Localized Distributed Sensor Deployment via Coevolutionary Computation

Xingyan Jiang
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
Email: xingyan@cs.mun.ca

Yuanzhu Peter Chen
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
Email: yzchen@cs.mun.ca

Tina Yu
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
Email: tinayu@cs.mun.ca

***Abstract*—Sensor placement of a wireless sensor network has a significant impact on the network performance, such as sensing coverage, communication costs and connectivity. Depending on the application domains, the deployment method of sensor networks may vary. In autonomous sensor networks, the deployment is typically realized via cooperative movement among the nodes themselves. In this paper, we propose a sensor deployment protocol, LODICO, to optimize the sensor placement with arbitrary initial positions. In essence, it is a distributed evolutionary algorithm executed cooperatively among all sensor nodes via local information exchange. The simulation results show that LODICO can provide a very high coverage rate in a short convergence time.**

I. Introduction

A wireless sensor network consists of a large number of small sensor nodes distributed over an area of interests. Such networks are capable of observing and sensing the environment and sending the collected data to a data sink for further processing. Sensors must be deployed before they can provide useful data. Therefore the deployment of static or mobile sensors is an important basis for sensor networking. A good placement yields a high utilization of network resources.

Various techniques have been proposed to optimize sensor placement. The primary consideration of node placement in sensor networks is sensing coverage. It is the area that can be monitored collectively by the sensors in the network. Energy conservation is another critical issue in sensor networking. The energy costs in operating a sensor network include moving nodes, sensing events in the environment, and transferring information. The lifetime of a sensor network is limited by the battery capacity of the nodes. In many applications where the replacement of power is impossible, minimizing energy consumption is extremely important during the deployment of sensors.

Deployment of sensor networks can be carried out in two major ways: pre-deployment and post-deployment. The goal of both approaches is to meet certain critical networking objectives, such as maximizing coverage or lifetime. Pre-deployment approach calculates or estimates the number or positions of the sensors before they are actually deployed. This is typically used for static sensors in a known environment. In contrast, in some dangerous or unaccessible environments, mobile sensors are placed randomly in the network at the beginning. This initial random placement does not usually give a good coverage and, thus automated adjustments is necessary. This is the post-deployment approach.

Research on pre-deployment approach of sensors mostly takes a centralized approach, because distributed algorithms are not necessary since a computer program can be run on a powerful computer before physical deployment. For example, in Isler et al. [4], two characteristics of sensor networks, coverage and connectivity, are considered in the pre-deployment process. They use computational geometry to deploy sensors and guarantee the coverage. Once the sensors are deployed, a suitable communication range is calculated in order to guarantee the network connectivity. Jourdan and de Weck [5] study the deployment problem using a multi-objective genetic algorithm. Their goal is to balance two conflicting objectives, maximizing the network coverage while minimizing the energy consumption in the network. A Pareto front is generated after the execution of the algorithm and produces a solution set for users to choose from. Hu et al. [3] considered a hybrid sensor network which consists of a mixture of regular small sensors and more powerful micro-servers. They employed tabu search to decide where the micro-servers should be placed so that the lifetime of the network can be maximized.

Post-deployment approach of sensors has been studied using a variety of techniques. Howard et al. [2] described an incremental algorithm which deploys one sensor at a time. Each sensor node uses the positions of previously deployed nodes to determine its own position. Zou and Chakrabarty [11] propose a virtual force based algorithm to enhance the coverage after an initial random deployment. Their algorithm is a cluster-based algorithm, and the clusterheads are responsible for coordinating the distributed computation. The algorithm combines attractive and repulsive forces to determine virtual motion paths, and a one-time movement is carried out when the positions of sensors are identified to conserve energy. Wang et al. [9] focuses on the coverage holes when calculating target positions of sensors. They optimize the coverage within a short deploying time and limited movement using three Voronoi diagram based deployment protocols, VEC, VOR, and Mini-Max. Chellappan et al. [1] propose a flip-based algorithm and optimize both the coverage and the total number of flips. More

recently, it has also been demonstrated that computational intelligence techniques, such as fuzzy logic [8] and swarm intelligence [10] can be effective in sensor deployment.

In this paper, we propose LODICO, a localized distributed algorithm, to maximize sensor network coverage. The algorithm facilitates sensors to construct their partial solutions based on local information exchange within neighborhood. This is a powerful extension of the existing framework of cooperative coevolutionary algorithms [7]. The rest of the paper is organized as follows. We first give a brief background of evolutionary algorithms in Section II. Then, in Section III the features of LODICO are highlighted, followed by a detailed description in Section IV. Simulation studies are presented and explained in Section V. Last, we conclude this paper with discussion on some future research in Section VI.

II. Brief Background of Evolutionary Algorithms

An evolutionary algorithm (EA) is a search method based on the idea of the Darwinian principle of survival of the fittest. It is a powerful optimization technique for finding a global solution to, typically, extremely complex problem where finding a solution is very time-consuming. EAs solve a problem by first generating a large number of *individuals*, each of which represents a candidate solution to the problem. The set of individuals are grouped in a *population*. An individual can be represented using various data structures, which is its genotype. A genotype can contain one or more chromosomes. This study will use two-chromosome for each genotype. Usually, a linear structure is employed to resemble the biological chromosome in natural systems. The fitness of an individual is evaluated by a *fitness function* that takes the genotype as an input and yields a scalar value. A number of operations can be applied to one or multiple individuals, such as *crossover* and *mutation*, in order to produce modified individuals. These modified individuals are called *offspring* and the original ones are *parents*. A crossover involves exchanging the genetic materials in the genotypes of two or more parents. A mutation on an individual is a random change in its genotype with a small probability. An EA is essentially an iterative reproduction of generations of individuals. A parent generation produces a set of offspring, and fitter individuals among the parents and offspring are selected to survive for the next generation. Frequently, the fitness of the population improves as the evolution continues until certain termination conditions are met. See [6] for more details.

Cooperative coevolutionary algorithm ($CCEA$) is a special evolutionary algorithm proposed in Potter [7]. Unlike the traditional EA which solves a problem by searching the whole solution space, CCEA divides the problem into subproblems and search the subsolution space simultaneously. Since the subsolution space is smaller, the algorithm may find better solutions faster. Multiple separate populations are created with their genotypic representations having no functional overlapping. Each population represents a different species corresponding to one solution component and an individual therein represents a solution to this subproblem. Only the individuals of the same species can mate to produce offspring. Each species evolves for a certain number of generations, after which genetic information is shared among all species via a representative from each species. This period of time is called an *ecosystem generation*. Using the most recent genetic information shared by other species, another ecosystem generation can be evolved by all species in parallel. An individual in a population is evaluated based on the combination of genotype and the representatives from other species. Fig. 1 gives an illustration of the idea. The outer evolutionary process is terminated when a certain termination condition is met.

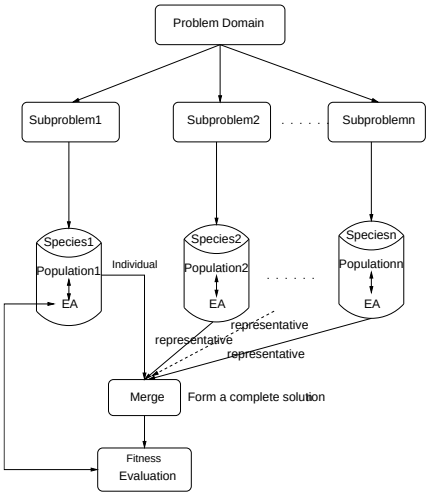


Fig. 1. A high-level view of CCEA

III. Features

The proposed sensor deployment protocol, LODICO, has two important features that identify itself as a localized distributed evolutionary algorithm:

1) LODICO is a completely localized distributed algorithm in that it requires each sensor to use and process information within its neighborhood. This is an essential requirement of distributed computing because every node in the system only has a local view of the environment. Global broadcasting of messages is possible but is considered infeasible due to the high computation overhead in such an environment. Sensor networks have limited resources and communication should be carried out locally to reserve energy. LODICO cooperates sensor nodes for self-deployment through localized information exchange and distributed evolutionary computing.
2) LODICO is a powerful extension of the existing CCEA framework in two important areas. First, the division of the general problem is flexible and dynamic. That is, every sensor node is responsible for dividing the global problem into a subproblem according to the most current sensor positions. In addition, as the deployment changes, so does the network structure. As a result, the division must be redone iteratively. Second, due to the

localized nature of the protocol, each sensor can only assume the availability of local information within its proximity. The fitness evaluation during the evolutionary process must tolerate the missing input from beyond the neighborhood. This is a salient contrast to the traditional CCEA, where fitness cannot be calculated without the information of all other subsolutions. The extended framework is depicted in Fig. 2.

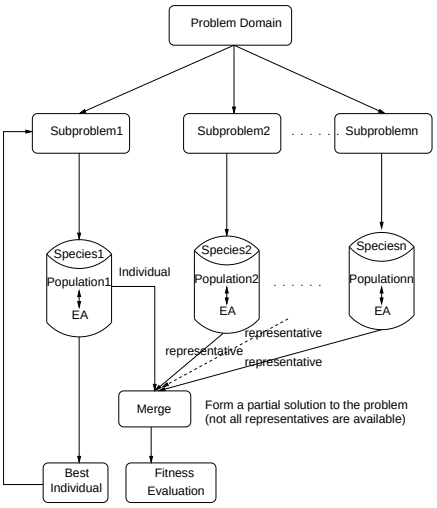


Fig. 2. High-level description of the localized distributed CCEA

IV. Algorithm Design

LODICO is executed on all sensors of the network in parallel for a number of iterations until a coverage requirement is met. In each cycle, a sensor first exchanges its location information with others within its communication range. Using this information, it prescribes a search space within its proximity in which it will find a target position and move to it at the end of the current ecosystem generation cycle. Within the search space, the sensor executes a local evolutionary algorithm to calculate the best target position using a fitness calculated from local information. Each sensor node then moves to its target position once it is calculated. As the network structure is altered, LODICO starts the next cycle by exchanging position information within neighborhoods. We make the following assumptions:

- Each sensor knows its own location.
- An sufficient number of sensors are deployed so that they can fully cover the entire area.
- Each sensor has a sensing range, R_s, a communication range, R_c, and $R_c \geq 3R_s$.

The LODICO algorithm consists of 3 major steps: planning, computing, and moving. We explain each step in the following sub-sections and the general flow of LODICO is given in Fig. 3.

A. Planning

At the beginning of the cycle, a sensor determines a partition of the entire deployment region to execute its local evolution-

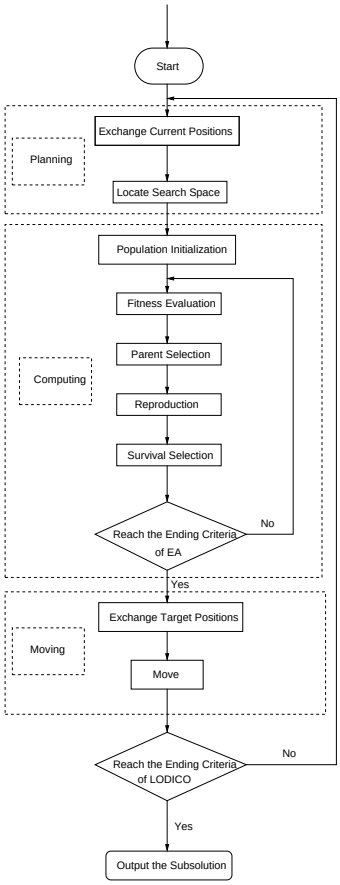


Fig. 3. LODICO flow chart

ary algorithm. To do that, it needs to know the positions of the neighboring nodes, i.e., those within its communication range, and to define a search space centered around its current position. The position information of each node is exchanged through a reliable wireless communication channel. The search space of a sensor is a limited scope within which the sensor can move in the current cycle.

The search space limit is important because excessive moving in a single cycle can make it hard for the algorithm to find good sensor locations. The reason is that sensors should cooperate with each other when positioning themselves. A target position is calculated using the latest position information within a neighborhood, so a drastic alteration of the neighborhood structure can invalidate the previous computation. In this work, we define the search space of a sensor to be equal to its sensing region, i.e. the circle of radius R_s centered at the node itself. Under the assumption that $R_c \geq 3R_s$, the search space limit of R_s ensures that the new coverage at a target position will not overlap with that of any node beyond its communication range, R_c. This is important for the fitness evaluation described in Section IV-B. The idea can be illustrated by the diagram in Fig. 4. Suppose node a has a communication range $R_c = 3R_s$. Centered at itself are these concentric circles of radii R_s, $2R_s$ and $3R_s$, denoted by C_1, C_2, and C_3, respectively. The search space

limit restricts node a's move within C_1, which implies that its new coverage will be restricted to C_2. For a non-neighbor node b, which is out of C_3, its coverage will not overlap with the new coverage of node a, no matter where it moves to within the range of its search space.

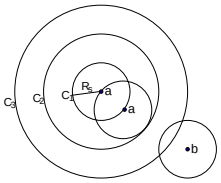


Fig. 4. Analysis of potential movement and overlaps

B. Computing

Each sensor executes an instance of a local EA to compute where it will move to at the end of the cycle. The local EA maintains a set of individuals, P, each of which corresponds to a positioning solution of the network. Here, an individual encodes its own position and those of its neighbors. To initiate these individuals, the sensor generates $|P|$ random positions uniformly distributed in its search space. Each position, along with those of the neighbors, is included in the genotype of an individual. Among these individuals, the $|Q|$ fittest are selected as parents, denoted by Q, to reproduce the same number of offspring Q'. Out of $P \cup Q'$, the $|P|$ fittest individuals survive and are carried over to the next generation. This local EA continues for k generations to determine the target position of the sensors, where k is a small integer as part of the EA configuration. The details of individual representation and its fitness evaluation are described at the following paragraph.

As a part of the network configuration, each sensor is given the information of the total number of sensors (n) in the network. We use a fixed length array of n elements to represent the genotype of an individual. Element i $(i = 1, 2, \ldots, n)$ is the position $\{x_i, y_i\}$ of sensor i in the deployment area (top diagram of Fig. 5). Note that for any given sensor, it only knows its own position and those of its neighbors. The elements in the genotype corresponding to non-neighbors contain invalid values. We use a second non-evolvable chromosome of length n to store the information of whether a sensor is inside or outside its neighborhood. Each element of this second chromosome can take a value from $\{0, 1, \star\}$, where 0 stands for non-neighbors, 1 stands for neighbors, and $\star$ stands for itself. Note that there is exactly one element with value $\star$ and that the number of 1's equals to the number of neighbors (bottom diagram of Fig. 5). Notice that we would use a variable-length genotype representation. However, our fix-length approach allows us to work on the problem under a much more general framework (Section VI). In this work, the only evolvable part of the genotype is its own position, but this can be naturally extended so that a sensor can "help to compute" the target positions of its neighbors. Using the described genotype, an offspring can be reproduced via arithmetic crossover, where the location value of an offspring is the mid-point of the gene values of its parents.

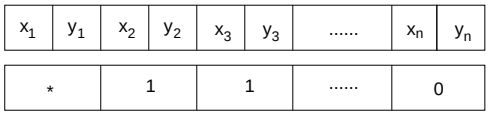


Fig. 5. The 2-chromosome genotype representation

The fitness of an individual is determined by the total coverage area induced by the new position and the total distance to travel to the new position. The goal is to find a target position with good coverage without excessive movement for energy conservation. And this should be evaluated using only local information. Assume that the sensing region of node i is A_i $(i = 1, 2, \ldots, n)$, each of which is a subset of the entire deployment area U, i.e. the universe. For a given node, only the sensing areas of its neighbors can be considered. To do that, we use the second chromosome in the genotype to filter the global information. Let $\mathcal{H} = \langle h_1, h_2, \ldots, h_n \rangle$ be the second chromosome of the sensor node. We define a companion vector $\overline{\mathcal{H}} = \langle \overline{h_1}, \overline{h_2}, \ldots, \overline{h_n} \rangle$, where $\overline{h_i} \in \{\emptyset, U\}$, for each $\mathcal{H}$. Specifically, $\overline{h_i} = U$ if $h_i \in \{1, \star\}$ and $\overline{h_i} = \emptyset$ if h_i = 0. Thus, the coverage unioned over a neighborhood of sensors is

$$\bigcup_{i=1}^{n} \left(\overline{h_i} \cap A_i\right)$$

For an individual represented by $\mathcal{H}$ and $\{A_i\}_{i=1}^{n}$ which is of distance d away from the current position, its fitness is

$$F = \left| \bigcup_{i=1}^{n} \left(\overline{h_i} \cap A_i\right) \right| - w \times d,$$

where w is a weight parameter for coverage-movement trade-off purposes.

Although the fitness evaluation of LODICO only uses local information from its neighboring nodes, the computed fitness value is able to drive the evolutionary search to find target position that gives good overall coverage and energy consumption.

C. Moving

Once the target position of a sensor is determined, the sensor moves to that location automatically using its actuation component. Then it broadcasts its new position and prepares for the next cycle. In some network scenarios, the assumption of $R_c \geq 3R_s$ can not be satisfied. In this case, the local coverage can not be calculated precisely. To alleviate this situation, an additional broadcast of the new location is necessary before the sensor starts to move to the new location. Further, a limited-scope flooding could be used alternatively.

V. Experimental Analysis

To evaluate the performance of LODICO, we implement a computer program to simulate the deployment of autonomous

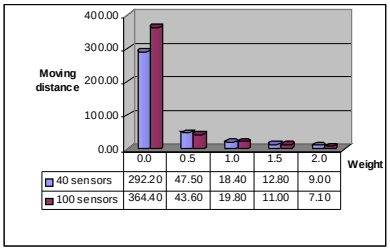
TABLE I
SIMULATION PARAMETERS

Parameters	Settings
Sensing range R_s	20m
Communication range R_c	60m
Deployment area size U	$100^2, 200^2, 300^2(\text{m}^2)$
Number of sensor nodes n	
area 100^2m^2	10, 12, 14, 16;
area 200^2m^2	40, 50, 60, 70;
area 300^2m^2	70, 80, 90, 100
Population size $\|P\|$	10
Generations g	5
Ecosystem generations g_e	30
Weight w	1.0

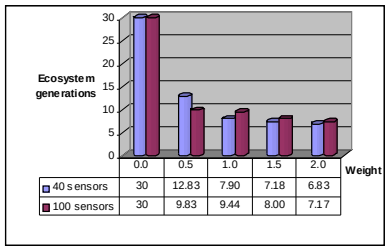
sensor networks with various initial positions. Three performance metrics, stable coverage, convergence time, and moving distance, are evaluated. Our experiment takes the average values of 20 runs and this section reports the experimental results.

The initial sensor positions are uniformly generated at random. We run simulations using various number of sensors in fields of different sizes. We adopt, respectively, 10, 12, 14, and 16 sensors in a $100 \times 100\text{m}^2$ square area, 40, 50, 60, and 70 sensors in a $200 \times 200\text{m}^2$ area, and 70, 80, 90, and 100 sensors in a $300 \times 300\text{m}^2$ area. Throughout the simulation, we use the same configuration for parameters: R_s, R_c, $|P|$, g, and g_e as shown in Table I.

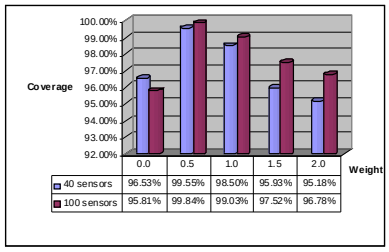
We use a set of preliminary experiments to study the effect of the weight parameter (w) for moving distance. The value of w can influence the network performance considerably. As shown in Fig. 6(a), the average moving distance of each sensor is large without moving distance control. Even though a small value of w can reduce the moving distance per sensor from 364.4m to 43.6m for 100 sensors deployed in a $300 \times 300\text{m}^2$ square region (second series in the chart). As w increases to 2, the moving distance per node is reduced to only about 2% compared to setting w to 0. The same trend holds for smaller networks such as 40 nodes in a $200 \times 200\text{m}^2$ square region (first series in Fig. 6(a)). When the algorithm convergence time is measured, we observe that a greater value of w leads to a smaller number of ecosystem generations needed to stabilize the final coverage because it can suppress excessive node movement effectively (see Fig. 6(b)). Note that in Fig. 6(b), when $w = 0$, the number of ecosystem generations takes to satisfy the convergence requirements is larger than 30, but we only plot them within 30 for better readability. We also measure the coverage of the sensors after 30 ecosystem generations when the algorithm either converges or fluctuates at a certain level. We see that, in Fig. 6(c), when $w = 0.5$ or 1, the highest stable coverage is reached at least 98.5%. Therefore, we set $w = 1.0$ in the following experiments to strike the balance. Meanwhile, when $w = 1.0$, we see that the program takes 7.9 and 9.44 ecosystem generations, respectively, to converge (Fig. 6(b)); Thus $g_e = 30$ is used in the last set of experiments.



(a) Moving distance



(b) Ecosystem generations



(c) Coverage

Fig. 6. Influence of distance weight

We next study the performance of LODICO under different network sizes. We use three configurations of smaller nodal density (i.e. 12 sensors in a $100 \times 100\text{m}^2$ region, 50 sensors in a $200 \times 200\text{m}^2$ region, and 80 sensors in a $300 \times 300\text{m}^2$ region) to show coverage improvement as sensors deploy themselves through localized interaction. Fig. 7 indicates that the coverage improves rapidly during the first few ecosystem generations and converges at around generation 7. When the populations stabilize, we evaluated three performance measurements, stable coverage (Fig. 8(a)), convergence time (Fig. 8(b), and moving distance (Fig. 8(c)), for 3 deployment area sizes ($100 \times 100\text{m}^2$, $200 \times 200\text{m}^2$, and $300 \times 300\text{m}^2$) and 4 different number of nodes for each deployment area size (Table I). The general observation from these experiments is that, as the sensor nodal density increases, so does the stable coverage, while the convergence time and moving distance decrease. This is reasonable as a larger number of sensors in the network makes it easier to cover a wider area of the deployed field.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a sensor deployment protocol, LODICO, to optimize the coverage of a wireless sensor network. LODICO is a completely localized algorithm which can be executed fully distributed and in parallel at each sensor

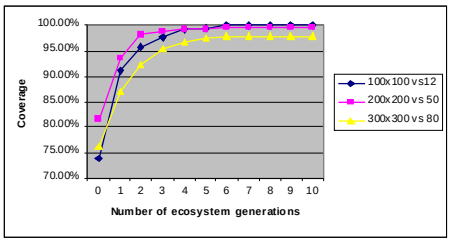
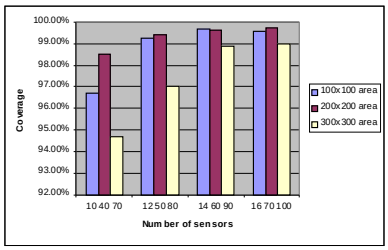
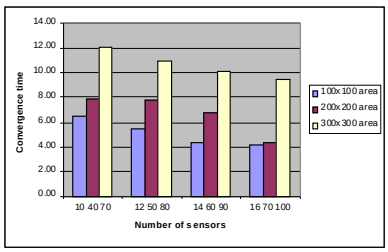


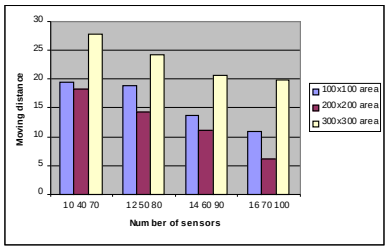
Fig. 7. Coverage improvement process



(a) Coverage



(b) Convergence time



(c) Moving distance

Fig. 8. Influence of number of sensors

node. The algorithm coordinates sensor nodes through local fitness evaluation and information exchange. In the experiments, we evaluate the stable coverage, convergence time, and the moving distance. The simulation results show that LODICO can achieve a very high coverage rate with short moving distances in a short period of time. As part of our future research, we plan to extend LODICO to consider the case where each sensor not only determines its own target position, but also suggests locations for the sensors in its neighborhood.

REFERENCES

[1] S. Chellappan, X. Bai, B. Ma, and D. Xuan. Sensor networks deployment using flip-based sensors. In *Proceedings of IEEE Mobile Sensor and Ad-hoc and Sensor Systems (MASS)*, November 2005.

[2] A. Howard, M. J. Mataric, and G. S. Sukhatme. An incremental self-deployment algorithms for mobile sensor networks. *Autonomous Robots, Special Issue on Intelligent Embedded Systems*, 13(2):113–126, Sep 2002.

[3] W. Hu, C. Chou, S. Jha, and N. Bulusu. Deploying long-lived and cost-effective hybrid sensor networks. In *Proceedings of the First Annual International Conference on Broadband Networking (BROADNETS)*, 2004.

[4] V. Isler, K. Daniilidis, and S. Kannan. Sampling based sensor-network deployment. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 2004.

[5] D.B. Jourdan and O.L. de Weck. Layout optimization for a wireless sensor network using a multi-objective genetic algorithm. In *IEEE Semiannual Vehicular Technology Conference*, Milan, Italy, May 2004.

[6] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, 1996.

[7] M. A. Potter. *The design and analysis of a computational model of cooperative coevolution*. PhD thesis, George Mason University, 1997.

[8] H. Shu, Q. Liang, and J. Gao. Distributed sensor network deployment using fuzzy logic systems. *International Journal of Wireless Information Networks*, 14(3):163–173, September 2007.

[9] G. Wang, G. Cao, and T. L. Porta. Movement-assisted sensor deployment. In *IEEE INFOCOM*, March 2004.

[10] X. Wu, J. Cho, B. J. d'Auriol, and S. Lee. Mobility-assisted relocation for self-deployment in wireless sensor networks. *IEICE Transactions*, 90-B(8):2056–2069, 2007.

[11] Y. Zou and K. Chakrabarty. Sensor deployment and target localization based on virtual forces. In *Proceedings of IEEE INFOCOM*, pages 1293–1303, 2003.