

p -Percent Coverage Schedule in Wireless Sensor Networks

Shan Gao

Department of Computer Science
Georgia State University
Atlanta, Georgia 30303
Email: sgao@cs.gsu.edu

Xiaoming Wang

Department of Computer Science
Shaanxi Normal University
Xian, Shaanxi, China 710062
Email: wangx@snnu.edu.cn

Yingshu Li

Department of Computer Science
Georgia State University
Atlanta, Georgia 30303
Email: yli@cs.gsu.edu

Abstract—We investigate the p -percent coverage problem in this paper and propose two algorithms to prolong network lifetime based on the fact that for some applications full coverage is not necessary and different subareas of the monitored area may have different coverage requirements. The first algorithm, CPCA, is a centralized algorithm which selects the least number of nodes to monitor p -percent of the monitored area. The second algorithm, DPCP, is a distributed algorithm which can determine a set of nodes in a distributed manner to cover p -percent of the monitored area. Both of the algorithms guarantee network connectivity. The simulation results show that our algorithms can remarkably prolong network lifetime, have less than 5% un-required coverage for large networks and employ nodes fairly for most cases.

I. INTRODUCTION

In Wireless Sensor Networks (WSNs), there are some scenarios with different surveillance requirements and deployment environments. In those cases, a few observation errors, missing data, communication delay and even partial failure of the network are tolerable or can be compensated in some way. Furthermore, network lifetime is their first concern. For example, in humid or rainy seasons, the occurrence possibility of fire in a forest is quite lower than that in dry seasons. Therefore, fully monitoring the entire forest is not necessary. One may consider lower the coverage levels for the subregions where fire will not occur with high probability and keep monitoring only a few important subregions. In this way, more sensors could be turned off and network lifetime could be prolonged.

Instead of focusing on full coverage, in this paper, we investigate partial coverage problems. We define the *Extended p -percent Coverage* (EPPC) problem. A particular required coverage percentage can be assigned to each subregion of the surveillance area such that the surveillance quality can be finely controlled and unnecessary energy consumption can be ulteriorly reduced. We propose two algorithms to solve the EPPC problem, CPCA and DPCP. The centralized algorithm, CPCA, groups sensor nodes into subsets, which provide necessary coverage in turn. Coverage contribution and residual energy of each node are the decisive factor to determine in which subset a node is. Since centralized algorithms may not be able to immediately react to the changes happened in the network when it is working, we develop DPCP, a distributed

algorithm. DPCP also works on turns. In each turn, a subset of nodes are selected, according to their *worth* (Section IV-B2) value, to construct the next working set.

Traditionally, network lifetime is defined as the period from the beginning of the network to the time the area cannot be fully covered. However, for the EPPC problems, due to the special coverage requirement, a new definition to *network lifetime* is introduced in Definition 3.1.

Our contributions include defining the EPPC problem and the SSPC problem and proposing two algorithms to solve the SSPC problem. Compared with the existing p -percent coverage algorithms, our algorithms have the following improvements: 1) our algorithms guarantee that at least p -percent of the target area can be covered; 2) our algorithms have lower computation complexity (CPCA) and communication overhead (DPCP); 3) the connectivity of the network is guaranteed; 4) we provide more flexibility to users by allowing users to specify a different coverage percentage for each subregion.

The rest of this paper is organized as follows. Section II surveys the related work. Section III introduces notations and define the EPPC problem and the problem of Sensor Scheduling for p -percent Coverage (SSPC). Section IV presents our algorithms, CPCA and DPCP. The simulation results are illustrated in Section V and Section VI concludes this paper.

II. RELATED WORK

The coverage problem in WSNs have been widely studied in the past years. Existing work on the coverage problem can be classified into three categories, the target coverage problem, the area coverage problem and the breach coverage problem [1]. In [2], the authors proposed an algorithm to solve the area coverage problem whose main goal is to monitor (cover) an area instead of known targets. Depending on the requirements of the coverage, the targets or the area may need to be k -covered ($k \geq 1$). In [3], Sensor Scheduling for the k -Coverage (SSC) problem was addressed and several greedy algorithms (PCL series) were proposed to provide k -coverage and maximize network lifetime. Many efforts have been spent on the area coverage problems. Most of them focus on the full coverage problems. According to our best knowledge, there are only a few papers on the p -percent coverage problem or similar topics. In [4], the authors proposed a design of the differential

random deployment. By strategically deploying more sensor nodes to the area with higher priority, the differential random deployment can effectively lower the possibility of missing interested events, as compared to the uniform random deployment. In [5], the authors used *faces* to model the monitored area. Then they proposed a $(1 + \ln(1 - q)^{-1})$ -approximation algorithm for the case when q -percent of the monitored area is required to be covered. Nevertheless, with a larger q , the performance of their algorithm becomes quite worse. Tan proposed a hexagon-based algorithm to maximize network lifetime in [6]. The monitored area is separated into many small hexagon regions. The deployed sensors are grouped according to which hexagon region they reside. Three rules were created to select an active sensor in each hexagon region in each turn. The network connectivity is guaranteed. As a byproduct, the portion of the covered area is determined by the size of hexagons. However, the value of this portion is not controllable and fluctuates in a range. Meanwhile, the network is uniformly activated approximately which means there is no way to enhance or degrade the coverage level of a certain region.

III. PRELIMINARIES AND PROBLEM DEFINITION

For convenience, we define necessary notations here.

N	The number of deployed sensors
S	The set of deployed sensors
s_i	The i th sensor
K	The number of subsets
S_k	The k th sensor subset
A	The area need to be monitored which is separated into J subregions
J	The number of subregions
A_j	The j th subregion in A
A'_j	The extended j th subregion
L	The network lifetime
L_i	The lifetime of s_i
D_j	The network diameter in the j th subregion
p	The predefined coverage percentage for A
p_j	The predefined coverage percentage for A_j
t_k	The lifetime of S_k
$t_{i,k}$	The working time of s_i in S_k
R_i	The region covered by s_i
$R_{j,i}$	The region in A_j covered by s_i
$R_{j,i,k}$	The region in A_j covered by s_i in S_k ; if s_i is not in S_k , $R_{j,i,k}$'s area is zero.

We consider a 2-dimensional area with N sensors deployed and no two sensors are deployed at the same location. Assume only $p\%$ of the target area A need to be covered on average. In addition, users may want to set a different threshold p_j as the coverage percentage for the special regions. Figure 1 shows a typical example. In this way, users can deploy more sensors in the critical regions such that the equipment cost can be reduced and network lifetime can be prolonged under the same hardware cost. On the other hand, separating A into subregions and assigning different coverage percentage to each subregion is a good strategy for large-scale networks.

Because for the EPPC problem each subregion has different importance, users may assign weight w_j for each subregion. Therefore, the area of the subregions and their weights are the two factors determining Network End-time. We define Network End-time as follows.

Definition 3.1: Network End-time of EPPC problem: Given an area A which is divided into J subregions and the

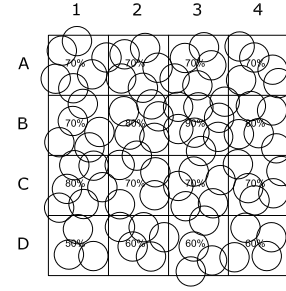


Fig. 1: Network Subregions

weight w_j assigned to the subregion A_j , network lifetime ends at the time that the percentage of weighted area of subregions whose coverage can be guaranteed by living nodes is less than the user-input f . (Represented by Formula (d) in Definition 3.3)

Definition 3.2: Extended p -Percent Coverage Problem: Given N sensors, a 2-dimensional area A which is divided into J subregions and a specific coverage percentage p_j for each subregion A_j , find a set of n sensors to cover A such that

$$p_j \| A_j \| \leq \| \bigcup_{i=1}^n R_{j,i} \|, \forall j = 1 \dots J$$

and meanwhile, n is minimized.

The EPPC problem seeks to find a subset to provide required coverage. However, considering the entire network lifetime, we have to take it into account the scheduling of sensors. Therefore, based on the EPPC problem we define the *Sensor Scheduling for p -Percent Coverage* problem as follows.

Definition 3.3: Sensor Scheduling for p -Percent Coverage (SSPC): Given a 2-dimensional area A and a set of sensors S ,

Objective: Maximize L

Subject to:

$$L = \sum_{k=1}^K t_k \quad (a)$$

$$t_k = \min(L_i), \forall work_{i,k} > 0, \forall i \in S, \forall k \in K \quad (b)$$

$$\sum_{k=1}^K (work_{i,k} \cdot t_k) \leq L_i, \forall i \in S \quad (c)$$

$$\frac{\sum_{j=1}^J \| A_j \| \cdot w_j \cdot F_{j,k}}{\sum_{j=1}^J \| A_j \| \cdot w_j} \geq f, \forall j \in J, \forall k \in K \quad (d)$$

$$F_{j,k} = \begin{cases} 1, & \text{if } \frac{\| \bigcup_{i=1}^N R_{j,i,k} \|}{\| A_j \|} \geq p_j; \\ 0, & \text{otherwise.} \end{cases} \quad (e)$$

$$work_{i,k} = \begin{cases} 1, & \text{if } s_i \text{ is in } S_k; \\ 0, & \text{otherwise.} \end{cases} \quad (f)$$

Formula (a) refers to network lifetime. Formula (b) deals with working time of S_k . Formula (c) guarantees the total working time of s_i is less than its lifetime. Formula (d) decides network end-time. Formula determines whether p_j is guaranteed in A_j .

IV. p -PERCENT COVERAGE SCHEDULING ALGORITHM

We assume there are enough sensors deployed into the monitored area A and all sensors have the same fixed sensing range R_s with different energy levels.

A. Centralized Algorithm

The basic idea of *Centralized p -Percent Coverage Algorithm (CPCA)* is to construct a node subset C iteratively. Each subregion A_j can be p_j -covered by nodes in this subset. During each iteration, a node which covers the most uncovered area in the subregion being considered is added into C .

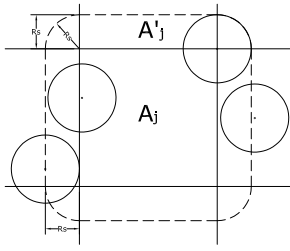


Fig. 2: Extended Subregion

Eventually, a connected node subset is constructed and can p_j -cover each subregion A_j .

The pseudo-code is shown in Algorithm 1. CS is the set of all subsets and C is the subset being constructed in the current iteration. From Line 7 to Line 15, in each iteration, CPCA checks whether A_j is p_j -percent covered. If not, a new node, which is connected with C and is the farthest neighbor from C , is chosen to be added into C . When A_j is done, CPCA tries to find the next subregion A_{j+1} to be considered. At Line 24, CPCA tries to find the next subregion by finding an *Extended Subregion* A'_{j+1} which is covered by C , yet is not p_j -percent covered by C . An Extended Subregion A'_j is an extension of A_j , as shown in Figure 2. A'_j actually contains all the nodes covering A_j and is also the maximum possible area covered by the nodes in A_j .

After several iterations, if p_j cannot be satisfied when all the nodes are included in C , this means p_j -percent coverage is impossible for this subregion. Then the status of this subregion is marked as *FAILED* (Line 19). CPCA determines the network end-time. CPCA stops when f is reached and returns CS .

If there is no *UNBUILT* subregion, it means all the subregions are either covered by C or of *FAILED* status. Then, we say the current C is successfully constructed. It is added into CS , nodes' energy information is updated and another new construction process begins from the very beginning.

After Line 25 in Algorithm 1, the node set C may not be able to cover every subregion of the surveillance area. There may exist a subregion A_j such that none of the nodes in C covers A_j . Thus, C can never span to A_j . For solving this problem, we need to actively add more nodes into C such that it can reach A_j . Algorithm 2 is designed for this purpose. It only handles one isolated *UNBUILT* subregion A_j every time. The purpose is to look for necessary nodes to build a communication path and add them into C such that C can span into A_j by executing the first part of Algorithm 1 (from Line 7 to Line 15) later.

Theorem 4.1: The time complexity of the CPCA algorithm is $O(N(N + D^2))$.

Due to the space limitation, the proof is omitted.

B. Distributed Protocol

Distributed p-Percent Coverage Protocol (DPCP) consists of 3 phases, *Discover*, *Construct* and *Connect*, as shown in Figure 3(a). Each phase is given a fixed period. The dashed line in each phase represents the actual finish time. Phase 2 and 3 have to begin at the predefined time despite the fact that their previous phases may finish earlier than their predefined

Algorithm 1 CPCA($p_j(1 \leq j \leq J, S)$)

```

1: Sort the nodes in  $S$  in non-increasing order according to their energy levels.
2: Set all subregions' statuses as UNBUILT
3:  $node = S.get(0)$ ,  $j =$  subregion where  $node$  resides
4:  $C.add(node)$ ,  $CS = null$ 
5: while  $|S| > 0$  do
6:   while  $A_j$  is UNBUILT do
7:     while  $\|R_{j,C}\| < p_j \|A_j\| \ \& \ |S| > 0$  do
8:        $node = C$ 's farthest connected neighbor in  $A'_j$ 
9:       if  $node = null$  then
10:        break
11:       else if  $node$  can increase  $R_{j,C}$  then
12:          $C.add(node)$ 
13:       end if
14:        $S.remove(node)$ 
15:     end while
16:   if  $R_{j,C} \geq p_j A_j$  then
17:     GridStatus[j] = BUILT
18:   else
19:     GridStatus[j] = FAILED
20:     if The area that can never be covered is greater than  $f|A_j|$  then
21:       return CS
22:     end if
23:   end if
24:   Find an  $A'_j$  which is covered by  $C$  but  $A_j$  is not  $p_j$ -percent covered by  $C$ .
25: end while
26: if there is an UNBUILT subregion then
27:   Connect( $C$ )
28:   Find an  $A'_j$  which is covered by  $C$  but  $A_j$  is not  $p_j$ -percent covered by  $C$ .
29: else
30:   Update lifetime of the nodes in  $C$ . If a node still has energy, add it back into  $S$ .
31:    $CS.add(C)$ ,  $C = null$ 
32:   Sort nodes and reset GridStatus as UNBUILT
33:    $node = S.get(0)$ ,  $j =$  subregion where  $node$ 
34:    $C.add(node)$ 
35: end if
36: end while
37: return CS

```

Algorithm 2 Connect(C)

```

1:  $j = C$ 's closest UNBUILT subregion
2:  $i = 0$ ,  $CN_i = C$ 
3: while  $R_{j,CN_i} = 0 \ \& \ |CN_i| > 0$  do
4:    $i++$ 
5:    $CN_i = neighborOf(CN_{i-1}) \cup \bigcup_{j=0}^{i-1} CN_j$ 
6: end while
7: if  $|CN_i| > 0$  then
8:   Choose a  $node$  in  $A'_j$  with the most energy among nodes in  $CN_i$ .
9:   Add  $node$  into  $C$ .
10:  while  $i \geq 0$  do
11:    Choose the farthest connected  $node$  to  $C$  in  $CN_i$  and add it into  $C$ .
12:     $i--$ 
13:  end while
14: else
15:   GridStatus[j] = FAILED
16: end if
17: return

```

beginning times. The tasks of the three phases are described as follows:

- 1) **Discover** Discover neighbors.
- 2) **Construct** Construct a subset of nodes to p -percent cover each subregion.
- 3) **Connect** Connect all subsets together.

In each subregion A_j , at the beginning of each turn, the following algorithm is executed to construct a subset of nodes to p_j -cover A_j . At the very beginning, a node is randomly chosen and set as the first *ROOT* node where the algorithm begins to run. The DPCP algorithm is described as follows:

- 1) Every node wakes up and sends a *HELLO* message to

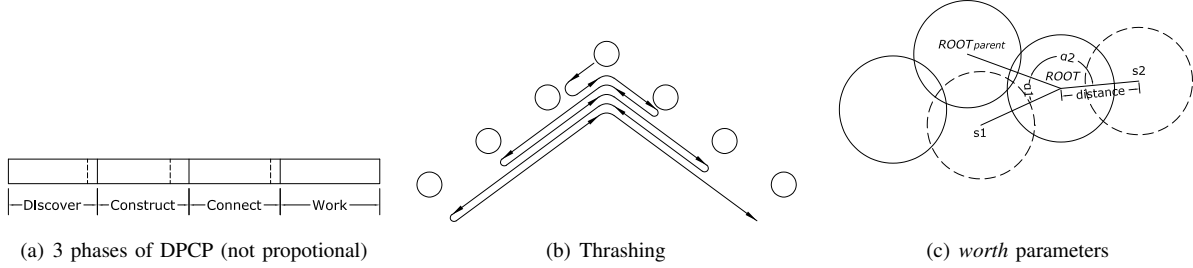


Fig. 3: DPCP, Thrashing and *worth*

all of its 1-hop neighbors. Every node will stay active for $T_{decision}$ time (see Figure 3(a)) and go to sleep if it is not involved in the next turn.

- 2) The *ROOT* node calculates the current coverage percentage cp of C as described in the Section IV-B1. If $cp \geq p_j$, a *SUCCESS* message including the information of C is sent to each node in C . Then, go to the Step 5. If not, go to Step 3.
- 3) The *ROOT* node calculates $worth_j$ for each node j in $Neighbor(ROOT) \setminus C$. The one with the highest *worth* among $Sibling(ROOT)$ and $Neighbor(ROOT) \setminus C$ is chosen as the next *ROOT* node, $ROOT_{next}$, and added into C . $Sibling(ROOT)$ is given in Step 4, and $Neighbor(ROOT)$ contains all of *ROOT*'s 1-hop neighbors. For the case where there is no $Sibling(ROOT)$ and no node in $Neighbor(ROOT)$, $ROOT_{parent}$ will be the $ROOT_{next}$.
- 4) A *BE_ROOT* message is sent from the current *ROOT* node to $ROOT_{next}$ with cp and the information of $Sibling(ROOT_{next})$ in $Neighbor(ROOT) \setminus C$, which is the node with the second highest *worth*. If the $ROOT_{next}$ is $Sibling(ROOT)$, the *BE_ROOT* message will not carry the sibling node information in order to prevent thrashing (shown in Figure 3(b)) from happening, which incurs heavy communication cost. When $ROOT_{next}$ receives a *BE_ROOT* message, it executes DPCP from Step 2 as the new *ROOT* node. If the receiver is not $ROOT_{next}$, it marks the sender as a *selected* node.
- 5) p_j is satisfied. The *ROOT* node inform each node in C broadcasts a *CONNECT* message. This message will be handled as follows.

Note: In the following description, A_k will be used to represent any *neighboring* subregion of A_j . Each message has a counter *stepCount* used to record the hop distance from the source.

- 1) If a message sent from A_j is received by a node in A_j 's C (each subregion constructs its own C), it will be dropped.
- 2) If a message sent from A_j is received by a node in A_j but not in C , the receiver will do:
 - if $stepCount < minStep(j)$, add itself into the message as a retransmitter, update $minStep(j)$ with $stepCount$, increase this message's $stepCount$ by one

and broadcast it. $minStep(j)$ means the minimum number of hops from the current node to A_j 's C .

- if $stepCount \geq minStep(j)$, drop it.
- 3) If a message sent from A_j is received by a node which is not in A_j and will not be active in the next turn, the receiver will do:
 - if A_j is not a neighboring subregion of the subregion in which the receiver is, or the message is not from A_j directly, then drop it.
 - if $stepCount < minStep(j)$, add itself into the message as a retransmitter, update $minStep(j)$ with $stepCount$, increase this message's $stepCount$ by one and broadcast it.
 - if $stepCount \geq minStep(j)$, drop it.
 - 4) If a message sent from A_j is received by a node which is not in A_j and will be active in the next turn, the receiver will do:
 - if A_j is not a neighboring subregion of the subregion in which the receiver is, or the message is not from A_j directly, then drop it.
 - if $stepCount < minStep(j)$, add itself into the message as a retransmitter, update message's *timestamp* with the receiving time, keep this message as $CMsg(j)$ and update $minStep(j)$ with $stepCount$.
 - if $stepCount \geq minStep(j)$, drop it.

When a node in A_k 's C has received m *CONNECT* messages from A_j , where m is a predefined threshold, it will send $CMsg(j)$ to all other nodes in this C . Upon receiving this message, the receiver need to immediately stop accepting *CONNECT* messages coming from A_j even if it has not received m *CONNECT* messages from A_j , and send its own $CMsg(j)$ to others. Eventually, each node in C will have multiple $CMsg(j)$ from other members of C . Only the one with the minimum $stepCount$ and the earliest *timestamp* will be chosen and the route information in that message will be used which means all retransmitters of this message will be added into both C s of A_j and A_k . A *CONNECT_PATH* message duplicated from the original *CONNECT* message will be sent to A_j from the original receiver. All nodes contained in the message's routing information will be notified to be involved in each region's C . The final receiver in A_j is the original sender of the corresponding *CONNECT* message, which is responsible for notifying other nodes in C about the new nodes for connecting with A_k .

1) *Localized Calculation of Coverage Percentage:* The energy consumption for radio communication is the major part of energy consumption in WSNs. Hence, we need to reduce the communication overhead on transmitting global information. Transmitting all nodes's data in C is too expensive. However, for calculating the area of covered region, C should be known by each new *ROOT* node in advance. The information of each node in C has to be transmitted to each new *ROOT* node during the execution of DPCP in order to calculate cp and decide whether more nodes are needed to cover more area.

We assume that the communication range is twice longer than the sensing range. Our idea is to have each node (not only those who will be/is *ROOT* node) know what happened to their neighbors. When a node received a *BE_ROOT* message which implies the sender is selected to be one member of C , it would mark this neighbor as a *selected* node. When it is also selected, it can know which neighbor is in C and how much area is only covered by itself. Therefore, cp can be resolved locally and the global information is not necessary.

2) *Calculation of worth:* DPCP uses *worth* to evaluate the contribution of each node in order to choose the best candidate to construct C . To evaluate a candidate node s_i , many factors need to be taken into account. Intuitively, the area covered only by s_i is one of the most important criteria. But, there are some other factors we need to consider.

Calculating covered area at each time a new *ROOT* node is selected results in very high overhead on computation and communication. Therefore, the method we use is to choose another parameter to approximately describe the coverage of s_i . The one we choose is the *distance* between the *ROOT* node and s_i . The larger the distance is, the more s_i can cover, intuitively. Nevertheless, the distance cannot represent the fact that a part of the covered area it represents might be also covered by ancestors of the current *ROOT* node. As shown in Figure 3(c), the dashed circles, s_1 and s_2 , are candidates. s_1 is farther from the *ROOT* node than s_2 and covers more area than s_2 does. However, the actual contribution of s_1 is less than that of s_2 because most area covered by s_1 is also covered by other nodes in C . Therefore, α is involved into the evaluation. A larger α means the node is farther from C and can provide more contribution on coverage in most cases. Additionally, the residual energy of each node should also be considered. Finally, we use three parameters to evaluate candidate nodes. *worth* is calculated by Formula (1).

$$worth(d, \alpha, e) = \lambda \min\left(\frac{d}{R_{s_i} + R_{s_j}}, 1\right) + \mu \frac{\alpha}{\pi} + \frac{e}{E} \quad (1)$$

λ and μ can be used to tune the weight of each of three parameters in Formula (1). Intuitively, a higher λ means with lower possibility of overlapping *distance* is the first concern. Similarly, users can use a higher μ to show that straight-line-like communication paths are preferred. In sparse networks, overlapping among ancestors and children seldom happens. So, the influence of α can be ignored. Nevertheless, in dense networks, a larger α results in less overlapping which means larger contribution on cp .

	CPCA and DPCP		GS
Deployment Parameters	Proportional Deployment f=50%, $\lambda = 1$, $\alpha = 1$		Density Control k = 1
	$p_{avg} = 0.4$	$p_{avg} = 0.8$	
p_0	0.3	0.3	
p_1	0.8	0.8	
p_2	0.9	0.9	
p_3	0.6	0.6	
p_4	0.2	0.92	
p_5	0.2	0.92	
p_6	0.2	0.92	
p_7	0.2	0.92	
p_8	0.2	0.92	
Std Dev	0.287	0.216	

TABLE I: Simulation Configuration

3) *Algorithm Performance:*

Theorem 4.2: The message complexity of DPCP for a subregion A_j is $2n_j + 3|C_j| + \min(8D_j, n_j) - 4$, where n_j is the number of nodes in A_j , D_j is the network diameter in A_j and C_j is the constructed working set in A_j .

Due to the space limitation, the proof is omitted.

V. SIMULATIONS

The surveillance area is a square and is separated into 3×3 subregions. Each subregion's size ranges from twice to 10 times of the sensing range. Nodes are proportionally deployed into each subregion according to p_j (*Proportional Deployment*). Each node has the same sensing range, $50m$, and the same communication range, $200m$. All simulations are repeated for 50 times.

A. Network Lifetime

Firstly, network lifetime is our primary concern. In our simulations, $f = 50\%$ and each region's weight is 1.

To determine the improvement, we compare CPCA and DPCP with GS [3], which is a k -coverage algorithm and generates close-to-optimal solutions. Using the same nodes deployment is not meaningful because the advantages of three algorithms cannot be shown. Thus, to show the best performance of GS, we set $k = 1$ and apply Density Control technique [3] to deploy nodes more reasonably in order to reduce the number of *unused nodes*. The simulation configuration is shown in Table I. Two simulation cases are designed. p_{avg} denotes the average coverage percentage of the whole area. The simulation results are shown in Figure 4. In Figure 4(a), we deployed 800 nodes. GS cannot construct one subset to cover the whole area even with all nodes when the size of subregion is larger than four times of R_s . This result once again shows that, for some applications without strict surveillance requirements, p -percent coverage is a quite good solution indeed. When $SS/R_s = 2$, where SS is the size of subregion, network lifetime of CPCA-0.4($p_{avg} = 0.4$), DPCP-0.4, CPCA-0.8($p_{avg} = 0.8$) and DPCP-0.8 are 7.6 times, 6.11 times, 4.12 times and 2.75 times longer than that of GS($k = 1$), respectively. Additionally, lifetime for $p_{avg} = 0.4$ is twice of that for $p_{avg} = 0.8$.

Considering CPCA-0.4 as an example, $SS/R_s = 2, 3$ and 4 , network lifetime drops sharply. The reason is SS/R_s only shows the linear relationship between the subregion's size and the sensing range. However, The relationship between the

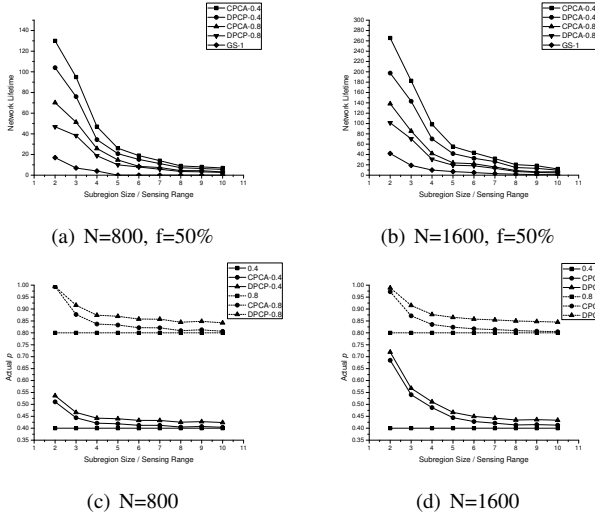


Fig. 4: Network Lifetime and Coverage Accuracy

subregion's area and the sensing range is quadric. The area increases 125% when SS/R_s changes from 2 to 3. And the network lifetime of CPCA-0.4 drops from 130 to 95, which is 73% of the original one. Theoretically, network lifetime should drop to 57.8, which is 44% of 130. However, because when $SS/R_s = 2$ the sensing range is relatively too large for the size of subregions. The overlapped area between nodes in different subregions is quite a lot. Thus, network lifetime is shorter than the theoretical value, and the loss, due to the increase of the area size, is not much. Nevertheless, when SS/R_s changes from 3 to 4, network lifetime drops a lot. The area of each subregion increases 77.8% and the theoretical loss on the network lifetime should be 43.8% which is close to our result, 50.5%.

B. Deviation from User Requirements

The goal of CPCA and DPCP is to cover p -percent of the monitored area. Denote p' as the actual coverage percentage obtained by algorithms and p as the user required percentage. Intuitively, larger p' induces energy waste, and lower p' means the coverage requirement is not satisfied. Therefore, the *deviation* of p' from p should be as small as possible. Meanwhile, p' should be no less than p .

The simulation setting is the same with the one in Section V-A except that $f = 100\%$ and we only compare results of CPCA and DPCP with user requirements. If $f = 50\%$, the global coverage percentage might be lower than 50% before the network is dead. According to the Definition 3.1, this is reasonable. Two sets of simulations with 800 nodes and 1600 nodes are executed, respectively. The results are shown in Figure 4.

In Figure 4(c), when $SS/R_s = 2$ or 3, p' is far from user requirements. The reason is that when SS is not much larger than the sensing range, one extra node newly added into C can result in a large change on p' . The covered area cannot be finely controlled when SS is relatively small comparing with R_s . With the increase of the subregion size, p' drops very soon. When SS/R_s is greater than 5, p' is very close to p . DPCP's

results are greater than CPCA's. The first reason is that, for reducing communication cost, the area in each subregion covered by nodes in neighboring subregions is ignored. Hence, when a ROOT node finds that p is reached, p' is greater actually. The second reason is that, for connecting with nodes in other subregions, redundant nodes have to be added into C . Additionally, in fact, DPCP provides more robust connection than CPCA does because the former tries to build connections with all neighboring subregions. However, when p is relatively large, the nodes for connection can also provide contribution on coverage such that the over-covered area becomes trivial.

C. Balance

We also conduct simulations to evaluate the balance of energy consumptions of nodes. Due to the space limitation, we only present the results in statements rather than figures. The *balance* is defined as the standard deviation of the percentage of residual energy in each subregion. It shows that larger differences among subregions' coverage requirements result in worse balance. DPCP constructs more robust connectivity with neighboring subregions than CPCA does. Hence, in each subregion, more nodes are used to build communication paths. The cases that nodes in a subregion are never or seldom used will not happen in DPCP.

VI. CONCLUSION AND FUTURE WORK

We investigate the EPPC problem in this paper and propose CPCA and DPCP to solve the SSPC problem, which extends the EPPC problem by taking into account the scheduling issue. The simulation results show that our algorithms can remarkably prolong network lifetime compared with traditional full coverage algorithms. The obtained coverage levels have small deviation from user requirements. They can also make use of nodes fairly for most cases.

VII. ACKNOWLEDGMENT

This work is supported in part by the NSF CAREER Award under Grant No. CCF-0545667 and the National Science Foundation of China under Grant No.60773224.

REFERENCES

- [1] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE INFOCOM 2005*, 2005.
- [2] Z. Zhou, S. Das, and H. Gupta, "Connected k-coverage problem in sensor networks," in *Proceedings of the International Conference on Computer Communications and Networks*, 2004.
- [3] S. Gao, C. Vu, and Y. Li, "Sensor scheduling for k-coverage in wireless sensor networks," in *2nd International Conference on Mobile Ad-hoc and Sensor Networks*, 2006.
- [4] K. Xu, H. Hassanein, G. Takahara, and Q. Wang, "Differential random deployment for sensing coverage in wireless sensor networks," in *Global Telecommunications Conference*, 2006.
- [5] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Efficient energy management in sensor networks," in *Ad Hoc and Sensor Networks*, ser. Wireless Networks and Mobile Computing, Y. P. Y. Xiao, Ed., vol. 2. Nova Science Publishers, 2005.
- [6] H. Tan, "Maximizing network lifetime in energy-constrained wireless sensor network," in *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, 2006, pp. 1091–1096.