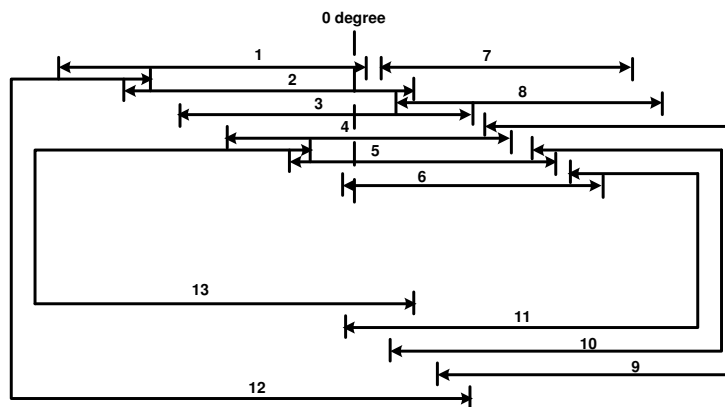


Fig. 4. An illustrative example

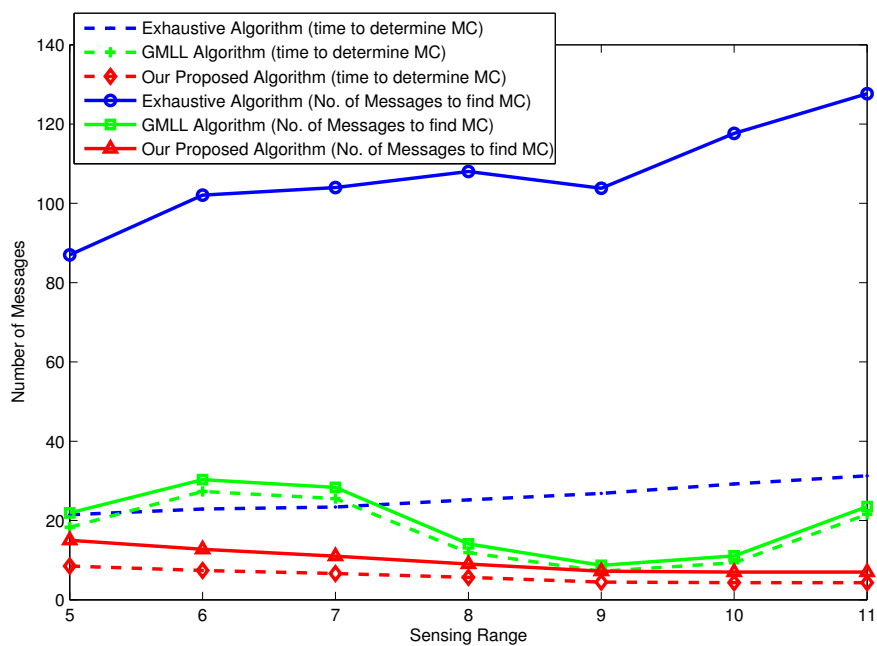


Fig. 5. Number of Messages Vs. Sensing Range with object radius 7

---

**Algorithm 1** Node  $q$  receives message  $\langle\langle q_1, s_1, f_1 \rangle, \dots \langle s_L, s_L, f_L \rangle\rangle$

---

```

1: Precondition:
2:  $q = GFN^K(q_m)$  for some  $k$ .
3:  $f_i = GFN^k(q_i)$ 
4:  $s_i = s_{q_i}$ 
5: /* Check whether an MC is identified. */
6: for  $i = 1$  to  $L$  do
7:   if  $q_i$  is a forward neighbor of  $f_i$  then
8:     /*  $MC(q_i)$  is an MC. */
9:     Inform  $q_i$  to start the real  $MC$  search and terminate.
10:   end if
11: end for
12: /* Determining whether search of  $MC(q_i)$  can be pruned. Initially, assume all the searches cannot be pruned. */
13: for  $i = 1$  to  $L$  do
14:   /* Prune  $MC(q_i)$  if  $GFN(f_i) = q$ . */
15:   if  $q = GFN(f_i)$  then
16:     Prune  $MC(q_i)$ .
17:     Continue
18:   end if
19:   for  $j = 1$  to  $L$  do
20:     if  $i \neq j$  and  $MC(q_i)$  and  $MC(q_j)$  have not been pruned then
21:       /* Check if they share the same greedy forward neighbor. */
22:       if  $FN(f_i) = GFN(f_j)$  then
23:         prune  $MC(q_j)$ , if  $s_i < s_j$ ;
24:         prune  $MC(q_i)$ , otherwise.
25:       end if
26:     end if
27:   end for
28: end for
29: /* Send the unpruned search of  $MC(q_i)$  to  $GFN(q)$ . */
30: Send  $\langle\langle q_i, s_i, GFN(f_i) \rangle\rangle$  to  $GFN(q)$  for every  $MC(q_i)$  that has not been pruned.

```

---

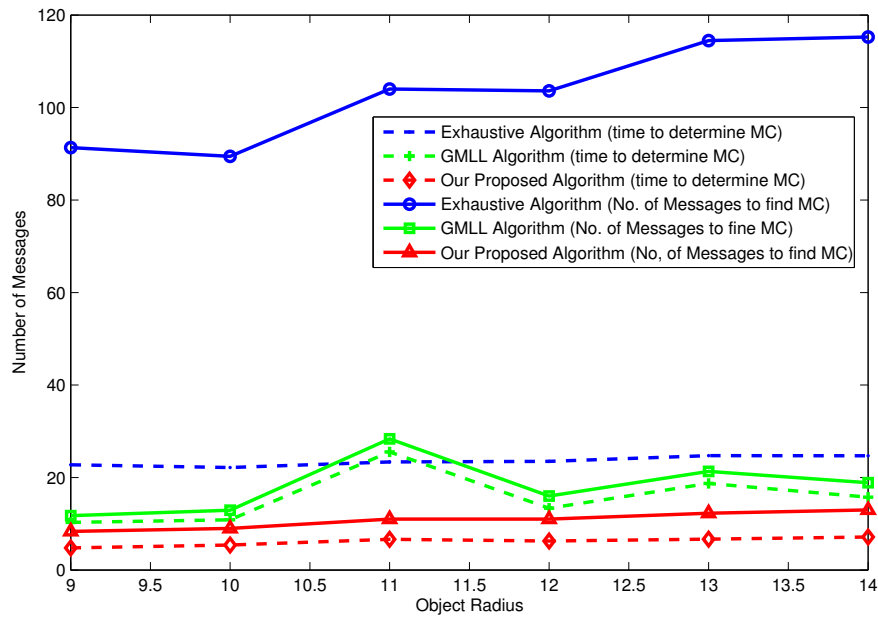


Fig. 6. Number of Messages Vs. Object Radius