DYNAMIC COOPERATIVE CO-EVOLUTIONARY AUTOMATED MOBILE SENSOR DEPLOYMENT VIA LOCALIZED FITNESS EVALUATION

XINGYAN JIANG

Dynamic Cooperative Co-evolutionary Automated Mobile Sensor Deployment via Localized Fitness Evaluation

by

© Xingyan Jiang

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

October 2008

St. John's Newfoundland

Abstract

A wireless sensor network is a self-organized network consisting 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.

Automated mobile sensor deployment of a wireless sensor network has a significant impact on the network performance, such as network sensing coverage, communication or mobile costs, and connectivity. Due to the small size of sensors, they are equipped with small batteries and have low-power computing and communication resources. The lifetime of a sensor is determined by its battery life and it can not operate for an infinite amount of time. Therefore, a good deployment yields a high utilization of power resources.

In this thesis, we propose an innovative cooperative co-evolutionary computation framework, Localized Distributed Coevolution (LODICO), to optimize the automated sensor deployment with arbitrary initial positions. LODICO is a fully distributed and localized algorithm. It can be executed on all sensors of the network in parallel. Meanwhile the information exchange has to be done locally as each sensor can only communicate with those within a distance. Further, we extend LODICO to LODICO/D to provide dynamic interaction to neighboring computing agents during the evolutionary process. It models the potential local interactions between computing agents, and uses the the imaginary neighboring movements to improve its local fitness

and to help escaping from local optima.

This thesis is a powerful extension work to the traditional Cooperative Coevolutionary Algorithm. One feature of it is the utilization of local fitness to achieve a global optimum, which makes co-evolutionary algorithms applicable to localized distributed environments, such as network computing. Another salient feature is that the proposed algorithms can adjust and adapt the frequent dynamic change of network structures due to the position changes or failures of computing agents. LODICO/D incorporates LODICO with mode D to help to escape local optima. Mode D creates the third feature of imaginary collaboration with the neighboring computing agents during the evolutionary process to improve its local fitness. Our experiments show that LODICO and LODICO/D are effective in obtaining good solutions under such dynamic, distributed, and localized condition constraints.

Acknowledgments

I would like to acknowledge the help and support of all people in making this thesis possible.

In particular, I would like to thank my co-supervisors, Dr. Yuanzhu Peter Chen and Dr. Tina Yu, for their continued encouragement, valuable suggestions, and patience, during my master's studies. They are always there to guide, to support, and to help. Thank you for giving me the chance to work with you, and thank you for what you taught me.

My special thanks go to Dr. Wolfgang Banzhaf, who kindly provides his expertise in the area of evolutionary computation despite his busy schedule.

I thank Ting Hu, my classmate and friend, for her help of graph plots and discussions. I also thank my officemate, Anne Ngugi. I enjoy all the discussions we had, work related or not.

My appreciation also goes to the friendly administrative staffs of the Department of Computer Science: Elaine Boone, Sharon Deir, Darlene Oliver, Nolan White, Paul Price, and Peter Howell. Thank you all for your kind support.

Last but not least, I want to express my deepest gratitude to my family: my parents, my husband, and my daughter, for their endless love, encouragement, and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Wireless Sensor Networking

A wireless sensor network is usually composed of a large number of small sensor nodes, also known as motes, distributed over an interested area [27]. It can be used to monitor a certain physical phenomenon, such as temperature, humidity, vocality, motion and so on, from the environment.

1.1.1 In a nutshell

The main components of a sensor node include an antenna, a transceiver, a storage, a controller, a sensing unit, and a power source. Each component has a specified capability. The antenna and transceiver transmit and receive information in a wireless channel. The storage saves data temporarily and the controller governs data processing. Different sensing units, such as acoustic sensor and seismic sensor, have capabilities of sensing different events. The power source is to provide sensor energy.

Depending on the different type of applications, sensor nodes can be either stationary or mobile. *Static sensors* are not capable of changing their positions after deployment, whereas *mobile sensors* with actuation components can move under their own control.

In such a network, each sensor has sensing, computing, and communicating capabilities. It first senses the environment and collects data, then processes and transmits the gathered data information to a powerful *sink node* (or base station). Next, the information will be forwarded to the Internet or other networks for data further processing. Figure 1.1 shows an example process of a wireless sensor network. Sensor S senses a fire event. It then transmits the sensed information to one of its neighbors. The neighbor then relays the message to one of its neighbors. Via multiple hops, this message reaches the data sink to be transfered to a different network.

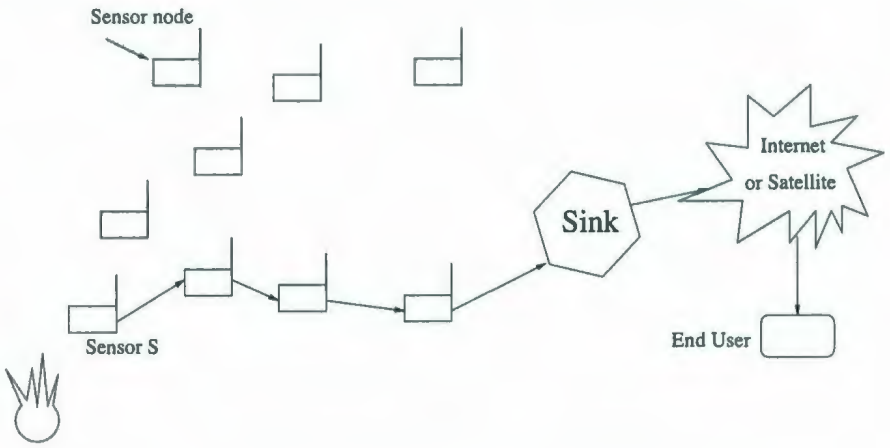


Figure 1.1: A wireless sensor network

1.1.2 Design considerations

Some critical goals of a wireless sensor network include: to provide satisfactory network sensing coverage, to preserve energy, to ensure network connectivity, and to use less number of sensors. Each objective is explained as follows.

- *Maximizing network sensing coverage*

  The sensing coverage of a sensor is defined as a disk area with itself in the center. The radius of the disk is called *sensing range*. Network sensing coverage is the union of the disks induced by all sensors in the network. In other words, it is the area that can be monitored collectively by the sensors in the network. Network sensing overage can be used to measure the quality of service of a sensor network [10], as a sensor network with a good coverage rate is able to provide more information of the environment it monitors. The question is how we can maximize the network coverage. We know that maximizing the network coverage means minimizing the overlaps between sensor nodes. In order to reducing the overlaps between the sensing coverage of sensors, we may hope that sensors are deployed as far apart from each other as possible. In other words, network sensing coverage can be improved by constructing sparse networks. Decreasing overlapping area not only increases the network sensing coverage, but also reduces the signal interference and message collisions. Less interference results in fewer retransmissions of lost message. A number of algorithms have been developed to optimize sensing coverage [4][12][21][39][35][42][43].

- *Minimizing energy consumption*

  Low power consumption requirement is one of the most important constraints

on the operation of sensor networks. The energy costs in operating a sensor network include moving nodes, sensing events in the environment, and transferring information. In a single-hop network topology, a sensor node can directly communicate and exchange messages with any other sensors in the networks. However, a sensor network is often too large and the long distance transmission by sensor nodes is not energy efficient. It is impossible for each sensor node to directly exchange messages with every other node. It is therefore necessary that sensor nodes can transmit their data using a way of multi-hop communication. In such a multi-hop fashion, sensors can communicate with others via the relay of some intermediate sensors. However, excessive communication between sensors would consume much energy. 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, preserving energy in order to increase the lifetime of sensor nodes is extremely challenging. Many researches have been carried out focusing on how to reduce the energy consumption [24][9][10][7][13][39][42].

- *Network connectivity and data transportation*

  Sensors communicate in a wireless channel using the communication technology like Bluetooth, ZigBee, Ultra Wideband (UWB) and so on. Each sensor node has a certain transmission power and a higher transmission power allows a sensor to send message over a longer distance. *Communication range* is used to measure the transmission power of a sensor. The larger the communication range, the better the transmission power. We say a network is a connected network if each sensor node is able to communicate directly to its *neighbors* which are

the sensors within its communication range, and communicate indirectly to other nodes within the network. In order to eventually sending the collected information from all sensors in the network to the base station, the sensor network has to be constructed as a connected network [24][8][41].

One critical goal of a sensor network is to forward the sensed data to a sink node. Routing the message created by a sensor node to a sink node may have multiple paths due to the large scale of a sensor network. Therefore, data routing is a very important issue in wireless sensor networking. Flooding and gossiping are two conventional routing protocols. In flooding, each sensor node rebroadcasts every received data packet to all of its neighbors and continues this process until the packet arrives at the destination. Flooding is very easy to implement, but has the drawbacks of implosion due to duplicated messages are sent to the same node, and overlaps caused by sensing the same event and sending similar packets to the same neighbor. Gossiping is a slightly enhanced version of flooding which can significantly reduce the number of routing message sent by sending the packet only to a randomly selected neighbor rather than broadcasting [1].

Many routing protocols have been proposed and they include flat-based routing, hierarchical-based routing, and energy-aware routing [2]. Flat-based routing includes SPIN, Directed Diffusion, and so forth. SPIN incorporates negotiation before transmitting data and ensures that only useful information will be transfered. Directed Diffusion is different from SPIN in terms of the on demand data querying mechanism it has. The sink node broadcasts interest messages to

find sources. The interest messages are the descriptions of a task. Each sensor that receives the interest sets up a gradient toward the neighbors from which it receives the interest. The gradient specifies both the direction where to forward the data and the status of the demand. In flat-based routing, all nodes are assigned equal roles, while they play different roles in hierarchical-based routing. Routing in sensor networks has attracted a lot of interest and many new routing mechanisms are developed by considering the characteristics of sensor nodes and the limitations and requirements of sensor networks. Detailed surveys are described in [1][2].

- *Minimizing number of sensors needed*

  Statistically, we know that more sensors will lead to a better network sensing coverage given a random distribution of sensor nodes. Since there is always a cost associated with any type of sensor node, we can not afford to deploy an arbitrarily large number of sensors. Therefore, minimizing the number of sensors needed for a certain application is also an important issue in sensor networking.

It is very challenging to develop algorithms to satisfy the above goals at the same time since some of them are conflicting.

Sensing coverage and energy conservation are two conflicting objectives. In order to maximize the network sensing coverage (i.e. minimize the overlaps between each sensing coverage), it is desirable that sensors are deployed as far apart from each other as possible. However, this cause that some sensors have to move to new positions and consume some power energy. The very energy constrained nature of

such networks makes the tradeoffs between sensing coverage and energy consumption especially necessary.

It is obvious that the more sensors are deployed, the more network coverage obtained. Network sensing coverage and the number of sensors deployed are conflicting as well.

1.1.3 Applications

A wireless sensor network may carry different types of sensors and can be used in various applications of different domains, such as military, medical, and environmental.

1. Military Applications

   A wireless sensor network usually has densely and randomly distributed, and self-organized characteristics. It is particularly suitable for the application in bad battlefields, including tracking the movements of warfare entities, monitoring the military strength, equipments, and materials, and assessing the opponents' quality, quantity, and possibly, intention.

2. Medical Applications

   Some medical applications of sensor networks include using medical sensors to help doctors and nurses to monitor the status of patients from a remote site. A number of wireless medical sensors, such as pulse oximeters, blood pressure monitors, and heart rate monitors have been designed and developed [28].

3. Environmental Applications

   Wireless sensor networks can trace the migration of birds and insects, observe the effects of the environmental changes to crops, monitor the quality of air,

and so on. It can also be used to detect forest fire or flood at a high spatial resolution and in a much more timely fashion.

In 2002, UC Berkeley and Intel researchers embedded 32 sensors in and around the burrows of the Leach's Storm Petrels on the Great Duck Island. They successfully used the sensor network to collect climate and burrow activity information and to estimate the habitat of the Leach's storm petrels. They calculated that the sensors have sufficient power to operate for the next six months [22]. In the summer of 2002, 43 nodes were deployed to the island. This time they operated the sensor network for four months to see how the system would perform. They collected and analyzed environmental data. The monitoring showed very high node failure rates, yet yielded valuable insight into sensor network operation which is not obtainable in an indoor deployment [25].

Wireless sensor networks have very wide application future. It not only has application value in the above mentioned fields, but also is able to be applied to many other fields, such as home and traffic etc. Wireless sensor networking is a fast-growing and exciting research area, and has attracted much attention and scientific interests during the past decade. We can forecast that wireless sensor network will be everywhere in the future.

1.2 Motivation for Optimizing Sensor Deployment

One of the most important issues of wireless sensor networking is the deployment of static or mobile sensors in the region of interest. Sensors must be deployed before they can provide useful data. An optimal deployment can let the network to collect

more data, while provide the maximum possible utilization of power resources.

With different applications considered, the deployment of sensor networks may vary. In some environments, the positions of sensors can be predetermined and placed one by one manually or deterministically using, say, a robot in the interested field. This is typically for static sensors. But in some dangerous or unknown environment, it is not possible to manually or deterministically deploy sensors. Therefore, mobile sensors can be deployed by dropping from an aircraft. This random deployment does not always cover the given area well, so an automated position adjustment after this initial deployment is necessary. In the former case, the positions of static sensors can be calculated before their actual deployment using more powerful computers other than sensor nodes. While in the latter case, mobile sensors have to cooperate with each other to fine-tune their positions. We call it automated sensor deployment and this thesis is motivated by our interest in this perspective.

We know that wireless sensor networking has a number of technical challenges. Automated sensor deployment, thus, also ought to be coupled tightly as part of the solution. Due to the small size of sensors, they are equipped with small batteries and have small energy resources. The lifetime of a sensor is determined by its battery life and it can not operate for an infinite amount of time. Therefore, sensors are limited in communication and mobility. It is not practical for a sensor to communicate directly to another sensor far away, even if it could, since this will consume much energy. Due to the failures of some nodes, mobile sensors may need to move to replace the failed nodes. It is also not feasible for a mobile sensor to travel a long distance. Therefore power conservation becomes one of the biggest challenges of automated sensor deployment. Another key issue is to handle data locally since a

sensor network is usually large in scale. Each sensor should operate based on its local view of the entire network to conserve energy. Additionally, sensor positions are changed periodically as they deploy themselves. They must have self-configuring capabilities to adjust and adapt the dynamic changes of their environments. It is more challenging to design an automated sensor deployment algorithm which is able to conserve power energy, process data locally, and be adaptive to dynamic change of environment.

The automated sensor deployment lately has been studied in such fields like computational geometry, robotics, fuzzy logic, and swarm intelligence. Although there are some solutions to this problem, my interest in tackling this problem is to extend the traditional Cooperative Coevolutionary Algorithm (CCEA) to be applicable to distributed and localized computing. An existing cooperative coevolutionary algorithm is a distributed evolutionary algorithm but its computation needs global information. Therefore, it can not be directly applied to localized distributed computing problem. This motivates us to develop a more flexible and powerful CCEA model which is more suitable for localized distributed environments.

1.3 Objectives & Contributions

The first and most important objective of this thesis is to develop a new localized distributed system based on the traditional cooperative coevolutionary algorithm. To achieve this first objective, we study and analyze the traditional model of CCEA. We discover that the existing CCEA does not satisfy our localized requirements since some computation in it is still based on the global information which is not available

in the environment of localized distributed computing. In this thesis, we present a new CCEA model that is able to support dynamic, localized, and distributed network applications and utilizes local information only to achieve a global objective.

The second objective is to apply this new model to optimize the automated sensor deployment problem. To achieve this objective, we first do a survey on this problem to see what researches have been done in this field. Localized algorithms are a primary design goal in wireless sensor networks. We have found that a simple distributed computing algorithm for automated sensor deployment would require sensors to construct their local partial solutions based on local information only and to periodically exchange the results of local computation with the neighboring sensors. We hope that the proposed model is effective and efficient by providing it to the automated sensor deployment, a typical application of Localized distributed system.

In the thesis, we develop two innovative coevolutionary computation frameworks, called LODICO and LODICO/D, for optimization tasks in distributed computing. They are completely localized distributed algorithms and can be applied to a broader application domains in localized distributed environment. Both LODICO and LODICO/D have the following three important features.

1. Localized and distributed evolutionary algorithm

   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 automated deployment through localized information exchange and distributed evolutionary computing.

2. Flexible and dynamic problem decomposition

   Every sensor node is responsible for dividing the global problem into a subproblem according to the most current sensor positions. Since the sensor positions change as the deployment progresses, the network structure also changes. As a result, the decomposition of the problem must be redone iteratively. This is contrast to the traditional evolutionary algorithm where the solution each population evolves is fixed throughout the execution of the algorithm. One consequence of this dynamic problem decomposition is that the populations that collaborate for fitness evaluation also change during the algorithm execution.

3. Energy efficient partial fitness evaluation

   Because each population 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 from all other subsolutions.

LODICO/D is an extension of LODICO. Therefore, it inherits all features of LODICO. Additionally, it allows the interaction among neighboring computing agents during the evolutionary process by providing two operation modes for effective and efficient evolutionary search. Under the LODICO/D algorithm, it models the poten-

tial neighboring interactions and uses that to improve fitness and to help sensors to escape their local optima, which is contrast to the CCEA where each local evolutionary algorithm is executed in isolation. We believe that this thesis is an important contribution to CCEA. We have implemented the LODICO and LODICO/D algorithms to solve the automated sensor deployment problem, and the simulation results show that they are effective in solving this type of problem.

1.4 Organization

In the thesis, we propose LODICO and LODICO/D, two localized distributed algorithms, to optimize sensor network deployment problem. It explores many aspects associated with using LODICO and LODICO/D in distributed networking environments.

We first introduce the automated mobile sensor deployment problem of wireless sensor networks and review related works in Chapter 2. Then, in Chapter 3 the background of traditional evolutionary algorithms and cooperative coevolutionary algorithms is provided. We emphasize on issues why the traditional cooperative coevolutionary algorithms can not be utilized directly in localized environments and what kind of model is expected to satisfy the localized distributed constraints. We present the first algorithm, LODICO, in Chapter 4. In Chapter 5, we go ahead with LODICO/D, which extends LODICO to more general cases. It facilitates local interactions between the neighboring computing components during the evolutionary process to help sensors to escape from local optimal positions. In Chapter 6, we test LODICO and LODICO/D using computer simulation. We observe that both

algorithms can be applied to the applications of localized distributed environments. Last, we conclude the thesis with a summary of contributions and future directions in Chapter 7.

Chapter 2

Literature Review on Sensor Deployments

The deployment of sensor nodes is the first step in establishing a sensor network. Once sensor nodes are deployed, networks are established automatically. The function of sensor networks is to collect data from the environment they are in and to periodically transmit the data to a base station. It will be a productive sensor network if each sensor in the network can collect plentiful data without overlapping the data collected by other sensors. Therefore the positions of sensors influence significantly their capabilities of collecting information from the environment. Each sensor has a small battery and therefore needs to minimize power consumption in order to extend its lifetime. Various techniques have been proposed to optimize sensor deployment [5][7][11][13][18][20][31][33].

The deployment of a wireless sensor network can be carried out in two major ways: pre-deployment and post-deployment. The goals of both approaches are to meet

some critical networking objectives, such as maximizing network sensing coverage, minimizing energy consumption, ensuring the network connectivity, and minimizing the number of sensors deployed.

2.1 Pre-deployment Approach

Pre-deployment approach calculates or estimates the number and the positions of the sensors before they are actually deployed. *This approach is typically used for static sensors deployment in a known environment.* After network topology is determined, the actual deployment is then carried out by human beings or mobile robots [9][15][18][21].

Research on pre-deployment methods mostly takes a centralized approach. Distributed algorithms are not necessary since a computer program can be run on a powerful computer before the physical deployment.

Liu and Mohapatra [21] develop a sensor network pre-deployment method for linear topology. They introduce two problems, IDEAL and HIE, with the same objective of maximizing the total coverage given the lifetime requirement. In IDEAL, each sensor's energy supply is heterogeneous. Total energy constraint is given and energy can be allocated arbitrarily among the nodes. The network dies only when there is no energy left in any node. In contrast, HIE assumes the network is a homogeneous energy network in which each sensor has the same fixed initial energy. Greedy algorithms are used to solve these two problems.

In Isler et al. [15], 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 [18] study the deployment problem using a multi-objective genetic algorithm. Their goal is to balance two conflicting objectives, maximizing the network sensing 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. [13] consider a hybrid sensor network which consists of a mixture of regular small sensors and more powerful micro-servers. They employ tabu search to decide where the micro-servers should be placed so that the lifetime of the network can be maximized.

2.2 Post-deployment Approach

In some dangerous or unaccessible environments, it is impossible to manually deploy sensors. Therefore mobile sensors are placed randomly in the field initially. This initial random placement does not usually give a good coverage and, thus automated adjustments of their locations is necessary. This is the post-deployment approach and *we call it automated mobile sensor deployment.*

Post-deployment approach of sensors has been studied using a variety of techniques. Howard et al. [11] describe 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 [42] 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. When two sensors are too close, the repulsive force intends to apart from each other. While when two sensors are far from each other, the attractive force intends to pull them closer. A one-time movement is carried out when the positions of sensors are identified to conserve energy.

Wang et al. [32] focus on repairing 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 MiniMax.

Chellappan et al. [6] propose a flip-based algorithm and optimize both the coverage and the total number of flips. Flip-based sensors have limited mobilities. They can flip only once to a new location and the flip distance is bounded. Their objective is to determine optimal movement plan for sensors so that the coverage is maximized while minimizing the total number of flips required. They construct a virtual graph based on the initial deployment and determine the optimal movement plan from the virtual graph.

Krause et al. [20] address deployment of role assignment of sensor nodes to maximize network lifetime while preserving the coverage. More recently, it has also been demonstrated that computational intelligence techniques, such as fuzzy logic [29], swarm intelligence [40] and evolutionary computation [17][16] can be effective in sensor deployment.

2.3 Our Approach

In general, our work in this thesis shares similar objectives as the above works on post-deployment problem. We optimize the network sensing coverage as well as the energy consumption. The different feature of our work from theirs is that we develop a new cooperative coevolutionary computation model motivated by the application of automated mobile sensor deployment. The model is an extended work of the current cooperative coevolutionary algorithms. As presented in Chapter 6, computer simulation shows our model is very effective for directing mobile sensors to find their target locations with good coverage while using less energy consumption.

Chapter 3

Evolutionary Computation Models

3.1 Traditional Evolutionary Algorithms

Evolutionary Algorithms (*EAs*) are search methods 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 [3].

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. 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 as an output. With the goal of finding the best solution to an optimization problem, evolutionary algorithms combine selection, fitness evaluation, crossover, and muta-

tion operators to develop generations of populations. Figure 3.1 illustrates the basic steps in an evolutionary algorithm.

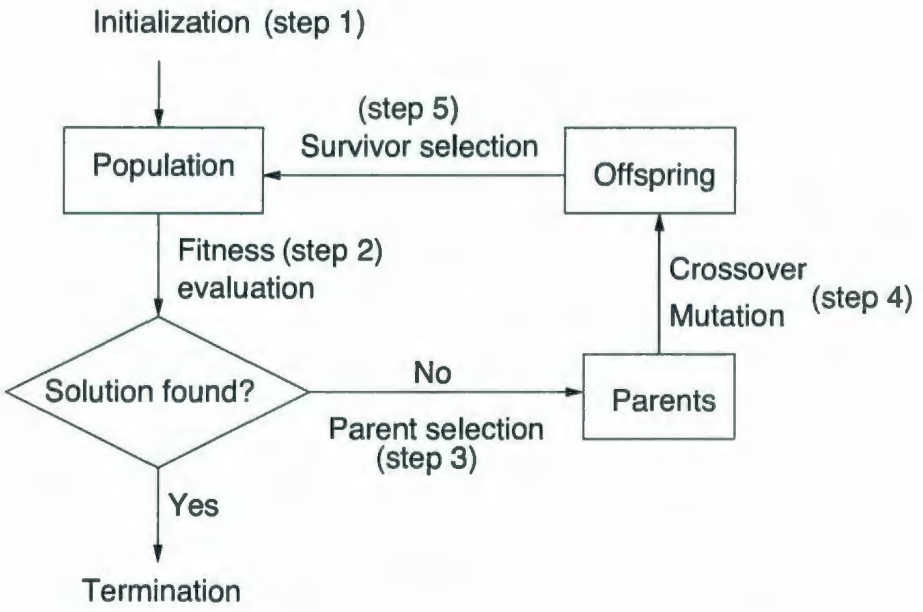


Figure 3.1: Flow chart of a simple evolutionary algorithm

First, a number of individuals in a population are randomly generated (step 1). At each evolutionary step, or each generation, the fitness of each individual is calculated based on a fitness function (step 2). Individuals with better fitness are selected as parents (step 3). Then their genetic representations (or genotypes) are recombined through crossover and mutation (step 4) to produce new solutions, called offspring. A crossover involves exchanging the genetic materials in the genotypes of two or more parents. A mutation is a random change of an individual's genotype to produce offspring. These offspring then compete with each other and with the previous best solutions to survive in the following generation (step 5). An EA is essentially an

iterative reproduction of generations of individuals. Frequently, the fitness of the population improves as the evolution continues. The process continues until certain termination conditions are met. The pseudocode of a traditional evolutionary algorithm is shown in Figure 3.2.

```
Begin
    Initialize population
    Evaluate the fitness of each individual in the population
    Repeat until termination condition is met
        1. Select parents to reproduce;
        2. Breed through crossover and mutation to create offspring;
        3. Evaluate the fitness of the offspring;
        4. Select individuals for the next generation;
End
```

Figure 3.2: Pseudocode of a general evolutionary algorithm

The following subsections describe the details of each components of an EA.

3.1.1 Components of evolutionary algorithms

1. Data Representation

   The first step of implementing an evolutionary algorithm is to transferring the real world problem in hand to a format of EA. Data representation defines a

set of solutions that form the search space of the given problem. Either a fixed or a variable length representation may be used to encode candidate solutions.

- Genotype and Phenotype

  Genotype is an EA solution representation of a real problem. Phenotype is the behavior of the genotype it represents. The crossover and mutation operators take place in the genotypes while the fitness evaluation is applied on the phenotypes.

- Population, Individual, Parents, and Offspring
  - Population: is a set of candidate solutions.
  - Individual: is a candidate solution in a population.
  - Parents: are two or more selected individuals for reproduction.
  - Offspring: are new candidate solutions produced from the selected parents.

2. Population Initialization

   The initial population of individuals are normally generated randomly. It is also possible to bias the initial population to sample a particular area of the problem search space.

3. Selection

   - Parent Selection and Survivor Selection

     An individual's selection can take place at two different stages of evolution: parent and survival. In both cases, the better an individual's fitness is, the more chance it would be selected.

- Parent selection: extracts individuals from an existing population for reproduction.
- Survivor selection: selects from one generation to create the basis of the next generation. It extracts individuals from parents and offspring populations to produce a new population.

- Selection Methods

  There are many different ways to select fitter individuals. Here we summarize several popular selection methods [3][23].

  - Roulette wheel selection

    Each individual is given a probability of being selected, which is proportional to its fitness. The fitter individuals have a greater chance of being selected than the weaker ones. When the fitness between the individuals differs greatly, the fittest individual may be over selected but other individuals have very little chance to be selected.

  - Tournament selection

    A group of individuals are randomly selected from the population. The best individual is the winner. The larger the tournament size, the stronger the selection pressure.

  - Rank selection

    All individuals in the population are sorted by their fitness and their ranks (instead of fitness) are used for selection. Rank selection will work better than the roulette wheel selection when the fitness of population differs greatly, as roulette wheel selection can over select but

rank selection would not.

- Elitism

  Some number of the best individuals are kept at each generation and are copied over to the population of the next generation.

4. Crossover

   It combines two parent solutions to create one or two new solutions with some of the features of each parent. The idea behind crossover is that the generated offspring may be better than its parents if it takes the best characteristics from its parents. There are many different types of crossover methods, such as uniform crossover, single point crossover, two point crossover, and arithmetic crossover. Arithmetic crossover generates offspring by a linear combination of the parents and we use the arithmetic crossover in this thesis as you will see in the next chapter.

5. Mutation

   It randomly modifies some of the genetic material of an individual to produce new solutions. Mutation introduces new materials to the population pool, hence it can be used to maintain the genetic diversity of the population.

6. Fitness Evaluation

   Fitness gives the performance of a candidate solution. Each individual is assigned a fitness value based on how well it solves the given problem. Individuals with a higher fitness value have a higher probability of contributing good solutions in the next generation.

7. Termination Condition

   Sometimes EA may run forever without reaching a satisfied solution, therefore a termination limit is necessary to stop an algorithm.

3.2 Cooperative Co-Evolutionary Algorithms

Cooperative Co-Evolutionary Algorithm ($CCEA$) is a special evolutionary algorithm proposed in [14][26]. Unlike the traditional EA [23], which solves a problem by searching the whole solution space, CCEA divides the problem into many subproblems and searches the subsolution space simultaneously. The subsolutions are then combined to form the whole solution to the problem. Since the subsolution space is smaller, the algorithm may find better solutions faster.

Coevolutionary search involves two or more populations. 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, which is equivalent to *one ecosystem generation.* At the end of each ecosystem generation, the genetic information at a population is shared among all species via a representative from each species for fitness evaluation. Figure 3.3 gives a high-level flow of CCEA, where R_i is the representative of species i. This figure only gives an illustration of cooperation of fitness evaluation for species 1. As you see in the figure, in order to evaluate each individual, we have to collect information from *all* other species to form a complete solution to the problem. Other species have the

common process of fitness evaluation. The outer evolutionary process is terminated when a certain termination condition is met.

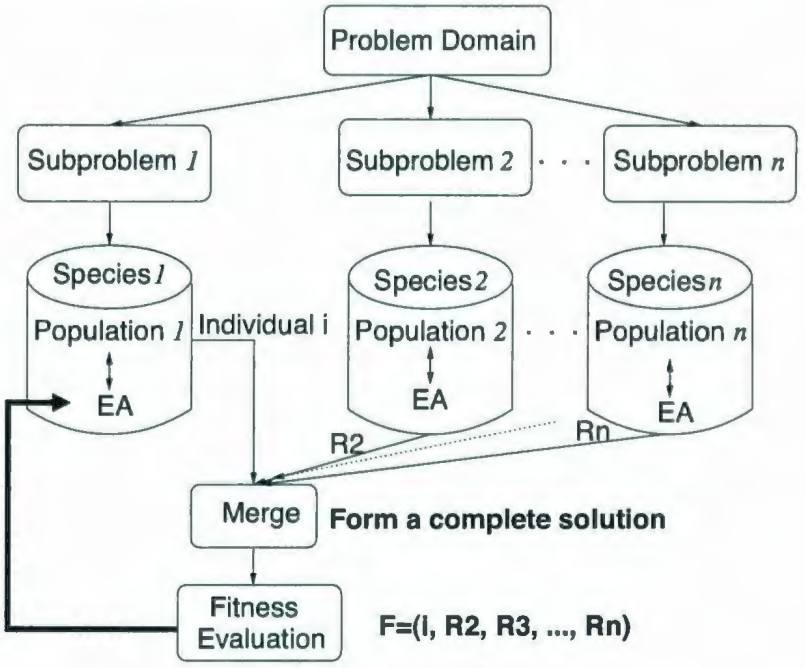


Figure 3.3: A high-level view of CCEA

3.2.1 Additional components of CCEA over EA

1. Subpopulation & Species

   In EAs, one population is employed to evolve the solutions. However in CCEA, a problem is decomposed into a number of subproblems, or species. For each subproblem, there is a separate population, which contains the set of candidate solutions to that subproblem.

2. Sub-search Space

   EA searches the whole solution space while CCEA searches subsolutions in a number of sub-search space simultaneously.

3. Ecosystem Generation

   Each ecosystem generation in CCEA involves a number of generations as introduced in Section 3.1.

4. Fitness Evaluation

   The fitness of an individual in CCEA depends on its ability to collaborate with representatives from other species, while in EA it is evaluated in isolation and based on its own genotype. There are many ways to select a representative, such as the current best individual or a random individual.

3.2.2 Related work on CCEA

CCEA has been successfully used in some applications. In [34], Wang and Wu use CCEA for robot path planning of collision avoidance problem. The algorithm can be executed in parallel and asynchronously while the representatives from each species are selected synchronously. Tan et al. [30] present a cooperative coevolutionary algorithm to co-evolve multiple subsolutions for a multi-objective optimization problem. They propose a distributed cooperative coevolutionary algorithm (DCCEA) for concurrent computing while there is no direct communication among species and all communications are performed between the species and a central server. The difference between the above work and our work is that their fitness evaluation is based on the global information while we evaluate individuals using local information only.

There has been research investigating *problem decomposition* and the efficiency of single-best collaboration during the evolution. Wiegand and colleagues [38][37] 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 [36] proposed dynamically merging the species when inter-dependency of variables in cross populations was detected. Kim and Ryu [19] went farther by allowing not only merging but also splitting the species when the inter-dependency no longer exist during the evolution. Our cooperative coevolutionary framework also provides dynamic division of species.

3.3 Desired CCEA and Features of Our Work

To work with the constraints of automated mobile sensor deployment: dynamic change of the network, local information exchange, and energy conservation, the following mechanisms have been developed so that coevolutionary algorithms can be applied effectively in localized and distributed environments, such as network computing. The expected framework is depicted in Figure 3.4, where R_i is the representative of species i. For better readability, we only illustrate the process of the fitness evaluation for species 1. For each individual in species 1, its fitness is decided by the combination of its genotype and the representative genotypes from neighboring species. Species 1 has two neighbors: node 2 and node 3, so its fitness is evaluated as $F = (i, R_2, R_3)$. Other species have the similar evaluation method.

In this thesis, we develop two novel CCEA models, called LODICO and LOD-

ICO/D, and their features are given below.

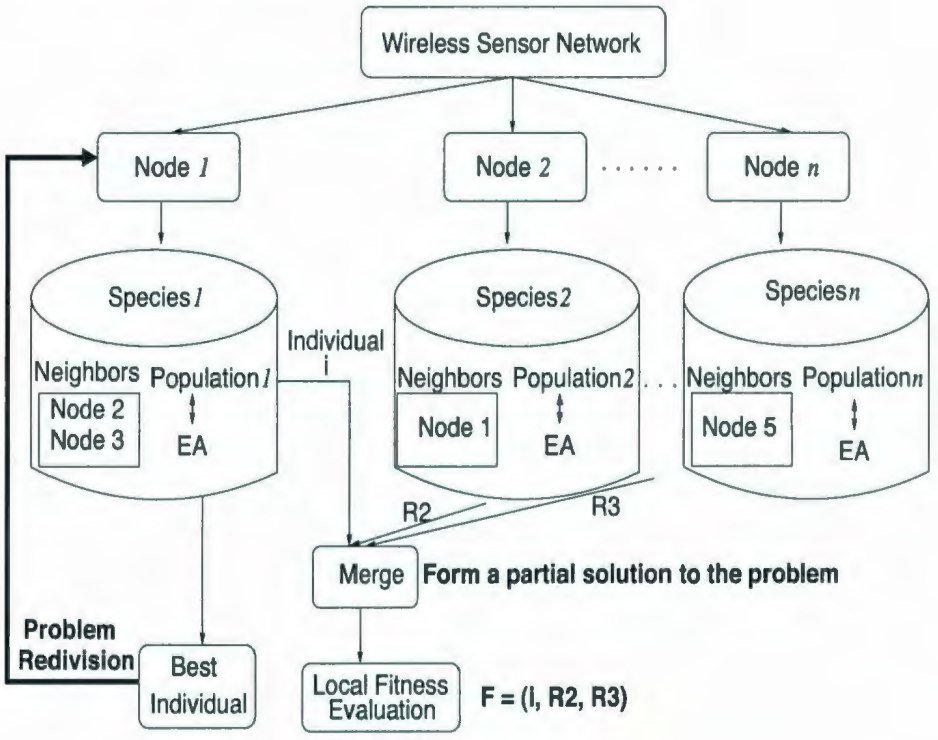


Figure 3.4: A high-level view of the expected CCEA framework

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**

   We alternate two operation modes to help sensors to escape local optima. In spirit, the first mode (mode I) is similar to the *splitting species* proposed in [19] and the second mode (mode D) is similar to the *merging species* proposed in [36]. 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 one ecosystem generation cycle. Alternating these two modes can accelerate the search process while avoiding local optima.

Chapter 4

Localized Distributed Sensor Deployment via Coevolutionary Computation (LODICO)

LODICO is a completely localized distributed algorithm in that each *local 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. In this chapter, we detail the design of LODICO and defer experimental analysis of it for Chapter 6.

4.1 LODICO Overview

LODICO consists of three major stages: *planning*, *computing*, and *moving*. A complete pass of the three steps is called an *ecosystem cycle*. LODICO is executed on all

sensors of the network in parallel for a number of iterations until a coverage requirement is met.

In the planning stage of 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 cycle. In the computing stage, the sensor executes a local evolutionary algorithm 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 of each sensor needs to be recalculated as the network structure is altered. LODICO starts the next ecosystem cycle by exchanging position information within neighborhood 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 are reached. Figure 4.1 gives the high-level flow of the implementation. The implementation is based on the following assumptions:

- Each sensor knows its own location by using the global positioning system (GPS) or some positioning algorithms.
- A sufficient number of sensors are deployed so that they can potentially 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 executes a sequence of ecosystem cycles, where each cycle

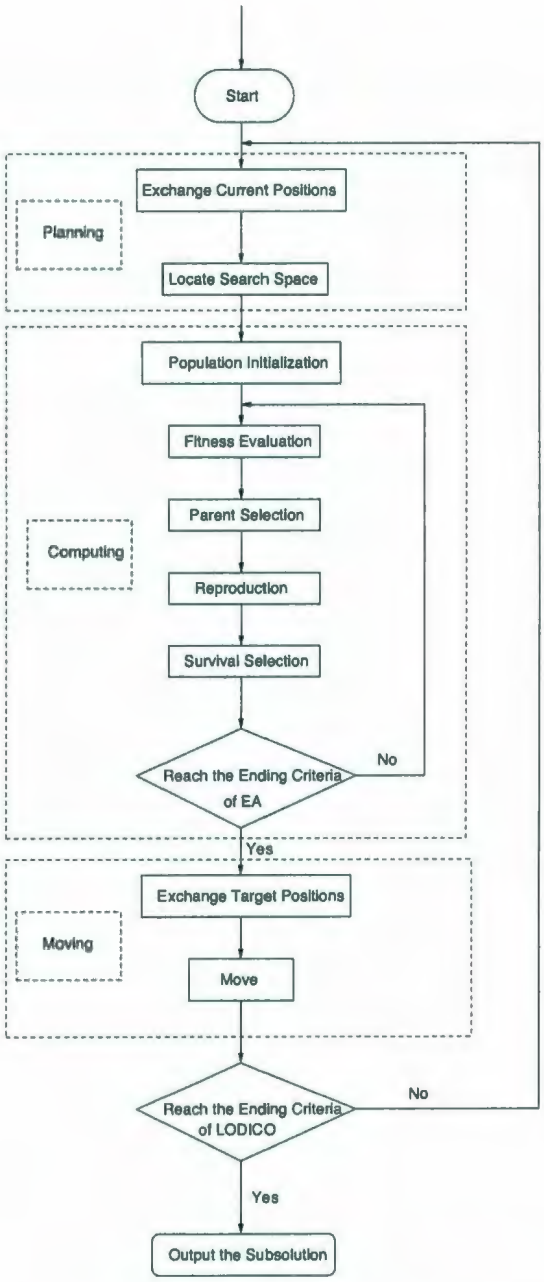


Figure 4.1: LODICO flow chart

consists of 3 steps: planning, computing, and moving. We explain each step in the following sections.

4.2 Planning

In the planning stage, a sensor determines a partition of the entire deployment region to execute its local evolutionary 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 ecosystem cycle.

At the beginning of each ecosystem cycle, LODICO decomposes the entire deployment area based on the current sensor locations in the network: each search space 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 ecosystem 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 thesis, we define the search space of a sensor to be equal to its sensing

region. 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 4.3.

The idea of the limit of search space can be illustrated by the diagram in Figure 4.2. 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 restricts node a to 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 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 limitation of search space can guarantee that the new coverage of a sensor node only have overlaps with neighboring coverage which is obtainable.

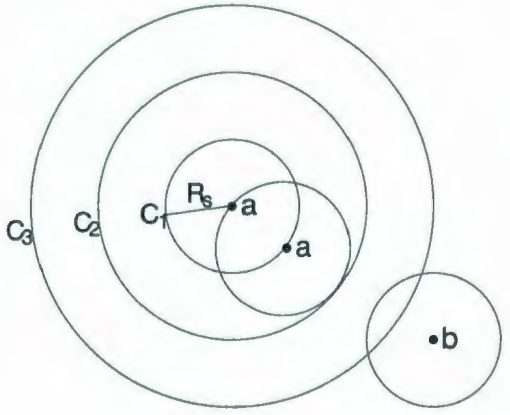


Figure 4.2: Analysis of potential movement and overlaps

4.3 Computing

Each sensor executes an instance of a local EA to compute where it will move to at the end of the cycle in the computing stage. This section describes each component of the local evolutionary algorithm.

4.3.1 Problem Representation

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, where n is the total number of sensors in the network. You may want to ask why we use such a long representation with unobtainable information. This representation is suitable for the dynamic change of the network structure after each sensor movement. We do not need to redefine a new representation for each new environment. It is also able to be used for our extended work as described in the next chapter.

Each element i ($i = 1, 2, \ldots, n$) is the position $\{x_i, y_i\}$ of sensor i in the deployment area (See top diagram of Figure 4.3). 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 Figure 4.3). It stores 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 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. Note that there is exactly one element with value $\star$ and that the number of 1's equals to the number of neighbors (See bottom diagram of Figure 4.3). Notice that we would use a variable-length genotype representation. However, our fix-length approach provides the flexibility to facilitate the dynamic problem division. 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.

Figure 4.3: The 2-chromosome genotype representation

4.3.2 Evolution

Each sensor population (*local population*) maintains a set of individuals, P, each of which corresponds to a sensor positioning which is a subsolution to the entire network. Here, an individual encodes its own position and those of its neighbors. Each sensor initiates its individuals by generating $|P|$ 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' via arithmetic crossover, where the location value of an offspring is the mid-point of the gene values of its parents. 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 selected as the target position of the sensor, where g is a small integer as part of the EA configuration. At the end of the g generation, the sensor moves to the target positions.

4.3.3 Fitness Evaluation

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. For a given node, the sensing coverage is the union of its sensing area and the sensing areas of its neighboring sensors. 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. 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

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

As we discussed in the last section, the search space limitation can ensure that the new coverage of a sensor node would not overlap with non-neighboring nodes. This is very important for the coverage fitness evaluation as the change of local network structure would not affect or invalidate the fitness evaluation.

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

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

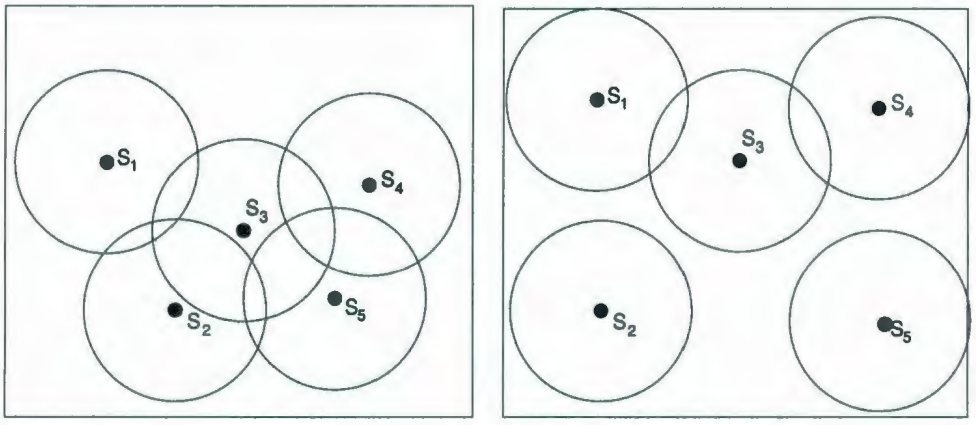
where w is a weight parameter for coverage-movement tradeoff purposes.

Although the fitness evaluation of LODICO only uses local information from its neighboring nodes, it will be shown (See Chapter 6) that the computed fitness value is able to drive the evolutionary search to find target positions that give good overall coverage and energy consumption.

4.4 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. Figure 4.4 gives the illustration of sensors before and after their movement. Each sensor in the network tries to move away from their neighbors to increase its local coverage.

As sensor nodes move to new positions, the network structure is changed. The previous neighbors may not be neighbors in this new ecosystem cycle, and some non-neighbors may become neighbors. As a result, each node has to re-decompose



(a) Before move (b) After move

Figure 4.4: Sensor positions before and after movement

the problem based on the current sensor positions. The consequence of this dynamic change of network structure is that the search space for each node is changed and each local population is also changed during the algorithm execution. This is different than the traditional evolutionary algorithm where the solution each population evolves is fixed throughout the execution of the algorithm. LODICO can reflect and manipulate the dynamic changes of network environments effectively.

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.

Chapter 5

LODICO with Dynamic Interaction of Neighboring Species (LODICO/D)

LODICO is a completely localized distributed algorithm and it is flexible for the dynamic changes of the network structure. With the constraints of that each sensor only has local information within its neighborhood, LODICO can still direct the evolutionary search to find a good solution based on the local fitness evaluation.

LODICO/D is an extension of LODICO. It inherits all features of LODICO. Additionally, it allows the interaction between neighboring species during the evolutionary process by providing another operation mode, mode D, for effective and efficient evolutionary search. It models the potential neighboring interactions and uses the imaginary neighboring moving plan to improve its local fitness and to help sensors to escape from their local optimal positions. This is a powerful extension to the

traditional CCEA, where each species evolves in isolation.

5.1 Local Optima

We have implemented the LODICO to solve the automated sensor deployment problem (See Chapter 6). The evolutionary process of LODICO can be very fast since the search space of each sensor is small. However, in some special situations, sensors may get stuck in their local optima and do not move anymore even though they have not reached their global target. For example, in Figure 5.1, $S_1, S_2, \ldots, S_6$ are six sensors

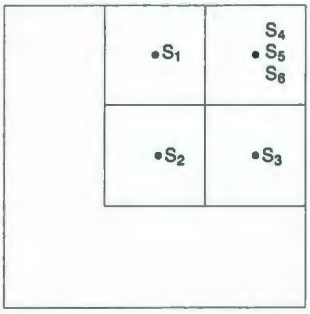


Figure 5.1: An example of local optima

used to cover an area, where S_4, S_5, and S_6 have the identical location. For easier understanding, we use a square area as the sensing region of a sensor. It is obvious that the sensing coverage would increase if some sensors move to the left or the lower region of the deployment 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 very beginning of the cycle. Wherever a sensor moves to within its search space, its fitness can not be improved.

This happens for non-square sensing region as well. Sensors would not move if the fitness improvement is within a certain threshold, say ϵ, as the little improvement of fitness may be established on a large moving distance with much power energy consumption.

In order to obtain locations that give a better coverage than the current ones do without much power consumption, the neighboring sensors need to have different locations. LODICO/D provides this flexibility by allowing both the locations of a sensor and its neighboring sensors to evolve, and helps the populations to escape the local optimum. For the local optimal example given in Figure 5.1, Figure 5.2 illustrates a possible result of escaping local optimal positions after running LODICO/D for one ecosystem cycle.

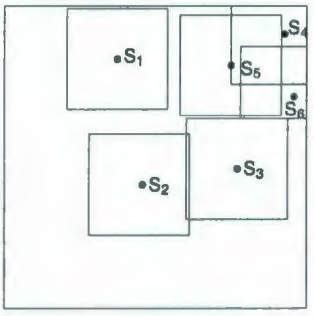


Figure 5.2: Escaping local optima

5.2 Bi-modal Operation

In LODICO/D, each local population is executed using one of two operation modes: mode I (Independent) and mode D (Dynamic). Mode I executes LODICO in Chap-

ter 4. That is, it evolves only a sensor's position. Mode D enables imaginary neighboring moving to realize local optimum escape. In this case, a local population evolves the positions of its neighbors along with that of its own in each ecosystem cycle. It is called imaginary because the modification of a neighbor's position only facilitates the calculation but has no physical effect on the new position of that neighbor. Regardless of the operation mode, the fitness of an individual always considers part of the entire sensor network.

5.3 Mode D

Mode D, as mode I, has the three stages: planning, computing, and moving, as in LODICO. They are only different in the planning stage.

5.3.1 Planning for extended search space

The search space of a sensor is a limited scope within which the sensor can move at the end of the ecosystem cycle. 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 reasonably fast. In mode D, the search space of a local population is multiple-dimensional: the x, y locations of a sensor and its neighboring sensor. Unlike mode I where the neighboring sensor locations that are used for fitness evaluation are fixed throughout the ecosystem cycle, the neighboring sensor locations are also evolved potentially. It models the potential local interactions between species and uses that to improve the local estimate of fitness and to help escaping local optima.

In either case, a sensor exchanges its location information with others within its communication range. Using this information, mode I 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 contrast, a sensor in mode D defines a search space not only includes its own position but those of its neighbors.

5.3.2 Uniform computing

The computing stage of LODICO/D can be executed the same way as LODICO.

- Fixed-length Representation

  Recall that in LODICO, we use a fixed length genotypic representation for each individual in each local population. There are a number of advantages by using it. First, it is suitable for *all* sensor nodes in the network *with different local network structures*. Second, it adapts the environment changes of sensor networks. After a sensor moves to a new position, its network structure changes and this representation can still reflect such change. Third, it fits both LODICO and the extended algorithm, LODICO/D. Though there are redundant information in this representation, the *one* representation fits many different cases.

- Uniform Crossover and Mutation

  We use the arithmetic crossover in LODICO, where an offspring is generated by taking the average value of its two parents' gene values. For a non-evolved neighbor, its gene values are same for all individuals. Therefore the average value of two identical value remains unchanged. That is, the arithmetic crossover does not change the genes other than the current sensor of mode I. For mode

D, since both the sensor along with its neighbors are evolved, crossover does make change to the gene values of both the sensor and its neighbors. We do not use mutation during the evolution in LODICO and LODICO/D. The frequent changes of network structures make the search spaces changed frequently, thus the population diversities are still maintained. The designed uniform evolutionary process is able to be used for both mode I and mode D.

5.3.3 Individual moving

In mode D, each population evolves the positions of its own sensor and those of its neighbors. In other words, there exists information overlaps between neighboring populations. This poses a question of how to resolve the conflicts and which decision we are going to adopt or use. A species can either adopt its own evolved decision or make a compromised decision based on its target position with those from its neighbors incorporated. Here, when a sensor is to move, it only adopts its own decision though the decision has been obtained by evolving a neighborhood of sensors. 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.

Each sensor node then moves to its target position in the moving stage once it is calculated. As the network structure is altered, LODICO/D starts the next ecosystem cycle by exchanging position information within its neighborhood and selecting an operation mode for the next cycle.

5.3.4 Combination with mode I

Each run of LODICO/D is a combination run of mode I and mode D. Mode I can run very fast as each node has a small search space. Mode D can not be run independently. The increase of its local fitness does not always lead to the improvement of global fitness since the target position is based on the imaginary movements of neighbors. Therefore, it has no idea when the algorithm converges. This is solved by detecting the *coverage holes* around the sensors. A coverage hole is an area that is not covered by any sensor in its neighborhood. If there is any hole, the local evolutionary algorithm is switched to mode D for one ecosystem cycle and switched back to mode I the following cycle, since mode I runs faster than mode D. This check is carried out for each sensor population. LODICO/D combines mode I and mode D to accelerate the process of evolution as well as escaping from local optima. The general flow of LODICO/D is given in Figure 5.3.

5.4 Discussion

We take this opportunity to discuss our fixed-length representation and the moving decision of mode D.

1. Fixed-length representation for a uniform design framework

   In LODICO and LODICO/D, we use a fix-length genotypic representation for individuals in each local population. We argued that this genotypic representation is not only suitable for both proposed algorithms, i.e. LODICO and LODICO/D, but also it lays down a foundation for a uniform representation for

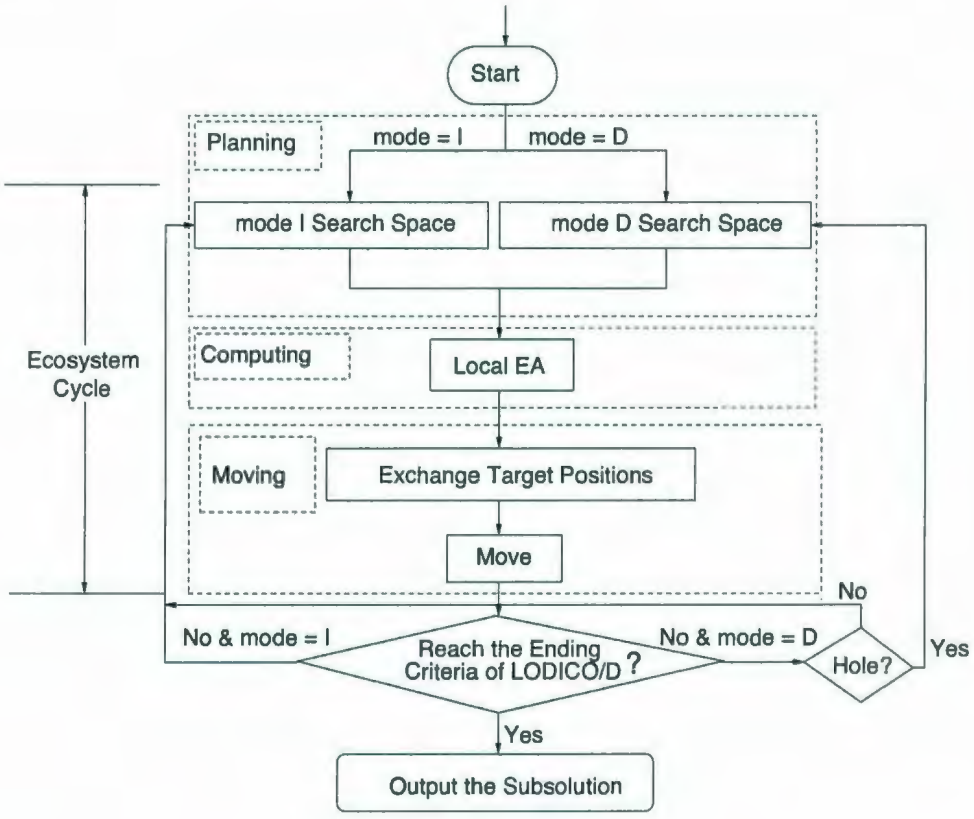


Figure 5.3: LODICO/D flow chart

all sensors in the network despite that their local neighborhood network structures are different and changed frequently. In addition, the use of a separate genotype reveals a significant generalization of fitness evaluation in EA.

The separate genotype enables a much more general fitness evaluation. A neighbor's neighbors may provide some useful information. Because such information has been propagated via multiple hops, it can be dated to a degree. Therefore, it would be beneficial to include its contribution to fitness in a "fuzzy" way. This can be realized by using values in [0, 1] to populate this second genotype. The extreme values of 0 and 1 correspond to contributions from non-neighbors and neighbors in LODICO and LODICO/D. In contrast, a value in between can control the level of contribution from an "informative" non-neighbor, i.e. the closer to 1 the value is, its contribution is considered more accurate. From the perspective of designing evolutionary algorithms, this reveals another area of future work. That is, fuzzy fitness evaluation and tolerance of missing inputs for global fitness computation.

2. Moving Decision

   In LODICO/D, we evolve both the positions of a sensor and its neighbors. The movement plan is decided by the current sensor only and is imaginary for its neighbors. Alternatively, We could also incorporate the moving suggestions from neighbors. To do this, the node should be informed of the suggested new position of itself from its neighbors. A simple comprise is to take a weighted average of these new positions including its own. Since the current sensor has a better idea of its environment than neighbors, its own decision can take a greater

weight in the final decision while considering the suggestions of its neighbors with a lesser weight. This would be another interesting extension in future work.

Chapter 6

Experimental Analysis

To evaluate the performance of LODICO and LODICO/D, we have implemented a computer program to simulate the deployment of autonomous mobile sensor networks with various initial positions. The experimental settings and results are given in the following sections.

6.1 Experimental Settings

We run computer simulations using various numbers of sensors in three different size of fields: $100 \times 100\text{m}^2$ (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. The initial sensor positions are uniformly distributed at random. Table 6.1 summarizes the parameter values used to carry out our simulation.

Three metrics, *moving distance*, *convergence time*, and *sensing coverage*, are used to evaluate the experimental results. Moving distance is the average distance that

Table 6.1: Simulation Parameters

Parameters	Settings
Sensing range R_s	20m
Communication range R_c	60m
Deployment area size U	$100^2, 200^2, 300^2(\mathrm{m}^2)$
Number of sensor nodes n	
Small size area: $100^2\mathrm{m}^2$	10, 12, 14, 16;
Medium size area: $200^2\mathrm{m}^2$	40, 50, 60, 70;
Large size area: $300^2\mathrm{m}^2$	70, 80, 90, 100
Population size $\|P\|$	10
Number of offspring $\|Q\|$	5
Ecosystem cycles g_e	30
Number of generations in each ecosystem cycle g	5

a sensor in the network has to travel from its initial position to the final position. Convergence time is the number of ecosystem cycles it takes for *all* sensor populations to converge, i.e. the best individual fitness stops improving. Sensing coverage is the percentage of the deployment field that is covered by the deployed sensors. The experimental results are presented and analyzed in the following sections.

6.2 Preliminary Study

We use two sets of preliminary experiments to study the effect of the weight parameter (w) on the algorithm performance. The first set deploys 40 sensors to the medium size field and the second set deploys 100 sensors to the large size field. Each set of run is conducted using 5 different w values: 0, 0.5, 1.0, 1.5, and 2.0. Figure 6.1 gives the performance results averaged over 30 runs and evaluated using the three metrics (moving distance, convergence time, and sensing coverage). The results show that the value of w can influence the network performance considerably.

Figure 6.1(a) shows that without the weight control (w=0), the sensors travel a long distance from their initial positions to the final positions, which consumes a lot of the battery power. The situation improves dramatically when w is increased; even a small value of w can reduce the moving distance per sensor from 364.4m to 43.6m for 100 sensors deployed in a large size field (second series in the chart). As w further increases to 2, the moving distance per sensor node is reduced to only about 2% of the moving distance when w is 0. The same trend holds for the smaller networks such as 40 nodes deployed to a medium size field (first series in Figure 6.1(a)).

When considering the algorithm convergence time, a greater value of w leads to

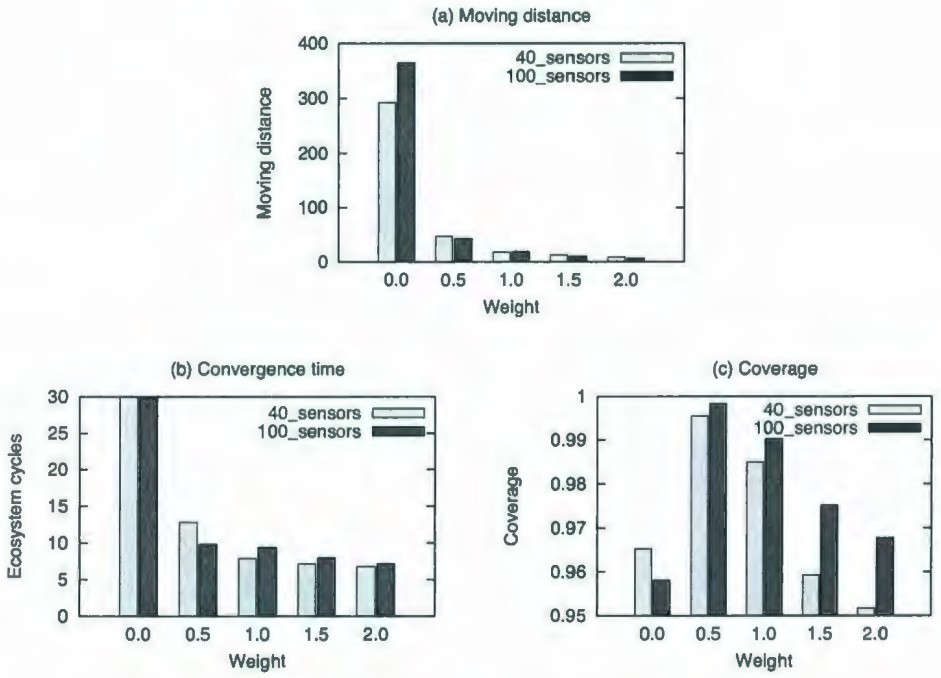


Figure 6.1: Influence of distance weight (w) on the 3 performance metrics

a smaller number of ecosystem cycles needed for the populations to converge to the final sensor positions, because it can suppress excessive node movement effectively (see Figure 6.1(b)). Figure 6.1(c) shows that $w = 0.5$ gives the largest sensing coverage of 99.5% at the time when all populations are converged. As the w value continues to increase, the network coverage is decreased.

When $w = 0$, the average number of ecosystem cycles takes to satisfy the convergence requirements is larger than 30 (Figure 6.1(b)), but we only plot them within 30 ecosystem cycles for better readability. We also measure the sensing coverage of the sensors after 30 ecosystem cycles and the population either converges or the fitness fluctuates at a certain level. This indicates that traveling a large distance does not help sensors locate positions that provide good coverage.

The 99% confidence intervals on the means of the three metrics with the 5 different w values are given in Figure 6.2.

When all three metrics are considered, $w = 1.0$ gives a good balance between large coverage, small moving distance and convergence time. We therefore **use $w = 1.0$ to conduct the rest of our experiments**. Meanwhile, when $w = 1.0$, we see that the populations take 7.9 and 9.44 ecosystem cycles to converge (Figure 6.1(b)) for the two sets of experimental runs. Figure 6.2(b) shows that its 99% confidence intervals are within 10 ecosystem cycles. Thus $g_e = 30$ **is a reasonable choice to use for the rest of our experiments**.

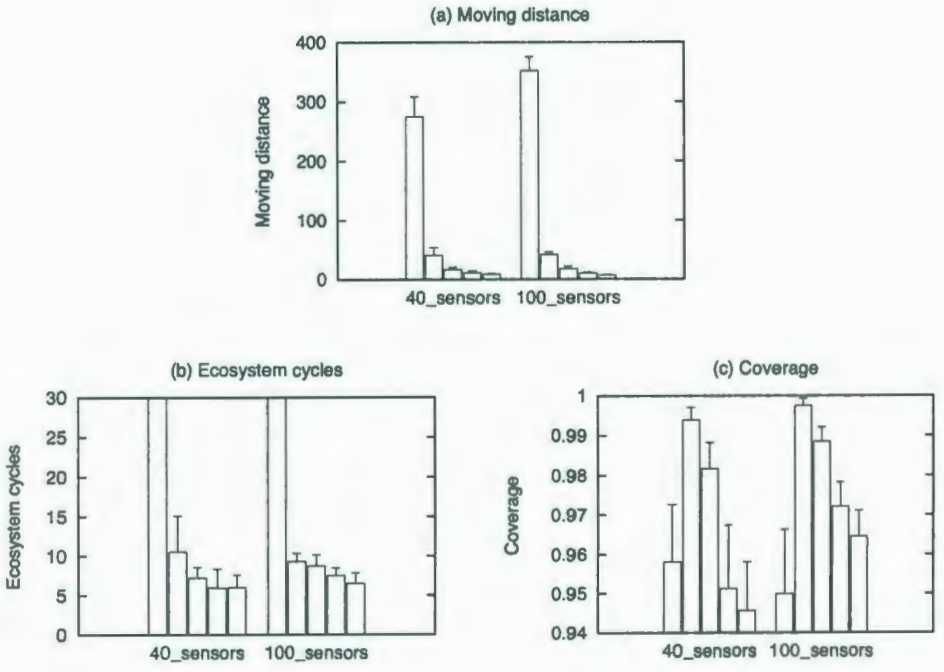


Figure 6.2: 99% confidence intervals on the means

6.3 Studies on LODICO

We study the performance of LODICO under different network sizes using a different number of sensors as that given in Table 6.1. The results are analyzed in the following sub-sections.

6.3.1 Fitness improvement

We first use three configurations of smaller nodal density (i.e. 12 sensors in a small size region, 50 sensors in a medium size region, and 80 sensors in a large size region) to show coverage improvement. The results of random one run for each configuration is given in Figure 6.3. The curves indicate that the global network coverage improves rapidly during the first few ecosystem cycles and the populations converge around the 7th cycle. You may notice that, for the medium and large size network, the populations converge at around 98% coverage of the deployed field. In Section 6.4, we will show that LODICO/D can improve the coverage to 100%.

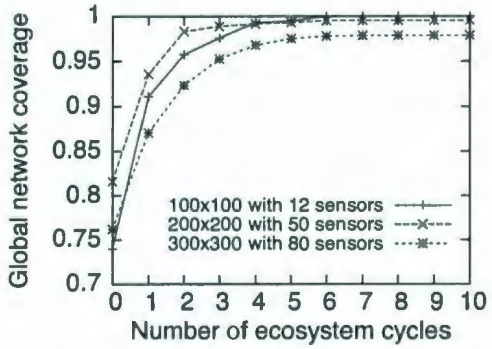


Figure 6.3: Coverage improvement under LODICO

Figure 6.4 is the results averaged over 20 runs for the three configurations, respectively. We plot 10 ecosystem cycles and the average global coverage can be achieved at least 97% at the 10th ecosystem cycle. The evolutionary search is able to quickly find a solution that gives good coverage.

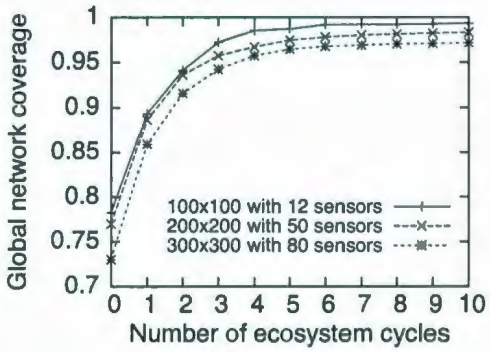


Figure 6.4: Average coverage improvement under LODICO

6.3.2 Coverage vs. moving distance

To study the global network coverage and the moving distance over time, we make one run deploying 10 sensors to a small size field, one run deploying 50 sensors to a medium size field, and one run deploying 100 sensors to a large size field. Figure 6.5 shows that they all have a similar pattern: 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.

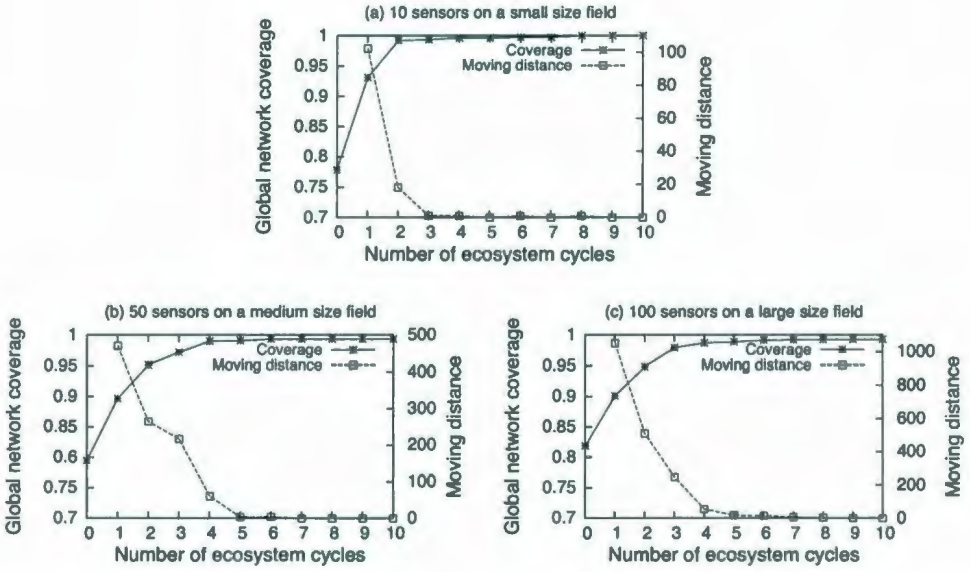


Figure 6.5: Coverage vs. moving distance

6.3.3 Global network fitness & local network fitness

We know that each ecosystem cycle includes a number of generations. In this single run experiment, we randomly deploy 10 sensors to a small size field to investigate the change of the local fitness of each sensor at the end of each ecosystem cycle and each generation.

Since a sensor moves to a new position at the end of each cycle, the induced new local network also changes. Figure 6.6 shows the increase or decrease of the local fitness of each sensor at the end of each ecosystem cycle as the change of local network structure. Within each ecosystem cycle, the best individual local fitness always increases or remains unchanged. Figure 6.7 shows the best individual fitness improvement within each ecosystem cycle for sensor 1. All other nodes have the similar patterns. The global fitness keeps on climbing up as the evolution progresses

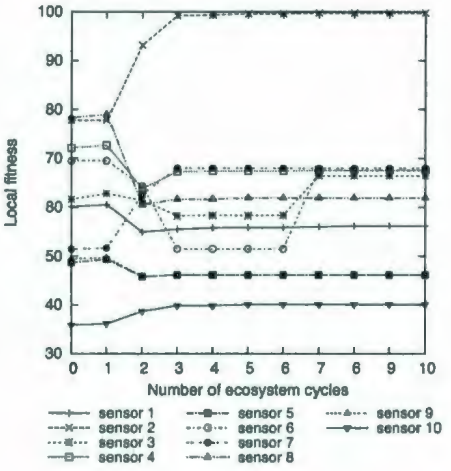


Figure 6.6: Local fitness changes after each ecosystem cycle

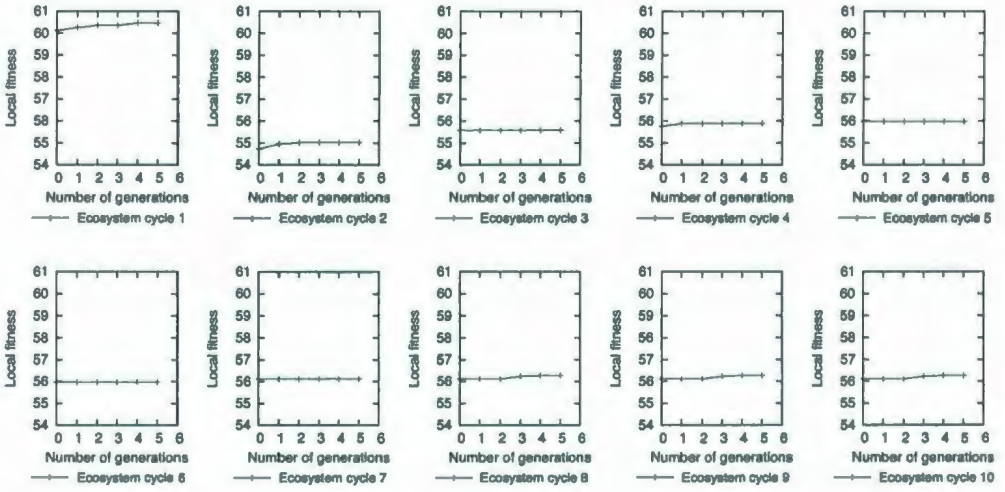


Figure 6.7: Sensor 1: the best individual local fitness improvement at each ecosystem cycle

as shown in Figure 6.8. This is because the fitness of each individual is based on how well it *collaborates* with its neighbors to provide local network coverage. This designed local fitness function is able to direct the evolutionary search of each local population toward a target position which can cooperate well with other sensors. As a result, a global network with good coverage can be generated.

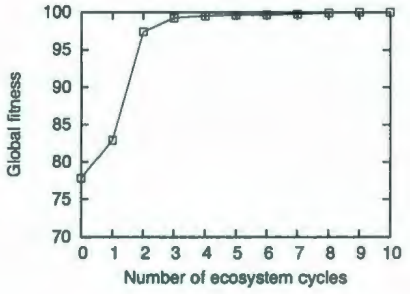


Figure 6.8: Global fitness improvement after each ecosystem cycle

6.3.4 Influence of the number of sensors

In this group of experiments, we want to see how nodal density affects the deployment performance. We randomly deploy 4 different numbers of sensor nodes (Table 6.1) to the 3 different size of deployment fields (small, medium, and large). When the best individual in all local populations stops improving, we evaluate the three performance metrics: network coverage (Figure 6.9(a)), convergence time (Figure 6.9(b)), and moving distance (Figure 6.9(c)) averaged over 30 runs. The 99% confidence intervals on the means for the three metrics are given in Figure 6.10.

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

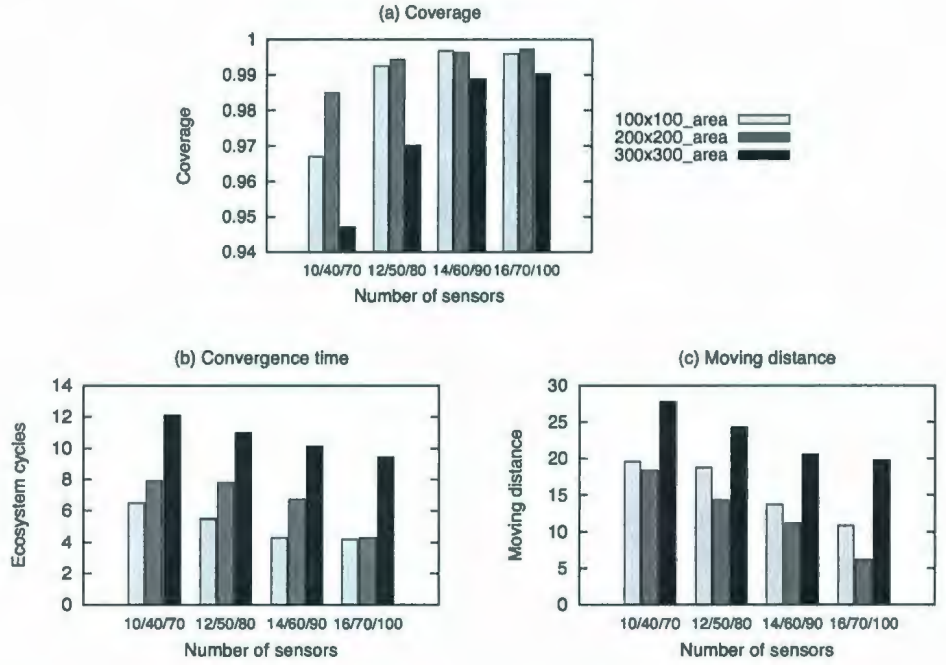


Figure 6.9: Influence of number of sensors

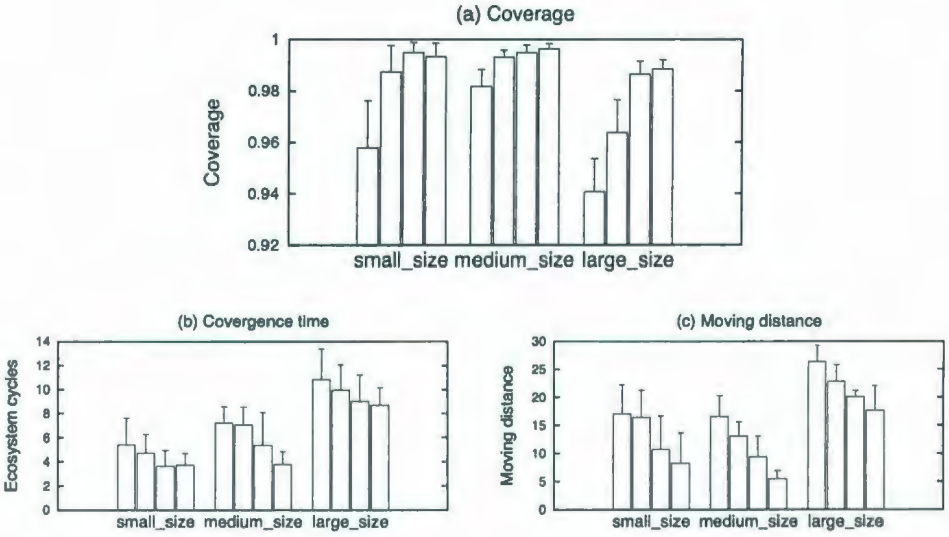


Figure 6.10: 99% confidence intervals on the means

moving distance decrease. This is reasonable as a larger number of sensors in the network make it easier to cover a wider area of the deployed field with a smaller amount of time and a shorter moving distance.

6.4 Studies on LODICO/D

In this section, we compare the performance of LODICO and LODICO/D, and investigate how LODICO/D helps sensor nodes to escape local optimal positions.

6.4.1 Comparison of LODICO & LODICO/D

As shown in Section 6.3.1, LODICO can provide deployments of sensor networks with good coverage. However, it does not always give 100% coverage for the deployed field. In the case where 100% coverage is required, incorporating mode D in the algorithm may help providing better coverage.

To investigate the benefit of mode D in helping the populations to escape local optima and deliver better solutions, we make 30 runs for each deployment of 40, 50, 60, and 70 sensors in a medium $200 \times 200\text{m}^2$ square area using LODICO and LODICO/D respectively. The two sets of experiments are carried out as follows: one operates LODICO and the other operates LODICO/D where mode I and mode D are alternated with 5 and 1 ecosystem cycles intervals, i.e. 5 mode I cycles followed by 1 possible mode D cycle.

This alternation is 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

Table 6.2: Average Coverage Comparison Between LODICO and LODICO/D

	LODICO		LODICO/D	
sensors	average coverage	100% covered	average coverage	100% covered
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

holes. 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 Chapter 5). This check is carried out for each sensor population, and only those whose best individual induces a local network with a coverage hole operate mode D while others remain operating mode I in the following ecosystem cycle.

The average coverage of 30 runs and the numbers of runs achieving 100% coverage are given in Table 6.2. Overall, both setups provide very good coverage. Nevertheless, LODICO/D with the alternation of mode I & mode D delivers a larger number of runs that produces 100% coverage.

6.4.2 Local optima

To validate our hypothesis that mode D improves performance by helping the populations escape local optima, we conduct another single experiment deploying 10 sensors to a small size field, where the sensor initial locations give a local optimum coverage (64%).

The simulation is carried out by alternating 2 cycles of mode I followed by 1 possible cycle of mode D. The best global fitness (See Figure 6.11) shows that after 2 cycles of no fitness improvement, the fitness declines after the execution of mode D at the third ecosystem cycle. This fitness decline is caused by a large moving distance (See Figure 6.12), indicating that the sensor has escaped the local optimum. After that, the global fitness starts to climb and eventually reaches 100% coverage.

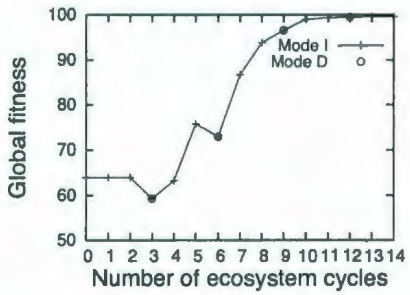


Figure 6.11: Global fitness under Mode I and D

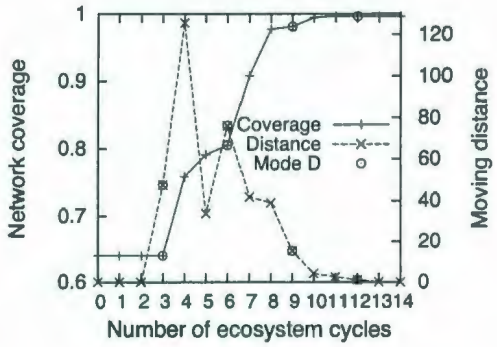


Figure 6.12: Network coverage vs. moving distance

6.4.3 Initial corner deployment

To evaluate the system performance under the situation where all sensors are initially deployed to a corner in the deployment field, we conduct another single experiment with randomly initialized 10 sensors in the corner of $40 \times 40\text{m}^2$ area in a $100 \times 100\text{m}^2$ field (See Figure 6.14(a)).

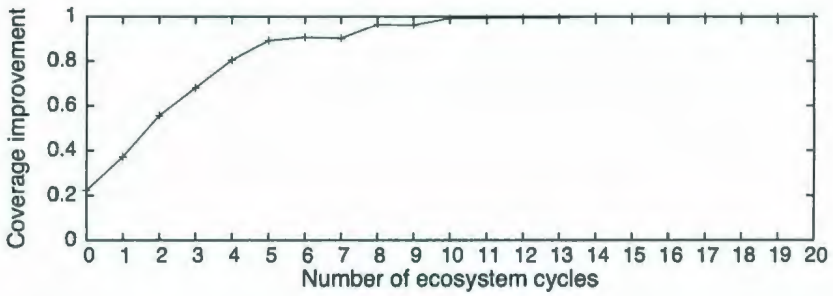
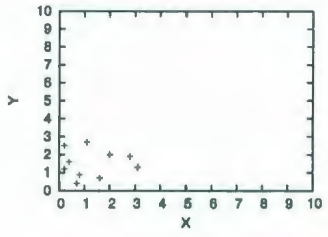
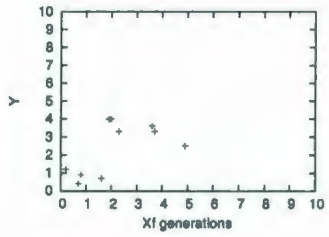


Figure 6.13: Coverage improvement of deploying sensors in a corner

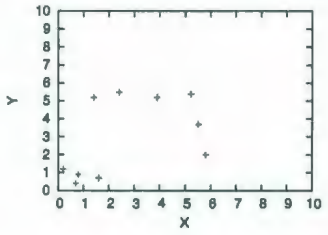
This is a hard situation for any deployment algorithm as the initial network coverage is only 22.28% as shown in Figure 6.13. The alternation of mode I and mode D has improved the fitness quickly, and after 5 ecosystem cycles, almost 90% of the given field is covered by the sensors. Further, at the 14th ecosystem cycle, the whole area is fully covered. Figure 6.14 shows the sensors movement process at the end of each ecosystem cycle.



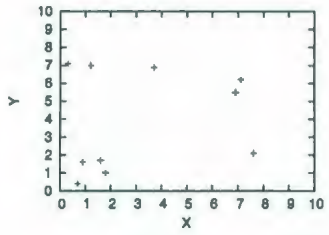
(a) Ecosystem cycle 0



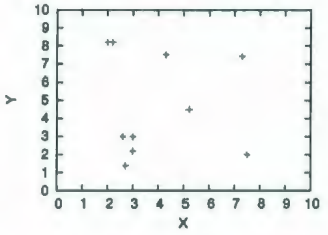
(b) After ecosystem cycle 1



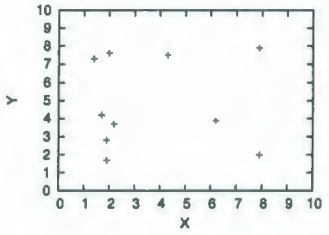
(c) After ecosystem cycle 2



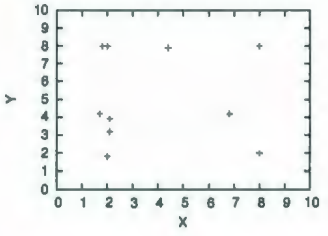
(d) After ecosystem cycle 3



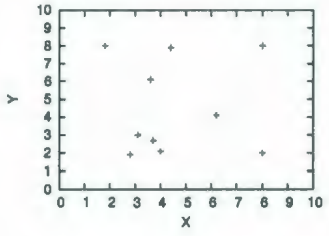
(e) After ecosystem cycle 4



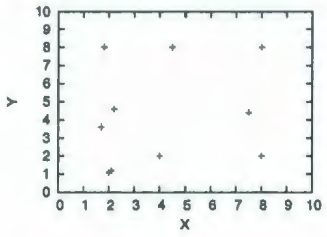
(f) After ecosystem cycle 5



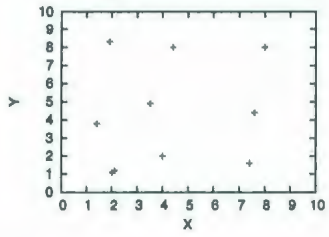
(g) After ecosystem cycle 6



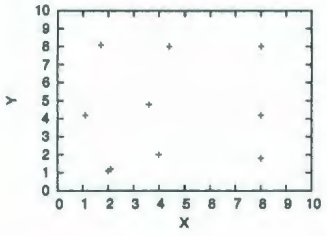
(h) After ecosystem cycle 7



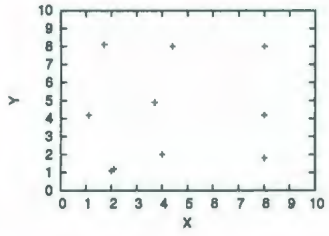
(i) After ecosystem cycle 8



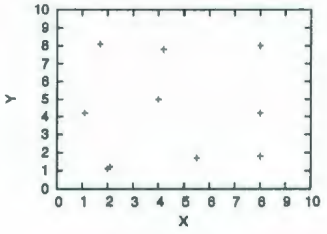
(j) After ecosystem cycle 9



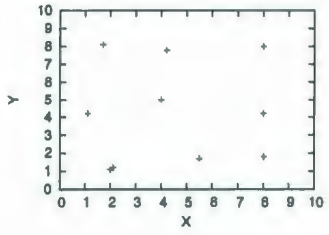
(k) After ecosystem cycle 10



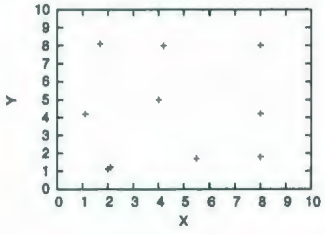
(l) After ecosystem cycle 11



(m) After ecosystem cycle 12



(n) After ecosystem cycle 13



(o) After ecosystem cycle 14

Figure 6.14: Position changes of sensors

Chapter 7

Conclusion and Future Work

In this thesis, we propose two innovative CCEA models, LODICO and LODICO/D, to optimize the automated mobile sensor deployment of a wireless sensor network. They are completely localized algorithms which can be executed fully distributed and in parallel at each sensor node.

LODICO and LODICO/D are powerful extensions to the traditional Cooperative Coevolutionary Algorithms in three aspects. First, CCEA evaluates individuals based on the collaboration of *all* species, i.e. a complete solution has to be generated for fitness evaluation, while LODICO and LODICO/D coordinate sensor nodes through localized partial fitness evaluation and information exchange. This makes CCEA applicable to fully distributed computing applications. Second, LODICO and LODICO/D propose a scheme of frequent, distributed, and dynamic problem division, in which no center control is needed and each sensor node divides its own subproblem based on its local information only. This is particularly suitable for the dynamic changes of mobile sensor networks. Third, we know that in CCEA, the

only interaction between species is the fitness evaluation. Yet, LODICO/D models local interactions between neighboring species during the variation of individuals to improve its fitness and help populations escaping local optima.

The simulation results show that LODICO and LODICO/D are effective in obtaining good solutions under dynamic, distributed, and localized condition constraints. They can achieve very high coverage rates with short moving distances in short period of times.

In the future work, we plan to extend mode D to incorporate a sensor's target position with its neighboring moving suggestions. Further, we want to use the fixed-length representation with fuzzy information of farther sensors to form an estimated global view of the network while still using local communication only. This will fully investigate the power of partial and fuzzy fitness evaluation. And yet, more interesting findings are to be made.

Bibliography

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[2] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11:6–28, 2004.

[3] T. Back, D. B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation1*. Institute of physics publishing, Philadelphia, PA, USA, 2000.

[4] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2006.

[5] E. S. Biagioni and G. Sasaki. Wireless sensor placement for reliable and efficient data collection. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003.

[6] 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)*, Nov. 2005.

[7] K. Dasgupta, M. Kukreja, and K. Kalpakis. Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks. In *Proceedings of the 8th IEEE Symposium on Computers and Communications*, Kemer-Antalya, Turkey, Nov. 2002.

[8] R. D'Souza, D. Galvin, C. Moore, and D. Randall. Global connectivity from local geometric constraints for sensor networks with various wireless footprints. In *Proceedings of the fifth international conference on Information processing in sensor networks (IPSN'06)*, pages 19–26, New York, NY, USA, 2006.

[9] D. Ganesan, R. Cristescu, and B. Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. In *Proceedings of Symposium on Information Processing in Sensor Networks (IPSN '04)*, Berkeley, California, Apr. 2004.

[10] N. Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(1):78–92, 2005.

[11] 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.

[12] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of Distributed Autonomous Robotic Systems*, pages 299–308, 2002.

[13] 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.

[14] P. Husbands and F. Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In *Proceedings of ICGA*, pages 264–270, 1991.

[15] 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.

[16] X. Jiang, Y. P. Chen, and T. Yu. Dynamic cooperative co-evolutionary sensor deployment via localized fitness evaluation. In *Proceedings of Parallel Problem Solving from Nature (PPSN)*, Dortmund, Germany, September 13-17 2008.

[17] X. Jiang, Y. P. Chen, and T. Yu. Localized distributed sensor deployment via co-evolutionary computation. In *Proceedings of the IEEE International Conference on Communications and Networking (ChinaCom)*, Hangzhou, China, August 25-27 2008.

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

[19] M. W. Kim and J. W. Ryu. An efficient coevolutionary algorithm using dynamic species control. In *Proceedings of Third International Conference on Natural Computation (ICNC)*, 2007.

[20] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placement: Maximizing information while minimizing communication cost. In *Proceedings of IPSN*, 2006.

[21] X. Liu and P. Mohapatra. On the deployment of wireless sensor nodes. In *Proceedings of international Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks (SenMetrics)*, 2005.

[22] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM workshop on wireless sensor networks and applications*, Atlanta, Georgia, USA, September 28 2002.

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

[24] S. Panichpapiboon, G. Ferrari, and O. K. Tonguz. Optimal transmit power in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(10):1432–1447, 2006.

[25] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson. Analysis of wireless sensor networks for habitat monitoring. *Wireless sensor networks*, pages 399–423, 2004.

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

[27] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, editors. *Wireless sensor networks.* Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[28] V. Shnayder, B. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh. Sensor networks for medical care. In *Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University*, 2005.

[29] 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.

[30] K. C. Tan, Y. J. Yang, and C. K. Goh. A distributed cooperative coevolutionary algorithm for multiobjective optimization. In *Proceedings of the IEEE trasactions on evolutionary computation*, 2004.

[31] G. Wang, G. Cao, and T. L. Porta. A bidding protocol for deploying mobile sensors. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, Nov 2003.

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

[33] G. Wang, G. Cao, and T. L. Porta. Proxy-based sensor deployment for mobile sensor networks. In *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2004.

[34] M. Wang and T. Wu. Cooperative co-evolution based distributed path planning of multiple mobile robots. *Zhejiang University SCIENCE*, 6A(7):697–706, 2005.

[35] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks*, 1(1):36–72, Aug 2005.

[36] K. Weicker and N. Weicker. On the improvement of coevolutionary optimizers by learning variable interdependencies. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1627–1632, 1999.

[37] R. P. Wiegand. *An analysis of cooperative coevolutionary algorithms*. PhD thesis, George Mason University, 2004.

[38] R. P. Wiegand, W. C. Liles, and K. A. D. Jong. The effects of representational bias on collaboration methods in cooperative coevolution. In *Proceedings of the Seventh Conference on Parallel Problem Solving from Nature (PPSN)*, pages 257–268, 2002.

[39] J. Wu and S. Yang. Smart: A scan-based movement-assisted sensor deployment method in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, Mar 2005.

[40] 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.

[41] B. Yener, M. Magdon-Ismail, and F. Sivrikaya. Joint problem of power optimal connectivity and coverage in wireless sensor networks. *Wireless Networks*, 13(4):537–550, 2007.

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

[43] Y. Zou and K. Chakrabarty. Uncertainty-aware and coverage-oriented deployment for sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):788–798, 2004.

Index