

# On the Coverage Problem for Myopic Sensors

Mohamed Aly, Kirk Pruhs, Taieb Znati  
Department of Computer Science  
University of Pittsburgh  
Pittsburgh, PA 15260  
Email: {maly, kirk, znati}@cs.pitt.edu

Brady Hunsaker  
Department of Industrial Engineering  
University of Pittsburgh  
Pittsburgh, PA 15261  
Email: hunsaker@enr.pitt.edu

## ABSTRACT

*The objective of the coverage problem is to organize the monitoring of targets by sensors in an energy efficient manner so as to maximize the lifetime of coverage. We consider the coverage problem in a network of myopic sensors, such as video sensors and acoustic sensors, which are only able to cover one target at any one time. We show how to formulate the problem of finding the lifetime as a linear program. We show that the actual coverage schedule can be found by iteratively finding maximum cardinality matchings in the underlying weighted bipartite graph, where the weights are derived from the solution of the linear program. We show experimentally that this algorithm is practical for moderate sized instances, depending on various properties of the instance.*

## I. INTRODUCTION

Recent advances in wireless communication networks, coupled with the convergence of technological and application trends, have resulted in exceptional levels of interests in wireless sensor networks. These networks are composed of large number of low-cost, low-power, multi-functional, wireless devices spread over a geographical area. Individually, sensing devices are resource-constrained and, therefore, are only capable of a limited amount of processing and communication. It is the coordinated effort of a large number of these devices, however, that bears promises for a significant impact, not only on science and engineering, but equally importantly on a broad range of civil and military applications, including managing complex physical systems, object tracking and environmental monitoring.

Harnessing the potential of wireless sensor networks brings about a number of fundamental challenges, the most critical of which is finding new ways to increase the longevity of such networks. Since sensors will either have to be powered by small non-renewable batteries, or by the modest amounts of energy that can be harvested from the environment, developing energy-efficient algorithms and mechanisms to optimize the use of battery power, while satisfying different and often contradictory performance metrics, is the most critical issue in the design of the network protocols for wireless sensor networks. This is particularly true for the sensor coverage problem which arises naturally in many sensor applications

related to the management of multi-sensor systems for target tracking [1].

Generally speaking, the coverage problem is how to apportion or balance the workload among the sensors so as to extend the lifetime of coverage. Most of the literature to date assumes that sensors can cover all targets in a fixed area at a fixed cost (which may depend on the measure of the area) [2]. However, for some sensors this is not a valid assumption. In particular, video sensors and acoustic sensors may be viewed as unidirectional, that is, at least in some settings, they can only be focused on one target. We call such sensors *myopic*.

In this paper, we consider the coverage problem for networks of myopic sensors. We show how to formulate the problem of finding the lifetime as a linear program. We show that the actual coverage schedule can be found by iteratively finding maximum cardinality matchings in the underlying weighted bipartite graph, where the weights are derived from the solution of the linear program.

In order to validate its practicality, we implemented our algorithm and tested it on synthetically generated networks with various properties. We consider the case of steady-state uniform networks, and the case of a hot spot, perhaps representing an emergency, where many of the targets are concentrated in one area. We used a standard LP solver, *Soplex*, which uses primal and dual solving routines [3]. To find maximum cardinality matchings, we implemented the matching algorithm presented in [4]. All of our code, including the code for generating the experimental instances, can be found on the project web page

[www.cs.pitt.edu/~maly/Myopic-Coverage/](http://www.cs.pitt.edu/~maly/Myopic-Coverage/).

Some of our more interesting findings are:

- The optimal schedules that our algorithm constructs have a surprisingly simple structure. This property makes the underlying technique generalizable to other wireless sensor network settings.
- The problem is more computationally difficult in the case of hot spots, or if the ratio of the number of sensors to the number of targets is small.
- The algorithm is practical for networks of a few hundred nodes, but may become impractical for networks with thousands of nodes, depending on various properties of the network environment.

The rest of the paper is organized as follows: the second section discusses related work. The third section presents our

formalization of the coverage problem for myopic sensors. A description of an algorithm to find a coverage schedule is also provided. Experimental results are presented in the fourth section. The last section of this paper provides a summary of this research work and discusses future work.

## II. RELATED WORK

The coverage problem arise naturally in applications in wireless sensor networks. In its most general form, the problem was shown to be NP-hard [5]. Several heuristics were proposed to solve different formulation of this problem. The setting in most of the literature related to the coverage problem is a collection set of sensors  $S = \{S_1, \dots, S_s\}$  in some metric space, usually the Euclidean plane. Further, each sensor  $S_i$  has an associated area  $A_i$  that it can cover, most commonly a circle of radius  $r_i$ . A non-myopic sensor  $S_i$  can cover all of  $A_i$  if the sensor is turned on.

We survey what is known for non-myopic sensors. In [1], an efficient algorithm is given to determine whether each point in the plane is covered by at least  $k$  sensors,  $k$ -covered or not, assuming all  $r_i$ 's are equal [1]. In [6], a simple scheme for maximizing the lifetime of coverage is proposed. In particular, the space and the sensors are partitioned so that all the sensors in a particular partition can cover all the space in the partition. The sensors in this partition then equally share the monitoring of the region of space. In [2], a distributed heuristic for computing a coverage plan is proposed. Using almost the same approach as [2], [7] introduced another heuristic to organize the coverage. Further, [2] presented a polynomial time solution in terms of the number of sensors. None of these schemes is applicable to the case of myopic sensors, because the basic concept of intersection of coverage areas used in these papers doesn't apply to myopic sensors.

The idea of modeling a sensor network as a weighted bipartite graph has been used in some previous studies, such as [8].

## III. THE FORMULATION OF THE MYOPIC SENSOR COVER PROBLEM

The setting is a collection  $S = \{S_1, \dots, S_s\}$  of myopic sensors, and a collection of targets  $T = \{T_1, \dots, T_t\}$ . Sensor  $S_i$  is powered by a battery that initially contains  $e_i$  units of energy. There is a monitoring cost  $C_{i,j}$ , measured in energy per unit time, that sensor  $S_i$  incurs for monitoring target  $T_j$ . That is, if  $S_i$  is assigned to  $T_j$  for  $P$  units of time, then  $C_{i,j}P$  units of energy are used from  $S_i$ 's battery. Note that we allow that  $C_{i,j}$  may be infinite, thus capturing the possibility that it may not be feasible for some sensors to cover some targets. Sensors are assumed to use no energy when they are not monitoring a target. The fact that the sensors are myopic means that each sensor can monitor at most one target at any time. We assume that all targets must be covered, that is, at all times each target must be monitored by a sensor.

The coverage at any particular time can be viewed as a maximum cardinality matching  $M$  in the complete bipartite graph  $G$  with bi-partitions  $S$  and  $T$ . There is an edge  $(S_i, T_j)$

in the matching if, and only if, sensor  $S_i$  is monitoring target  $T_j$ . The rate at which the network uses energy at a particular time is the sum of the costs of the edges in the matching, that is,  $\sum_{(S_i, T_j) \in M} C_{i,j}$ . The objective of the coverage problem is to maximize the lifetime of the network, that is, the length of time for which it is possible to cover all the targets. An actual coverage schedule specifies a matching for each unit of time in the lifetime of coverage.

The most obvious way to model this as a mathematical program is as follows. Let  $w_{i,j}$  be the fraction of time that sensor  $S_i$  is assigned to target  $T_j$ . So  $P \cdot w_{i,j}$  is the total time that sensor  $S_i$  monitors target  $T_j$ . There are three constraints: (1) that each sensor  $S_i$  should not expend more than  $e_i$  energy, (2) that every sensor covers at most one target during each time slot, and (3) that every target should be covered by at least one sensor at all times. Thus, we get:

$$\begin{aligned} & \text{Maximize} && P && (1) \\ & \sum_{j=1}^t P \cdot w_{i,j} \cdot c_{i,j} \leq e_i && i = 1, \dots, s && (2) \\ & \sum_{j=1}^t w_{i,j} \leq 1 && i = 1, \dots, s && (3) \\ & \sum_{i=1}^s w_{i,j} \geq 1 && j = 1, \dots, t && (4) \\ & w_{i,j} \geq 0 && && (5) \end{aligned}$$

There are two difficulties with the above mathematical programming formulation. Firstly, the program is not linear because of the  $P \cdot w_{i,j}$  terms. Secondly, it is not clear that the program captures all of the constraints of our problem. We handle these difficulties in turn.

To create an equivalent linear program, we introduce a new variable  $Q$  equal to  $1/P$ . The objective function then becomes minimizing  $Q$ , and the non-linear constraint becomes the linear constraint  $\sum_{j=1}^t w_{i,j} \cdot c_{i,j} \leq e_i Q$ . Thus, our new linear program is

$$\begin{aligned} & \text{Minimize} && Q && (6) \\ & \sum_{j=1}^t w_{i,j} \cdot c_{i,j} \leq e_i Q && i = 1, \dots, s && (7) \\ & \sum_{j=1}^t w_{i,j} \leq 1 && i = 1, \dots, s && (8) \\ & \sum_{i=1}^s w_{i,j} \geq 1 && j = 1, \dots, t && (9) \\ & w_{i,j} \geq 0 && && (10) \end{aligned}$$

If the solution of the above LP is  $Q' = 1/P'$ , then it is clear that the coverage lifetime is at most  $P'$ . But it is not so clear that a lifetime of  $P'$  is obtainable since the above LP doesn't specify which sensor should be assigned to which target at each time. Next, we constructively show how to transform a solution of the above LP into a schedule with lifetime  $P'$ , thus

showing that a coverage lifetime equal to the solution of the LP is always achievable.

We now define what we call the coverage graph and explain how to construct a schedule by finding perfect matchings in this coverage graph. One bipartition of the coverage graph consists of the  $s$  sensors. The other bipartition consists of the  $t$  targets, and  $s - t$  virtual targets. A sensor will be matched to a virtual target if it is not covering a real target. The coverage graph is complete in that there is an edge between every two vertices in opposite bi-partitions. The weight of an edge  $(S_i, T_j)$  for a sensor  $S_i$  and a target  $T_j$  is the value of the variable  $w_{i,j}$  from our optimal LP solution. One can see from the LP that (before we consider virtual targets) the sum of the weights of the edges incident on any particular vertex is at most 1. The weight of the edges incident on the virtual targets are set in such a way that the sum of the weights on the edges for every vertex is equal to 1. This can be accomplished using the following algorithm:

---

#### Set\_Weights()

---

**Begin**

**Let**  $w(v)$  be the the aggregate weight of the edges incident to vertex  $v$ .

**While** (there is a sensor  $S_i$  with  $w(S_i) < 1$ ) **do** {

**While** ( $w(S_i) < 1$ ) **do** {

**Let**  $VT_j$  be lowest index virtual target with  $w(VT_j) < 1$

**Do** Increase the weight of the edge  $(S_i, VT_j)$  **until** either  $w(S_i)$  or  $w(VT_j)$  reaches 1 } }

**End**

---

We now show how to find a coverage schedule by iteratively finding perfect matchings in the coverage graph. The basic steps of the algorithm to produce the schedule can be described as follows:

---

#### Coverage\_Schedule()

---

**Begin**

**Do** Initialize time  $t = 0$

**While** ( $\exists$  a perfect matching in the coverage graph) **do** {

**Let**  $w_{i,j}$  be the smallest weight of an edge in  $M$

**Let**  $M$  be in the coverage plan from time  $t$  to time  $t + w_{i,j}$

**Let**  $t = t + w_{i,j}$

**Do** Deduct  $w_{i,j}$  from the weight of each edge in  $M$  in the coverage graph }

**End**

---

To prove the correctness of our algorithm, we need to argue that until all edges have 0 weight, there is always a perfect matching in the current coverage graph. To see why this is the case, scale up the edge weights of the coverage graph so that all weights are integer, and hence will stay integer throughout the algorithm. The weight of an edge can be thought of as the multiplicity of the edge in a multi-graph. Note that the above algorithm maintains the invariant that the aggregate weights of the edges incident on a vertex in the coverage graph (or equivalently the degree in the corresponding multi-graph), is the same for all vertices. Thus, the cardinality of the neighborhood of any subset  $S$  of vertices in this multi-graph is at least the cardinality of  $S$ . Hence, the coverage graph will always have a perfect matching by Hall's Theorem [9].

#### IV. EXPERIMENTAL RESULTS

The sensor network environment consisted of the number of sensors and targets in a two dimensional square, with side length 10,000 units. We simulated networks having 10 to 1000 targets. We assumed that the sensors were uniformly distributed, which is a natural configuration for sensors. We assumed each sensor had a starting energy level of 100,000 units. We assumed that the cost  $C_{i,j}$  of coverage is proportional to the square of the Euclidean distance between  $S_i$  and  $T_j$ , which is a natural assumption in the acoustic setting where the inverse square law applies [10].

The goal of our first set of experiments was to determine the practicality of our algorithm for different distributions of targets within the service area. Therefore, we simulated two types of target distributions. In the first case, we assumed that the sensors were uniformly distributed in the service area. This was meant to represent some steady-state normal configuration of the network. In the second case, we assumed that there is some hot spot with an unusually large number of targets that need to be monitored. This case might arise if some locations are more critical, for example, if there is a disaster or some other important event occurring there. In these cases, we assumed that the number of sensors is double that of targets. For the hot spot case, we assumed an environment with a single hot spot, consisting of a square of 500 units a side, that contains 50% of the targets. Targets inside and outside of the hot spot are uniformly distributed. We ran the simulation for 100 different instances of each size. We measured the solution time, the number of the iterations taken by the LP solver, the solution value, the total network lifetime, and the running time used to find the optimal coverage schedule. The results of the simulations are shown in the following figures. Note that we only show some of our findings due to space constraints.

**Figure 1** represents the average running time taken by the LP solver in the case of uniformly distributed targets, as well as the case of a service area having a hot spot. In both cases, the solution time is linear in terms of the number of targets, which is a reasonable bound on the running time. Note that **Figure 1** is a semi-logarithmic graph. For the first case, the running time is fairly small, even for large networks. Although still linear, there exists a difference in the order of magnitude

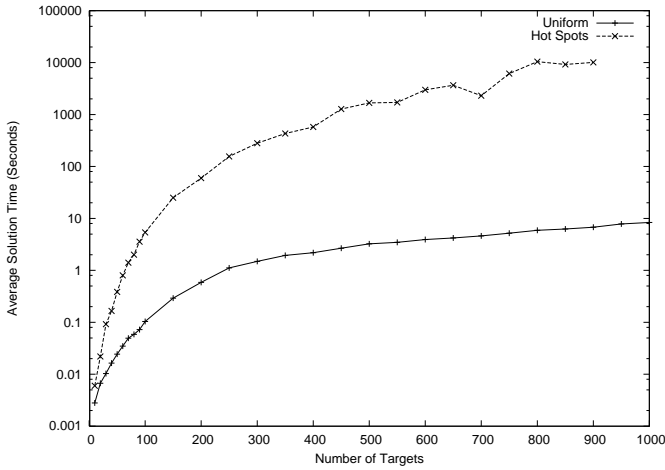


Fig. 1. Average solution time for uniformly distributed targets vs. the hot spot case

increase in the running time for the hot spot instances. The curves representing the number of iterations performed by the LP solver are similar to the running time curves shown in Figure 1.

A possible explanation for this difference between the two types of instances has to do with whether sensors are likely to cover only nearby targets. In the uniformly distributed case, sensors and targets are distributed in similar ways, so it is likely that sensors will almost always cover one of the few nearest targets. In the hot spot case, however, many of the targets in the hot spot must be covered by relatively distant sensors outside the hot spot. These sensors, in turn, cannot completely cover the targets nearest them, and other more distant sensors will help with coverage of these targets. Thus, almost all of the sensors outside of the hot spot must divide coverage among nearby and distant targets. Changes in one part of the coverage solution are thus more likely to cause changes for distant sensors. In addition, many targets in the hot spot are roughly the same distance from a sensor, causing many solutions to have very similar objective values. These difficulties may lead to more simplex pivots to find an optimal solution, even though we suspect that near-optimal solutions may still be found relatively quickly.

**Figure 2** represents the average number of different matchings in the optimal schedule. It was surprising to us that the number of matchings is so small and did not grow as the size of network grows. Our hypothesis is that the number of matchings is roughly a function of the ratio of the number of sensors to the number of targets. This would explain why the number of matchings is constant as both the number of targets and the number of sensors grow in a fixed ratio. We do not understand why the number of matchings is so low. From Figure 2, we can observe that the average number of matchings of the service area with a hot spot is less than that of the uniform service area, although this difference is not large.

The running time of actually finding the schedule from the

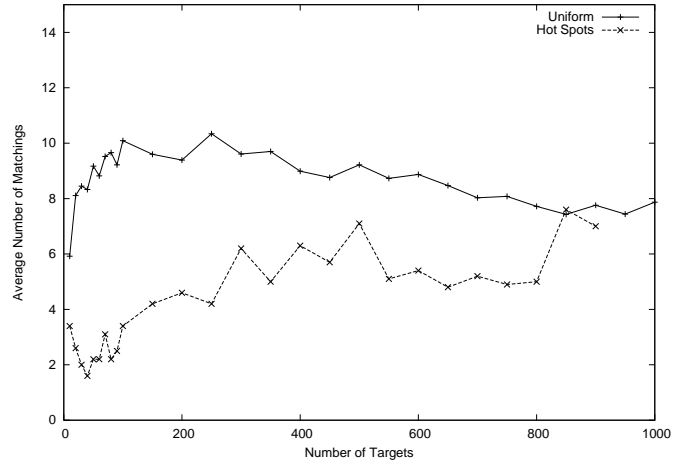


Fig. 2. The average number of matchings for uniformly distributed targets vs. the hot spot case

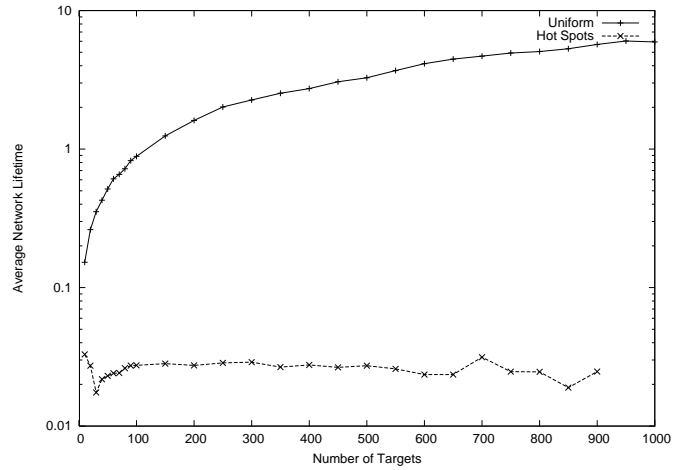


Fig. 3. Average network lifetime for uniformly distributed targets vs. the hot spot case

LP solution was in the order of fractions of a second for all runs of both types of service areas. This is because the number of matchings that had to be found was small, and because the matching algorithm is very fast, running in time  $(n^{1.5} \sqrt{\frac{m}{\log n}})$  [4].

**Figure 3** shows the average network lifetime for both the uniform targets case and the hot spot case. For the hot spot case, increasing the size of the instance didn't affect the lifetime much. Increasing the number of targets does not change the percentage of targets lying in the hot spot area. Therefore, the lifetime remains the same for all network sizes. The lifetime depends on the initial amount of energy of sensors. In the case of uniformly distributed targets, some efficiency is gained by increasing the size of the network, so that at least initially the lifetime scales roughly linearly with the size of the network.

We next address the issue of how the ratio between the number of sensors and the number of targets, which up until

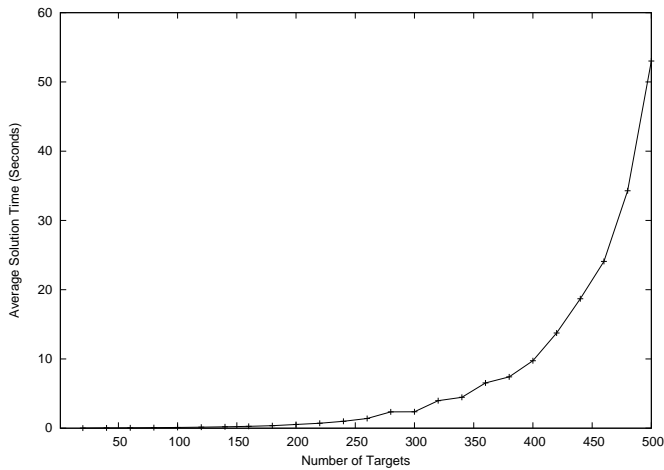


Fig. 4. Average running time as a function of the number of targets

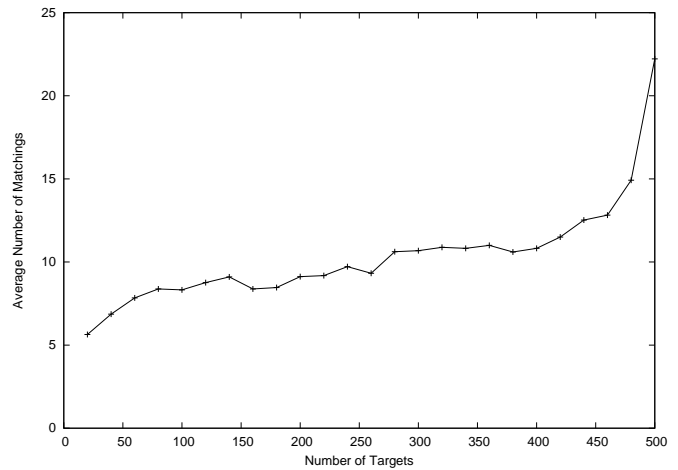


Fig. 5. Average number of matchings as a function of the number of targets

now has been fixed at 2, affects our results. We ran another set of experiments in which we fixed the number of sensors to be 500, while changing the number of targets. We used the same parameters used in the first set of experiments, except that in these experiments we ran the simulation 50 times for each target size. For the sake of simplicity, we only tested service areas with uniform distributions of both targets and sensors. The following two figures present the results of these experiments.

**Figure 4** presents the average running time of the LP solver for networks with a fixed number sensors and a variable number of targets. The solution time is negligible for small numbers of targets and significantly increases when the number of targets exceeds the half of that of sensors.

A possible explanation for this observation has to do with the number of sensors whose coverage is in question. With a small number of targets, most sensors will have one target that is clearly closest and will cover only that target in the optimal solution. A smaller number of sensors will be near two or more targets and the optimization will focus on determining the best coverage for these sensors. As the number of targets approaches the number of sensors fewer and fewer sensors will have this clear single-target solution, and the optimization process takes longer.

**Figure 5** shows the variation in the average number of matchings needed to obtain the optimal lifetime for the same network setup described above. The number of matchings increases slowly with the increase in the number of targets until the number of targets becomes 90% of that of sensors. The number of matchings significantly increases thereafter.

The result of Figure 5 may be explained with a similar argument to that of Figure 4. Note that only sensors that divide coverage among several targets contribute to increasing the number of matchings. In a network with a small number of targets, many sensors may cover a single target and therefore not increase the number of matchings. As the number of targets approaches the number of sensors, more and more sensors may divide their coverage, increasing the number of matchings

required.

## V. CONCLUSION AND FUTURE WORK

In this paper, we presented an algorithm to solve the coverage problem for a network of myopic sensors. We experimentally validated the practicality of this algorithm for a range of networks. Our technique can be generalized to solve many related problems. For example, it can be used if the number of sensors required to cover a target was dependent on the target, or if the energy in the sensors batteries varied from target to target.

## ACKNOWLEDGMENT

This work has been supported in part by NSF grants CCR-0098752, ANI-0123705, CNS-0325353, ANI-010292 and CCF-0448196.

## REFERENCES

- [1] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *IEEE Infocom*, 2001, pp. 1380–1387.
- [2] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *ACM Int. Workshop on Wireless Sensor Networks and Applications WSNA*, 2002.
- [3] R. Wunderling, "Paralleler und objektorientierter simplex-algorithmus," Ph.D. dissertation, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996.
- [4] H. Alt, N. Blum, K. Mehlhorn, and M. Paul, "Computing a maximum cardinality matching in a bipartite graph in time  $o(n^{1.5} \sqrt{\frac{m}{\log n}})$ ," *Information Processing Letters*, vol. 37, no. 4, pp. 237–240, 1991.
- [5] M. Marengoni, B. Draper, A. Hanson, and R. Sitaraman, "System to place observers on a polyhedral terrain in polynomial time," *Image and Vision Computing*, vol. 18, no. 10, pp. 773–780, 2000.
- [6] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE Int. Conf. on Comm. ICC*, Helsinki, June 2001, pp. 472–476.
- [7] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *ACM Int. Workshop on Wireless Sensor Networks and Applications WSNA*, 2003.
- [8] N. Sadagopan and B. Krishnamachari, "Decentralized utility based sensor network design," in *2nd Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, March 2004.
- [9] *Matching Theory*. North-Holland, 1986.
- [10] *Acoustics, An Introduction to Its Physical Principles and Applications*. Acoustical Society of America, 1981.