On Computing Mobile Agent Routes for Data Fusion in Distributed Sensor Networks

Qishi Wu, *Member*, *IEEE*, Nageswara S.V. Rao, *Senior Member*, *IEEE*, Jacob Barhen, *Member*, *IEEE*, S. Sitharama Iyengar, *Fellow*, *IEEE*, Vijay K. Vaishnavi, *Fellow*, *IEEE*, Hairong Qi, *Member*, *IEEE*, and Krishnendu Chakrabarty, *Senior Member*, *IEEE*

Abstract—The problem of computing a route for a mobile agent that incrementally fuses the data as it visits the nodes in a distributed sensor network is considered. The order of nodes visited along the route has a significant impact on the quality and cost of fused data, which, in turn, impacts the main objective of the sensor network, such as target classification or tracking. We present a simplified analytical model for a distributed sensor network and formulate the route computation problem in terms of maximizing an objective function, which is directly proportional to the received signal strength and inversely proportional to the path loss and energy consumption. We show this problem to be NP-complete and propose a genetic algorithm to compute an approximate solution by suitably employing a two-level encoding scheme and genetic operators tailored to the objective function. We present simulation results for networks with different node sizes and sensor distributions, which demonstrate the superior performance of our algorithm over two existing heuristics, namely, local closest first and global closest first methods.

Index Terms—Genetic algorithms, mobile agents, distributed sensor networks.

1 Introduction

Multiple sensor systems have been the target of active research since the early 1990s [1] with a particular emphasis on the information fusion methods for *distributed sensor networks* (DSNs) [2], [3], [4]. Recent developments in the sensor, networking, and computing areas now make it possible to deploy a large number of inexpensive and small sensors to "achieve quality through quantity" in complex applications. In an important subclass of DSNs that are deployed for remote operations in large unstructured geographical areas, wireless networks with low bandwidth are usually the only means of communication among the sensors. These sensors are typically lightweight with limited processing power, battery capacity, and communication bandwidth. The communication tasks consume the limited power available at such sensor nodes and, therefore, in order to ensure their sustained operations, the power consumption must be kept to a minimum. Furthermore, the massively deployed sensors typically generate a huge amount of data of various modalities, which makes it critical to collect only the most important data and to collect it efficiently. In addition, despite the abundance of deployed sensors, not all sensor data is critical to ensuring the quality of fused information such as adequate detection energy for target detection or tracking.

In conventional fusion architectures, all the sensor data is sent to a central location where it is fused. But, the transmission of noncritical sensor data in military DSN deployments increases the risk of being detected in addition to consuming the scarce resources such as battery power and network bandwidth. To meet these new challenges, the concept of *mobile agent-based distributed sensor networks* (MADSNs) has been proposed by Qi et al. [5] wherein a mobile agent selectively visits the sensors and incrementally fuses the appropriate measurement data. Mobile agents are special programs that can be dispatched from a source node to be executed at remote nodes. Upon arrival at a remote node, a mobile agent presents its credentials, obtains access to local services and data to collect needed information or perform certain actions, and then departs with the results. One of the most important aspects of mobile agents is the security, which is not addressed here, but is being actively investigated [6], [7]. The transfer of partial fusion results by a mobile agent may still have the risk of being spied on with hostile intent; the serial data collection process employed by the mobile agent obviously decreases the chance of exposing the individual raw data. Although there are advantages and disadvantages of using mobile agents [8] in a particular scenario, their successful applications range from e-commerce [9] to military situation awareness [10]. They are found to be particularly useful for data fusion tasks in DSN. The motivations for using mobile agents in DSN have been extensively studied [5]. For a particular multiresolution data integration application, it is shown

- *Q. Wu, N.S.V. Rao, and J. Barhen are with the Center for Engineering Science Advanced Research, Computer Science and Mathematics Division, Oak Ridge National Laboratory, One Bethel Valley Road, PO Box 2008, MS-6016, Oak Ridge, TN 37831-6016. E-mail: {wuqn, raons, barhen}@ornl.gov.*
- *S.S. Iyengar is with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803. Email: iyengar@bit.csc.lsu.edu.*
- *V.K. Vaishnavi is with the Department of Computer Information Systems, Georgia State University, PO Box 4015, Atlanta, GA 30302-4015. E-mail: vvaishna@gsu.edu.*
- *H. Qi is with the Electrical and Computer Engineering Department, University of Tennessee, Knoxville, TN 37996. E-mail: hqi@utk.edu.*
- *K.Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Box 90291, 130 Hudson Hall, Durham, NC 27708. E-mail: krish@ee.duke.edu.*

Manuscript received 11 June 2003; revised 4 Dec. 2003; accepted 16 Jan. 2004. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0092-0603.

that a mobile-agent implementation saves up to 90 percent of the data transfer time due to savings in avoiding the raw data transfers. Also, the conditions under which an MADSN performs better than a DSN are analyzed and the conditions for an optimum performance of the former are determined in [5]. In this paper, we direct our research efforts to the networking aspects of MADSNs with a focus on a new routing method for mobile agents.

The order and number of nodes on the route traversed by a mobile agent determine the energy consumption, path loss, and detection accuracy and, hence, have a significant impact on the overall performance of a MADSN. Low energy consumption, low path loss, and high detection accuracy are always the main goals of a DSN. The computation of a suitable route, in practice, involves a trade off between the cost (energy consumption and path loss) and the benefit (detection accuracy): Visiting more sensors improves the quality of fused data, but also raises the communication and computing overheads. Previously, algorithms based on the *local closest first* (LCF) and *global closest first* (GCF) heuristics have been used to compute routes for mobile agents in MADSNs [5]. Their performance has been quite satisfactory for small DSNs that are systematically deployed in simple environments. However, their performance deteriorates as the network size grows and the sensor distributions become more complicated. In some practical DSNs deployed for target detection and tracking, these two heuristics could generate particularly unsatisfactory solutions since they only consider the spatial distances between sensor nodes. In these applications, several other factors, such as sensor detection signal levels and link power consumption, play a very significant role and, indeed, could be far more important than the physical distance alone. It is very critical to consider all these factors simultaneously in the data fusion process for computing a satisfactory route of a mobile agent.

We consider DSNs with geographically distributed sensors that are deployed for the purposes of target classification and tracking, both of which require the fusion of measurements from a number of sensors. The measurements from the sensors that receive strong signals from the target are often most useful in the fusion process and, thus, the fusion step is preceded by identifying a subset of sensor nodes that "strongly" indicate the presence of a target. By identifying such sensor nodes, the complexity and amount of sensor data for fusion operation can be significantly reduced. Thus, we focus on identifying a route for a mobile agent through such sensor nodes by utilizing the signal strengths of the sensor nodes. The mobile agent then visits these nodes and performs the required fusion of the sensor information available at these nodes. Each sensor can transmit messages with certain energy costs, and the transmissions are subject to wireless propagation losses. We formulate the *mobile agent routing problem* (MARP) in an MADSN as a combinatorial optimization problem involving the cost of communication, path loss due to wireless propagation, and signal energy received by the sensors. The overall routing objective is to maximize the sum of the signal energy received at the visited nodes while minimizing the power needed for communication and the path losses. In particular, the objective function is directly proportional to the signal energy and inversely proportional to the path loss and cost of communication. We show MARP to be NP-complete by using a reduction from a variation of the 3D traveling salesman problem. We then propose an approximate solution based on a *genetic algorithm* (GA) that employs a two-level encoding scheme and genetic operators tailored to the objective function, which aims to achieve the total signal energy equal to or above a given level. Simulation results for networks with different node sizes and sensor distributions show that our algorithm has superior performances compared to LCF and GCF.

The rest of the paper is organized as follows: In Section 2, models for sensor nodes and wireless communication links are described and then the mobile agent routing problem is formulated. The details of the genetic algorithm are discussed in Section 3, including a two-level encoding scheme, derivation of the objective function, and implementations of genetic operators. Simulation results are presented and discussed in Section 4. Concluding remarks are provided in Section 5.

2 Mobile Agent Routing Problem

We now briefly describe the architecture of an MADSN to motivate the formulation of the route optimization problem. An MADSN typically consists of three types of components: *processing elements* (PE), *sensor nodes*, and *communication network* [11]. The processing elements and sensors are usually interconnected via a wireless communication network. A group of neighboring sensor nodes that are commanded by a single PE forms a *cluster*.

2.1 Sensor Nodes

A sensor node, also referred to as a *leaf node*, is the basic functional unit for data collection in an MADSN. A sensor node may have several channels with different sensory devices connected to each of them. Sensor nodes are geographically distributed and collect measurements of different modalities such as acoustic, seismic, and infrared from the environment. The data acquisition is controlled by a sampling subsystem, which provides the acquired data to the main system processor [12]. The signal energy from each channel can be detected individually and processed in the analog front end. A mobile agent migrates among the sensor nodes via the network, integrates local data with a desired resolution sequentially, and carries the final result to the originating PE. The fused data may be used to derive appropriate inferences about the environment based on the application. The *setup* time of a PE accounts for loading the mobile agent code and performing other initialization tasks.

The amount of signal energy that reaches an individual sensor is an effective indicator of how close the node is to a potential target. Once the signal is captured and processed by a sensor node, the strength level of the detected signal is broadcast (through an omnidirectional antenna) to the whole cluster so that a PE in any location has a knowledge of various levels of signal energy detected by all active sensor nodes within the coverage area. In target detection applications, a leaf node with higher signal energy carries

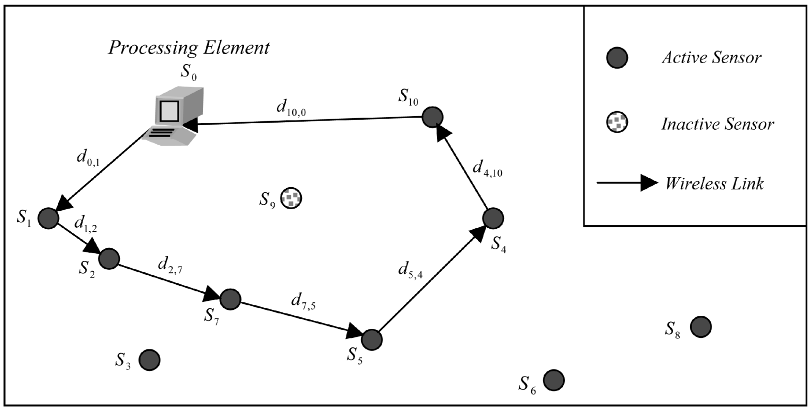


Fig. 1. A simple MADSN with one PE and 10 leaf nodes.

more information and should have higher priority of being visited. To simplify computation, we use a quantitative value to represent the level of signal energy detected by a local sensor node.

2.2 Communication Links

Wireless communication links need to be established between neighboring nodes as the mobile agent migrates along a route. The embedded RF modems on sensor nodes provide such a networking capability with low power requirement. For example, on WINS NG 2.0 platform [12], each node is equipped with two RF modems, both of which support 2.4 GHz *frequency-hopped spread spectrum* (FHSS) communication. The different *clusters* select different "network numbers," which correspond to separate hopping pseudonoise sequences to avoid interferences. We define an abstract link class with only the parameters we are interested in. The detailed radio configuration and wireless link establishment is beyond the scope of the paper.

It is worthwhile to note that the message transmission time between two sensor nodes depends not only on the physical distance between them, but also on the channel bandwidth and the data packet loss rate as well as the size of the message, which includes partially integrated data and mobile agent code itself. In general, the electromagnetic propagation time is almost negligible in short-range wireless communication. Hence, the physical distance is not explicitly considered in our model, but is incorporated as a part of the path loss representing the signal attenuation. The received signal strength below a certain level due to path loss may not be acceptable. The system loss factor is a parameter of the free space propagation model, which is not necessarily related to the physical propagation [13].

2.3 Mobile Agent Routing

A mobile agent is dispatched from a *processing element* and is expected to visit a subset of sensors within the *cluster* to fuse data collected in the coverage area. Generally, the more sensors visited, the higher the accuracy achieved using any reasonable data fusion algorithm will be [14]. But, it is important to select an appropriate route so that the required signal energy level can be achieved with a low cost in terms of total energy consumption and path loss.

An MADSN with a simple configuration is shown in Fig. 1 for illustrative purposes. The sensor network contains one PE, labeled as S_0, and $N = 10$ leaf nodes, labeled $S_i, i = 1, 2, \ldots, N$, one of which is down. The sensor nodes are spatially distributed in a surveillance region of interest, each of which is responsible for collecting measurements in the environment. The signal energy detected by sensor node S_i is denoted by $s_i, i = 1, 2, \ldots, N$. Sensor node S_i takes time $t_{i,acq}$ for data acquisition and time $t_{i,proc}$ for data processing. The wireless communication link with physical distance $d_{i,j}$ between sensor node S_i and S_j has channel width W bits and operates at frequency B Hz. Some sensor nodes may be down temporarily due to intermittent failures, as sensor S_9 in Fig 1.

The routing objective is to find a path for a mobile agent that satisfies the desired detection accuracy while minimizing the energy consumption and path loss. The required path is computed based on the current state of DSN and the mobile agent traverses the nodes along the path while performing the fusion operation. The energy consumption depends on the processor operational power and computation time and the path loss is directly related to the physical length of the selected path. We define these quantities in the next section.

2.4 Objective Function

The objective function for the mobile agent routing problem is based on three aspects of a routing path: energy consumption, path loss, and detected signal energy:

1. **Energy consumption**. The energy consumption at a sensor node is determined by the processing speed and the computation time. If an energy-driven *Real-Time Operating System* (RTOS) is installed on the sensor node, the processor speed can be dynamically scaled depending on workload and task deadlines [15]. For wireless message transmissions, the energy consumption depends on the sensor's transmission power and message transmission time. We assume that the message includes the mobile agent code of size M bits and measured data of size D bits. For a given desired resolution, a fixed data size D is used to store the

partially integrated data at each sensor. The time for the message to be transmitted over a wireless channel of bandwidth BW bps is calculated as:

$$t_{msg} = \left\lceil \frac{M+D}{BW} \right\rceil.$$

The energy consumption EC of path P, consisting of nodes $P[0], P[1], \ldots, P[H-1]$, is defined as:

$$EC(P) = a \cdot (t_{0,setup} + t_{0,proc}) \cdot F_0^2 + P_{0,t} \cdot t_{msg} + \sum_{k=1}^{H-1} \left\{ b \cdot \left\{ (t_{P[k].acq} + t_{P[k],proc}) \cdot F_{P[k]}^2 \right\} + P_{P[k],t} \cdot t_{msg} \right\},$$

where the kth leaf node $S_{P[k]}$ on path P has data acquisition time $t_{P[k],acq}$, data processing time $t_{P[k],proc}$, operational level $F_{P[k]}$, transmitting power $P_{P[k],t}$, and node $P[0]=0$ corresponds to the PE. Here, the operational level refers to the processor operating frequency, the square of which determines its corresponding operating power level. Coefficients a and b are suitably chosen to "normalize" the operational level to its power level.

2. **Path loss**. The power received by sensor S_j has the following relation with the power transmitted by sensor S_i according to the Friis free space propagation model [13]:

$$P_{j,r}(d_{i,j}) = P_{i,t} \cdot \frac{G_{i,t} G_{j,r} \lambda^2}{(4\pi)^2 d_{i,j}^2 \beta},$$

where $G_{i,t}$ is the gain of sensor S_i as a transmitter and $G_{j,r}$ is the gain of sensor S_j as a receiver. Wavelength λ is the ratio of speed of light c and carrier frequency f and β is the system loss factor. The physical distance $d_{i,j}$ between S_i and S_j is computed from their spatial locations. *Path loss* (*PL*) represents the signal attenuation as a positive quantity measured in dB and is defined as the difference (in dB) between the effective transmitted power and the received power:

$$PL(d_{i,j}) = 10 \log \frac{P_{i,t}}{P_{j,r}} = 10 \log \left[\frac{(4\pi)^2 \beta}{G_{i,t} G_{j,r} \lambda^2} \cdot d_{i,j}^2 \right].$$

Therefore, the total path loss along path P can be calculated as:

$$PL(P) = \sum_{k=0}^{H-1} \left[10 \log \left(\frac{(4\pi)^2 \beta}{G_{P[k],t} G_{P[(k+1) \bmod H],r} \lambda^2} \cdot d_{P[k],P[(k+1) \bmod H]}^2 \right) \right].$$

3. **Signal energy**. A sensor node detects a certain amount of energy emitted by the potential target, which may be used by a mobile agent for data integration. A mobile agent always tries to accumulate as much signal energy as possible for accurate decision in target detection. The sum of the detected signal energy SE along path P is defined as:

$$SE(P) = \sum_{k=1}^{H-1} s_{P[k]},$$

where $s_{P[k]}$ is the signal energy detected by the kth sensor node on path P.

By combining the above three aspects of a routing path, we consider an objective function as follows:

$$O(P) = SE(P) \left(\frac{1}{EC(P)} + \frac{1}{PL(P)} \right),$$

wherein the terms $SE(P)$, $EC(P)$, and $PL(P)$ are assumed to be "normalized" to appropriately reflect the contribution by various loss terms. Intuitively, this objective function prefers paths with higher signal energies by penalizing those with high path losses and energy consumption. For the detection applications, we are interested in concluding the presence of a target in the monitoring area, which is determined by a threshold level of detected energy. For example, this threshold can be determined to maximize the probability of detection while keeping the false alarm rate below a specified quantity. In particular, a path providing high signal energy at the expense of a considerable amount of energy consumption and path loss may not be preferable. Note that alternative objective functions may be used as long as they correctly reflect the trade off between detected signal energy, energy consumption, and path loss.

To facilitate the design of the genetic algorithm in Section 3, we define a *fitness function* based on the above objective function as follows:

$$f(P) = O(P) + g,$$

where g is the penalty function for overrunning the constraint and is defined by:

$$g = \begin{cases} 0, & SE(P) \geq E \\ \delta \cdot (SE(P) - E)/E & SE(P) < E, \end{cases}$$

where E is the desired detection accuracy or signal energy level and δ is a properly selected penalty coefficient.

2.5 NP-Completeness of Mobile Agent Routing Problem

An MADSN can be represented by a completely connected graph $G = (V, E)$, where each node corresponds to a sensor node specified by its location in plane and each edge corresponds to a communication link. In addition, each node is associated with two parameters representing signal level and energy consumption and the path loss is associated with each edge.

The *mobile agent routing problem* (MARP) is to compute a path P in a MADSN such that $O(P) > \tau$ and the *k-hop mobile agent routing problem* (k-MARP) additionally requires that the path P has exactly k edges. We first show that MARP and k-MARP are polynomially equivalent and then show that the latter is NP-complete by reducing the *3D Maximum Traveling Salesman Problem* (3MTSP) to it (thus establishing the NP-completeness of MARP).

An instance of MARP can be reduced to n instances of k-MARP, namely, 1-MARP, 2-MARP, …, n-MARP, such that the answer to the former is obtained by simply OR-ing those from all n instances. A solution to the former exists if and only one of the instances yields a solution and, thus, MARP is polynomially reducible to k-MARP. To show the equivalence, we also need to show the reverse. Given an instance of k-MARP, we generate an instance of MARP by defining its parameters as follows:

$$EC^1(P) = EC(P) + k\Delta_1, PL^1(P) = PL - k\Delta_2,$$
$$\text{and } SE^1(P) = (k\Delta_2 + PL_{\min}),$$

where Δ_1 and Δ_2 satisfy the following conditions:

$$\begin{aligned}&\Delta_1 = (EC_{\min} - SE_{\max}/\tau)/k\\ &\Delta_2 = (PL_{\max} - SE_{\min}/\tau)/k,\\ &k\Delta_2 + PL_{\min} > 1,\end{aligned}$$

where

$$\begin{aligned}EC_{\min} &= \min_{e\in E} EC(e),\\ PL_{\max} &= \max_{e\in E} PL(e),\\ SE_{\max} &= \max_{v\in V} SE(v),\end{aligned}$$

and $SE_{\min} = \min_{v\in V} SE(v)$.

These conditions ensure that a solution of MARP yields a path P with k edges and satisfies the condition $O(P) > \tau$ if and only if k-MARP has a solution. This establishes the polynomial equivalence of MARP and k-MARP.

We now show the polynomial reduction from k-MARP to MTSP. The definition of MTSP is as follows: Given a completely connected graph $G = (V, E)$ and a nonnegative real number α, does there exist a closed-loop path P, with nodes $P[0], P[1], \ldots, P[n-1], P[n] = P[0]$ such that $\sum_{i=0}^{n-1} l_3(P[i], P[(i+1) \bmod n]) \geq \alpha$? Here, each vertex corresponds to a point in three-dimensional Euclidean space R^3. The starting point $v_{P[0]}$ and ending point $v_{P[n]}$ in the space refer to the same vertex in the graph.

The quantity

$$l_3(P[i], P[i+1]) = \sqrt{(x_{P[i]} - x_{P[i+1]})^2 + (y_{P[i]} - y_{P[i+1]})^2 + (z_{P[i]} - z_{P[i+1]})^2}$$

is the Euclidean distance between two adjacent vertices $P[i]$ and $P[i+1]$ on path P. The MTSP under Euclidean distances in R^d for any fixed $d \geq 3$ is proven to be NP-hard in [16]. The conventional traveling salesman problem requires that path length be minimized and the cities are defined for dimension $d = 2$. On the contrary, 3MTSP requires maximization of path length and is known to be intractable only in three or higher dimensions. Note that k-MARP requires the maximization of $O(P)$, but is defined for $d = 2$, which makes a direct reduction from 3MTSP somewhat nontrivial.

Given an instance of 3MTSP, we generate an instance of k-MARP as follows: We create a graph for the k-MARP with $k = n$ using only the x and y coordinates of the vertices of MTSP (without the loss of generality, we assume that all coordinate values are distinct). We consider the objective

$$O(P) = \frac{SE(P)}{PL(P)} = \frac{\sum_{i=0}^{k-1} s_{P[i]}}{PL(P)}$$

by ignoring the energy consumption component. Recall that the path loss is given by

$$PL(P) = \sum_{i=0}^{k-1} 10 \cdot \log(A \cdot d^2_{P[i],P[(i+1) \bmod k]}).$$

Let $e_{P[i]}$ represent the edge between vertices $P[i]$ and $P[i+1]$ on path P. We define

$$\begin{aligned}s_{P[i]} &= s(e_{P[i]})\\ &= l_3(P[i], P[i+1]) + \tau\big(10 \cdot \log(A \cdot d^2_{P[i],P[(i+1) \bmod k]})\big) - \frac{\alpha}{k}.\end{aligned}$$

A solution to the k-MARP is a path P with k hops such that

$$O(P) = \frac{\sum_{i=0}^{k-1} s_{P[i]}}{\sum_{i=0}^{k-1} d(v_{P[i]}, v_{P[i+1]})} \geq \tau,$$

τ is a given nonnegative real number. After reorganizing, the objective function can be rewritten as:

$$\begin{aligned}&\sum_{i=0}^{k-1}(s(e_{P[i]} - \tau \cdot d(e_{P[i]})) =\\ &\sum_{i=0}^{k-1}(l_3(P[i], P[(i+1) \bmod k]) - \alpha/k) > 0,\end{aligned}$$

which guarantees the condition

$$\sum_{i=0}^{n-1}(l_3(P[i], P[(i+1) \bmod n]) \geq \alpha$$

necessary for a solution to the corresponding MTSP instance.

On the other hand, if there exists a solution to the MTSP, i.e., a closed-loop path P consisting of n edges such that $\sum_{i=0}^{n-1}(l_3(P[i], P[(i+1) \bmod k]) \geq \alpha$, this path can be used to solve the corresponding n-MARP such that $O(P) \geq \tau$. Note that the above reduction from MTSP to n-MARP is polynomial-time computable and, hence, NP-completeness of the latter follows from that of the former.

The restriction of n-MARP is studied in [5], where two heuristics LCF and GCF are proposed. In the next section, we propose a genetic algorithm method for MARP and empirically show that it outperforms LCF and GCF.

3 Genetic Algorithm for Mobile Agent Routing Problem

3.1 Introduction to Genetic Algorithm

A genetic algorithm is a computational mechanism that "simulates" the process of genetic selection and natural elimination in biological evolution. Pioneering work in this field was conducted by Holland in the 1960s [17], [18]. Compared to traditional search algorithms in *artificial intelligence* (AI), a genetic algorithm is often able to automatically acquire and accumulate implicit knowledge about the search space during its search process and self-

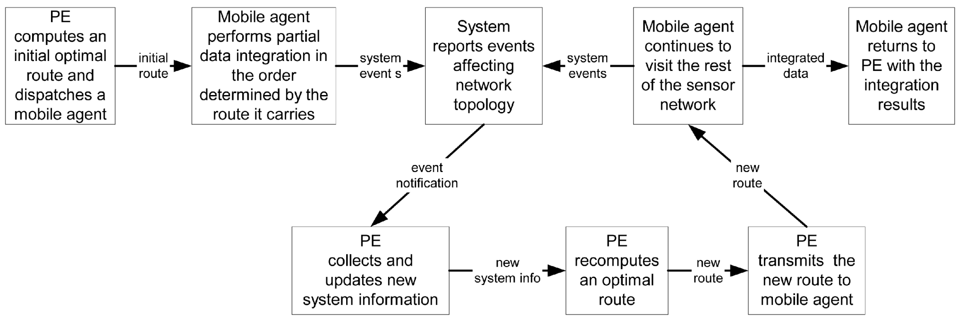

PE computes an initial optimal route and dispatches a mobile agent
initial route
Mobile agent performs partial data integration in the order determined by the route it carries
system event s
System reports events affecting network topology
system events
Mobile agent continues to visit the rest of the sensor network
integrated data
Mobile agent returns to PE with the integration results
event notification
new route
PE collects and updates new system information
new system info
PE recomputes an optimal route
new route
PE transmits the new route to mobile agent

Fig 2. Activity diagram of the adaptive semidynamic routing method.

adaptively control the search process through a random optimization technique. It is often able to yield the global optimal solution and avoid the combinatorial explosion which may occur if the inherent knowledge within the search space is ignored. It has been frequently used to solve combinatorial optimization problems and nonlinear problems with complicated constraints or nondifferentiable objective functions.

3.2 Adaptive Routing Strategy Based on Genetic Algorithm

Mobile agent routing algorithms can be classified as dynamic or static routing according to the place where routing decisions are made. A dynamic method determines the route locally on the fly at each hop of the migration of a mobile agent among sensor nodes, while a static method uses centralized routing, which computes the route at the PE node in advance of mobile agent migration. For a sensor network application at hand, one has to judiciously choose between the dynamic and static methods. For example, it might be sufficient to use a static routing for target classification, but a target tracking task might require a dynamic routing to follow a moving target. In this paper, we propose an event-driven adaptive method to implement a semidynamic routing strategy based on a two-level genetic algorithm whose activity diagram is illustrated in Fig 2.

The fitness function constructed earlier is intended to meet the desired detection accuracy while minimizing the energy consumptions and path losses in a global sense. Consequently, a genetic algorithm needs to gather the relevant system-wide information to perform the route computation. Since a mobile agent always starts its data collection journey from the PE node, which can usually be equipped with more powerful computing resources than regular sensor nodes, it is reasonable to compute the route at the PE node. The mobile agent simply follows the route computed by the genetic algorithm according to the fitness function. The PE node has the predetermined knowledge necessary for performing the global optimization, such as the geographical locations (through GPS interfaces) and transmitting/receiving parameters of sensor nodes. Once a sensor node has acquired some measurements, it broadcasts the detected signal level (not the raw measurement data with the highest resolution) within the coverage area so that the PE node is able to construct a global energy map based on the broadcast signal level information.

In this scheme, the routing code resides exclusively on the PE node in order to keep the mobile agent code as compact as possible. The mobile agent carries the precomputed route that determines the order of sensor nodes to be visited. However, if the network system is notified of some events (for example, some nodes are shut down or activated, do not have enough remaining energy to transmit the signal along the previously designated link, or the detected signal levels vary due to target movements), which cause the previously computed route to be invalid, the routing code is rerun at the PE node and the new route is transmitted to the mobile agent.

3.3 Two-Level Genetic Encoding

We convert the problem parameters into the individuals made up of genes in the genetic domain. Such mapping is known as *genetic encoding*. Here, the problem domain means the space made up of phenotype individuals, i.e., effective candidate solutions, while the genetic domain is the space made up of genotype individuals of the genetic algorithm. Due to the robustness of a genetic algorithm, it does not impose critical requirements on encoding technique as long as a minimum of three encoding criteria, i.e., completeness, soundness, and nonredundancy, are satisfied [19].

We employ a two-level encoding to adapt the genetic algorithm for the mobile agent routing problem in MADSN. The first level is a numerical encoding of the sensor (ID) label sequence L in the order of sensor nodes being visited by mobile agent. For the MADSN shown in Fig. 1, the sensor label sequence L has the following contents:

0	1	2	3	7	5	6	8	4	10	9

The first element is always set to be "0" for the reason that a mobile agent starts from the PE S_0. The mobile agent returns to S_0 from the last visited sensor node, which is not necessarily the last element of the label sequence if there exist inactive sensor nodes in the network. This sequence consists of a complete set of sensor labels because it takes part in the production of a new generation of solutions through the genetic operations. It is desired to inherit as much information as possible in the new generation from the old one. For example, in Fig. 1, although nodes 3, 6, 8, and 9 are not visited in the given solution (the second-level sequence is designed to do so), they or some of them may likely make up a segment of a better solution than the current one in the new generation.

The second level is a binary encoding of the visit status sequence V in the same visiting order. For the MADSN in Fig. 1, the visit status sequence V contains the following binary codes:

1	1	1	0	1	1	0	0	1	1	0

where "1" indicates "visited" and "0" indicates "unvisited." The first bit corresponds to the PE and is always set to be "1" because the PE is the starting point of the itinerary. If a sensor is inactive, its corresponding bit remains "0" until it is reactivated and visited.

Masking the first level of numerical sensor label sequence L by the second level of binary visit status sequence V results in a candidate path P for mobile agent. In the above example, the path P is obtained as:

0	1	2	3	7	5	6	8	4	10	9

These two levels of sequences are arranged in the same visiting order for the purpose of convenient manipulations of visited/unvisited and active/inactive status in the implementation of the genetic algorithm. The number of hops H in a path P can be easily calculated from the second level of binary sequence as follows:

$$H = \sum_{i=0}^{N} V[i], \begin{cases} V[i] = 1, & \mathit{sensor}\ S_i\ \mathit{is\ active\ and\ visited} \\ V[i] = 0, & \mathit{sensor}\ S_i\ \mathit{is\ inactive\ or\ unvisited.} \end{cases}$$

3.4 Implementations of Genetic Operators

We now describe the genetic operators. These operators are similar to those used in the conventional solution to *Traveling Salesman Problem*. However, we incorporate the details corresponding to the current routing problem.

3.4.1 Selection Operator

The purpose of the selection operation is to select good individuals and, at the same time, eliminate bad individuals from the population based on the evaluation of individual fitness. It is also called reproduction operation. Its aim is to inherit good individuals directly from last generation or indirectly from the new individuals produced by mating the old individuals. In our implementation, each pair of individuals is selected randomly from the old generation to perform the crossover, mutation, and inversion operations. The fitness is computed for every newly generated child for evaluation. To maintain the same population sizes for each generation, the fitness of every newly generated child is compared with the minimum fitness of the whole population. If it is bigger than the minimum fitness value, then this child is added to the population and the individual with the minimum fitness is removed; otherwise, the new child is discarded.

3.4.2 Crossover Operator

Similar to gene recombination, which plays an essential role during the process of natural biological evolution, crossover is the most significant operation in the genetic search strategy. It determines the major behavior of optimization process. Several crossover schemes are used in the literature, such as one-point crossover, two-point crossover, and multipoint crossover. However, as a common criterion, any crossover operator should ensure that the proper genes of good individuals be inherited by the new individuals of the next generation. A big crossover probability may improve a genetic algorithm's capability to search new solution space, while increasing the probability of undermining the combination of good genes. If the crossover probability is set too small, the search process may be trapped in a dull status and is prone to ceasing.

In our implementation, a two-point crossover is applied to both levels of sequences. These two crossover points are selected randomly. Given two parents as follows:

- Parent 1: First level sequence: 0-2-7-3-/-5-1-6-/-4-9-8 and second level sequence: 1-0-1-1-/-1-0-0-/-1-1-1.
- Parent 2: First level sequence: 0-3-5-2-/-9-6-4-/-1-7-8 and second level sequence: 1-0-0-0-/-1-0-1-/-1-0-1,

where "/" represents the crossover points, the crossover operator produces two children

- Child 1: First level sequence: 0-9-6-4-2-7-3-5-1-8 and second level sequence: 1-0-1-1-/-1-0-1-/-1-1-1.
- Child 2: First level sequence: 0-5-1-6-3-2-9-4-7-8 and second level sequence: 1-0-0-0-/-1-0-0-/-1-0-1.

For the first level of label sequence, the crossover portion (between the two crossover points) of one individual is copied and inserted at the front of the other individual (immediately after label 0). All the duplicate genes in the resulting individual are removed to guarantee that each node appears exactly once in that individual. For the second level of visit status binary sequence, the crossover portions are simply exchanged between two individuals.

3.4.3 Mutation Operator

The main purpose of the mutation operation is to prevent losing any single important gene segment and, thus, to maintain the variety of the solution population. Because the frequent use of the mutation operation may tend to make the genetic algorithm conduct a random search, a relatively small probability for mutation operation is favorable. To perform the mutation operation, two mutation points are selected randomly and the values of these two points are exchanged. As an example, consider the following parent individual:

First level sequence: 0-2-9-3-7-1-4-5-8-6

Second level sequence: 1-0-1-1-0-1-0-0-1-1

↑ ↑

two selected gene loci

The mutation operator produces the following child:

- First level sequence: 0-2-9-3-8-1-4-5-7-6 and
- Second level sequence: 1-0-1-1-1-1-0-0-0-1.

3.4.4 Inversion Operator

The crossover operator has a wide span of activity in the feasible solution space, while, due to the pressure of genetic selection and natural elimination, the mutation operation also has difficulty in carrying out local search activity (especially at the late stage of computation when the genetic algorithm tends to converge). The inversion operation is a special form of mutation which is designed to carry out a reordering operation and improve the local search for the genetic algorithm.

At any time, two inversion points are selected randomly to determine the inversion portion of the individual. The inversion operation is executed by reversing the order of the inversion portion of the original individual. Given one parent as follows:

- First level sequence: 0-5-7-/-1-2-8-9-/-6-3-4 and
- second level sequence: 1-0-1-/-1-0-1-1-/-0-0-1,

where the inversion portions are enclosed by two "/" signs, the inversion operator produces the following child:

- First level sequence: 0-5-7-/-9-8-2-1-/-6-3-4 and
- second level sequence: 1-0-1-/-1-1-0-1-/-0-0-1.

3.5 Parameter Selection for Genetic Algorithm

We usually select a high probability value above 0.9 for a genetic operator like crossover, which controls the main direction of the evolution process. A low probability value below 0.1 is appropriate for genetic operators like mutation or inversion to reduce the risk of destroying the good gene segments in later generations. Observed from the experimental data, a small variation of these probabilities does not have a significant impact on the performance of genetic algorithm. With respect to the maximum generation number, we select different values for different test examples in order to ensure that the optimization process approaches a steady state eventually. The difference of the best fitness values between two adjacent generations may be used as an alternative convergent indicator. In this case, the genetic algorithm does not have to wait for a long time to reach the prespecified maximum generation number if the optimization process converges quickly. Its disadvantage is that the program may prematurely terminate if the optimization process does not converge quickly.

4 Algorithm Evaluation

4.1 Simulation Results

We compare the search results of the genetic algorithm with those computed by the LCF and GCF methods. In most cases, LCF is able to deliver a satisfactory route, hence it is comparable to GA. GCF may find a path with less number of hops than LCF, but it usually has significantly longer path length, which results in unacceptable path loss levels. A series of experimental networks of different sensor node sizes and distribution patterns are created. The spatial locations of all the nodes are randomly selected. About 1-10 percent of the sensors are shut down uniformly over the surveillance region. The center node is selected as the PE node, i.e., the starting point of the route traversed by the mobile agent. All sensor parameters of data acquisition and wireless channel in the MADSN use the real-life data of the field demo listed in Table 1. The carrier frequency band used by sensor nodes with Hitachi SuperH processor SH-4 architecture to transmit their data is 2.4 GHz.

In order to make a visual comparison of the search performances, we plot the routes computed by GA, LCF, and GCF for the first experimental sensor network in Figs. 3, 4, and 5, respectively. This relatively small network consists of 200 sensor nodes, eight of which are put in the sleep state. A quantified amount of signal energy associated with each active sensor ranging from 0 to 64 is marked under the corresponding circle-shaped node. The PE node located in

TABLE 1
Parameters of the MADSN

Sensor node processor type	Hitachi SuperH processor SH-4 architecture
Sensor node processor speed	200 MHz
Mobile agent sizes	400 bytes
Ave. data sizes	100 bytes
Carrier frequency band	2.4 GHz
Transmitting power	100 mW
Transmitter gain	2
Receiving power	80 mW
Receiver gain	2
Channel operation frequency	20 kHz
Channel width	16 bits
Data sampling rate	20 kHz
Sample data format	16-bit

the center of the region particularly possesses a rectangular shape to differentiate from the regular leaf nodes. The minimum acceptable amount of signal energy detected by an individual sensor node is five and inactive nodes in the sleep state do not detect any signal energy. A single target is arbitrarily placed in this surveillance region. The area of elliptic shape encompassing the target represents the sensing zone within which the sensor nodes usually detect higher signal energy than others far away from the target. The total signal energy detected by all sensor nodes in this network is 1,467 units and the acceptable signal level adequate for inferring correct information about the environment is set to be 1,200 units. In other words, the mobile agent may return to the PE node immediately once the amount of measurement data it collects has reached 1,200 units.

For the GA-based routing method, we set the maximum generation number to be 200 as the convergent indicator, which informs the program when to stop its search process. It has been observed that the optimization process of the genetic algorithm moves forward rapidly in the beginning and becomes slow and stable in the later stages of computation, especially after the generation number reaches 100. The search result visualized in Fig. 3 shows that the GA has successfully discovered a clean route that goes through every node within the sensing zone of the target while avoiding the nodes in sleep state. The global random optimization strategy makes GA capable of exploring any potential sensing zones and visiting nodes in the target neighborhoods with the highest priority for an optimal detection performance in addition to preserving an energy-saving route. This salient feature of the GA-based routing method is evident in the search results for all experimental sensor networks we tested.

Starting from the PE node, the LCF algorithm selects the neighbor node closest to the current location as the next-hop destination. The search result visualized in Fig. 4 shows that the LCF algorithm is also able to obtain a relatively orderly route for the mobile agent. However, since LCF is based entirely on local information, it lacks the capability of identifying sensing zones in a global manner. The mobile agent that follows the route displayed in Fig. 4 reaches the sensing zone, but returns to the PE node, leaving out some nodes with high signal levels in the sensing zone. In some extreme cases, the mobile agent may have to visit all the rest of the nodes with low signal levels in the network before it reaches the sensing zone.

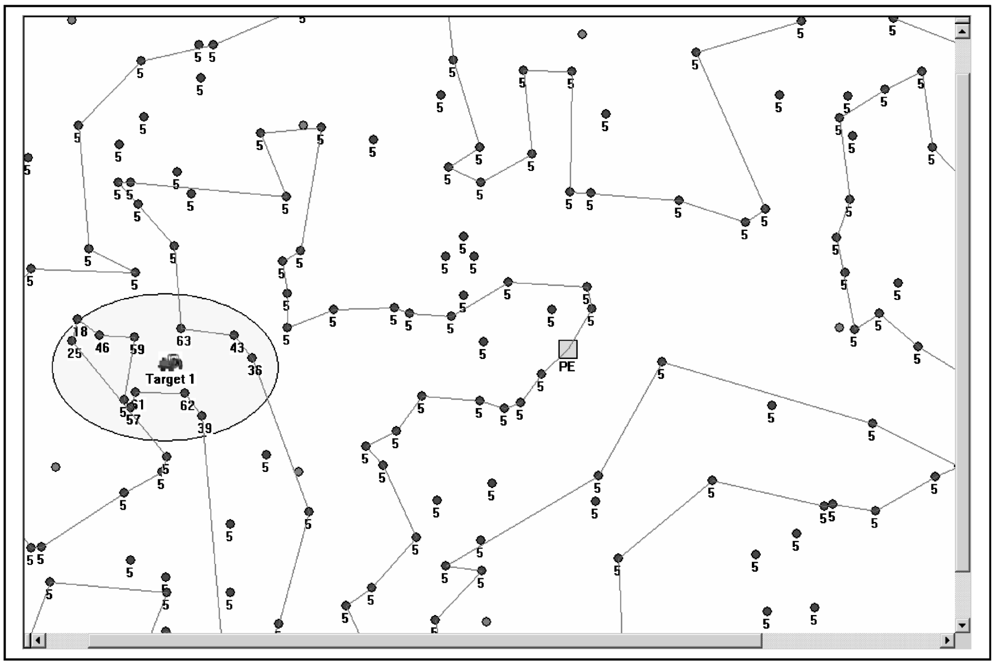


Fig. 3. Visualization of the search results computed by GA for an MADSN with 200 nodes.

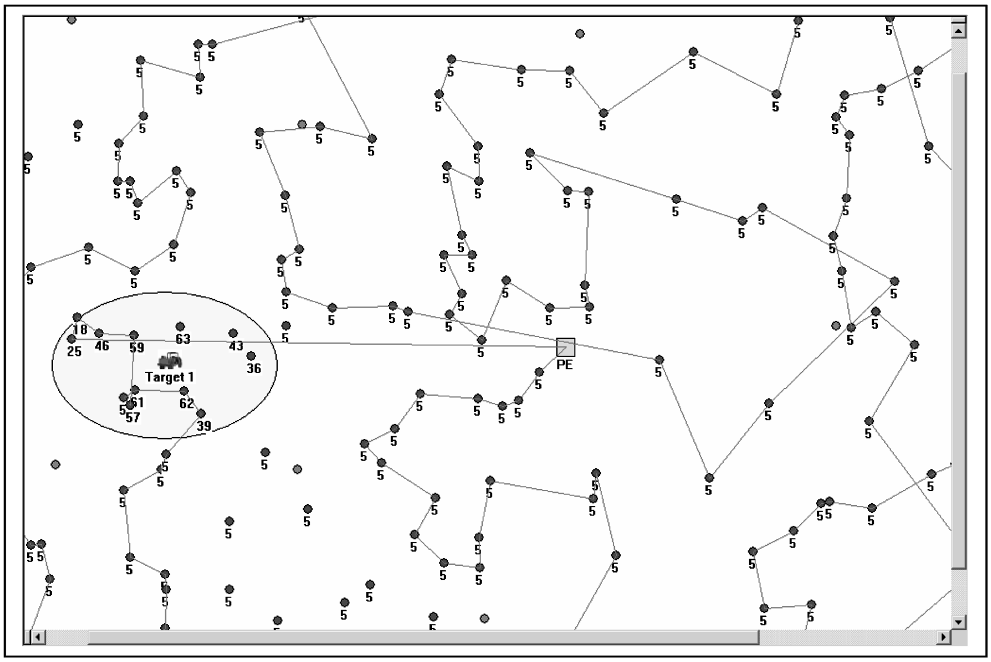


Fig. 4. Visualization of the search result computed by LCF for an MADSN with 200 nodes.

The GCF algorithm also starts its route search process from the PE node and selects the node closest to the center of the region as the next-hop destination. GCF is a global search algorithm making use of only the geographical information. Consequently, similarly to LCF, GCF is also not able to recognize sensing zones of potential targets. It is an interesting observation that, in most cases we studied, GCF produced messier routes than GA and LCF, which resulted in much higher path losses. Moreover, since the GCF algorithm moves its search process from the center to the edge of the network, it often exhibits undesirable performance when the targets are located in a remote portion of the surveillance region.

The simulation results computed by GA, LCF, and GCF for all experimental sensor networks are tabulated in Table 2, which consists of three parts. The upper part of Table 2 contains the general information about network configurations, such as node size, number of dead or inactive sensors, number of potential targets in the region, total detected signal energy, and acceptable signal energy level for correct environmental inferences. Note that the total detected signal energy refers to the sum of the energy

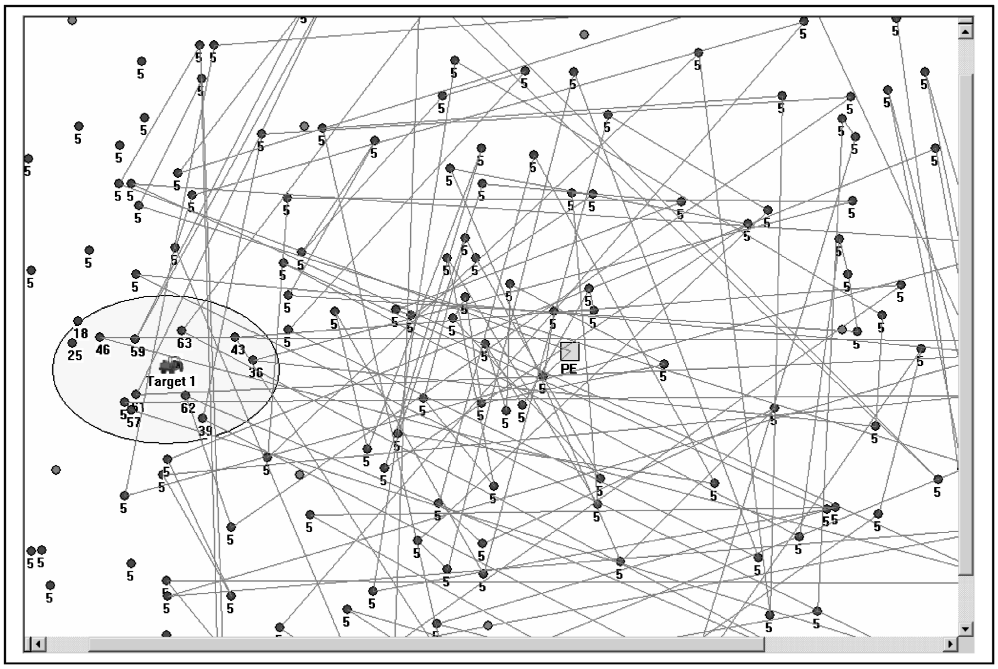


Fig. 5. Visualization of the search result computed by GCF for an MADSN with 200 nodes.

detected by all active sensors deployed in the region under surveillance, while the acceptable signal energy level is a given value desired for a specific application. The middle part of Table 2 keeps track of four quality components of a route computed by each algorithm, i.e., number of hops, path loss, energy consumption, and achieved signal energy. For instance, with regard to the first network of 200 nodes, GA uses 140 hops to achieve the acceptable signal level, while LCF uses 168 hops and GCF uses 147 hops, respectively; the path losses of GA, LCF, and GCF are 22,919 units, 26,864 units, and 28,748 units, respectively; the migration of mobile agent along the path computed by GA consumes 407 units of energy, while LCF consumes 489 units of energy, and GCF consumes 428 units of energy. The bottom part of Table 2 is used to compare the overall performance of GA, LCF, and GCF in terms of the same objective function as defined in Section 2.4.

In order to make a more straightforward performance comparison between these three methods, the data in the middle and bottom parts of Table 2 is plotted in Figs. 6, 7, 8, and 9, illustrating the corresponding curves of node sizes versus number of hops, node sizes versus path loss, node sizes versus energy consumption, and node sizes versus objective values, respectively. Since the routing criteria such as energy consumption, path loss, and detection accuracy considered in the GA fitness function are closely related to the number of hops of a route, which is, in general, determined by the node size of a network, the advantages of the GA-based method become more evident as the node size increases.

Fig. 6 shows that the routes computed by GA have the smallest number of hops, except for the networks with small node sizes, and LCF has a larger number of hops than GCF. Fig. 7 shows that GCF experiences much higher path losses than GA and LCF, especially when the node size increases. This trend is consistent with the observation made in the first experimental network. Fig. 8 shows that GA often finds the most energy-saving route, while LCF incurs the highest energy consumption. The curves illustrated in Figs. 6, 7, and 8 clearly justify that in most cases, GA is able to find a satisfying path with fewer hops, less energy consumption, and less path loss than LCF and GCF. Fig. 9 also shows that the GA has a superior overall performance over the two algorithms in terms of the objective function defined in this implementation.

The current GA program was implemented in C++ using MFC with GUI and per-generation result display. The GA-based routing code takes a few seconds to run the first 100 generations for a network of hundreds of nodes. GA runs much faster and its executable code size decreases significantly when GUI is not implemented. In such cases, the GA runtime may not be a serious problem for semidynamic routing, as further discussed in the next section.

4.2 Computational Paradigm Using Mobile Agent

The new emerging technology of agent programming has provided a flexible and complementary way of managing resources of distributed systems. Although the computational paradigm using mobile agent has not yet matured to be a proper area in computer science, it has attracted a great deal of attention from many researchers and an increasing amount of related works have been found in the literature. The general advantages of using a mobile agent, such as reducing network load, overcoming network latency, achieving robust and fault-tolerant performance, are summarized in [20].

The mobile agent programming, however, has been mainly technology driven so far with a focus on the implementation of mobile agent runtime platforms and practical applications including electronic commerce, telecommunications services, network management, group work, and work flow management. Papavassiliou et al. [21] presents an implementation framework for the development of effective market-based routes for brokering

TABLE 2
Comparisons of Search Results of GA, LCF, and GCF for Networks of Different Node Sizes and Distribution Patterns

Case #	Node sizes	# of dead sensors	# of potential targets	Total detected signal energy	Acceptable signal energy level
1	200	8	1	1467	1200
2	300	4	2	2985	2750
3	400	5	2	4000	3680
4	500	8	3	4080	3710
5	600	9	4	4980	4800
6	700	10	5	5340	5190
7	800	11	4	6200	5950
8	900	13	3	7000	6380
9	1000	14	5	7525	7000
10	1100	13	4	8515	7990
11	1200	18	5	10050	9425
12	1300	16	4	11410	9600
13	1400	15	4	11500	9800
14	1500	19	5	12380	11000
15	1600	25	6	13505	12210

Case #	GA				LCF				GCF			
	No. hops	Path loss	Energy cnsmpt	Achieved signal energy	No. hops	Path loss	Energy cnsumpt	Achieved signal energy	No. hops	Path loss	Energy cnsumpt	Achieved signal energy
1	140	22919	407	1202	168	26864	489	1215	147	28748	428	1204
2	216	34154	670	2785	273	42027	796	2757	228	43191	664	2763
3	300	47100	875	3691	379	57842	1106	3684	291	55897	849	3688
4	424	64302	1237	3716	460	68645	1342	3721	424	82817	1237	3714
5	483	76205	1389	4815	561	82627	1637	4807	490	95417	1430	4819
6	544	88344	1552	5195	663	96370	1935	5197	566	110972	1652	5194
7	613	98722	1803	5963	757	109877	2186	5952	708	139179	2093	5958
8	703	113790	2079	6400	847	121082	2473	6390	774	151392	2260	6385
9	838	122087	2481	7001	948	134053	2768	7003	929	183379	2713	7004
10	931	137906	2704	8002	1059	150332	3123	7990	988	207397	2845	8005
11	1067	152397	3116	9425	1146	161248	3347	9460	1117	221073	3262	9450
12	1009	145899	2946	9650	1208	171032	3528	9600	1138	231898	3301	9610
13	1105	158471	3227	9840	1287	175492	3693	9800	1174	239843	3459	9820
14	1270	170029	3634	11005	1381	192300	4099	11019	1315	258942	3832	11004
15	1369	182382	3823	12210	1534	211317	4480	12210	1495	295711	4366	12240

Case #	GA objective value *O(P)=SE(P)(1/EC(P)+1/PL(P))*	LCF objective value *O(P)=SE(P)(1/EC(P)+1/PL(P))*	GCF objective value *O(P)=SE(P)(1/EC(P)+1/PL(P))*
1	3.005763	2.52989	2.854965
2	4.238259	3.529169	4.225116
3	4.296651	3.394613	4.409913
4	3.061832	2.826934	3.047271
5	3.529708	2.994646	3.420435
6	3.406098	2.739716	3.190872
7	3.367668	2.776951	2.88944
8	3.134647	2.63668	2.867397
9	2.87919	2.582226	2.619838
10	3.017345	2.611586	2.852306
11	3.086556	2.885079	2.939742
12	3.34177	2.777218	2.95268
13	3.111365	2.709512	2.879914
14	3.093068	2.745518	2.914104
15	3.260774	2.783227	2.844873

purposes in the future multioperator network marketplace. On this platform, a genetic algorithm is used to identify optimal resource allocation strategies and the agent-based network management approach is used to represent an underlying structure for the multioperator network model and facilitate the collection and dissemination of the required management data. The ant routing mechanism, a routing method in the agent technology, is described in [22], which conducted an analysis on the growing and jumping behaviors of an agent-based routing algorithm. Some analytical results are presented such as the maximum number of agents in a node and the maximum number of jumps of an agent. Umezawa et al. [23] describe the architecture of a mobile agent-based framework and the implementation of its prototype for self-configurable sensor networks that enables sensor networks to be dynamically reconfigured to suit the requirements of applications and changes in the environments. This mobile agent-based framework provides an approach to the easy development of adaptive and application-specific software for sensor nodes.

The MADSN proposed in [5] is used to perform multiresolution data integration and fusion. The mobile agent moves the processing code to leaf sensor nodes from a central processing site. The distributed computation avoids transferring a large set of raw data for each leaf node and, therefore, saves limited bandwidth and prolongs battery life. Furthermore, the agent-based communication models show great promise to operate in unpredictable, harsh, and even adversarial environments.

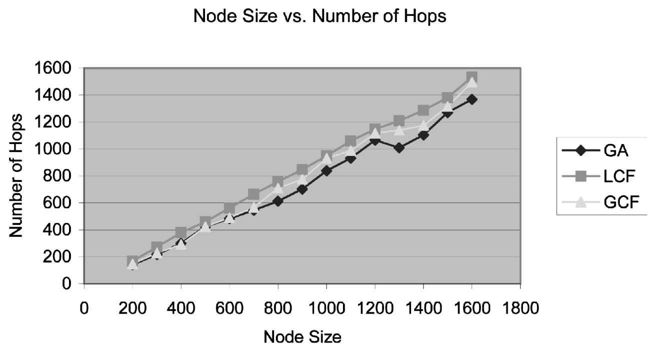


Fig. 6. Node sizes versus number of hops for the three algorithms.

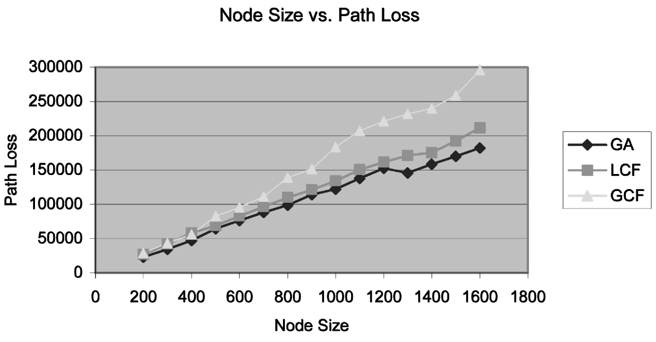


Fig. 7. Node sizes versus path loss for the three algorithms.

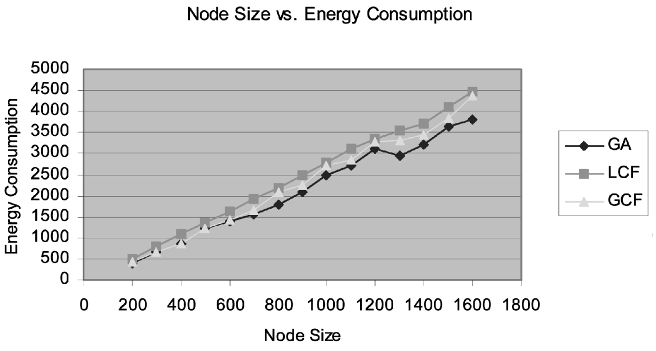


Fig. 8. Node sizes versus energy consumption for the three algorithms.

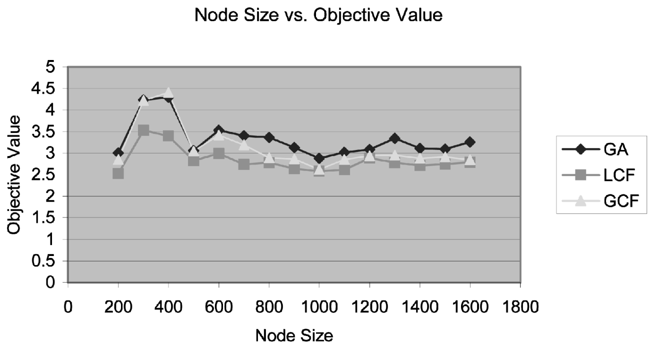


Fig. 9. Node sizes versus objective values for the three algorithms.

4.3 Discussion on Routing Algorithms Using LCF, GCF, and GA

GCF is relatively simple and fast but suffers from poor performance in terms of path loss. GCF essentially utilizes sorting to compute the path. Its computational complexity is $O(N \log N)$ if using a comparison based sorting algorithm. LCF has the computation complexity of $O(N^2)$ if the closest neighbor node is obtained by simple comparison in each step. The analysis for the computation complexity of the genetic algorithm is more complicated. After making some simplifications on its implementation, the computational complexity for the genetic algorithm is $O(NMG)$, where N is the number of nodes in the network, M is the initial population size, and G is the maximum generation number used to indicate the end of the computation.

As is the case for GCF, the performance of LCF also depends significantly on the network structure [5]. In some bad cases, it may end up with unacceptable results. Comparatively, the network structure has much less influence on the performance of the genetic algorithm due to its random search technique.

Unlike LCF and GCF, it is not necessary to specify a starting node for the implementation of the genetic algorithm. Actually, any active sensor node can be designated as a starting node in the genetic algorithm, while the performances of LCF and GCF are crucially dependent on the location of its starting node. In addition, no matter in what order the nodes are visited, the genetic algorithm always comes up with a closed route for the mobile agent to come back to its starting node.

LCF is suitable for carrying out dynamic routing because each step of its computation depends only on the location of the current node, while GCF is in favor of static routing, whose computation can be carried out offline based on the global network structure. Both LCF and GCF are deterministic routing algorithms, which always supply the same path between a source/destination pair in a given network. The genetic algorithm collects information about the network status from all sensor nodes so that it is able to conduct adaptive routing. However, broadcasting the detected signal energy produces extra communication overhead.

The adaptive semidynamic routing scheme is supported by the robustness of a sensor network. According to the experience from the field demo, sensor nodes usually function well once brought up and the network may remain stable continuously for 1-2 hours of sessions. The time taken for the proposed routing method based on the two-level genetic algorithm to compute a satisfactory route is always in the order of seconds for a typical network and rarely goes beyond one minute, even for networks with large node sizes. Consequently, there is adequate time for the PE node to rerun the GA routing code to produce a new route for mobile agent in the case of relatively infrequent occurrences of system events.

5 Conclusions

We presented a mobile agent-based paradigm for data fusion in distributed sensor networks. By utilizing a simplified analytical model for a distributed sensor network, we formulated a route computation problem for the mobile agent in terms of maximizing the received signal strength while keeping path loss and energy consumption low. This route computation problem turned out to be NP-hard thereby making it highly unlikely to develop a polynomial-time algorithm to compute an optimal route. Hence, we proposed a genetic algorithm to solve this problem by employing a two-level genetic encoding and suitable genetic operators. Simulation results are presented for comparison between our GA and existing LCF and GCF heuristics. Various aspects of the proposed algorithm such as computational complexity, impact of network structure and starting node, and dynamic and static routing, are discussed.

Future research work may be focused on exploring more complex routing models with more general objective functions. For example, in the current model, we assume that the sensor locations are fixed once they are manually deployed, which is the case in the field demo. However, in a real-world sensor network, sensors could be airborne or mounted on vehicles or robots. The mobility of sensors brings new challenges to the design of dynamic routing algorithms for the mobile agent. Besides, instead of using the simple free space propagation model to compute path loss, more complex empirical propagation models may be studied and applied in the construction of objective function.

Acknowledgments

The authors would like to thank four anonymous reviewers for their insightful suggestions and constructive comments that greatly improved the presentation of the revised paper. This research was supported in part by the US Defense Advanced Research Projects Agency under grants No. N66001-00-C-8046 and No. K153, US Office of Naval Research under grant No. N000140110712, and by Engineering Research Program, US Department of Energy under contract No. DE-AC0500OR22725 with UT-Batelle, LLC.

References

[1] A.K. Hyder, E. Shahbazian, and E. Waltz, *Multisensor Fusion.* Kluwer Academic, 2002.

[2] D.N. Jayasimha and S.S. Iyengar, "Information Integration and Synchronization in Distributed Sensor Networks," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 21, no. 5, pp. 1032-1043, Sept./Oct. 1991.

[3] Y.F. Zheng, "Integration of Multiple Sensors into a Robotics System and Its Performance Evaluation," *IEEE Trans. Robotic Automation,* vol. 5, pp. 658-669, Oct. 1989.

[4] R.C. Luo and M.G. Kay, "Multisensor Integration and Fusion in Intelligent Systems," *IEEE Trans. System, Man, and Cybernetics,* vol. 19, pp. 901-931, 1989.

[5] H. Qi, S.S. Iyengar, and K. Chakrabarty, "Multi-Resolution Data Integration Using Mobile Agents in Distributed Sensor Networks," *IEEE Trans. Systems, Man, and Cybernetics Part C: Applications and Rev.,* vol. 31, no. 3, pp. 383-391, Aug. 2001.

[6] T. Sander and C. Tschudin, "Protecting Mobile Agents against Malicious Hosts," *Mobile Agent and Security,* G. Vigna, ed., pp. 44-60, Springer-Verlag, 1998.

[7] S. Berkovits, J.D. Guttman, and V. Swarup, "Authentication for Mobile Agents," *Mobile Agents and Security,* G. Vigna, ed., pp. 114-136, Springer-Verlag, 1998.

[8] C.G. Harrison and D.M. Chess, "Mobile Agents: Are They a Good Idea?" Technical Report RC 1987, IBM T.J. Watson Research Center, Mar. 1995, http://www.research.ibm.com/massive/mobag.ps.

[9] P. Dasgupta and N. Narasimhan, "Magnet: Mobile Agents for Networked Electronic Trading," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 4, pp. 509-525, July/Aug. 1999.

[10] K.N. Ross and R.D. Chaney, "Mobile Agents in Adaptive Hierarchical Bayesian Networks for Global Awareness," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics,* pp. 2207-2212, 1998.

[11] S.S. Iyengar, D.N. Jayasimha, and D. Nadig, "A Versatile Architecture for the Distributed Sensor Integration Problem," *IEEE Trans. Computers,* vol. 43, no. 2, pp. 175-185, Feb. 1994.

[12] Revision A, WINS NG 2.0 User's Manual and API Specification, Sensoria Corp., 30 May 2002.

[13] T.S. Rappaport, *Wireless Communications Principles and Practice,* second ed. Prentice Hall PTR, 2002.

[14] N.S.V. Rao, "Multisensor Fusion Under Unknown Distributions: Finite Sample Performance Guarantees," *Multisensor Fusion,* A.K. Hyder, E. Shahbazian, and E. Waltz, eds., 2002.

[15] V. Swaminathan and K. Chakrabarty, "Real-Time Task Scheduling for Energy-Aware Embedded Systems," *Proc. IEEE Real-Time Systems Symp. (Work-In-Progress Sessions),* Nov. 2000.

[16] A. Barvinok et al. "The Geometric Maximum Traveling Salesman Problem," *J. ACM,* combined journal version of previous paper and Fekete's 1999 SODA paper on the maximum TSP, including a faster algorithm for arbitrary polyhedral metrics, submitted for publication in 2002.

[17] J.H. Holland, *Adaptation in Nature and Artificial Systems.* Univ. of Michigan Press, reprinted by MIT Press, 1992.

[18] D.A. Coley, *An Introduction to Genetic Algorithms for Scientists and Engineers.* World Scientific, 1999.

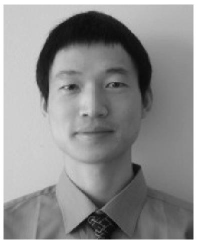
[19] D.E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, 1989.

[20] D.B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents," *Comm. ACM,* vol. 42, no. 3, pp. 88-89, Mar. 1999.

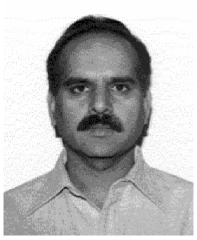
[21] S. Papavassiliou, A. Puliafito, O. Tomarchio, and J. Ye, "Mobile Agent-Based Approach for Efficient Network Management and Resource Allocation: Framework and Applications," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 4, pp. 858-872, May 2002.

[22] J. Sum, H. Shen, C.S. Leung, and G. Young, "Analysis on a Mobile Agent-Based Algorithm for Network Routing and Management," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 3, pp. 193-202, Mar. 2003.

[23] T. Umezawa, I. Satoh, and Y. Anzai, "A Mobile Agent-Based Framework for Configurable Sensor Networks," *Proc. Fourth Int'l Workshop Mobile Agent for Telecomm. Applications,* pp. 128-140, Oct. 2002.



Qishi Wu received the BS degree in remote sensing and GIS from Zhejiang University, People's Republic of China in 1995, the MS degree in geomatics from Purdue University in 2000, and the PhD degree in computer science from Louisiana State University in 2003. He is currently a postdoctoral research fellow in the Computer Science and Mathematics Division at Oak Ridge National Laboratory. His research interests include computer networks, distributed sensor networks, algorithms, and artificial intelligence. He is a member of the IEEE and IEEE Computer Society.



Nageswara S.V. Rao received the BTech degree in electronics and communications engineering from the Regional Engineering College, Warangal, India, in 1982, the ME degree from the School of Automation, Indian Institute of Science, Bangalore in 1984, and the PhD degree in computer science from Louisiana State University in 1988. He is currently a distinguished research staff member at Oak Ridge National Laboratory, where he has worked since 1993. He was an assistant professor of computer science at Old Dominion University during 1988-1993. His research interests include network transport dynamics, statistical approaches to transport control, information and sensor fusion, robot navigation, and fault diagnosis. He is a senior member of the IEEE and the IEEE Computer Society.

Jacob Barhen received the DSc degree from the Technion-Israel Institute of Technology, Haifa, in 1978. He is the director of the Center for Engineering Science Advanced Research (CESAR) at the Oak Ridge National Laboratory (ORNL). In March 1999, he was named Corporate Fellow. He is also a nonresident affiliate of the California Institute of Technology's Jet Propulsion Laboratory (JPL) and an adjunct professor in the Department of Computer Science at the University of Tennessee, Knoxville. From 1987 to 1994, Dr. Barhen was the head of the Neural Computation and Nonlinear Science Group at Caltech/JPL. He began his career at ORNL, where he headed the Machine Intelligence Group from 1978 to 1987. At both institutions, he established world-class research groups in artificial neural information processing and computational science. He has been the principal investigator of numerous basic and applied research projects funded by US Government agencies. Currently, the Missile Defense Agency, the US Department of Energy's Office of Science, NASA, and the US Army support his work. Dr. Barhen's research interests include global optimization, neural networks, emerging computational systems, sensitivity and uncertainty analysis, optical information processing. He has authored more than 160 scientific papers, and holds eight US patents. He has received three NASA Space Act awards for major contributions to the National Space Program, 11 NASA awards for Technical Innovation, and a 1998 R&D_100 award, for the invention of the TRUST global optimization method. He is a member of the editorial boards of *Neural Processing Letters* (Kluwer Academic), *Neural Networks* (Pergamon Press), *Mathematical and Computer Modeling* (Pergamon Press), and *Concurrency* (J. Wiley). He is a member of the AAAS, the IEEE, SPIE, the International Neural Networks Society, and the Planetary Society.

S. Sitharama Iyengar is the chairman and Roy Paul Daniels Chaired Professor of Computer Science and is also Satish Dhawan Chaired Professor at the Indian Institute of Science. He has been awarded the Distinguished Alumnus Award at the Indian Institute of Science in March 2003. He is the Distinguished Research Master Award winning professor of the Computer Science Department at Louisiana State University. He was involved with research in high-performance algorithms, data structures, sensor fusion, data mining, and intelligent systems since receiving his PhD degree (in 1974 at Mississippi State University) and his MS degree from the Indian Institute of Science (1970). He has served on several review panel committees and as a principal investigator on research projects supported by various government agencies. His publications include 13 books (authored or coauthored textbooks; Prentice-Hall, CRC Press, IEEE CS Press, John Wiley & Sons, etc.) and more than 280 research papers in refereed journals and conference in areas of high-performance parallel and distributed algorithms and data structures for image processing and pattern recognition, and distributed data mining algorithms for biological databases. His books have been used at Berkeley, Purdue, the University of Southern California, the University of New Mexico, etc. He was a visiting professor at the Jet Propulsion Laboratory-Cal Tech, Oak Ridge National Laboratory, the Indian Institute of Science, and the University of Paris. Dr. Iyengar has served as an associate editor for the IEEE, as guest editor for several publications, and as a series editor for *Neuro-Computing of Complex Systems* for CRC Press. He has been on the prestigious National Institute of Health-NLM Review Committee in the area of medical informatics for four years. Dr. Iyengar has received several awards including, in 1998, the prestigious IEEE CS Technical Achievement Award for outstanding contributions to data structures and algorithms in image processing and sensor fusion problems. He has been a distinguished lecturer for several associations, and a member of the ACM accreditation committee. He is also a fellow of the ACM, the IEEE, the AAAS, a Williams Evans Fellow, an IEEE Distinguished Visitor, etc., and a member of the European Academy of Sciences and the New York Academy of Sciences. He has given more than 50 plenary talks and keynote lectures at numerous national and international conferences.

Vijay K. Vaishnavi received the BE degree (with distinction) in electrical engineering from Jammu and Kashmir University and the MTech and PhD degrees in electrical engineering (with a major in computer science) from the Indian Institute of Technology, Kanpur. He has also done post-doctoral work in computer science for two years at McMaster University, Canada. He is currently a professor of computer information systems and computer science at Georgia State University. His current areas of research interest include interorganizational systems (semantic interoperability, directory services, Web-based virtual communities, coordination, security), software development (object-oriented metrics, software specifications and their maturity, object-oriented modeling and design), and data structures and algorithms (multisensor networks and fusion). The US National Science Foundation and private organizations, including IBM, Nortel, and AT&T, have supported his research. He has authored numerous papers in these and related areas. His papers have appeared in the *IEEE Transactions on Software Engineering*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Computers*, *SIAM Journal on Computing*, *Journal of Algorithms*, and several other major international journals and conference proceedings. Dr. Vaishnavi is an IEEE Fellow. He is also member of the IEEE Computer Society, the ACM, and the AIS.

Hairong Qi received the BS and MS degrees in computer science from Northen JiaoTong University, Beijing, in 1992 and 1995, respectively, and the PhD degree in computer engineering from North Carolina State University in 1999. She is now an assistant professor in the Department of Electrical and Computer Engineering at the University of Tennessee, Knoxville. Dr. Qi is a member of the IEEE and Sigma Xi. Her current research interests are collaborative signal and information processing in sensor networks, biomedical imaging, and automatic target recognition.

Krishnendu Chakrabarty received the BTech degree from the Indian Institute of Technology, Kharagpur, in 1990 and the MSE and PhD degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in computer science and engineering. He is now an associate professor of electrical and computer engineering at Duke University. During 2000-2002, he was also a Mercator Visiting Professor at the University of Potsdam in Germany. He is a recipient of the US National Science Foundation Early Faculty (CAREER) award and the US Office of Naval Research Young Investigator award. His current research projects include: design and testing of system on-chip integrated circuits, embedded real-time systems, distributed sensor networks, modeling, simulation and optimization of microfluidic systems, and microfluidics-based chip cooling. Dr. Chakrabarty is a coauthor of two books: *Microelectrofluidic Systems: Modeling and Simulation* (CRC Press, 2002) and *Test Resource Partitioning for System-on-a-Chip* (Kluwer, 2002), and the editor of *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation* (Kluwer 2002). He has published more than 150 papers in journals and refereed conference proceedings, and he holds a US patent in built-in self-test. He is a recipient of a best paper award at the 2001 Design, Automation, and Test in Europe (DATE) Conference. He is also a recipient of the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Germany. He is an associate editor, editor, and editorial board member of several publications. He has also served as an associate editor *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*. He is a member of the ACM and ACM SIGDA and a member of Sigma Xi and also a senior member of the IEEE and the IEEE Computer Society. He serves as vice chair of technical activities on IEEE's Test Technology Technical Council, and is a member of the program committees of several IEEE/ACM conferences and workshops.