

Cover Set Problem in Directional Sensor Networks

Yanli Cai^{*†}, Wei Lou^{*}, Minglu Li[†]

^{*}Department of Computing

The Hong Kong Polytechnic University, Hong Kong

[†]Department of Computer Science and Engineering

Shanghai Jiao Tong University, China

{cai-yanli, li-ml}@cs.sjtu.edu.cn, csweilou@comp.polyu.edu.hk

Abstract—A directional sensor network consists of a number of *directional sensors*, which can switch to several directions to extend their sensing ability to cover the interested targets in a given area. Because a directional sensor has a smaller angle of sensing range or even does not cover any target when it is deployed, how to cover the interested targets becomes a major problem in directional sensor networks. In this paper, we address the *directional cover set problem (DCS)* of finding a cover set in a directional sensor network, in which the directions cover all the targets. We propose both centralized and distributed algorithms for the DCS. We also introduce two applications that utilize these algorithms to extend the network work time while maximizing the coverage of the targets. Simulation results are presented to demonstrate the performance of these algorithms and the applications.

I. INTRODUCTION

A directional sensor network is a set of *directional sensor* nodes deployed to monitor a given area or some interested targets [1], [2]. In contrast to an *omni-directional sensor* that has an omni-angle of sensing range, a directional sensor has a limited angle of sensing range due to technical constraints or cost considerations, such as video sensors [3], ultrasonic sensors [4] and infrared sensors [5]. To extend the sensing ability of directional sensors to monitor their omni-angle vicinity, several ways can be adopted. One way is to put several directional sensors of the same kind on one sensor node, each of which faces to a different direction. Another way is to place the sensor node onto a mobile device so that the node can move around. The third way is to equip the sensor node with a device that enables the sensor node to switch or rotate to different directions. We adopt the third way in this paper so that a sensor can face to several directions.

For simplicity, the following assumptions and notations are adopted in this paper. In the directional sensor model, the sensing region of each direction of a directional sensor is a sector of the sensing disk centered at the sensor with a sensing radius. When the sensors are randomly deployed, each sensor initially faces to one of its directions. Some interested targets with known locations are deployed in a two-dimensional Euclidean plane. Each sensor node equips exactly one sensor on it. Therefore, the terms sensor and node are not differentiated in this paper. A number of directional sensors are randomly scattered close to these targets. If a directional sensor faces to a direction, we say that the sensor works in

this direction and the direction is the work direction of the sensor. When this sensor works in a direction and a target is in the sensing region of the sensor, we say that the direction of the sensor covers the target.

How to monitor or cover the interested targets in directional sensor networks is a challenging problem, which is also a *coverage* problem. This is because a directional sensor has a smaller angle of sensing range than an omni-directional sensor or even does not cover any target when it is deployed. We call a subset of directions of the sensors as a *cover set*, in which the directions cover all the targets. Because no more than one direction of the same sensor can work at the same time, they cannot be in a cover set. We call the problem of finding a cover set in a directional sensor network as the *directional cover set problem (DCS)*.

We propose a centralized algorithm and a distributed algorithm for the DCS that schedule directional sensors to face to certain directions to cover the targets. In the proposed algorithms, we do not put restrictions on the number of selected sensors. Enhanced algorithms can be proposed based on these algorithms for different optimization purposes such as power conservation or interference avoidance. We introduce two applications that utilize the proposed algorithms to extend the network work time while maximizing the coverage of the targets.

The rest of the paper is organized as follows: Section II briefly surveys the related work in the literature. In Section III, the definition of the DCS is given. In Section IV, we describe our algorithms for the DCS. In Section V, we describe two applications that use these algorithms. Simulation results are presented to show the performance of the proposed algorithms in Section VI. We conclude the paper in Section VII.

II. RELATED WORK

The coverage of a sensor network represents the quality of monitoring. According to the objectives to be monitored, different coverage models have been studied in both omni-directional and directional sensor networks.

The area coverage represents the fraction of the region that is covered by sensors [6], [7], [8]. In [9], [10], [11], both area coverage and communication connectivity are considered for omni-directional sensor networks. If the communication radius is at least twice of the sensing radius, complete area coverage

of a convex region implies communication connectivity among the active sensors [9], [10].

The target coverage problem in omni-directional sensor networks is also studied in literature when a set of targets are deployed in a region [12], [13], [14]. [12] assumes that a sensor can watch only one target at a time, and builds a target watching timetable for each sensor to maximize the network lifetime. Sensors are organized into subsets that are activated successively to extend the network lifetime [13], [14].

The coverage problem in directional sensor networks has attracted intense interest recently. In [1], the authors present a directional sensor model where each sensor is fixed to one direction and analyze the probability of the full area coverage. In [2], [15], a directional sensor model where a sensor is allowed to work in several directions is proposed. In [2], heuristic and distributed algorithms are also proposed to find a minimal set of directions, in which the directions cover the maximal number of targets. The problem of finding non-disjoint cover sets and allocating the work time for each of them to maximize the network lifetime is discussed in [15]. To solve this problem, three heuristic algorithms based on Linear Programming are proposed and evaluated in [15].

III. DIRECTIONAL COVER SET PROBLEM

We use the following notations and definitions throughout the paper.

- M : the number of targets.
- N : the number of sensors.
- W : the number of directions per sensor.
- a_m : the m^{th} target, $1 \leq m \leq M$.
- s_i : the i^{th} sensor, $1 \leq i \leq N$.
- $d_{i,j}$: the j^{th} direction of the i^{th} sensor, $1 \leq i \leq N$, $1 \leq j \leq W$. We define $d_{i,j} = \{a_m | a_m \text{ is covered by } d_{i,j}, \forall a_m \in A\}$ and $s_i = \{d_{i,j} | j = 1 \dots W\}$. Hence, if $a_m \in d_{i,j}$, a_m is covered by $d_{i,j}$.
- A : the set of targets. $A = \{a_1, a_2, \dots, a_M\}$.
- S : the set of sensors. $S = \{s_1, s_2, \dots, s_N\}$.
- D : the set of the directions of all the sensors. $D = \{d_{i,j} | i = 1 \dots N, j = 1 \dots W\}$. Notice that $\bigcup_{i=1}^N s_i$ is a non-overlapped partition of D .

Definition 1. Cover Set: Given a collection D of subsets of a finite set A and a partition S of D , a *cover set* for A is a subset $D' \subseteq D$ such that every element in A belongs to at least one member of D' and every two elements in D' cannot belong to the same member of S .

Definition 2. Directional Cover Set Problem (DCS): Given a collection D of subsets of a finite set A and a partition S of D , find a cover set for A .

The DCS has been proven to be NP-complete by reduction from the 3-CNF-SAT problem [15].

IV. SOLUTIONS TO DCS

In this section, we first present a centralized greedy algorithm named *DCS-Greedy* that has high possibility to find a

cover set, considering that the DCS is NP-complete. Then, we propose a distributed algorithm named *DCS-Dist*. We will show in Section V how to utilize these algorithms to extend the network work time while maximizing the coverage of the targets.

A. DCS-Greedy Algorithm

In this subsection, we propose a greedy algorithm named *DCS-Greedy*. Given a directional sensor network with a set A of M targets, a set S of N sensors and a set D of directions. We define a tuple $G = (D_G, A_G)$, where A_G is the set of targets and $A_G = A$ initially, and D_G is the set of directions that cover at least one target in A_G , i.e., $D_G = \{d_{i,j} | a_m \in d_{i,j}, \exists a_m \in A, \forall d_{i,j} \in D\}$. If more than one direction of a sensor is in D_G , we say that these directions *conflict* with each other and are *conflicting directions*. Otherwise, if only one direction of the sensor is in D_G , we say that this direction is a *non-conflicting direction*. For example, the directions $d_{i,j}$ and $d_{i,j'}$ of the same sensor s_i conflict with each other if they are both in D_G . We need to select a set of non-conflicting directions from D_G to be a cover set. We denote D_s as such a selected set of directions.

In this algorithm, when selecting directions from D_G to D_s , we consider two cases, Case 1 and Case 2. For either case, we specify a *pivot policy* to pick a direction among the candidate directions in D_G that satisfy this case. The pivot policy we use here is to find a direction to cover the target that can be covered by minimal number of directions. Other pivot policies can also be adopted according to specific application requirements.

Case 1. Each target in A_G is covered by at least one direction in D_G and there exist non-conflicting directions in D_G .

We handle this case as the following to select non-conflicting directions into D_s . Pick a non-conflicting direction d_{i^*,j^*} in D_G using the pivot policy. We denote U as the set of the targets in A_G that are covered by d_{i^*,j^*} . Remove the targets in U from A_G . After the targets in U are removed from A_G , there are some directions in D_G that cover no targets in the current A_G , including the direction d_{i^*,j^*} . We denote the set of these directions as V . Remove the directions in V from D_G . If a direction $d_{i,j}$ in V conflicts with the directions neither in D_s nor in D_G , we add $d_{i,j}$ to D_s . Remove $d_{i,j}$ from V and repeat to select a new direction from V into D_s until the remaining directions in V conflict with the directions either in D_s or D_G .

Case 2. Each target in A_G is covered by at least one direction in D_G and no non-conflicting direction exists in D_G .

We handle this case as the following to select a direction and remove its conflicting directions from D_G . Apply the pivot policy to select a direction d_{i^*,j^*} . Remove the directions that conflict with d_{i^*,j^*} from D_G .

The *DCS-Greedy* algorithm works as follows: Firstly, while Case 1 is satisfiable, repeat to use the pivot policy to select non-conflicting directions into D_s . Secondly, if Case 2 is satisfiable, use the pivot policy to select one direction and

remove its conflicting directions. Repeat to handle Case 1 and Case 2 and get a set of directions D_s . When A_G is empty, the algorithm succeeds to find a cover set D_s . When D_G is empty but A_G is not empty, it fails to find a cover set, but it still returns D_s , which is a set of non-conflicting directions, as the work directions of the sensors.

The *DCS-Greedy* algorithm is shown below:

DCS-Greedy algorithm

```

1:  $A_G = A$ ,  $D_G = \{d_{i,j} \mid a_m \in d_{i,j}, \exists a_m \in A, \forall d_{i,j} \in D\}$ ,
    $G = (D_G, A_G)$ ,  $D_s = \emptyset$ 
2: while  $D_G \neq \emptyset$ 
3:   while Case 1 is satisfiable
4:     Pick a non-conflicting direction  $d_{i^*,j^*}$  in  $D_G$  using
     the pivot policy
5:      $U = \{a_m \mid a_m \in d_{i^*,j^*}, \forall a_m \in A_G\}$ ,  $A_G = A_G - U$ 
6:      $V = D_G - \{d_{i,j} \mid a_m \in d_{i,j}, \exists a_m \in A_G, \forall d_{i,j} \in
     D_G\}$ ,  $D_G = D_G - V$ 
7:     for each  $d_{i,j} \in V$ 
8:       if  $d_{i,j}$  conflicts with the directions neither in  $D_s$ 
       nor  $D_G$ 
9:          $D_s = D_s \cup \{d_{i,j}\}$ 
10:      end if
11:    end for
12:  end while
13:  if  $D_G \neq \emptyset$ 
14:    if Case 2 is satisfiable
15:      Pick  $d_{i^*,j^*}$  in  $D_G$  using the pivot policy
16:       $D_G = D_G - \{d_{i^*,j} \mid j \neq j^*, \forall d_{i^*,j} \in D_G\}$ 
17:    end if
18:  end if
19: end while
20: if  $A_G \neq \emptyset$ 
21:    $D_s = \emptyset$ 
22: end if
23: return  $D_s$ 

```

B. DCS-Dist Algorithm

In this subsection, we propose a distributed algorithm called *DCS-Dist* to the DCS for the applications when centralized algorithms are inapplicable. In this algorithm, a sensor only cooperates with its neighbors in its communication range. There are two stages in this algorithm, the deployment stage and the decision stage.

In the deployment stage, each sensor scans the targets that its directions can cover and assigns a priority to each target locally. The fewer times a target can be covered by the directions of a sensor and its neighbors, the higher priority it is assigned to. Initially, each sensor is in the active state. First, each sensor s_i scans the environment to detect the targets, denoted by $M_{i,j}$, that can be covered by each of its directions $d_{i,j}$. Sensor s_i maintains the sets of $M_{i,j}$, for $j = 1 \dots W$ locally. Then, s_i broadcasts a message indicating each of its directions $d_{i,j}$ and the targets $M_{i,j}$ that it can cover to its neighbors, denoted as N_i . After waiting for a period for the broadcasted messages of its neighbors, s_i assigns a priority p_m to each target a_m in $\bigcup_{j=1}^W M_{i,j}$, where $p_m = 1/T_m$ and

$T_m = 1 + |\{d_{i',j} \mid a_m \in d_{i',j}, \forall d_{i',j} \in s_{i'}, s_{i'} \in N_i\}|$. The variable T_m indicates how many times a target a_m in $\bigcup_{j=1}^W M_{i,j}$ can be covered by the directions of s_i and its neighbors.

After each sensor has assigned the priority to the targets that its directions can cover, it shifts to the decision stage. In this stage, a sensor probes the states of its neighbors and decides its work direction. First, each sensor s_i initializes a timer T_p as a value uniformly distributed in $[0, \delta_p]$ and goes to sleep. When the timer T_p decreases to zero, s_i wakes up and marks itself PREWORK. Note that the sensor in the PREWORK state does not respond to its neighbors. Then, s_i broadcasts a probing message and waits for a period for its neighbors' replies. On receiving the message, any active neighbor but not in the PREWORK state responds to s_i with a message indicating its work direction. At last, s_i makes a decision based on its neighbors' replies. If it finds out that its directions can cover some uncovered targets, it erases the PREWORK mark, and works in the direction that covers the uncovered target a_{m^*} with the highest priority p_{m^*} ; otherwise, it can simply go to sleep.

The *DCS-Dist* algorithm is shown below:

DCS-Dist Algorithm

```

1: Each sensor  $s_i$  detects the targets  $M_{i,j}$  that can be covered
   by each of its directions  $d_{i,j}$ , for  $j = 1 \dots W$ 
2:  $s_i$  broadcasts a message including  $d_{i,j}$  and  $M_{i,j}$ , for  $j =
   1 \dots W$ , to its neighbors  $N_i$ 
3:  $s_i$  waits for a period for the broadcasted messages of its
   neighbors
4: for each  $a_m \in \bigcup_{j=1}^W M_{i,j}$ 
5:    $s_i$  assigns a priority  $p_m = 1/T_m$  to  $a_m$ , where  $T_m =
   1 + |\{d_{i',j} \mid a_m \in d_{i',j}, \forall d_{i',j} \in s_{i'}, s_{i'} \in N_i\}|$ 
6: end for
7:  $s_i$  initializes the timer  $T_p$  and goes to sleep
8: if  $T_p \leq 0$ 
9:    $s_i$  wakes up and marks itself PREWORK
10:   $s_i$  broadcasts a probing message and waits for a period
   for replies
11:  for each  $s_{i'} \in N_i$ 
12:    if  $s_{i'}$  is active but not in the PREWORK state
13:       $s_{i'}$  responds to  $s_i$  to indicate its work direction
14:    end if
15:  end for
16:  if  $\exists a_m \in M_i$  that  $a_m$  is uncovered
17:     $s_i$  erases the PREWORK mark
18:     $s_i$  works in the direction that covers the uncovered
    target  $a_{m^*}$  with the highest priority  $p_{m^*}$ 
19:  else
20:     $s_i$  goes to sleep
21:  end if
22: end if

```

The time complexity of the *DCS-Greedy* algorithm is $O(N^2WM)$. The time complexity of the *DCS-Dist* algorithm is $O(NWM)$.

V. APPLICATIONS BASED ON ALGORITHMS FOR DCS

In this section, we show two applications that are based on the *DCS-Greedy* algorithm and the *DCS-Dist* algorithm respectively to extend the network work time while maximizing the coverage of the targets. We denote the lifetime of a sensor s_i as L_i , which is the time duration when the sensor is in the active state all the time. For simplicity, we assume each sensor initially has an equal lifetime.

Firstly, we briefly describe an application called *WT-Greedy* that is based on the *DCS-Greedy* algorithm. In each iteration of this application, using the *DCS-Greedy* algorithm, we compute at most one set of non-conflicting directions as the work directions of the sensors. We set the work time of each cover set as a fixed value Δt , which is determined according to application requirements. The residual lifetime of any selected sensor s_i is updated, i.e., $L_i = L_i - \Delta t$. The pivot policy when finding a set of non-conflicting directions here takes into consideration the residual lifetime of each sensor and works as follows: First, we find the uncovered target a_{m^*} that can be covered by minimal number of directions. Then, we find the sensor s_{i^*} with the longest residual lifetime among all the candidates whose directions can cover a_{m^*} . At last, the direction d_{i^*,j^*} of s_{i^*} that can cover a_{m^*} is selected. Repeat to compute the sets of non-conflicting directions until the residual lifetimes of all the sensors are less than Δt . These sets of non-conflicting directions work one after another for Δt each, according to the order when they are generated.

Secondly, we introduce the other application called *WT-Dist* based on the *DCS-Dist* algorithm. This application has two stages, the deploying stage and the monitoring stage. The deploying stage of this application works the same as the *DCS-Dist* algorithm. In the monitoring stage, sensors work in rounds. Each round lasts for a period of Δt , which is the same as the one in the *WT-Greedy*. At the beginning of one round, a sensor probes the states of its neighbors and decides its work direction the same as the decision stage of the *DCS-Dist* algorithm. For simplicity, we assume the duration of the decision stage of each round is a very small value compared to Δt and can be omitted. The sensor which decides to work in the decision stage works until the end of this round. Otherwise, it sleeps until the end of this round. Each sensor decides to work or sleep as described above in each round until its residual lifetime is less than Δt .

VI. SIMULATION RESULTS

We evaluate the performance of the *DCS-Greedy* and *DCS-Dist* algorithms and their applications through simulations running on a computer with 3 GHz CPU and 1 GB memory. N sensors with sensing radius r and M targets are deployed uniformly in a region of $R \times R$, where $R = 400$. Each sensor has W directions.

A. Comparison Across the Solutions to the DCS

Each algorithm runs 1000 times through random placement of sensors and targets. For the *DCS-Dist* algorithm, we assume the communication radius is twice of the sensing radius.

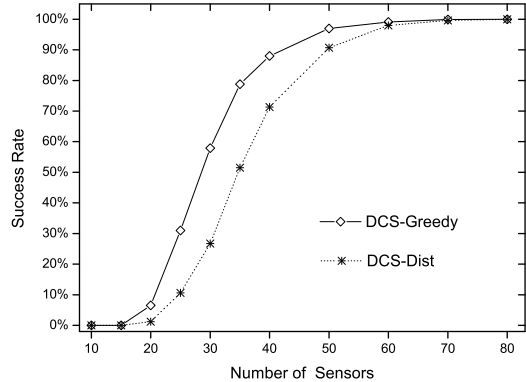


Fig. 1. Success rate vs. number of sensors N with $M = 40$, $r = 100$ and $W = 3$

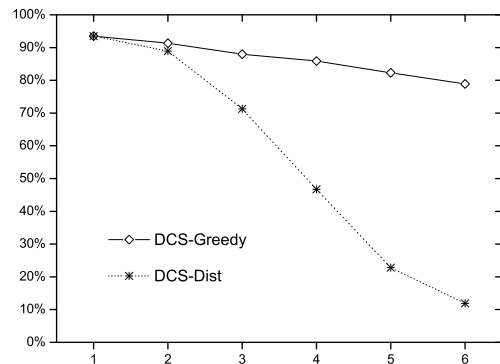


Fig. 2. Success rate vs. number of directions per sensor W with $M = 40$, $N = 40$ and $r = 100$

1) *Success Rate*: Fig. 1 shows the success rate of the *DCS-Greedy* and *DCS-Dist* algorithms. The success rate is the ratio of the number of samples where a cover set is successfully found by each algorithm to the total number of samples. We set $M = 40$, $r = 100$ and $W = 3$. From this figure we can see that the *DCS-Greedy* algorithm has a higher success rate than the *DCS-Dist* algorithm. Fig. 2 shows the relationship between the success rate of the three algorithms and the number of directions per sensor W . The success rate drops when W increases. Especially, the success rate of the *DCS-Dist* algorithm decreases much faster.

2) *Coverage Percentage*: Fig. 3 shows the coverage percentage of the *DCS-Greedy* and the *DCS-Dist* algorithms. The coverage percentage is the ratio of the number of covered targets to the total number of targets M . We set $M = 40$, $r = 100$ and $W = 3$. We can see from this figure that the coverage percentage of both algorithms increases quickly when N increases from 10 to 30 and relatively slowly after 30. The coverage percentage of the *DCS-Dist* algorithm is slightly smaller than that of the *DCS-Greedy* algorithm. Fig. 4 shows that the coverage percentage of the two algorithms drops when W grows. From this figure, we can see that the *DCS-Greedy* algorithm can still have relatively higher coverage percentage even when $W = 6$.

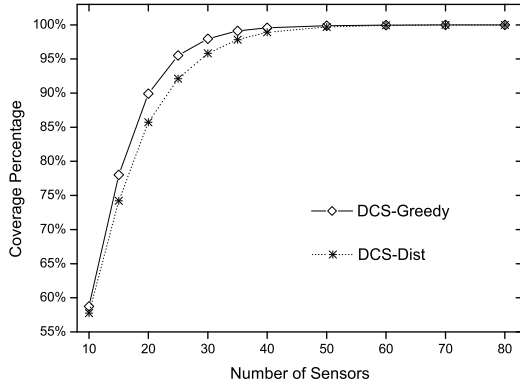


Fig. 3. Coverage percentage vs. number of sensors N with $M = 40$, $r = 100$ and $W = 3$

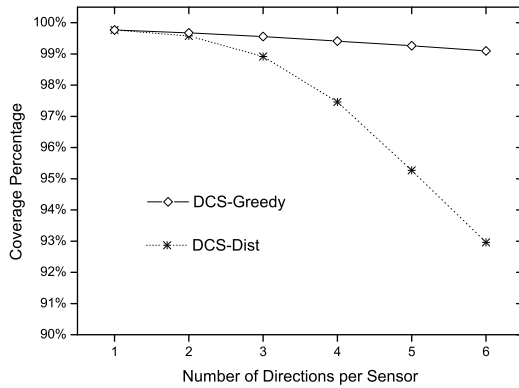


Fig. 4. Coverage percentage vs. number of directions per sensor W with $M = 40$, $N = 40$ and $r = 100$

B. Simulation for the Two Applications

In this simulation, the initial lifetime of each sensor is set as 1. Each application runs 10 times through random deployment of sensors and targets. Fig. 5 shows the relationship between the coverage percentage and the network work time for the *WT-Greedy* and *WT-Dist* when $M = 10$, $r = 100$, $W = 3$ and $\Delta t = 0.05$. As the time goes, the coverage percentage drops. In this figure, we can see that the average coverage percentage of the *WT-Dist* drops faster than the *WT-Greedy*.

VII. CONCLUSIONS

In this paper, we have studied the problem of finding a cover set. A centralized algorithm named *DCS-Greedy* and a distributed algorithm named *DCS-Dist* have been proposed for this problem. The *DCS-Greedy* algorithm has a higher possibility to find a cover set, and has a greater coverage percentage than the *DCS-Dist* algorithm. Based on the algorithms for the DCS, we have also introduced two applications to extend the network work time.

ACKNOWLEDGMENT

This work is supported in part by grants HKPU A-PH12, Z09M, Z0DF, PolyU-5236/06E, and the National Basic Re-

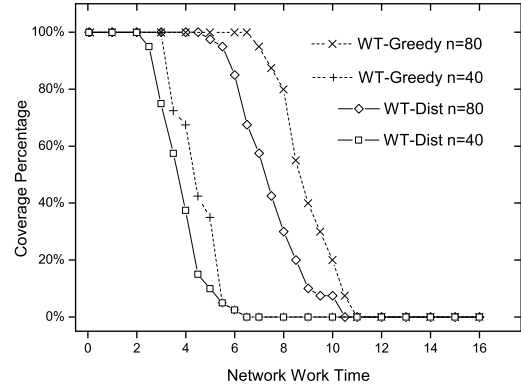


Fig. 5. Network work time vs. coverage percentage with $M = 10$, $r = 100$, $W = 3$ and $\Delta t = 0.05$

search Program of China (973 Program) under Grant No. 2006CB303000.

REFERENCES

- [1] H. Ma and Y. Liu, "On coverage problems of directional sensor networks," in *International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2005.
- [2] J. Ai and A. A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, no. 1, pp. 21–41, Feb. 2006.
- [3] M. Rahimi, R. Baer, O. I. Iroezzi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation in wireless sensor networks," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.
- [4] J. Djughash, S. Singh, G. Kantor, and W. Zhang, "Range-only slam for robots operating cooperatively with sensor networks," in *IEEE International Conference on Robotics and Automation*, 2006.
- [5] R. Szcwyczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *SenSys*, 2004.
- [6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *IEEE INFOCOM*, 2001.
- [7] B. Liu and D. Towsley, "A study of the coverage of large-scale sensor networks," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2004.
- [8] Y. Cai, M. Li, W. Shu, and M. Wu, "ACOS: An area-based collaborative sleeping protocol for wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 3, no. 1, pp. 77–97, 2007.
- [9] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *SenSys*, 2004.
- [10] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, 2005.
- [11] Y. Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Trans. Computers*, vol. 54, no. 8, pp. 978–991, 2005.
- [12] H. Liu, P. Wan, C. Yi, X. Jia, S. Makkai, and P. Niki, "Maximal lifetime scheduling in sensor surveillance networks," in *IEEE INFOCOM*, 2005.
- [13] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE INFOCOM*, 2005.
- [14] M. X. Cheng, L. Ruan, and W. Wu, "Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks," in *IEEE INFOCOM*, 2005.
- [15] Y. Cai, W. Lou, M. Li, and X. Li, "Target-oriented scheduling in directional sensor networks," in *IEEE INFOCOM*, 2007.