On Perimeter Coverage in Wireless Sensor Networks with Minimum Cost

K.-S. Hung

Department of Electrical and Electronic Engineering,
The University of Hong Kong,
Pokfulam Road, Hong Kong
E-mail: kshung@eee.hku.hk

K.-S. Lui*

Department of Electrical and Electronic Engineering,
The University of Hong Kong,
Pokfulam Road, Hong Kong
E-mail: kslui@eee.hku.hk
* Corresponding author

Abstract: One of the major applications of sensor networks is tracking and surveillance. Very often, a single sensor is sufficient to monitor a single target. However, when the object is very large, several sensors have to work together to monitor the object continuously. In this paper, we study how to identify a set of sensors that can cover the perimeter of a large target with the minimum cost. We develop a novel distributed algorithm that requires fewer messages than existing mechanisms. Our algorithm can be extended to solve the problem when the sensor range is adjustable. We provide a formal proof of correctness and convergence time analysis of our algorithm. We further demonstrate the performance through extensive simulations.

Keywords: perimeter coverage; sensor networks; minimum cost; circular-arc graph; distributed algorithm.

Reference to this paper should be made as follows: Hung, K.-S. and Lui, K.S. (2010) 'On Perimeter Coverage in Wireless Sensor Networks with Minimum Cost', *International Journal of Sensor Networks* , Vol. , Nos. , pp.

Biographical notes: Ka-Shun Hung did his BEng. (first class honors) in Electrical and Electronic Engineering from the University of Hong Kong, Hong Kong. He continued his MSc in Electrical Engineering from the Columbia University, New York, and then received his PhD in the 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 received the B.Eng. and M.Phil. degrees in Computer Science from the Hong Kong University of Science and Technology. After receiving her PhD degree from the University of Illinois at Urbana-Champaign, USA, she joined the Department of Electrical and Electronic Engineering of the University of Hong Kong. Her research interests include network protocol design and analysis, sensor networks, and Quality-of-Service issues.

1 Introduction

Wireless sensor networks have caught lots of attentions in recent years, because applications that are too dangerous for humans to operate could be performed by WSNs (Estrin et al. (2002)). In monitoring applications, small battery-powered sensor nodes are deployed in a large scale. Each sensor node can sense the environment up to a certain sensing range. Thus, each sensor can only cover a limited physical area and sensors usually cooperate to achieve a certain monitoring objective. The monitoring objective is usually transformed to a coverage problem. Area coverage and target coverage are are two common monitoring objectives widely studied (Cardei and Wu (2004)). An area is under monitoring in area coverage that any event occurs inside the area should be discovered. In target coverage, a number of target objects are needed to be observed.

In this paper, we study another coverage problem, called *perimeter coverage*, in which the perimeter of a large object has to be monitored. One typical application scenario is to monitor the entire perimeter of the white house so as to ensure its security. Each sensor is associated with a cost. To reduce the total cost for monitoring, we would like to identify a set of sensors that can cover the whole perimeter with the minimum cost. This problem is similar to the minimum weight circle-cover problem in a circular-arc graph. There are some centralized algorithms in the literatures(Andrews and Lee (1995); Atallah et al. (1995); Bertossi (1988); Ibarra et al. (1992)). Nevertheless, these algorithms do not tailor for the broadcast and distributed nature of wireless sensor networks. Previously, a distributed algorithm has been developed (Chow et al. (2007a)) to solve this problem. This mechanism suffers from a high message overhead because all the nodes passing through a reference point has to initiate the search. In this paper, we further enhance the distributed protocol by reducing the message overhead to be proportional to the size of network. We also provide a formal proof of correctness of our proposed algorithm.

Many existing networks assume that the sensing range of a sensor is fixed. However, recent research shows that adjustable sensing range is possible. It is generally assumed that the cost associated with a smaller range is smaller (Cardei and Du (2005); Cardei et al. (2005); Cardei et al. (2006); Dhawan et al. (2006); OPS (2009); Wang and Medidi (2007); Wu and Yang (2004); Zhou et al. (2004)). In view of this, we enhance our algorithm to consider also sensors with adjustable sensing range without increasing the message overhead.

The paper is organized as follows. In Section 2, we briefly discuss the related work in the literatures. Then, the network model and our distributed algorithms under the fixed sensing range scenario will be discussed in detail in Section 3 together with a formal proof of correctness. In Section 4, we present the modifications required to our distributed algorithm under the adjustable sensing range scenario. We carried out extensive simulations and the results are presented in Section 5. Finally, we conclude our schemes in Section 6.

2 Related Work

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

The two traditional coverage problems 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 (Carle and Simplot-Ryl (2004); Cao et al. (2004); Gupta et al. (2003); Gupta et al. (2006)), multiple area coverage (e.g., k-coverage (Huang and Tseng (2003); Huang and Tseng (2005))) and fractional area coverage (Hung et al. (2007); Ye et al. (2006)), etc. On the other hand, target coverage problem refers to the cover of a certain target object or a number of target objects within a certain area. Various algorithms have been proposed to tackle the node placement (Kar and Banerjee (2003)) and energy issues (Cardei and Du (2005); Cardei et al. (2006); Thai et al. (2007)) on the target coverage problem. While most of the existing work assume that a target object is a point in the area, (Olteanu et al. (2008)) consider the situation in which a target may take up a certain space in the target area, and the target is detected once any portion of it falls within the sensing area of the sensor.

In this paper, we are specifically interested in the perimeter coverage problem. Unlike that of Olteanu et al. (2008) which aims at detecting a shaped target object, we study how to find a set of sensors to cover the whole perimeter of a shaped target object based on some selection criteria. One of the fundamental criteria is to find the set with the minimum amount of sensors distributively (Chow et al. (2007b); Hung and Lui (2010a)). This problem is very similar to finding a minimum circle-cover of a circular-arc graph, which studies how to cover the whole circle using as few arcs as possible. Circular-arc graph problems have been studied for quite a long time. The centralized algorithms for finding the minimum circle-cover are described in (Atallah and Chen (1989); Bertossi (1988); Lee and Lee (1984); Yu et al. (1989)).

Unlike these previous works, our main focus in this paper is to find the minimum weight circle-cover of a weighted circular-arc graph in a distributed manner. In (Bertossi (1988)), a parallel algorithm was proposed to tackle minimum weight circle-cover problem. This was known to be the first parallel algorithm proposed. Other than that, some parallel algorithms have been developed (Atallah et al. (1995); Ibarra et al. (1992)) but they are centralized in nature and do not exploit the wireless nature of sensor networks. Previously, a distributed algorithm was developed to find the set of nodes with minimum cost that covers the entire perimeter of the target object in a sensor network (Chow et al. (2007a)). This is also known to be the first distributed algorithm developed. Unfortunately, the message overhead of this algorithm can be very high. In this paper, we propose another approach that can reduce the message overheads and then provide a formal proof of correctness of our developed approach.

On the other hand, adjustable sensing range technique arouses more and more attentions recently. Many studies show that the network lifetime can be prolonged if the sensing range is reduced to lower the sensing cost (Cardei and Du (2005); Cardei et al. (2005); Cardei et al. (2006); Dhawan et al. (2006); Wu and Yang (2004)). It is widely assumed that energy consumed by a sensor is related to the sensing range of the sensor. The longer the range or the larger the sensing area, the more the energy is needed. The Osiris photoelectric sensors OPS (2009) available in the market support this

assumption. Therefore, in our paper, we also consider the scenario where a sensor can adjust its sensing range and the cost of sensing is related to the sensing range.

3 Distributed Minimum Cost Cover Algorithms under fixed sensing range scenario

3.1 Notations and Definitions

Before we describe our distributed protocol, we define our problem in this section. Some of the notations are borrowed from Hung and Lui (2010a).

3.1.1 Cover Range, Cover and Cost of a Cover

The perimeter of the big object to be monitored can be in any irregular shape as long as it forms a loop. Each sensor can only monitor a continuous portion of the perimeter. The portion that a sensor can cover is the *cover range* of the sensor. How a sensor determines the range is application dependent and it is outside the scope of this paper. Interested readers are referred to (Lee and Aghajan (2006); McCormick et al. (2006); Tezcan and Wang (2008)).

As the perimeter forms a close loop, we need a reference point on the perimeter to represent the starting point and also the end point of the perimeter so that a cover range can be represented as $[x, y]$ according to the coordinate system used on the perimeter. A possible way to represent x and y is to use the distance on the perimeter from the reference point. In this paper, we follow the practice in the circular-arc graph problem that we use $[0°, 360°)$ to model the whole perimeter. The cover range of a sensor i is represented as $[s_i, t_i]$ and $[s_i, t_i]$ spans in the clockwise manner as illustrated in Figure 1. It is worth noting that our protocols can be applied to any perimeter coordinate system that represents the portion a sensor can cover as a range. If sensor i does not cover $0°$, $s_i < t_i$; otherwise, $t_i < s_i$.

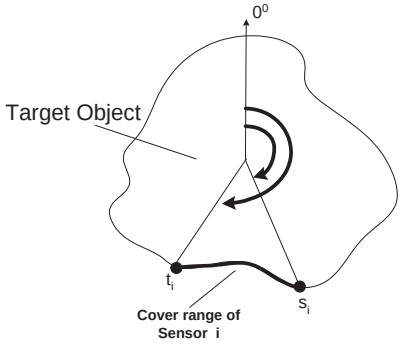

Figure 1 A sensor node i with cover range $[s_i, t_i]$ on the perimeter of the target object.

Given a set of sensors S, a subset of sensors $D \subseteq S$ is a *cover* if for each angle $\gamma \in [0^o, 360^o)$, there exists a sensor $j \in D$ such that $\gamma \in [s_j, t_j]$. In other words, $\bigcup_{i \in D}[s_i, t_i] = [0°, 360°)$. Figure 2 illustrates a scenario of 9 sensors surrounding a target object, where each arrow represents the cover range of a node.

Each sensor i is associated with a cost $f(i)$. In Figure 2, the number in circle represents the cost of each node. The cost of a cover D, $f(D)$, is the total cost of the sensors in the cover, that is, $f(D) = \sum_{i \in D} f(i)$. 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. The costs of the covers are 16, 22 and 17, respectively. Note that the cost of a sensor depends on the sensor application considered, and we do not specify any cost metric in this paper. Typical cost examples include the number of hop counts from the sensor to a sink for optimizing the message flow (Chow et al. (2007a)), or the remaining battery capacity of a sensor for extending the network lifetime (Chow et al. (2009)), or the cover range of a sensor for minimizing the amount of energy used for sensing (Hung (2010b)), etc.

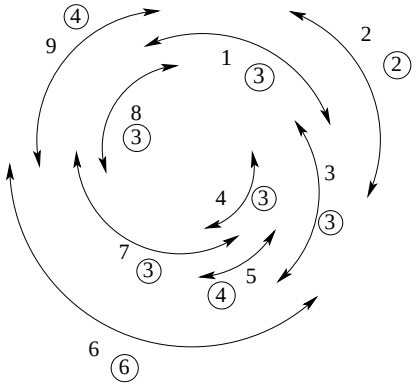

Figure 2 An example to illustrate the concept of minimum cost cover.

3.1.2 Minimum Cost Cover

Minimum Cost Cover (MCC) is a cover with minimum cost among all covers. Formally, M is a minimum cost cover if $f(M) \leq f(D)$ for every cover $D \subseteq S$. For example, in Figure 2, {1, 2, 4, 7, 8} is a Minimum Cost Cover, and the cost of the cover is 14. We denote $MCC(i)$ to be a cover that contains i while it is the minimum cost one among all covers that contain i. In other words, $f(MCC(i)) \leq f(D)$ for every cover D where $i \in D$.

3.1.3 Backward and Forward Neighbors

Two nodes are *neighbors* if their cover ranges overlap. Formally, when i and j both do not cover $0°$, i and j are neighbors if $s_i < s_j < t_i$ or $s_i < t_j < t_i$. The definition can be extended easily to ranges that cover $0°$ but we leave it out for the ease of discussion. With the assumption that the communication range is twice of the sensing range, each node can communicate directly with neighbors only. It is possible that the sensing range of a sensor is completely contained in another one, like

sensors 9 and 8 in Figure 2. When the sensing ranges of two sensors overlap and none of them is contained in the other, one sensor is a *backward neighbor* and the other sensor 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. As sensor 8 is completely contained in sensor 9, they are not backward and forward neighbors of each other. It is worth noting that 8 and 9 would not appear together in the same minimum cost cover.

3.2 Problem Statement

In this paper, we aim at developing a distributed algorithm to find the minimum cost cover (MCC) of the perimeter of a large object which forms a loop. (Note that if the perimeter is not a loop, the problem is reduced to finding the set of sensors with minimum cost to cover a range, which can be solved in reasonable overhead (Chow et al. (2007a)).) Each node $i \in S$ with cover range $[s_i, t_i]$ can only communicate with neighbor nodes who have overlapping cover ranges. By communicating with neighbors only, when the algorithm terminates, i can determine whether it is included in the MCC.

Theoretically, if we can find the cost of $MCC(i)$ for every $i \in S_0$, where S_0 contains the nodes with cover range passes through 0^o, we know the cost of the minimum cost cover. Therefore, we first describe how to find $MCC(i)$.

3.3 Finding MCC(i)

When i wants to identify $MCC(i)$, it initiates the search by sending a search message to all its forward neighbors. This message carries a cost value that keeps the total cost to cover the range starting from s_i to the node that receives the message. When the message goes through the set of nodes in the clockwise direction, it will eventually be sent back to i and i will know which backward neighbor is in $MCC(i)$ and the cost of $MCC(i)$. This algorithm allows each node to determine whether it is in $MCC(i)$ or not but no node, even i, knows all the nodes that are in $MCC(i)$.

We now describe the mechanism in more detail and provide an example. When node i wants to find $MCC(i)$, it sends the search message $< f(i), s_i, i >$ to all its forward neighbors. When a node j receives a message $< c, s_i, i >$ from its backward neighbor, it keeps track of the smallest c it receives so far. After j has received a message from *each* of its backward neighbors that *starts on or after* s_i, it sends $< c_{min} + f(j), s_i, i >$ to all its forward neighbors, where c_{min} is the minimum c among all the messages it received. Note that j needs to receive the messages from backward neighbors that start after s_i only. It is because nodes that start before s_i will receive the search message later. Subsequently, each node will send out a search message, and i will receive one search message from each of its backward neighbors. i can then determine which backward neighbor is in $MCC(i)$.

We use an example to illustrate the mechanism. Suppose Node 1 in Figure 3 wants to find $MCC(1)$. Note that Node 0 is ignored in this example as it is contained in Node 1. It initiates the search by sending $< f(1) = 3, s_1, 1 >$ to all of its forward neighbors, i.e., Nodes 2 and 3. Node 2 has only one backward neighbor that starts on or after s_1, i.e., Node 1. Therefore, Node 2 records $f(1)$ and sends $< f(1) + f(2) = 7, s_1, 1 >$ to its forward neighbors, which are Nodes 3 and 4. Now, Node 3 has received a message from both its backward neighbors, 1 and 2. Definitely, it finds the cost value carried by the message sent by 1 is smaller and so it sends $< f(1) + f(3) = 6, s_1, 1 >$ to its forward neighbors 4 and 5. This time, 4 receives messages from all its backward neighbors. Let the smaller cost among the messages that a node received be c. 4 sends $< c + f(4) = 6 + 2 = 8, s_1, 1 >$ to its forward neighbors 5 and 6. Subsequently, 5 sends out $< c + f(5) = 6 + 1 = 7, s_1, 1 >$ and 6 sends out $< c + f(6) = 7 + 2 = 9, s_1, 1 >$. Eventually, Node 1 would receive all messages from its backward neighbors and it can determine the cost of $MCC(1) = 7$.

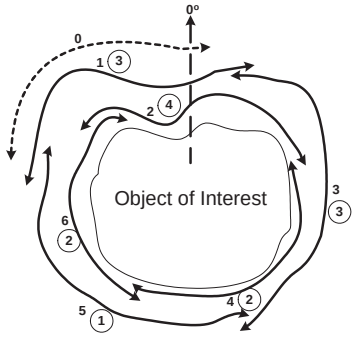

Figure 3 An $MCC(i)$ Search Algorithm Example.

The pseudocode of the $MCC(i)$ *search algorithm* of different states can be found in Algorithm 1 and Algorithm 2. When i wants to start the search, it enters the $INIT$ state. In this state, i sends a search message to all its forward neighbors and then transits to the $WAIT$ state. For other nodes n, when they start up, they enter the $INIT$ state. After initiating the variables in the $INIT$ state, it transits to the $WAIT$ state to wait for the search message from all its backward neighbors. Once it has received all backward neighbors' messages, it sends out the search message and enters FIN state. During this stage, node n has completed its task in searching for $MCC(i)$ and so it waits for $< SELECT >$ message to see if it is selected in the $MCC(i)$. Once i receives messages from all its backward neighbors, it can determine $MCC(i)$ by selecting the one with the minimum cost. Note that $previous(i)$ is used to store the id of the backward neighbor in which the search message with smaller cost comes from. At this stage, i can enter FIN state and send back the $< SELECT >$ message to

Algorithm 1 Pseudocode of distributed $MCC(i)$ Search Algorithm of source node i.

```
/* ———— INIT state ———— */
1: cost(i) = f(i)
2: previous(i) = ⊥
3: total_cost = ∞ /*current minimum cost*/
4: Send ⟨f(i), s_i, i⟩ to all forward neighbors. Then, i turns into
   WAIT state.
/* ———— WAIT state ———— */
1: /* Receiving Search Message ⟨c, s_i, i⟩ from Backward neighbor
   w */
2: if total_cost > c then
3:    /* Update the cost of MCC(i)*/
4:    total_cost = c
5:    previous(i) = w
6: end if
7: if all backward neighbors' messages are received then
8:    enter the FIN state
9: end if
/* ———— FIN state ———— */
1: /* f(MCC(i)) = total_cost */
2: send ⟨SELECT⟩ to previous(i)
```

Algorithm 2 Pseudocode of distributed $MCC(i)$ Search Algorithm of node n where $n \neq i$.

```
/* ———— INIT state ———— */
1: cost(i) = ∞
2: previous(i) = ⊥
3: enter the WAIT state.
/* ———— WAIT state ———— */
1: /* Receiving Search Message ⟨c, s_i, i⟩ from backward neighbor
   w */
2: if cost(i) > c + f(n) then
3:    /* Update the minimum cost to reach n from i */
4:    cost(i) = c + f(n)
5:    previous(i) = w
6: end if
7: if all backward neighbors' messages are received then
8:    Send ⟨cost(i), s_i, i⟩ to all forward neighbors.
9:    Enter FIN state.
10: end if
/* ———— FIN state ———— */
1: if receive < SELECT > then
2:    send ⟨SELECT⟩ to previous(i)
3: else if overhear < SELECT > sent to a neighbor that starts
   earlier than n then
4:    DONE (n is not selected)
5: end if
```

the node $previous(i)$ which denotes the node included in $MCC(i)$. When node n overhears a $< SELECT >$ message and it is the recipient of the message, it realizes that it is selected in $MCC(i)$. Then, it further sends a $< SELECT >$ message to the node $previous(i)$. This process continues until the node $previous(i)$ is node i. Nodes that contain i or contained in i are not involved in the search. When they overhear the search message and realized that it is initiated by i, they do not have to participate the search.

Since each node needs to send a single search message and only those nodes selected to be in the cover need to send the $< SELECT >$ message, the message complexity of the algorithm is $O(N)$ where N is the number of sensors in the network. We now formally prove the correctness of our mechanism.

Lemma 3.1: *MCC(i) Search Algorithm is correct.*

Proof:

To ease the discussion, we assume s_i is $0°$. We label the nodes according to their start angles. A node *a starts before b* if $s_a < s_b$. We also say *b starts after a*. We label i, the node that initiates the search, as n_0 and the nodes start after i as n_1, n_2, and so on where n_x starts before n_y if $x < y$. Note that apart from n_0, n_1 may have other backward neighbors. Nevertheless, n_1 does not have to receive messages from these backward neighbors before sending out its search message. We refer the backward neighbors whose search messages that a node needs to wait for as *important backward neighbors*. In general, the search messages that n_x must receive before it sends out a search message must be from the nodes in the set $\{n_0, n_1, ..., n_{x-1}\}$. Besides, n_y can never be an important backward neighbor of n_x if $y > x$.

We prove this lemma by induction. n_0 initiates the search by sending $< f(n_0), s_{n_0}, n_0 >$ to its forward neighbors. As n_0 is the only important backward neighbor of n_1, n_1 identifies the minimum cost c and sends out the search message $< c, s_{n_0}, n_0 >$ to its forward neighbors.

Assume that n_k can identify the minimum cost from n_0 to itself. It should be noted that n_j where $j < k$ should have sent out its message too. When n_k sends out the search message, n_{k+1} should have received a search message from each of its important backward neighbors since every node n_j where $j \leq k$ has sent a message. Thus, n_{k+1} can identify which important backward neighbor offers the minimum cost from n_0 to itself.

By induction, we prove that the minimum cost to reach n_{k+1} can also be determined correctly after n_k sends out its search message. Eventually, the search message will pass back to n_0, and n_0 can identify which backward neighbor should be included in the cover and no more search message is generated. Therefore, when the algorithm terminates, the cover $MCC(i)$ is identified.

□

3.4 Distributed Minimum Cost Cover Algorithm (DMCC)

One way to find the minimum cost cover is to find the cost of each $MCC(i)$ where $i \in S_0$ and identify the minimum one. The overhead would be $O(|S_0|N)$ where there are N sensors in the network. Since all nodes need to send out a search message for every individual $MCC(i)$ search, it is possible to reduce the overhead by "combining" the messages of several searches into a single message to reduce overhead. We now describe our DMCC algorithm that can identify the minimum cost cover with $O(N)$ number of messages in detail.

3.4.1 Algorithm Description

We call a node that covers 0° a *zero node*. That is, every node in S_0 is a zero node. The main idea of our algorithm is that every node $u \in S$ does not send out a search message until it can identify the minimum cost from each zero node i to itself. The search message sent by u should contain information about all $MCC(i)$. The major challenges of this idea are: Who should initiate the algorithm? How should u determine it has obtained all the necessary information to construct its search message? What should the search message look like? Can we prune the search of some $MCC(i)$? When will the algorithm terminate?

Combined Search Message Format: We first describe the search message. In the $MCC(i)$ search algorithm, each message is of format $< c, s_i, i >$. The first element represents the cost, the second and the third values represent the start angle of the node that initiates the search and the identity of the node, respectively. To carry the minimum cost information of several searches in one message, we enhance the message to $<< c_{q_1}, s_{q_1}, q_1 >, ..., < c_{q_j}, s_{q_j}, q_j >, ..., < c_{q_{|S_0|}}, s_{|S_0|}, q_{|S_0|} >>$ where c_{q_j} represents the minimum cost to reach the current node from the zero node q_j.

Message Combining Mechanism: We label the nodes in S_0 according to the ascending order of their start angles as q_1, q_2, ..., $q_{|S_0|}$. The nodes in S_0 are neighbors of each other since they all cover 0°. Therefore, by exchanging cover range messages, they can identify which node is q_1, which is q_2, etc.

The algorithm is best to be initiated by q_1. It is because if q_j, where $j \neq 1$, initiates the process, the process cannot combine the search message of $MCC(q_k), \forall k = 1, \ldots, j-1$. In this case, the number of messages to find MCC will increase as extra messages are needed to carry the search of $MCC(q_k)$, $\forall k = 1, ..., j-1$. The implication of this is that the number of messages required is minimum if the process is initiated by q_1. q_1 initiates the search by sending out $< f(q_1), s_{q_1}, q_1 >$. Based on Lemma 3.1, we know that q_2 knows the minimum cost to reach itself from q_1 after receiving the search message of q_1. At this moment, it can combine $MCC(q_1)$ and $MCC(q_2)$. The combined message becomes $<<f(q_1) + f(q_2),\ s_{q_1},\ q_1>,\ <f(q_2),\ s_{q_2},\ q_2>>$. The first tuple is used to continue the search of $MCC(q_1)$, while the second one is for searching $MCC(q_2)$. Similar process will be carried out by q_3, ..., q_k, ..., $q_{|S_0|}$ accordingly. In other words, each node $q_k \neq q_1$ has two tasks. First, it records the minimum costs from its backward neighbors in S_0 to itself. Second, it initiates the search of $MCC(q_k)$. After q_k has received all the search messages from its important backward neighbors, it sends the combined search message to its forward neighbors. For all other non-zero nodes, they only have to carry out the first task.

Pruning Mechanism: It is obvious that if the combined message contains the information of the search of every zero node, we can identify the MCC finally. However, some of this information, in fact, can be pruned without affecting the correctness of the mechanism. After n has received the search messages from all its backward neighbors, n compares the costs among them. Let $< c_{q_i}, s_{q_i}, q_i >$ and $< c_{q_j}, s_{q_j}, q_j >$ be two tuples received by n. If q_i starts earlier than q_j ($s_{q_i} \leq s_{q_j}$) and $c_{q_i} \leq c_{q_j}$, $MCC(q_j)$ would not be the only minimum cost cover and n can prune the search of $MCC(q_j)$. n then sends the combined message which contains only the unpruned searches to all its forward neighbors.

Lemma 3.2: *MCC will not be pruned by n.*

Proof:
As q_i starts earlier, the range covered by q_j to n ($[q_j, n]$) must be contained in the range from q_i to n ($[q_i, n]$). Since $c_{q_i} \leq c_{q_j}$, if we replace the sensors in $MCC(q_j)$ that cover $[q_j, n]$ by the sensors in $MCC(q_i)$ that cover $[q_i, n]$, this new cover must have a cost not larger than $MCC(q_j)$. Therefore, n does not have to continue the search of $MCC(q_j)$. □

Search Termination: Based on Lemma 3.1, we know that when i initiates the search of $MCC(i)$, the search can be terminated once the message comes back to i through a set of nodes in the clockwise direction. Recall that zero nodes $\{q_1, q_2, ..., q_{|S_0|}\}$ are sorted according to their start angles. Since our algorithm is initiated by q_1, through a set of nodes in the clockwise direction, the combined search message will eventually come back to q_1. At this moment, the cost of $MCC(q_1)$ can be determined. Instead of terminating the algorithm, q_1 has to continue all the remaining unpruned searches and inform other zero nodes the cost of $MCC(q_1)$. Again, two tasks are performed by a zero node q_k once it has received all the search messages. First, q_k terminates the search of $MCC(q_k)$, determines the cost of $MCC(q_k)$, and announces the cost to other zero nodes if $MCC(q_k)$ is unpruned. In fact, to determine whether $MCC(q_k)$ is pruned, q_k can check if the search messages received still contains the entry about $MCC(q_k)$. Second, it continues the search of any unpruned $MCC(q_l)$, where $l > k$ by sending the combined message containing the tuple $< c_{q_l} + f(q_k), s_{q_l}, q_l >$. Therefore, the whole search process can be terminated when $q_{|S_0|}$ can determine the cost

of $MCC(q_{|S_0|})$. Since all zero nodes can overhear each other, when $q_{|S_0|}$ announces the cost of $MCC(q_{|S_0|})$, all zero nodes should be able to determine which $MCC(q_i)$ is the smallest. q_i can then inform its previous neighbor by a $< SELECT >$ message as described in Section 3.3.

Example: To better illustrate the algorithm, we describe an example. Refer back to Figure 3, Node 1 is q_1 and Node 2 is $q_{|S_0|}$. Node 1 initiates the DMCC algorithm by sending $< f(1), s_1, 1 >$ to its forward neighbors, in which $f(1) = 3$ in our example. After Node 1 sends out the search message, Node 2 records the minimum cost to reach Node 2 from Node 1 as $min\_cost(1) = f(1) + f(2) = 3 + 4 = 7$. Since Node 2 is also a node in S_0, it will also initiate the search of $MCC(2)$. Therefore, Node 2 sends out $<< min\_cost(1) = 7, s_1, 1 >, < f(2) = 4, s_2, 2 >>$ to its forward neighbors. Then, when Node 3 receives the search message from Nodes 1 and 2, it records the minimum cost to reach Node 3 from Node 1 as $min\_cost(1) = f(1) + f(3) = 3 + 3 = 6$ and that of Node 2 as $min\_cost(2) = f(2) + f(3) = 4 + 3 = 7$. At this moment, Node 3 receives all the search messages from its backward neighbors. In this case, Node 3 can prune the search of $MCC(2)$ as Node 2 starts later than Node 1 and with the minimum cost larger than that of Node 1. Hence, Node 3 sends out $< min\_cost(1) = 6, s_1, 1 >$ to its forward neighbors. Then, Node 4 receives two search messages. They are $< 4, s_2, 2 >$ from Node 2 and $< 6, s_1, 1 >$ from Node 3. It records the minimum cost to reach Node 4 from Node 1 and 2 as $min\_cost(1) = 6 + f(4) = 6 + 2 = 8$ and $min\_cost(2) = 4 + f(4) = 4 + 2 = 6$, respectively. Therefore, Node 4 sends out $<< min\_cost(1) = 8, s_1, 1 >, < min\_cost(2) = 6, s_2, 2 >>$ to its forward neighbors. Afterwards, Node 5 prunes the search of $MCC(2)$ and only sends out the search message of $MCC(1)$, i.e., $< min\_cost(1) = 7, s_1, 1 >$. Node 6 sends out the combine search messages for $MCC(1)$ and $MCC(2)$, i.e., $<< min\_cost(1) = 7 + 2 = 9, s_1, 1 >, < min\_cost(2) = 6 + 2 = 8, s_2, 2 >>$.

After receiving all the necessary search messages, Node 1 can determine the cost of $MCC(1)$ is 7 through Node 5. However, to complete the search of $MCC(2)$, Node 1 sends out $< min\_cost(2) = 8 + 3 = 11, s_2, 2 >$ to Node 2 together with the cost of $MCC(1)$. Node 2 then determines the cost of $MCC(2) = 8$ through Node 6 and it informs Node 1 the cost. At this moment, the search process is terminated and Node 1 knows that $MCC(1)$ is the minimum cost cover.

In the previous discussion, we assume the zero node that starts the earliest initiates the algorithm. However, it is possible that a zero node does not have any zero node backward neighbors even it does not start the earliest. An example is Node 0 in Figure 3. In this case, this zero node should initiate the algorithm by itself.

Theorem 1: *DMCC algorithm can find MCC correctly.*

Proof:
Since our mechanism combines the search messages without changing the method of finding $MCC(i)$ as discussed in Secton 3.3, Lemma 3.1 also implies the correctness of our mechanism except the pruning technique. Therefore, to prove the correctness of DMCC, we need to prove the correctness of the pruning technique. According to Lemma 3.2, we know that MCC still exists in either one of the unpruned search of $MCC(i)$, where the search is on going. Hence, by exchanging the cost of all the unpruned $MCC(i)$, the minimum one is still MCC. Therefore, DMCC algorithm is correct. □

DMCC algorithm requires every zero node to send out two search messages, while all other non-zero nodes need to send out one. This contributes to $O(N + |S_0|)$ number of messages. Since $|S_0| \leq N$, DMCC finds MCC in $O(N)$ number of messages.

4 Adjustable Sensing Range

When describing DMCC above, we assumed sensors are using a single sensing range only. We now describe how DMCC can be enhanced to find the minimum cost cover when sensors can adjust their sensing ranges to several different levels in which a longer range will result in a higher cost.

4.1 Problem Statement

Each node can adjust its sensing range to different levels. Different nodes can have different numbers of levels and different sets of sensing ranges. Recall that S denotes the set of sensors in the network. A sensor $i \in S$ can adjust its sensing range to r. This can be represented as an instance $i.r$ in the network. The set of all possible instances of sensors which are able to cover the perimeter of the target object is denoted as S'. The cover range of an instance $i.r \in S'$ is $[s_{i.r}, t_{i.r}]$ and the cost of this instance is $f(i.r)$. Under the adjustable sensing range environment, forward and backward neighbors concept defined in Section 3.1.3 applies to the instances of nodes instead of the nodes themselves. Given a cover I, if $i.r \in I$, $i.r' \notin I$ for all possible sensing ranges $r' \neq r$ of node i. The cost of a cover I, $f(I)$, is the total cost of the instances in the cover, that is, $f(I) = \sum_{i.r \in I} f(i.r)$. Suppose D denotes the set of all possible covers. $MCC \subseteq D$, such that $\forall I \subseteq D$, $f(MCC) \leq f(I)$.

In this scenario, our goal is to develop a distributed algorithm in which a node can identify whether any of its instance is included in the MCC.

4.2 Adjustable Distributed Minimum Cost Cover algorithm (ADMCC)

One of the major differences between the adjustable sensing range scenario and the fixed sensing range

scenario is that in the fixed sensing range situation, every subset in S is a valid selection. In contrast, some subsets in S' are not allowed since at most one instance of a node can be included in the cover. Fortunately, due to the properties among instances, our DMCC algorithm can be applied in S' with little modification.

4.2.1 DMCC under adjustable sensing range

Property 1: *Suppose that r and r' are two different sensing ranges of i where $i \in S$. If $r < r'$, then $i.r \in S'$ is contained inside $i.r' \in S'$.*

Property 1 shows that the instance of node i with the smaller sensing range is contained inside the instance of the same node with the larger sensing range. This property is illustrated in Figure 4.

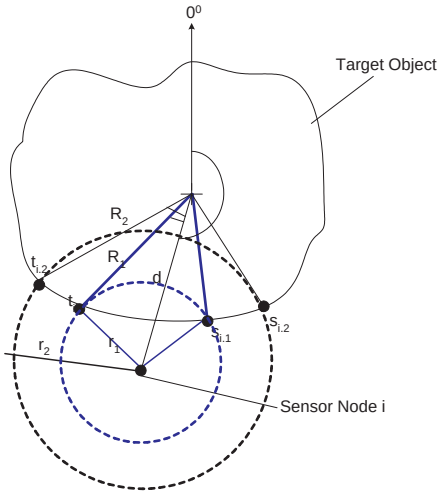

Figure 4 Adjustable Sensing Range Example

Since MCC would not contain two sensors where one is contained in the other, we have the following property.

Property 2: *An MCC of S' does not contain both $i.r$ and $i.r'$ for any $i \in S$ and two different sensing ranges r and r'.*

Due to Property 2, DMCC can still be directly applied to S' with a minor modification that a node $i \in S$ has to send and receive messages on behalf of its instances $i.r \in S'$.

4.2.2 Description of ADMCC

Unfortunately, applying DMCC to S' directly will incur $O(|S'|)$ messages. Therefore, the overhead is increased by a factor of the number of sensing levels. To reduce overhead, a node sends out a combined message for all its instances instead of sending one message for each instance.

Initiator Node: Recall that, in DMCC, the zero node that starts earliest or a zero node that does not have a zero backward neighbor should initiate the search. However, the node with the zero instance (an instance that covers 0°) in S' that starts earliest may not be the appropriate node to initiate the search if we would like to combine the message for all its instances. Consider the configuration in Figure 5. $j.r_1$ is the zero instance that starts the earliest. If j initiates the computation process, it also sends out a search message for $MCC(j.r_2)$. It is not correct since $j.r_2$ is a forward neighbor of $i.r_1$ and violates the condition in DMCC that a node should not send out a search message until it has received the search messages from all its zero backward neighbor instances. Before we describe which node should start the search, we first define several definitions.

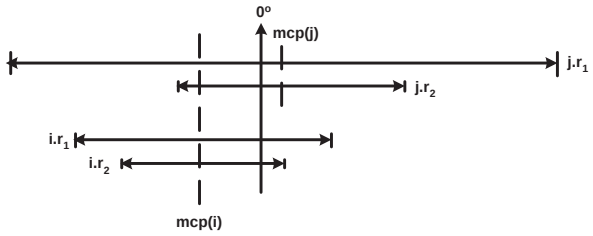

Figure 5 An example of $mcp(i) < mcp(j)$.

In the fixed range situation, we call a node *zero node* if it covers 0°. In the adjustable sensing range case, a node i is a *zero node* if there exists an instance of i that covers 0°. It is possible that not all instances of a zero node cover 0°. On the other hand, it is possible for a node i to have all its instances share a common mid-point as shown in Figure 4. Note that this assumption may not be true if the target object is in irregular shape. In this case, we only combine those instances that share a common mid-point. The *mid-cover-point* of node i, denoted as $mcp(i)$, is the center of its cover ranges in different instances. For any $i, j \in S$, we define $mcp(i) < mcp(j)$ if $mcp(i)$ is on the left of $mcp(j)$ along the circle as shown in Figure 5. We have the following property:

Property 3: *For any $i, j \in S$, if $mcp(i) < mcp(j)$, then $i.x$ would not be a forward neighbor of $j.y$ for any sensing level x and y.*

In the example shown in Figure 5 where i and j are both zero nodes, we say $mcp(i) < mcp(j)$ since $mcp(i)$ is to the left of $mcp(j)$. To show this property, suppose that there is an instance $i.x$ which is a forward neighbor of an instance $j.y$. That is, $s_{j.y} < s_{i.x}$. Since $mcp(j)$ is the mid-point of the cover range, $t_{j.y} = mcp(j) + mcp(j) - s_{j.y}$. Similarly, $t_{i.x} = 2 * mcp(i) - s_{i.x}$. Since $mcp(i) < mcp(j)$ and $s_{j.y} < s_{i.x}$, we have $t_{i.x} < t_{j.y}$, which means $i.x$ is contained in $j.y$. It leads to contradiction since $j.y$ is not a forward neighbor of $i.x$ according to our definition in Section 3.1.3.

Property 3 suggests that we should select a zero node with the smallest mid-cover-point to initiate the algorithm since the zero instances of this node would not be forward neighbors of other zero instances. We thus select node i to start the algorithm where node i satisfies:

1. i is a zero node
2. $mcp(i) < mcp(j)$ for any zero node j (Note that $x < y$ means x is to the left of y along the circle.)

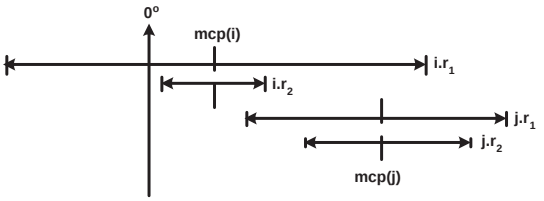

Figure 6 An initiator with both zero and non-zero instances.

We mentioned earlier that not all instances of a zero node are zero instances as well. Therefore, it is possible that the initiator node i contains an instance that does not cover 0° as shown in Figure 6. In this case, i should only carry the search information of those zero instances. In fact, if the non-zero instance of i is to the right of 0°, it can be pruned ($i.r_2$ in Figure 6). It is because a backward neighbor of this non-zero instance must cover 0°, which implies that there exist another zero node with mid-cover-point smaller than $mcp(i)$. However, this is not possible since $mcp(i)$ is the smallest. Hence, i is selected to start the algorithm.

Message Combining Mechanism: After node i sends out the search message that carries the information of $MCC(i.r)$ for every zero instance $i.r$, other nodes send out a combined search message after collecting a search message for each backward neighbor of each of its instances. We now explain there will be no deadlock in the whole procedure. That is, once the search has been started by i, every node will eventually receive enough messages from its backward neighbors and create its own search message.

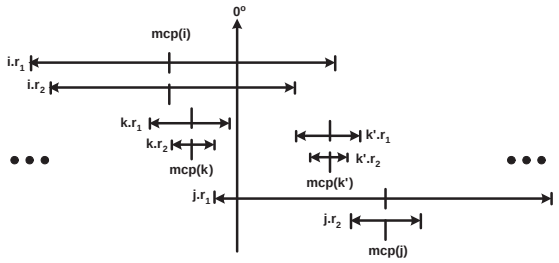

Figure 7 Order of nodes in ADMCC

Lemma 4.1: *Every node will eventually send out a search message in the ADMCC algorithm.*

Proof:
We label the nodes as $n_0, n_1, ..., n_{|S|}$. Let the initiator node be n_0. There are two cases:

1. $mcp(n_0)$ is to the right of 0°
   In this case, all the non-zero nodes k with $0 < mcp(k) < mcp(n_0)$ can be pruned. We arrange the nodes according to the mid-cover-points. That is, $mcp(n_x) < mcp(n_y)$ if $x < y$.
2. $mcp(n_0)$ is to the left of 0°
   In this situation, nodes are divided into two sets. Set A contains the non-zero nodes k with $mcp(n_0) < mcp(k) < 0°$ while $B = S \setminus A$. Note that $n_0 \in B$. We first arrange the nodes in B according to the mid-cover-point and then the nodes in A. For example, the nodes are arranged in Figure 7 according to the following order of mid-cover-points, i.e., $mcp(i)$, $mcp(k')$, $mcp(j)$, ..., $mcp(k)$.

By defining the order in this way, Property 3 ensures that the important backward neighbors of the instances of node n_x must be instances of nodes in the set $\{n_0, n_1, ..., n_{x-1}\}$.

We prove this lemma by induction. n_0 initiates the search by sending one combined search message for all its zero instances to its forward neighbors. If all the instances of n_0 are zero instances as shown in Figure 5, n_1 has received a search message from each of the important backward neighbors of its instances. If NOT all n_0's instances cover 0°, the non-zero instances are either all to the left of 0° or all to the right of 0°. If they are on the right, they have been pruned as in Figure 6 and should not be considered by n_1. If they are on the left, they are not important backward neighbors as defined in the proof of Lemma 3.1 and need not be considered as well. Therefore, we can conclude that n_1 has received all the search messages from all of its important backward neighbors and it can send out its search message.

Assume that n_k has received all the search messages on behalf of all its instances. It should be noted that n_j where $j < k$ should have sent out its combined search message too. When n_k sends out the combined search message, n_{k+1} should have received a combined search message from each of its important backward neighbors and it can send out its own search message. Therefore, there is no deadlock in the algorithm.

□

By Lemma 4.1, we know that a node i can send out one combined search message for all its instances after it has received messages from all their backward neighbors. By combining the search messages of all its instances, the number of messages can effectively be reduced to $O(N)$.

Search Termination: In ADMCC, when a zero node q_i receives all the search messages for all its instances, it can determine the cost of all unpruned $MCC(a)$, where a represents a zero instance of q_i. Instead of terminating the algorithm, it still has to continue the search of the remaining unpruned $MCC(b)$, where b is another zero instance of the zero node q_k and $mcp(q_i) < mcp(q_k)$. At the same time, q_i announces the costs of all the unpruned $MCC(a)$. The whole search process can be terminated when the zero node with the largest mid-cover-point receives search messages for

all its zero instances and then announces the costs. At this moment, the costs of all unpruned searches can be determined. By overhearing others' announcements, each zero node knows which instance is selected to form the minimum cost cover.

Search Message Format: In the combined message, we need to identify the searches of different instances of different nodes. In ADMCC, the search message format is $<< e_1, <$*combined search message of instance* $e_1>>$, $< e_2, <$*combined search message of instance* $e_2>>$, ... $>$ where e_k is an instance of the node which sends out the combined search message. The format of the combined search message of instance e_k is similar to the combined search message of the node in DMCC defined in Section 3.4.1.

Example: As shown in Figure 8, Node 1 has two zero Instances 1.1 and 1.2. Since Node 1 is the only zero node, it initiates the process of computing the minimum cost cover. Hence, Node 1 sends $<< 1.1, < f(1.1) = 5, s_{1.1}, 1.1 >>, < 1.2, < f(1.2) = 2, s_{1.2}, 1.2 >>>$ to all the forward neighbors. After receiving the search message from Node 1, Node 2 updates the minimum cost to reach Instance 2.1 from Instance 1.2 as $c\_min_{1.2,2.1} = f(1.2) + f(2.1) = 5 + 2.5 = 7.5$. It also determines the minimum cost to reach Instance 2.2 from Instance 1.1 as $c\_min_{1.1,2.2} = f(1.1) + f(2.2) = 2 + 6 = 8$ and that from Instance 1.2 as $c\_min_{1.2,2.2} = f(1.2) + f(2.2) = 5 + 6 = 11$. Since Node 2 has received the search messages of all of its instances (i.e., all the backward neighbors of its instances), Node 2 sends the combined message $<< 2.1, < c\_min_{1.2,2.1} = 7.5, s_{1.2}, 1.2 >, < 2.2 < c\_min_{1.1,2.2} = 8, s_{1.1}, 1.1 >, < c\_min_{1.2,2.2} = 11, s_{1.2}, 1.2 >>>$ to the forward neighbors.

Upon receiving the search message from Node 2, Node 3 determines the minimum cost to reach Instance 3.1 from zero Instances 1.1 and 1.2 as $c\_min_{1.1,3.1} = c\_min_{1.1,2.2} + f(3.1) = 8 + 3 = 11$ and $c\_min_{1.2,3.1} = c\_min_{1.2,2.2} + f(3.1) = 11 + 3 = 14$, respectively. Moreover, it determines the minimum cost to reach Instance 3.2 from Instances 1.1 and 1.2 as $c\_min_{1.1,3.2} = c\_min_{1.1,2.2} + f(3.2) = 8 + 8 = 16$ and $c\_min_{1.2,3.2} = c\_min_{1.2,2.2} + f(3.2) = 11 + 8 = 19$, respectively. At this moment, Node 3 has received all the search messages to make its own. It sends $<< 3.1, < c\_min_{1.1,3.1} = 11, s_{1.1}, 1.1 >, < c\_min_{1.2,3.1} = 14, s_{1.2}, 1.2 >>, < 3.2, < c\_min_{1.1,3.2} = 16, s_{1.1}, 1.1 >, < c\_min_{1.2,3.2} = 19, s_{1.2}, 1.2 >>>$ to the forward neighbors of its instances. Once Node 1 receives the combined search message from Node 3. It determines the cost of $MCC(1.2)$ is $c\_min_{1.2,3.1} = 14$, while the cost of $MCC(1.1)$ is $c\_min_{1.1,3.2} = 16$. Since Node 1 is the only zero node, the cost of MCC can be determined once the costs of $MCC(1.1)$ and $MCC(1.2)$ are determined. The one with the minimum cost is MCC which is $MCC(1.2)$.

We refer readers to (Hung and Lui (2008)) for a complete descriptions of pseudocodes and state diagrams.

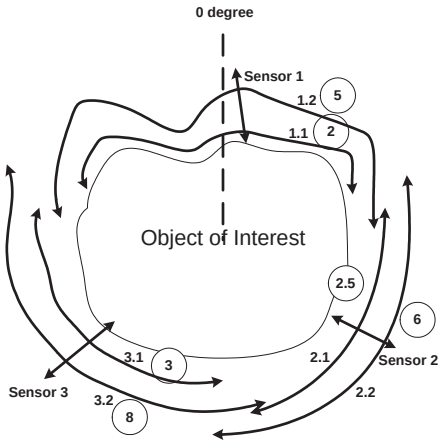

Figure 8 Adjustable Distributed Minimum Cost Cover Algorithm Example

5 Simulation

For comparison, we have implemented the algorithm described in (Chow et al. (2007a)) and our proposed algorithms (DMCC and ADMCC) in this paper. In the algorithm described in (Chow et al. (2007a)), each zero node i finds $MCC(i)$ individually and there is no combining of messages. We denote this algorithm as Exhaustive Distributed Minimum Cost Cover Algorithm (Ex-DMCC) in this literature.

5.1 Simulation Settings

Our simulation environment is similar to the one adopted in (Chow et al. (2007b)). We considered an area of 100 grids $\times$100 grids, in which each grid takes up an area of 1 unit^2 and every grid has a probability of 0.8 to contain a sensor. The average number of sensors is around 6500. The target object, which is a circle with radius of 25 units, is located at the center $(50, 50)$. A sensor can cover the portion of the perimeter of the target if that portion is within the sensing range of the sensor. We measure the performance of our mechanisms for different sensing range values. The cost value associated with each node is randomly generated from $1 - 2$. Under the adjustable sensing range scenario, we have $f(i.r) \propto r^2$ (Cardei and Du (2005); Cardei et al. (2005); Cardei et al. (2006)). We randomly generated 30 topologies for each set of results. Four performance metrics are measured and they are the total number of messages generated, the expected energy expenditure due to the transmission of messages, the minimum cost of the covers found by the algorithm, and the total time needed to find MCC.

5.2 Simulation Results

5.2.1 Fixed Sensing Range Scenario

We compare the message overheads and energy needed to find the minimum cost cover using Ex-DMCC and

DMCC to show the effect of combining messages. Figure 9 illustrates the log number of messages to find the covers with different fixed sensing ranges. The figure shows that Ex-DMCC requires a large amount of messages to find the MCC. This is because it requires each zero node q_i to look for $MCC(q_i)$ individually and $|S_0|$ can be large in some situations. On the other hand, DMCC greatly reduces the message overhead by combining the messages, so that most of the nodes only need to send out the combined search message once. Note the message overheads of DMCC with or without pruning are similar. Therefore, we show only the results with both combined messages and pruning in Figure 9. On the other hand, both algorithms show an increase in the number of messages with an increase in the sensing range. It is because when the sensing range is short, some sensors may not be able to cover the target perimeter and these sensors are not involved in the protocol. It can be observed that the overhead in Ex-DMCC increases faster than that of DMCC when the sensing range increases. This is because when nodes have a larger sensing range, more nodes would cover 0° and so $|S_0|$ is larger.

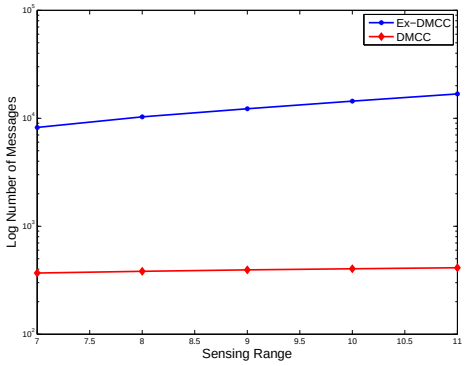

Figure 9 Log Number of Messages Vs. Sensing Range.

Other than message overheads, energy needed to find the minimum cost cover is also our concern. In fact, each Ex-DMCC message contains three tuples $< c, s_i, i >$, while each DMCC message contains several combined Ex-DMCC messages. Therefore, the size of a DMCC message can be multiple times of that of an Ex-DMCC message. Practically, c and s_i can be rounded to 1 byte integers with a certain loss of precision. i depends on the size of the network and it can be represented by an integer smaller than 2 bytes in general. Therefore, the size of an Ex-DMCC message is around 4 bytes, while that of DMCC message is around $|S_0| \times 4$ bytes if none of the search is pruned. In (Shnayder et al. (2004); Shnayder et al. (2009)), the authors suggested that CSMA should be performed before sending a message so as to avoid collision. The average CSMA time is around $41.0ms$. On the other hand, we know that transmitting with the maximum range incurs a current of $21.5mA$, while listening to the channel incurs a current of $7mA$. In other words, the transmitting and the listening operations incur a power of $64.5mW$ and $21mW$ under a $3V$ power supply. The transmission rate is around $16kbps$. The energy consumed for sending a message can be estimated by $P_t \times T_x + P_i \times I_x$, where P_t and P_i are the power required for transmitting and listening, respectively. T_x is the time durations required for transmitting the whole message and can be estimated by $\frac{message\ size}{transmission\ rate}$, while I_x is the average CSMA time. From the simulation, we know that $|S_0|$ is generally less than 50 after pruning. Therefore, a combined message contains approximately 200 bytes only. Based on the information above, the energy consumed for transmitting a 4-byte message and a 200-byte message are $1mJ$ and $7.3mJ$, respectively. Figure 10 illustrates the estimated transmission energy expenditure of Ex-DMCC, DMCC with the combining technique but without the pruning technique, and DMCC with both the combining technique and the pruning technique. The figure shows that DMCC with both the combining and the pruning techniques uses up the least amount of energy. This is mainly due to the reason that DMCC requires much fewer messages and smaller message size to find the minimum cost cover.

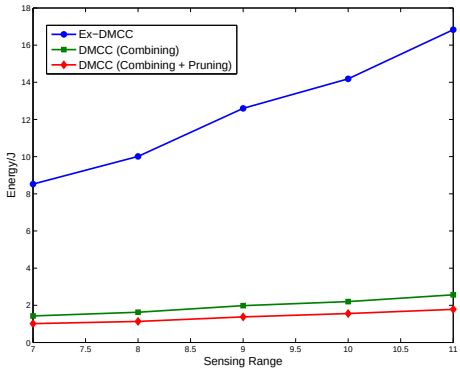

Figure 10 Estimated Transmission Energy Expenditure Vs. Sensing Range.

The total time needed to find MCC can be estimated by finding the total CSMA time and the total transmission time. These two parameters can be found by using the transmission rate, the mean CSMA time information, and the average size of a message. Note that we assume all $q_i \in S_0$ initiate the search of $MCC(q_i)$ simultaneously in the *Ex-DMCC* algorithm. Figure 11 illustrates the estimated total time needed of Ex-DMCC, DMCC with the combining technique but without the pruning technique, and DMCC with both the combining technique and the pruning technique. The figure shows that DMCC with both the combining and the pruning techniques is the fastest in finding MCC. This is mainly due to the reason that DMCC requires fewer messages (which affects the CSMA time) and smaller message size to find the minimum cost cover (which affects the transmission time).

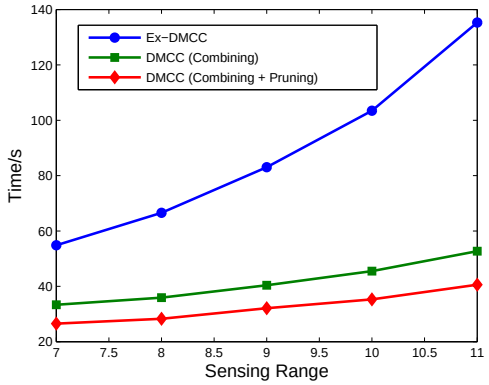

Figure 11 Estimated Total Time needed to find MCC Vs. Sensing Range.

5.2.2 Adjustable Sensing Range Scenario

In our simulations, we measure the performance of four situations which have different numbers of sensing levels. Particularly, 1SR, 2SR, 4SR, and 6SR represent one level, two levels, four levels, and six levels, respectively. We assume that there exists a maximum sensing range r_m. For xSR, the sensing range $r_i = \frac{i \times r_m}{x}$ is associated with the cost $f(i) = f(r_m) \times (\frac{r_i}{r_m})^2$, for $i = 1$ to x. This arrangement is based on the settings in (Cardei et al. (2006); Chow et al. (2007a)).

Figure 12 shows that the message overhead produced by DMCC increases as the number of the sensing levels increases. In contrast, the message overhead of ADMCC, in which a node sends out a combined message for all its instances, changes only a little. Figure 13 shows

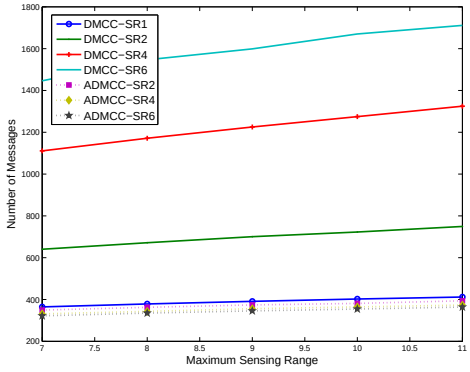

Figure 12 Number of Messages Vs. Maximum Sensing Range.

that the estimated transmission energy expenditure of both DMCC and ADMCC increases as the number of the sensing levels increases. ADMCC requires fewer messages than that of DMCC and this leads to smaller amount of transmission energy expenditure. Due to this reason, the energy expenditure differences between ADMCC and DMCC increases with the numbers of sensing levels as a lot of messages are required to find the minimum cost cover in DMCC, but nearly the same amount of messages are required to find the minimum cost cover in ADMCC.

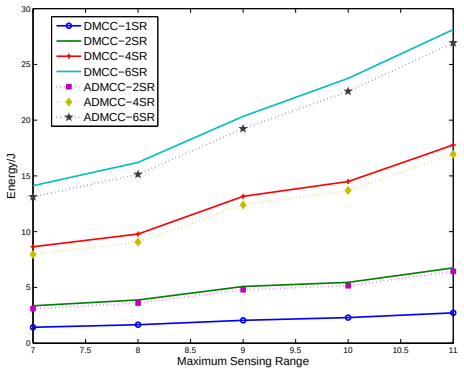

Figure 13 Estimated Transmission Energy Expenditure Vs. Sensing Range.

Figure 14 shows that the estimated total time needed of both DMCC and ADMCC increases as the number of the sensing levels increases. ADMCC requires fewer messages than that of DMCC and this leads to shorter total CSMA time. Due to this reason, the total time difference between ADMCC and DMCC increases with the numbers of sensing levels as a lot of messages are required to find the minimum cost cover in DMCC, but nearly the same amount of messages are required to find the minimum cost cover in ADMCC.

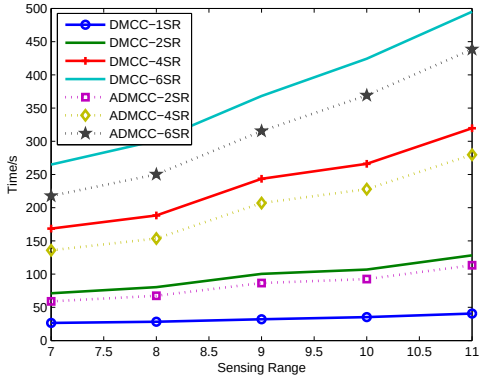

Figure 14 Estimated Total Time needed to find MCC Vs. Sensing Range.

Figure 15 illustrates the costs of the minimum cost cover of various maximum sensing range settings. As expected, the more the sensing levels, the smaller the cost of covers due to the reason that finer adjustment on sensing range can be applied to each node so as to cover the target object. However, the advantage in the cover costs becomes smaller when the number of sensing levels further increases. This is because the advantage brought by the finer adjustment of the sensing range becomes smaller and smaller.

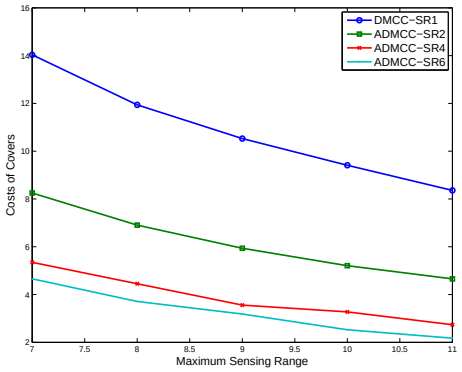

Figure 15 Cover Costs Vs. Maximum Sensing Range.

6 Conclusion

In this paper, we focus on the angle/perimeter coverage problem in which multiple sensors are expected to collaborate to monitor the whole perimeter of a large target object with the minimum cost. The algorithm is distributed in nature that when it terminates, every sensor can finally determine whether it is in the cover by communicating only with its neighbors. This coverage problem is very different from some existing coverage problems in which a certain area or point are to be monitored instead. To the best of our knowledge, we are the first to propose a distributed algorithm for finding the minimum cost cover with the communication complexity of O(size of the network). We also give the formal proof of correctness of our algorithm. Moreover, we are the first to study this problem under the adjustable sensing range scenario. Through extensive simulations, our proposed DMCC algorithm outperforms Ex-DMCC in terms of the communication overhead, and ADMCC can find a cover with smaller cost than the existing algorithms in the adjustable sensing range scenario.

References

Andrews, M. G. and Lee, D. T. (1995) 'Parallel algorithms on circular-arc graphs', *Computational Geometry – Theory and Applications*, Elsevier Science Publishers, 1995.

Atallah, M. J. and Chen, D. Z. (1989) 'An Optimal Algorithm for the Minimum Circle-Cover Problem', *Information Processing Letters*, Elsevier Science Publishers, vol. 32, 1989, pp. 159–165.

Atallah, M. J., Chen, D. Z. and Lee, D. T. (1995) 'An Optimal Algortihm for Shortest Paths on Weighted Interval and Circular-Arc Graphs, with Applications', *Algorithmica*, vol. 14, November, 1995, pp. 429–441.

Bertossi, A. A. (1988) 'Parallel Circle-Cover Algorithms', *Information Processing Letters*, Elsevier Science Publishers, vol. 27, 1988, pp. 133–139.

Cardei, M. and Wu, J. (2004) 'Coverage in Wireless Sensor Networks', *Handbook of Sensor Networks, M. Ilyas and I. Magboub (eds.)*, CRC Press, 2004.

Cardei, M. and Du, D.-Z. (2005) 'Improving wireless sensor network lifetime through power aware organization', *Wireless Networks*, Kluwer Academic Publishers, vol. 11, No. 3, May, 2005, pp. 333–340.

Cardei, M., Wu, J., Lu, M., and Pervaiz, M.O.(2005) 'Maximum Network Lifetime with Adjustable Range', *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, August, 2005.

Cardei, M., Wu, J., Lu, M. (2006) 'Improving network lifetime using sensors with adjustable sensing range', *International Journal of Sensor Networks*, 2006, pp. 41–49.

Carle, J. and Simplot-Ryl, D. (2004) 'Energy-efficient area monitoring for sensor networks', *Computer*, IEEE, Vol. 37, February, 2004, pp. 40–46.

Cao, G., Wang, G., La Porta, T., Phoha, S., Wang, G. and Zhang, W. (2004) 'Distributed algorithms for deploying mobile sensors', *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*, 2004, pp. 427–439.

Chow, K.-Y., Lui, K.-S. and Lam, Edmund Y. (2007a) 'Achieving 360 Angle Coverage with Minimum Transmission Cost in Visual Sensor Networks', *IEEE Wireless Communications and Networking Conference (WCNC)*, March, 2007, pp. 4112–4116.

Chow, K.-Y., Lui, K.-S. and Lam, Edmund Y. (2007b) 'Maximizing Angle Coverage in Visual Sensor Networks', *IEEE International Conference on Communications (ICC 2007)*, June, 2007, pp. 3516–3521.

Chow, K.-Y., Lui, K.-S. and Lam, Edmund Y. (2009) 'Wireless sensor networks scheduling for full angle coverage', *Multidimensional Systems and Signal Processing*, Vol. 20, No. 2, June, 2009, pp. 101–119.

Dhawan, A., Vu, C.T., Zelikovsky, A., Li, Y. and Prasad, S. K. (2006) 'Maximum Lifetime of Sensor Networks with Adjustable Sensing Range', *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing*, 2006.

Estrin, D., Culler, D., Pister, K. and Sukhatme, G. (2002) 'Connecting the physical world with pervasive networks', *Pervasive Computing*, IEEE, vol. 1, Jan–Mar, 2002, pp. 59–69.

Gupta, H., Zhou, Z., Das, S. R. and Gu, Q. (2003) 'Connected sensor cover: Self-organization of senor networks for efficient query execution', *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2003, pp. 198–200.

Gupta, H., Zhou, Das, S. R. and Gu, Q. (2006) 'Connected sensor cover: Self-organization of senor networks for efficient query execution', *IEEE/ACM Transactions on Networking (TON)*, Vol. 14, February, 2006, pp. 55–67.

Huang, C.-F. and Tseng, Y.-C. (2003) 'The coverage problem in a wireless sensor network', *ACM International Conference Wireless Sensor Networks and Applications (WSNA)*, 2003.

Huang, C.-F. and Tseng, Y.-C. (2005) 'The coverage problem in a wireless sensor network', *Mobile Networks and Applications*, 2005.

Hung, K.-S., Lui, K.-S. and Kwok, Y.-K. (2007) 'A Trust-based Geographical Routing Scheme in Sensor Networks', *IEEE Wireless Communications and Networking Conference (WCNC)*, March, 2007.

Hung, K.-S. and Lui, K.-S. (2008) 'On Perimeter Coverage of Wireless Sensor Networks', *Technical Report TR-2008-004*, Department of Electrical and Electronic Engineering, The University of Hong Kong, September, 2008.

Hung, K.-S. and Lui, K.-S. (2010a) 'On Perimeter Coverage of Wireless Sensor Networks', *IEEE Transactions on Wireless Communications*, To appear.

Hung, K.-S. (2010b) 'On Perimeter Coverage Issues in Wireless Sensor Networks', *PhD Thesis*, The University of Hong Kong, Feburary, 2010.

Ibarra, O. H., Wang, H. and Zheng Q. (1992) 'Minimum Cover and Single Source Shortest Path Problems for Weighted Interval Graphs and Circular-arc Graphs', *Proceedings of Allerton*, 1992.

Kar, K. and Banerjee, S. (2003) 'Node placement for connected coverage in sensor networks', *Proceedings of WiOpt: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003, pp. 14–17.

Lee, C. C. and Lee, D. T. (1984) 'On a Circle-Cover Minimization Problem', *Information Processing Letters*, Elsevier Science Publishers, Vol. 18, February, 1984, pp. 109–115.

Lee, H. and Aghajan, H. (2006) 'Vision-enabled node localization in wireless sensor networks.', *COGnitive systems with Interactive Sensors (COGIS)*, March, 2006.

McCormick, C., Laligand, P.-Y., Lee, H. and Aghajan, H. (2006) 'Distributed agent control with self-localizing wireless image sensor networks.', *COGnitive systems with Interactive Sensors (COGIS)*, March, 2006.

Olteanu, A., Xiao, Y., Wu. K. and Du, X. (2008) 'Weaving a Proper Net to Catch Large Objects.', *IEEE Globecom*, December, 2008. pp.133-137.

Osiris Photoelectric Sensors, *http://www.schneider-electric.ca/www/en/products/sensors2000/html/osiris.htm*, [Accessed 07/10/2009].

Shnayder, V., Hempstead, M., Chen, B., Allen, G. W. and Welsh, M. (2004) 'Simulating the Power Consumption of Large-Scale Sensor Network Applications', *Proceedings of the 2nd international conference on Embedded networked sensor systems (ACM Sensys)*, November, 2004. pp. 188–200.

Shnayder, V., Hempstead, M., Chen, B., Allen, G. W. and Welsh, M. (2009) 'PowerTOSSIM: Efficient Power Simulation for TinyOS Applications', *http://www.eecs.harvard.edu/ shnayder/ptossim*, [Accessed 07/10/2009].

Tezcan N. and Wang, W. (2008) 'Self-Orienting Wireless Multimedia Sensor Networks for Maximizing Multimedia Coverage', *IEEE International Conference on Communications (ICC 2008)*, May, 2008.

Thai, M. T., Li, Y., Wang, F. and Du, D.-Z. (2007) 'O(log n)-localized algorithms on the coverage problem in heterogeneous sensor networks', *IEEE International Performance Computing and Communications Conference (IPCCC 2007)*, April, 2007. pp. 85–92.

Wang, J. and Medidi, S. (2007) 'Energy Efficient Coverage with Variable Sensing Radii in Wireless Sensor Networks', *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2007.

Wu, J. and Yang, S. (2004) 'Coverage and Connectivity in Sensor Networks with Adjustable Ranges', *International Workshop on Mobile and Wireless Networking (MWN)*, August, 2004, pp. 188–200.

Ye, M., Chan, E., Chen, G. and Wu, J. (2006) 'Energy Efficient Fractional Coverage Schemes for Low Cost Wireless Sensor Networks', *26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS Workshops)*, 2006, pp. 79–79.

Yu, M. S., Chen, C. L. and Lee, R. C. T.(1989) 'Optimal parallel circle-cover and independent set algorithms for circular arc graphs', *Proc. 1989 International Conference on Parallel Processing*, 1989, pp. 126–129.

Zhou, Z., Das, S. and Gupta, H. (2004) 'Variable Radii Connected Sensor Cover in Sensor Networks', *IEEE SECON*, October, 2004.