Dynamic Cooperative Coevolutionary Sensor Deployment via Localized Fitness Evaluation

Xingyan Jiang, Yuanzhu Peter Chen, and Tina Yu

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
{xingyan,yzchen,tinayu}@cs.mun.ca

Abstract. We propose an innovative cooperative co-evolutionary computation framework, Dynamic Cooperative Coevolution (DCC), which provides dynamic coupling of neighboring species for the fitness evaluation of individuals. One feature of DCC is the utilization of local fitness to achieve a global optimum, which makes it possible for co-evolutionary algorithms to be applied in localized distributed environments, such as network computing. This work is motivated by our interest in autonomous sensor deployment, where a sensor can only communicate with those within a limited range. Our experiments show that DCC is effective in obtaining good solutions under such distributed and localized conditions.

1 Introduction

A wireless sensor network consists of a large number of sensor nodes distributed over an area of interest. 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 transmit data. The deployment of static or mobile sensors, hence, is an important basis for sensor networking. A good placement yields high utilization of the network resources.

Two metrics are frequently used to evaluate the quality of sensor placement. The first one is *sensing coverage*, which is the area that the sensors in the network can monitor collectively. The second one is *energy consumption* during the sensor deployment. The energy cost in operating a sensor network includes 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 battery is impossible, minimizing energy consumption during the sensors deployment is extremely important.

Autonomous sensor deployment has been studied using a variety of techniques. Howard et al. [2] described an incremental algorithm which deployed one sensor at a time. Each sensor node used the positions of previously deployed nodes to determine its own position. Zou and Chakrabarty [13] proposed a virtual force based algorithm to expand sensing coverage after the initial random deployment. The sensor movements were determined by the combined attractive and repulsive forces and the movements were coordinated by a cluster head.

Wang et al. [9] focused on repairing coverage holes when calculating sensors target positions using three Voronoi diagram based deployment protocols, VEC, VOR, and MiniMax. Chellappan et al. [1] proposed a flip-based algorithm to optimize both the coverage and the total number of flips. More recently, it has been demonstrated that computational intelligence techniques, such as fuzzy logic [8] and swarm intelligence [12] can be effective in sensor deployment.

In this paper, we propose *DCC*, a dynamic cooperative co-evolutionary framework, for autonomous sensor deployment. The algorithm facilitates sensors to construct partial network structures based on the local information exchanged by sensors within their neighborhood, i.e. *communication range*. Step by step, the global network structure is constructed to achieve the goal of *sensing coverage maximization* and *energy consumption minimization*. The paper is organized as follows. We first give a brief background of cooperative co-evolutionary algorithms in Section 2. In Section 3, the features of DCC are introduced, followed by a detailed description in Section 4. Simulation studies are presented with results analyzed in Section 5. Finally, we conclude this paper in Section 6.

2 Cooperative Co-evolutionary Algorithms

Cooperative co-evolutionary algorithm ($CCEA$) is a special evolutionary algorithm proposed in [3, 7]. Unlike the traditional EA [6], which solves a problem by searching the entire solution space, CCEA divides the problem into subproblems and searches the sub-solution spaces simultaneously. Since the sub-solution space is smaller, the algorithm may find better solutions faster.

In CCEA, multiple separate populations are created with their genotypic representations having no functional overlapping. Each population represents a different species and an individual therein represents a solution to the subproblem. Only the individuals of the same species can mate to produce offspring. However, the fitness of an individual is evaluated on the combination of its genotype and the representative genotypes of *all* other species. Each population evolves for a certain number of generations, which is equivalent to one *ecosystem generation*. At the end of each ecosystem generation, one representative is selected from each population and their genotypes are shared with other populations for fitness evaluation. The high-level flow of CCEA is given in Fig. 1, where R_i is the representative of species i.

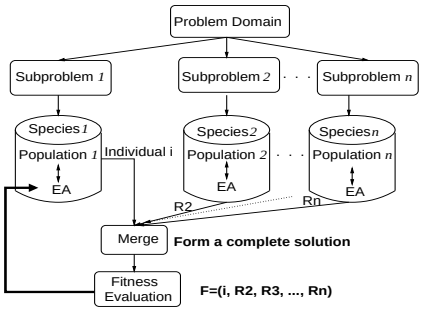

Fig. 1. A high-level view of CCEA.

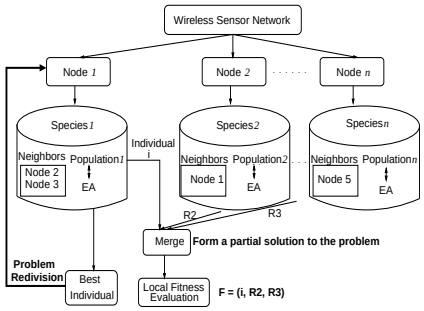

Fig. 2. High-level view of DCC.

There are researchers investigating *problem decomposition* and the efficiency of single-best collaboration during the evolution. Wiegand and colleagues [11] argued that when a problem is divided in such a way that there exists contradictory cross-population epistasis (inter-dependency), single-best collaboration would not produce good solution. To address the inter-dependency issue, Weicker and Weicker [10] proposed dynamically merging the species when inter-dependency of variables in cross populations was detected. Kim and Ryu [5] went farther by allowing not only merging but also splitting the species when the inter-dependency no longer exist during the evolution. Our cooperative co-evolutionary framework also provides dynamic division of species. The main features of the framework are described in the following section.

3 Dynamic Cooperative Coevolution Framework

DCC is a completely localized distributed algorithm in that each population only collaborates with populations within its neighborhood for fitness evaluation. This is an essential requirement for distributed computing where 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 required. To work with such constraints, the following mechanisms have been developed so that co-evolutionary algorithms can be applied effectively in localized and distributed environments, such as network computing. The DCC framework is depicted in Fig. 2.

1. **Flexible and dynamic problem division**. Under distributed environments where the location of each node may change dynamically, the partitioning of the problem (i.e. the sub-solution that each population evolves) also changes. This is contrast to the CCEA where the solution each population evolves is fixed throughout the execution of the algorithm. One consequence of this dynamic problem division is that the populations that collaborate for fitness evaluation also change during the algorithm execution.
2. **Energy efficient partial fitness evaluation**. Because each population can only assume the availability of local information within its proximity, the fitness evaluation must tolerate the missing input from beyond the neighborhood. This is a salient contrast to CCEA, where fitness cannot be evaluated without the information from all other populations.
3. **Two operation modes for effective and efficient evolutionary search**. In spirit, the first mode (mode I) is similar to the *splitting species* proposed in [5] and the second mode (mode D) is similar to the *merging species* proposed in [10]. If evolutionary search reaches a local optimum, merging species helps escaping the local optimum and making the search more effective. If evolutionary search reaches the basin of a global optimum after escaping a local optimum, splitting species helps the search find the global optimum faster. We developed a simple method to detect that a population might have reached a local optimum by checking the existence of coverage holes in the neighborhood. If one or more holes exist, operation is switched to mode D for 1 ecosystem generation cycle. Alternating these two modes can accelerate the search process while avoiding local optima.

4 Algorithm Design for Autonomous Sensor Deployment

We have implemented the DCC concept to solve the autonomous sensor deployment problem[1]. DCC consists of 3 major stages: *planning*, *computing*, and *moving*. A complete pass of the 3 steps is called an *ecosystem cycle*. In the planning stage, a sensor first divides the problem and 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 cycle. In the computing stage, the sensor executes a local EA within its search space to calculate the best target position using a fitness calculated from local information. Finally it moves to the target position in the moving stage. Once the movement is completed, the new search space for each sensor is calculated. The sensor may switch its operation mode (described in the following paragraph) if needed and then starts a new ecosystem cycle to search for the next position that the sensor would move to next. This process repeats many times until the specified number of ecosystem cycles is reached. Fig. 3 gives the high-level flow of the implementation.

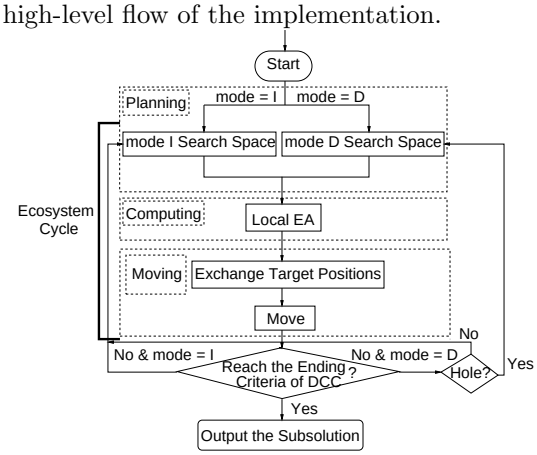

Fig. 3. DCC flow chart

After the sensors are randomly distributed in the field initially, one *local population* is used to evolve one sensor's target position. Each local population can be executed using one of two operation modes: mode I (Independent) evolves only a sensor's position and mode D (Dynamic) evolves the positions of a sensor and its neighboring sensors. In the first case, the fitness of an individual is evaluated on the combination of its genotype and the representative genotypes from the neighboring sensor populations. In the latter case, the fitness of an individual is evaluated on its own genotype, which contains the positions of a sensor and its neighboring sensors. Regardless of the operation mode, the fitness of an individual only covers a partial network of the entire sensor network.

Under mode I, the search space of a local population is two-dimensional: the x, y location of a sensor. With each local population searching a 2-dimensional space separately and simultaneously, the global sensor network can be obtained

[1] The source code is located in http://www.cs.mun.ca/∼xingyan/ppsn/DCC.tar.gz.

reasonably fast. However, occasionally sensors may get stuck in a local optimum. For example, in Fig. 4, $S_1, S_2, \ldots, S_6$ are 6 sensors used to cover an area, where S_4, S_5 and S_6 have identical location[2]. It is obvious that the sensing coverage would increase if some sensors move to the left or the lower region of the field. However, this would never happen because the current sensor locations give the best coverage (the union of the sensing region of all sensors), based on the neighboring sensor positions provided at the beginning of the ecosystem cycle. In order to obtain locations that give a better coverage than the current ones do, the neighboring sensors need to have different locations. Mode D provides this flexibility by allowing both the locations of a sensor and its neighboring sensors to evolve and helps the populations escape the local optimum.

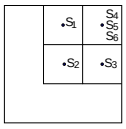

Fig. 4. An example of local optimum.

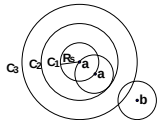

Fig. 5. Potential movement and overlaps.

In mode D, the search space of a local population is multiple-dimensional: the *x, y* locations of a sensor and its neighboring sensors. Unlike mode I where the neighboring sensor locations that are used for fitness evaluation are fixed throughout the ecosystem cycle, the neighboring sensor locations also evolve. It models the potential local interactions and uses that to improve the local estimate of fitness. Note that the evolved neighboring sensor positions are only used for fitness evaluation. They have no impact on the neighboring sensors' new positions, which are only decided by the "fittest" individual in the neighboring sensor populations.

The implementation is based on the following assumptions: 1) each sensor knows its own location. 2) a sufficient number of sensors are deployed so that they can potentially cover the entire area. 3) each sensor has a sensing range, R_s, a communication range, R_c, and $R_c \geq 3R_s$. DCC algorithm executes a sequence of ecosystem cycles, where each cycle consists of 3 steps: planning, computing, and moving. We explain each step in the following sub-sections.

4.1 Planning: Problem Division

At the beginning of each ecosystem cycle, the entire deployment area is partitioned based on the current sensor locations in the network: each sub-area is the *sensing region* of a sensor, i.e. the circle of radius R_s centered at the position of the sensor. A local evolutionary algorithm is executed for each sensor to locate a new position within the region where the sensor will move to at the end of the cycle. Under the assumption that $R_c \geq 3R_s$, the new coverage of a sensor and its non-neighboring sensors would never overlap no matter where they move to. For example, in Fig. 5 node a has a communication range $R_c = 3R_s$ and centered

[2] We use a square area to indicate a sensor's sensing region for simplicity.

at itself are three circles of radii R_s, $2R_s$ and $3R_s$, denoted by C_1, C_2, and C_3, respectively. The search space restricts node a to move within C_1, which implies that its new coverage will be restricted to C_2. For a non-neighboring node b, which is out of C_3, its sensing coverage will not overlap with the new coverage of node a, no matter where it moves to within the range of its search space. This restriction is important for the fitness evaluation described in Section 4.2.

For each local population, the individual with the highest fitness at the end of each cycle is selected and the sensor position information is exchanged with all its neighboring sensors (i.e., those within its communication range) populations through a reliable wireless communication channel. At the initial cycle where individuals in the population were randomly generated, representatives are selected randomly.

4.2 Computing: New Position Exploration

This section describes each component of the evolutionary algorithm.

Representation We used a fixed length array of n elements to represent the genotype of an individual, where n is the total number of sensors in the network. Each element i ($i = 1, 2, \ldots, n$) is the position $\{x_i, y_i\}$ of sensor i in the

x_1	y_1	x_2	y_2	x_3	y_3	.....	x_n	y_n
$\star$		1		1		.....	0	

Fig. 6. The 2-chromosome genotype representation

deployment area (see top diagram of Fig. 6). Since a sensor only has position information of its neighboring sensors, the elements in the genotype corresponding to non-neighboring sensors contain invalid values. To distinguish a neighboring sensor from a non-neighboring one, a second non-evolvable chromosome of length n is used (see bottom diagram of Fig. 6). There, a value 0 indicates that the corresponding element in the first chromosome is a non-neighbor while 1 indicates that it is a neighbor and $\star$ indicates the sensor itself. This 2-chromosome genotype representation provides the flexibility to facilitate the dynamic problem division explained in Section 3. When a sensor is switched from being a neighbor to a non-neighbor (or vice visa) for a particular sensor after movement, an update of the second chromosome can reflect such change.

Fitness Evaluation The fitness of an individual (sensor position) is determined by the total sensing coverage induced by the position and the travel distance between this and the current position of the sensor. Assume the sensing region of node i is A_i ($i = 1, 2, \ldots, n$), each A_i is a subset of the entire deployment area U, i.e. the universe. For a given node, the sensing coverage is the union of its sensing area and the sensing areas of its neighboring sensors. Let $\mathcal{H} = \langle h_1, h_2, \ldots, h_n \rangle$ be the second chromosome of the sensor's genotype. To calculate its coverage, 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 the neighborhood of a sensor is:

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

For an individual with sensor position which is d away from the current position, its fitness F 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 to adjust the tradeoff between coverage and movement. Although the fitness evaluation of DCC only uses local information from its neighboring nodes, it will be shown (see Section 5) that the computed fitness value is able to drive the evolutionary search to find target positions that give good overall coverage and requires a small amount of energy consumption.

Selection and Genetic Operations Among a population of P individuals, the $|Q|$ fittest are selected as parents, denoted by Q, to reproduce the same number of offspring Q' via arithmetic crossover. The Q individuals are paired based on their ranks: the first rank is paired with the second rank, the second rank is paired with the third rank and so on. The arithmetic crossover takes the average of the two parents' gene values as the gene value of its offspring.

Out of $P \cup Q'$, the $|P|$ fittest individuals survive and are carried over to the next generation. This process continues for g generations and the fittest individual at the end is the target position where the sensor moves itself to.

4.3 Moving: Automatic Sensor Relocation

Once the new 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.

5 Experimental Analysis

We used the implemented DCC algorithm to simulate the autonomous sensor deployment under various initial conditions: sensors are distributed uniformly to three different sizes of field: $100 \times 100\text{m}^2$ (small), $200 \times 200\text{m}^2$ (medium) and $300 \times 300\text{m}^2$ (large). For the small size field, 10, 12, 14 and 16 sensors are deployed; for the medium size field, 40, 50, 60 and 70 sensors are deployed; for the large size field, 70, 80, 90 and 100 sensors are deployed. Table 1 summarizes the parameter values used to carry out our simulation.

We use 3 metrics to evaluate the experimental results averaged over 30 runs: *moving distance*, *convergence time* and *sensing coverage*. Moving distance is

Table 1. Simulation parameters

Parameter	Setting	Parameter	Setting
Deployment area size U	$100^2, 200^2, 300^2(\text{m}^2)$	Sensing range R_s	20m
No. of sensor nodes n		Communication range R_c	60m
area 100^2m^2	10, 12, 14, 16;	Population size $\|P\|$	10
area 200^2m^2	40, 50, 60, 70;	No. of offspring $\|Q\|$	5
area 300^2m^2	70, 80, 90, 100	No. of runs	30
No. of eco cycles g_e	30	No. of gen in each eco cycle g	5

the average distance that a sensor in the network has to travel from its initial to final position. Convergence time is the number of ecosystem cycles it takes for *all* sensor populations to converge, i.e. the best individual fitness stopped improving. Sensing coverage is the percentage of the deployment field that is covered by the deployed sensors. Also, to select a weight parameter (w) that balances the evolutionary force toward solutions that give *large* coverage and *small* moving distance, we conducted a preliminary study and chosen $w = 1$ [4].

5.1 Simulation Under Mode I Only

We study mode I performance under different network sizes (small, medium, large) using a different number of sensors as that given in Table 1. Fig. 7 shows that the global network coverage improves rapidly during the first few ecosystem cycles and the populations converge around generation 7. Fig. 8 gives the global network coverage and the moving distance over time for one run on a medium size field. It shows that the coverage increases while the moving distance decreases as the evolution progresses. The selected w (1) is able to balance the two conflicting objectives and direct the evolutionary search to find a good solution.

When the best individual in all populations stopped improving, the 3 metrics were evaluated (see Fig. 9). The general observation from these experiments is that, as the sensor nodal density increases, so does the induced network 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 under a smaller amount of time and moving distance.

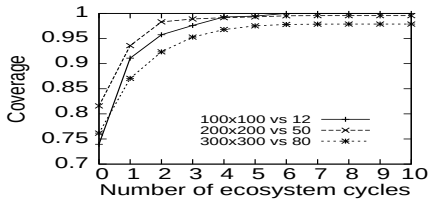

Fig. 7. Coverage improvement.

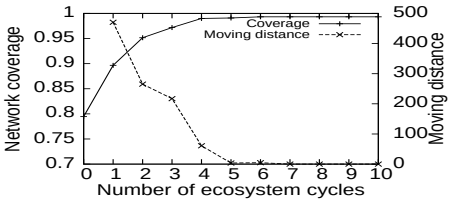

Fig. 8. Coverage vs. moving distance.

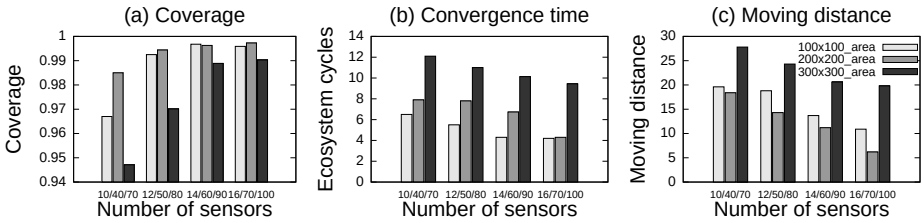

Fig. 9. Performance of Mode I evaluated by 3 different metrics.

5.2 Simulation Under the Alternation of Mode I & D

To investigate the benefit of mode D in helping the populations escape local optima and deliver better solutions, we carried out two sets of experiments: one operated mode I only and the other alternated mode I & D with 5 and 1 ecosystem cycles intervals, i.e. 5 mode I cycles followed by 1 possible mode D cycle. This alternation was selected because a population is not likely to reach a local optimum during the first 5 cycles, hence should be operated under mode I. At the end of the 5th cycle, the best individual in each population is checked for *coverage holes* (an area that is not covered by any sensor in its neighborhood). If there is any hole, the local GA is switched to mode D for 1 cycle and switched back to mode I the following cycle, since mode I runs faster than mode D (see Section 4). This check is carried out for each sensor population. The average coverage of 30 runs and the numbers of runs achieving 100% coverage are given in Table 2. Overall, both setups provide very good coverage. Nevertheless, alternating mode I & D delivers a higher number of runs that produced 100% coverage.

Table 2. Coverage Comparison Between Mode I and Mode I & D

	Mode I		Mode I & D	
sensors	coverage	100% cover	coverage	100% cover
40	98.50%	0	99.33%	1
50	99.44%	0	99.88%	15
60	99.63%	0	99.98%	25
70	99.73%	0	99.99%	27

To validate our hypothesis that mode D improves performance by helping the populations escape local optima, we conducted another experiment with 10 sensors initialized to locations that give a local optimum coverage (64%) and deploying them to a medium size field. The simulation was carried out by alternating 2 cycles of mode I followed by 1 possible cycle of mode D. The best global fitness (see Fig. 10) shows that after 2 cycles of no fitness improvement, the fitness declined after the execution of mode D, which is caused by a large moving distance (see Fig. 11), indicating the sensor has escaped the local optimum. After that, the global fitness starts to climb and eventually reaches 100% coverage.

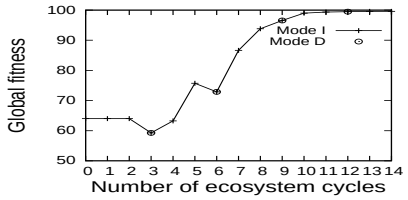

Fig. 10. Global fitness under Mode D & I.

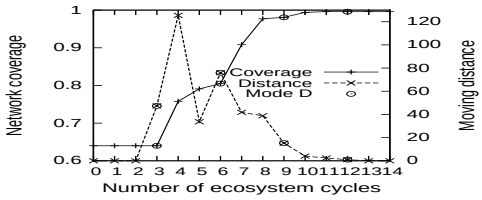

Fig. 11. Coverage vs. moving distance.

6 Conclusions

We have presented an innovative cooperative coevolutionary framework, DCC, for optimization tasks in localized and distributed environments. By supporting *dynamic problem division*, *partial fitness evaluation* and *2 operation modes*, DCC is shown to be effective in the autonomous sensor deployment task, where high coverage and low energy consumption were achieved in a short period of time.

References

1. S. Chellappan, X. Bai, B. Ma, and D. Xuan. Sensor networks deployment using flip-based sensors. In *Proceedings of IEEE 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. P. Husbands, and F. Mill. Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling. In *Proceedings of ICGA* 1991: 264-270.
4. X. Jiang and P. Y. Chen and T. Yu. Localized Distributed Sensor Deployment via Co-evolutionary Computation. In *Proceedings of the IEEE International Conference on Communications and Networking*, 2008.
5. M. W. Kim and J. W. Ryu. An efficient coevolutionary algorithm using dynamic species control. In *Proceedings of ICNC*, 2007.
6. M. Mitchell. *An introduction to genetic algorithms.* MIT Press, 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. K. Weicker and N. Weicker. On the improvement of coevolutionary optimizers by learning variable interdependencies. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1627–1632, 1999.
11. R. P. Wiegand, W. C. Liles and K. A. De Jong. The effects of representational bias on collaboration methods in cooperative coevolution. In *Proceedings of the Seventh Conference on Parallel Problem Solving from Nature*, pages 257-268. 2002.
12. X. Wu, J. Cho, B. J. d'Auriol, and S. Lee. Mobility-assisted relocation for self-deployment in wireless sensor networks. *IEICE Trans*, 90-B(8):2056–2069, 2007.
13. Y. Zou and K. Chakrabarty. Sensor deployment and target localization based on virtual forces. In *Proceedings of IEEE INFOCOM*, pages 1293–1303, 2003.