Mobile Netw Appl (2011) 16:782–793
DOI 10.1007/s11036-010-0269-z

A Genetic Algorithm Approach to Multi-Agent Itinerary Planning in Wireless Sensor Networks

Wei Cai · Min Chen · Takahiro Hara · Lei Shu · Taekyoung Kwon

Published online: 15 October 2010

Abstract It has been shown recently that using Mobile Agents (MAs) in wireless sensor networks (WSNs) can help to achieve the flexibility of over-the-air software deployment on demand. In MA-based WSNs, it is crucial to find out an optimal itinerary for an MA to perform data collection from multiple distributed sensors. However, using a single MA brings up the shortcomings such as large latency, inefficient route, and unbalanced resource (e.g. energy) consumption. Then a novel genetic algorithm based multi-agent itinerary planning (GA-MIP) scheme is proposed to address these drawbacks. The extensive simulation experiments show that GA-MIP performs better than the prior single agent algorithms in terms of the product of delay and energy consumption.

Keywords mobile agent · wireless sensor networks · genetic algorithm · itinerary planning

1 Introduction

A wireless sensor network (WSN) [1] consists of spatially distributed sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. The development of WSNs is initially driven by military applications such as battlefield surveillance. WSNs are now used in many industrial and civilian application areas, including industrial process monitoring and control, environment and habitat monitoring, healthcare applications, home automation, and traffic control.

There is a burgeoning interest in using a Mobile Agent (MA) [2] in a WSN, which can address the software (re)installation issue for dynamically changing application functionalities (or their requirements) [3–5]. As a special kind of software, a MA migrates among sensor nodes to carry out task(s) autonomously. For instance, collecting sensory data from a number of source nodes with some processing defined on demand, and adaptively handling sensory data depending on time-varying network dynamics are typical application requirements of the MA dispatcher (i.e., the sink node) in WSNs. Using MAs has been shown to be an efficient approach to enhance such capabilities of WSNs [6, 7].

The MA design in WSNs can be decomposed into a few components [4]: (1) an overall framework, (2) itinerary planning, (3) a middleware system design, and (4) agent cooperation. Among these components, itinerary planning determines the order of sensory data source nodes to be visited during the MA migration, which has a significant impact on the performance of MA-based WSNs. Thus, finding out an optimal itinerary for a MA to visit the given set of the source nodes is crucial. However, finding an optimal itinerary is NP-hard [8], and heuristic algorithms have been proposed to compute itineraries with sub-optimal performance.

A number of itinerary planning schemes have been proposed in recent studies [6–9], but most of them focus only on the single MA itinerary in WSNs. Although

W. Cai · M. Chen · T. Kwon (✉)
School of Computer Science and Engineering,
Seoul National University, Seoul, Korea
e-mail: tkkwon@snu.ac.kr

T. Hara · L. Shu
Department of Multimedia Engineering,
Osaka University, Osaka, Japan

the MA technique can help to achieve the flexible (re)installation of application software on demand, using only a single MA may also bring some shortcomings, e.g., the long latency and the unbalanced energy consumption. In order to address these shortcomings of single MA itinerary planing (SIP), the potential advantage of using multi-agent itinerary planning is illustrated in [10]. In this paper, a novel genetic algorithm (GA) based multi-agent itinerary planning (GA-MIP) scheme is proposed, which seeks to reduce the delay by dispatching multiple MAs in parallel.

To realize the GA-MIP algorithm, we contrive to represent the itineraries of multiple MAs by encoding a Source Ordering Code and a Source Group Code (see Section 4.2 for details) into digits, which can be interpreted as a particular gene for genetic evolution. First, we set up a searching space from randomly selected genes. Then, we perform an iterative genetic evolution process. At each iteration, evolution operators such as crossover and mutation are applied to increase the variety of the genes. After this process, a fitness function is used to choose the better genes to survive for the next generation, which is analogous to the natural-selection in the real world. After a number of evolutionary iterations, the solution(s) corresponding to an efficient itinerary planning will be determined.

The contributions of this paper are summarized in the following:

– *Novelty*: Though the use of a genetic algorithm in planning the itinerary for a MA is presented in [9], only using a single agent is considered. To the best of our knowledge, this paper is the first effort to solve the multi-agent itinerary planning (MIP) problem based on the genetic algorithm.
– *Performance*: The previous MIP paper [10] breaks down the MIP solution procedure into three phases: (1) determining the number of MAs, (2) grouping source nodes for individual MAs, and (3) determining the visiting sequence for each MA. Intrinsically, these three phases merely transform a MIP problem to a repetition of SIP problems. This may lead to substantially sub-optimal solutions. In contrast, the proposed GA-MIP scheme in this paper considers the three phases simultaneously as a single problem for better performance.

The remainder of the paper is organized as follows. Related work is introduced in Section 2. Network model and problem statement are represented in Section 3. Then we describe the novel genetic algorithm-based multi-agent itinerary planning scheme in Section 4. Simulation experiments are performed in Section 5. Finally, Section 6 concludes this paper.

2 Related work

A number of studies have been conducted for MA itinerary planning in WSNs [6–11]. Among these heuristic proposals, Local Closest First (LCF) and Global Closest First (GCF) are simple approaches [7] for MA itinerary planning. LCF searches for the next sensorydata source node with the shortest distance to the current node while GCF selects the next closest node to the sink node as its next source node. MADD [6] is similar to LCF, but selects the farthest source node as the starting point of the itinerary.

IEMF extends LCF by estimating the communication cost in MA migration. It quantifies the increase in the MA packet size and then models the energy consumption of MA migration by *Estimated Cost*. In [9], a genetic algorithm for single MA itinerary planning is proposed to exploit the information of sensor detection signal levels and link power consumption.

However, all of these studies only focus on the single MA itinerary planning (SIP) problem. Thus, they have some intrinsic shortcomings of using a single MA, which aggravates as the number of source nodes becomes large [11] such as:

– *Delay Scalability Issue*: A single MA roams in a network for data collection. The duration of visiting all of the given source nodes in sequence will incur a large delay, especially as the network size increases;
– *Potential Route Inefficiency*: Often, the source nodes to be visited may be distributed in a clustered manner. A single MA then migrates from one cluster to another, carrying the data it retrieved before. The ever increasing MA packet size will consume substantially increased energy if the clusters are distant;
– *Traffic Load Issue*: In the perspective of the network lifetime, it is better to distribute the traffic load across the network. However, sensor nodes in the agent itinerary will deplete their energy faster than others. Note that the MA packet size will be increased as it collects more and more data from the sensor sources.

To address these problems, a multi-agent itinerary planning (MIP) is first proposed in [10], in which a circular area with the center of a cluster (i.e. the group of densely located source nodes) and some radius is determined as the coverage of each MA. This process repeats until all of the given source nodes are covered. In other words, a MIP problem is divided into individual SIP ones, which can be solved by the previous SIP solution repeatedly. However, this greedy approach may lead to a substantially sub-optimal MIP solution.

3 Network model and problem statement

We consider a WSN that consists of a number of randomly deployed static sensor nodes. A static sink node is deployed at the central location of the WSN, with almost infinite power supply and strong computational capability. A number of sensor nodes are battery-powered, each of which knows its geographical location. All the sensor nodes have the same transmission radius, and any two directly connected (1-hop) sensor nodes have a stable bi-directional link. The density of the sensor nodes deployed is assumed to be high enough to guarantee that each sensor node has at least two 1-hop neighbor nodes. A number of MAs can be issued by the sink node to visit the set of given source nodes simultaneously with different itineraries. That is, each MA has its own itinerary for visiting a subset of the source nodes. This paper seeks to reduce the task duration (or the delay of the MA migration to collect data from the given sources), which may be critical application requirements.

4 Genetic algorithm based multi-agent itinerary planning

In this section, we introduce the GA-MIP scheme. The encoding methods, the operators for the evolution, and the fitness function are the three key components in GA-MIP, to be detailed in Sections 4.2, 4.3, and 4.4.

4.1 Introduction to genetic algorithm

A genetic algorithm [12] (GA) is an adaptive heuristic search algorithm based on the evolutionary theory of genetics and natural selection, which simulates the survival of the fittest principle. In a GA system, each solution to the problem is described as an individual with a particular genetic instance (or simply a gene) in the nature. The solutions (or parents) produce children that inherit mixed gene from their parents. Meanwhile, an opportunistic mutation may happen to generate new individuals. Through the evaluation by a fitness function, the better individuals could survive. As time goes on, the survivals contain the better genes which represent the better solutions to the problem.

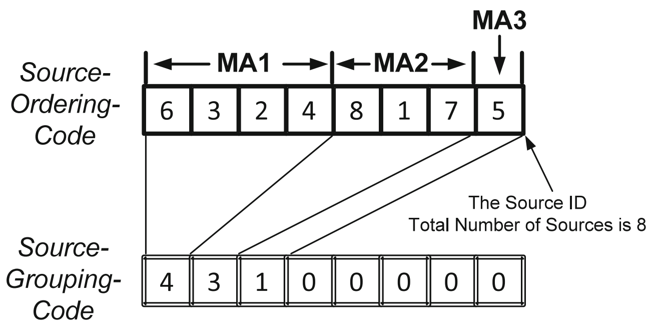
4.2 Encoding method

To enable GA implementation, we represent a solution to the (MIP) problem as digits representing a gene. In GA-MIP, we devise a gene consists of *Source-Ordering-Code* and *Source-Grouping-Code*, which have the following definitions:

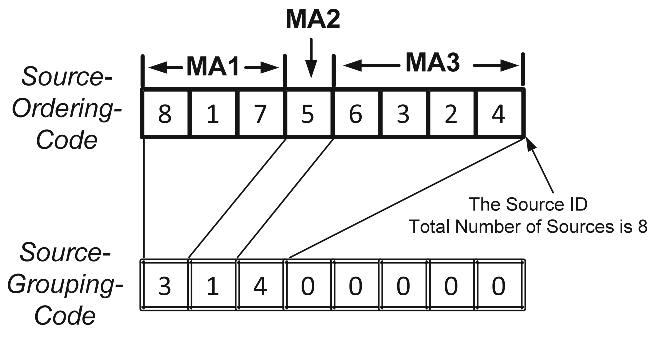
1. *Source-Ordering-Code*. It is an array of the source nodes' identifier numbers. Actually, the array is the concatenation of the sources to be visited; more precisely, the array is the concatenation of the segments, each of which is the group of sources. That is, each segment of the array will be covered by each MA, where the MA will traverse the given sources of the segment from left to right. A segment is called a Source Node Group, which is the sources to be visited by a particular MA.
2. *Source-Grouping-Code*. It is an array of numbers, each of which indicates the number of source nodes that are allocated in the corresponding Source Node Group. In other words, each number in the array is the segment size.

Note that the combination of *Source-Ordering-Code* and *Source-Grouping-Code* represents a single MIP solution or a gene.

Figure 1a shows an example for the substantiation of the encoding. The number of non-zero elements in *Source-Grouping-Code* represents the number of MAs;



(a) The codes with 3 MAs and 8 source nodes



(b) The codes of the same solution with different permutation

Fig. 1 The encoding examples

thus, there are three MAs in Fig. 1a. Furthermore, the value of each non-zero element indicates the number of source nodes to be visited by the corresponding MA. For example, the three non-zero elements (i.e., 4, 3, 1) in the *Source-Grouping-Code* mean the following information: (1) The first MA (MA1 in Fig. 1a) will visit 4 source nodes in the order of 6, 3, 2, 4; (2) The second MA (i.e., MA2) will visit the source nodes in the order of 8, 1, 7; (3) The third MA (i.e., MA3) will visit the source node with ID of 5.

Additionally, the elements in the *Source-Grouping-Code* must be sorted by a descending order, in order to guarantee the uniqueness of a source grouping solution. Since each *Source-Ordering-Code* is associated with a *Source-Grouping-Code*, such uniqueness can eliminate the duplicated genes. Otherwise, the combinations of the two codes likely produce duplicate solutions. For instance, without sorting the elements in the source-grouping code, the GA technique may generate two solutions of the same gene (see Fig. 1b is equivalent to Fig. 1a).

4.3 Operators

In this section, we describe the genetic operators for the GA-MIP scheme. Like the conventional operators for GA, we have the *crossover* and *mutation* operators.

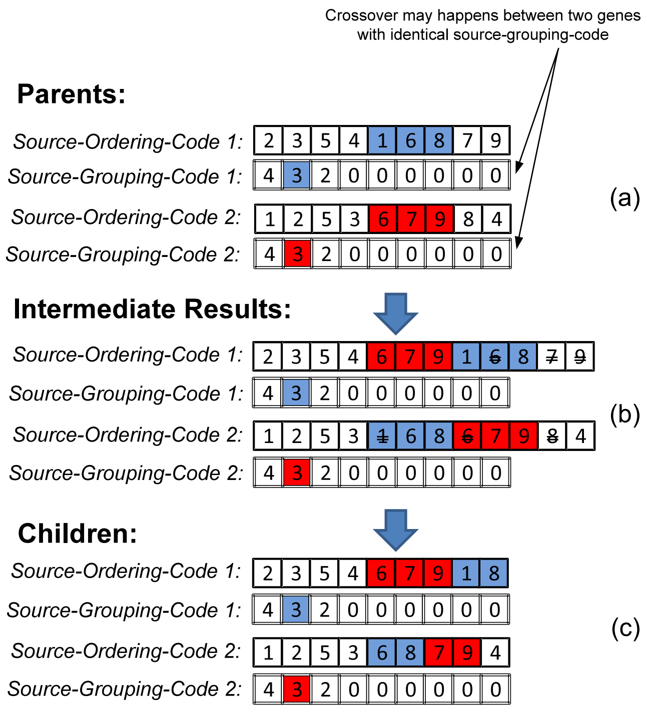

Crossover may happens between two genes with identical source-grouping-code
Parents:
Source-Ordering-Code 1: 2 3 5 4 1 6 8 7 9
Source-Grouping-Code 1: 4 3 2 0 0 0 0 0 0
Source-Ordering-Code 2: 1 2 5 3 6 7 9 8 4
Source-Grouping-Code 2: 4 3 2 0 0 0 0 0 0
(a)
Intermediate Results:
Source-Ordering-Code 1: 2 3 5 4 6 7 9 1 6 8 7 9
Source-Grouping-Code 1: 4 3 2 0 0 0 0 0 0
Source-Ordering-Code 2: 1 2 5 3 1 6 8 6 7 9 8 4
Source-Grouping-Code 2: 4 3 2 0 0 0 0 0 0
(b)
Children:
Source-Ordering-Code 1: 2 3 5 4 6 7 9 1 8
Source-Grouping-Code 1: 4 3 2 0 0 0 0 0 0
Source-Ordering-Code 2: 1 2 5 3 6 8 7 9 4
Source-Grouping-Code 2: 4 3 2 0 0 0 0 0 0
(c)

Fig. 2 An example for source-ordering-code crossover

4.3.1 Crossover operators

A crossover operator is a key component in GA. It imitates the way of natural biological evolution. There are several crossover schemes have been proposed, such as one-point crossover [13] and multi-point crossover [14].

In our approach, crossover is only applied between the two segments in two *Source-Ordering-Codes* whose corresponding sizes in the *Source-Grouping-Codes* are equal as shown in Fig. 2 (in this case 3). Suppose there are two parents (or solutions). We first randomly select a non-zero element (denoted by m) that is common in the two parents' *Source-Grouping-Codes*; that is, each parent should have a segment of the same length m.[1] Thus, m is mapped to a segment in the *Source-Ordering-Code* of each of the two parents (the second segments in both parents in Fig. 2a). The selected segment of one parent is inserted to that of the other parent and vice versa as shown in Fig. 2b. Note that the duplicate source IDs after the crossover should be eliminated to ensure the correctness of the *Source-Ordering-Code*.

4.3.2 Mutation operators

The mutation operator is used to accommodate the variety of the genes so that the discovery of new (hopefully better) solutions is possible. In our approach, both *Source-Ordering-Code* mutation and *Source-Grouping-Code* mutation are incorporated independently.

For the *Source-Ordering-Code*, the operator randomly selects two elements in the code and switches their positions. Figure 3 illustrates the mutation of a *Source-Ordering-Code*, where the second and the fifth elements transpose their positions. Let an array $soc[]$ with size of N denote the *Source-Ordering-Code* to be mutated. Note that the the elements are located from $soc[1]$ to $soc[N]$. Its mutation pseudo-code is given in Algorithm 1.

Algorithm 1 Source-Ordering-Code Mutation
```
1: Pick two integers randomly $r_1, r_2 \in [1, N], r_1 \leq r_2$
2: $Temp \leftarrow soc[r_1]$
3: $soc[r_1] \leftarrow soc[r_2]$
4: $soc[r_2] \leftarrow Temp$
5: return
```

[1]Even if they do not have the same length segment currently, *Source-Grouping-Codes* are changed by the mutation process at each iteration, to be detailed later. Thus, they will have the same length segment at some later iteration.

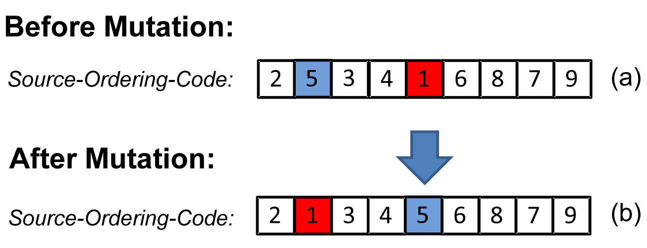


Fig. 3 The example for source-ordering-code mutation

The mutation of the *Source-Grouping-Code* is performed by an incremental approach. Two elements in the *Source-Grouping-Code* will be randomly selected, one will be incremented by 1 and the other will be decremented by 1. Notice that we have to ensure the the values of the elements in the *Source-Grouping-Code* after the mutation should still be within the range of $[0, N]$, where N is the number of total source nodes to be visited. Denote the array of the elements in the *Source-Grouping-Code* by $sgc[]$. In an extreme case, there will be one MA for one source node. Then $sgc[]$ will have N elements. To allow even this unlikely case to happen, we consider maximum N elements from $sgc[1]$ to $sgc[N]$. As illustrated in Fig. 4, the (randomly selected) third and fifth elements are decremented and incremented, respectively. So, the third element decreases to 0 and the fifth element increases to 2, which is shown as the "Intermediate Result" in Fig. 4b. As the values of the elements are not listed in a numerical order, we perform sorting to produce the final *Source-Grouping-Code* in Fig. 4c. The pseudo-code of the mutation on *Source-Grouping-Code* is given in Algorithm 2.

Algorithm 2 Source-Grouping-Code Mutation
```
1: repeat
2:     Pick two integers randomly $r_1, r_2 \in [1, N]$
3: until $sgc[r_1] > 0$ and $sgc[r_2] < N$
4: Decrease $sgc[r_1]$ by 1
5: Increase $sgc[r_2]$ by 1
6: Sort $sgc[]$ by Descending Order
7: return
```

4.4 Fitness function

For the purpose of selecting the better genes to survive for the evolution at each iteration, we propose a fitness function to evaluate the performance of a gene. The fitness function should estimate the communication cost of each solution, which represents the itineraries

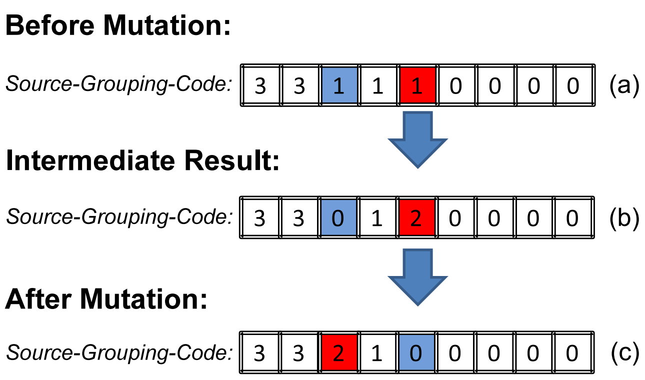


Fig. 4 The example for source-grouping-code mutation

of MAs determined by the combination of the *Source-Ordering-Code* and the *Source-Grouping-Code*.

As in [8], we employ the similar model to estimate the MA migration energy cost for each MA. The cost of k-th MA's itinerary is denoted by $E_k, k = 1, 2, ..., K$, where K is the total number of MAs dispatched in the current solution.

In order to calculate E_k, we first calculate the MA size, which increases as it traverses the source nodes since it accumulates the data from the sources that have visited. Let S_{data} denote the size of raw sensory data at a source node. Also, let S_{proc} and S_{head} denote the size of the processing code of the MA, and the MA header size, respectively. To model this MA increase, first let $S^0_{\text{ma}} = S_{\text{proc}} + S_{\text{head}}$ denote the size of MA when dispatched from the sink. Then, let S^i_k denote the size of the k-th MA when it leaves after processing data at i-th source in the source list assigned to this MA. Let r denote the reduction ratio by agent assisted local processing, and let ρ denote the aggregation ratio for redundancy elimination among the sensory data collected in different sources. Then S^i_k is given by:

$$\begin{aligned} S^i_k &= S^{i-1}_k + (1-\rho)(1-r)S_{\text{data}} \\ &= S^0_{\text{ma}} + [1 + (i-1)(1-\rho)](1-r)S_{\text{data}} \end{aligned}$$

Let m_{tx} and m_{rx} be the energy consumption for receiving and transmitting a data bit, respectively. Let e_{ctrl} denote the energy consumed for control messages (e.g. an ACK frame). We assume that m_{tx}, m_{rx} and e_{ctrl} are identical at each node without power control. Let S_{rx} and S_{tx} be the size of a received packet and that of a transmitted packet. The communication energy consumption at a source by receiving the MA (whose size is S_{rx}) and then transmitting MA (whose size is S_{tx}) can be calculated by:

$$e(S_{rx}, S_{tx}) = m_{rx} \cdot S_{rx} + m_{tx} \cdot S_{tx} + e_{ctrl}$$

Let $H(i-1, i)$ denote the hop counts between two adjacent sources in the itinerary of an MA (say, $i-1$-th source to i-th source in the MA's list). Also, let m_p denote the energy consumption for processing a bit. Finally $E^i_{i-1}(S^{i-1}_k)$ is the communication energy cost when a MA migrates from its $i-1$-th source to i-th source with the MA size S^{i-1}_k, which is given by:

$$E^i_{i-1}(S^{i-1}_k) = m_p \cdot S_{\text{data}} + e\left(0, S^{i-1}_k\right) + H(i-1, i) \cdot e\left(S^{i-1}_k, S^{i-1}_k\right) + e\left(S^{i-1}_k, 0\right)$$

We divide the whole itinerary of the k-th MA into three phases:

1. *Code-conveying phase*: the goal of this phase is to convey the processing code to the target region. In the phase, the the k-th MA migrates from the sink to its first source node s^1_k. Let $E1_k$ denote the communication energy consumption in this phase. Then, $E1_k = H(t, s^1_k) \cdot e(S^0_{\text{ma}}, S^0_{\text{ma}})$, where t denote the sink node.
2. *Roaming phase*: starting from the time when the k-th MA leaves s^1_k to the time when it visits its last source node $s^{n(k)}_k$, where $n(k)$ denote the number of source nodes needed to be visited by the k-th MA; let $E2_k$ denote the communication energy consumption in this phase. Then, $E2_k = \sum_{i=2}^{n(k)} E^i_{i-1}(S^{i-1}_k)$.
3. *Returning phase*: starting from the time when the k-th MA finishes visiting all the assigned source nodes to the time when it returns to the sink. The communication energy consumption in this phase is denoted by $E3_k$, i.e., $E3_k = m_p \cdot S_{\text{data}} + e(0, s^{n(k)}_k) + H(s^{n(k)}_k, t) \cdot e(s^{n(k)}_k, s^{n(k)}_k)$.

Finally, the communication energy for the migration of the k-th MA can be estimated by $E_k = E1_k + E2_k + E3_k$. Let $E_{\text{migration}}$ denote the total migration cost of all the MAs. Thus, $E_{\text{migration}} = \sum_{k=1}^{K} E_k$.

Consequently, the fitness function considers the energy cost of the migration of the MAs by $f = E_{\text{migration}}$. During the GA evolution, the j-th gene corresponds to a particular itinerary solution, whose cost is denoted by $f(j)$, $j = 1...P$, where P is the number of selected solutions (or genes) at each iteration. After the crossover and mutations, the population of genes is expanded by a certain factor (say, α); that is, the number of genes is increased to $(1+\alpha)P$. The larger is α, the better genes can be selected for the next generation at the cost of the more computation overhead. In our simulation, α is set to 1. In order to construct the population of the next generation (or iteration), the selection operator chooses the P best genes among $(1+\alpha)P$ genes according to the lower (or better) value of their fitness function values. This procedure will be repeated for I iterations.

4.5 Pseudo-code of GA-MIP

We summarize the GA-MIP procedure as pseudo-code in Algorithm 3. A particular solution (or gene) is denoted simply by $G[]$, which is actually the combination of $soc[]$ and $sgc[]$. Note that as in the nature world, both crossover and mutation occur with some probability distributions. In GA-MIP, crossover is performed only on *Source-Ordering-Code* by the probability p_{OC}. Likewise, the probabilities of mutation on *Source-Ordering-Code* and *Source-Grouping-Code* are denoted by p_{OM} and p_{GM}, respectively. Recall that we denote the number of selected solutions (or genes) at each iteration by P and the number of iterations by I.

Algorithm 3 GA-MIP Algorithm

```
 1: Source-Ordering-Code random list of $N$ sources
 2: Source-Grouping-Code random initialization
 3: for $i = 1$ to $P$ do
 4:   Random Gene Initialization $G[i]$
 5: end for
 6: for $k = 1$ to $I$ do
 7:   for $i = 1$ to $P$ do
 8:     select $r_1, r_2, r_3 \in [0, 1]$, from a uniform distribution
 9:     if $r_1 \le p_{OC}$ then
10:       for $j = 1$ to $P$ do
11:         if $sgc[]$ in $G[j]$ equals $sgc[]$ in $G[i]$ then
12:           Source-Ordering-Code Crossover on $G[i]$, $G[j]$
13:           Break
14:         end if
15:       end for
16:     end if
17:     if $r_2 \le p_{OM}$ then
18:       Source-Ordering-Code Mutation on $G[i]$
19:     end if
20:     if $r_3 \le p_{GM}$ then
21:       Source-Grouping-Code Mutation on $G[i]$
22:     end if
23:     Produce Child Gene $G[N+i]$
24:   end for
25:   Sort $G[]$ by descending order by fitness function
26:   Select from $G[1]$ to $G[P]$
27: end for
28: return the best gene in $G[]$
```

Table 1 Simulation parameters for mobile agent system

Raw data size	2,048 bits
MA code size	1,024 bits
MA accessing delay	10 ms
Data processing rate	50 Mbps
Raw data reduction ratio	0.8
Aggregation ratio	0.9
Network size	1,000 × 500
Radio transmission range	60 m
Number of sensor nodes	800
MAC layer standard	802.11 b

5 Simulation

5.1 Simulation setup

We implemented the proposed GA-MIP algorithm as well as several existing SIP algorithms, such as LCF [7], MADD [6] and IEMF[8]. We use OPNET[2] and perform simulations. The same network model in [10] is adopted, in which the nodes are uniformly deployed within a 1,000 × 500 m field, and the sink node is located at the center of the field and multiple source nodes are randomly distributed in the network. To verify the scaling property of our algorithms, we select a large-scale network with 800 nodes. The parameters for MA system are shown in following Table 1.

5.2 Evaluation metrics

In order to evaluate the time and energy efficiency from the simulation results, we consider the following four performance metrics as follows:

- *Estimated energy cost*: The value of estimated energy cost calculated by the fitness function. Even though these values are just approximate energy cost in the network, they still could be a criterion to evaluate the convergence of the GA evolution progress.
- *Task duration*: In a SIP algorithm, it is equivalent to average end-to-end report delay, which is the average delay from the time when a MA is dispatched by the sink to the time when the agent returns to the sink. In an MIP algorithm, since multiple agents work in parallel, there must be one agent which returns to the sink at last. Then, the task duration of an MIP algorithm is the delay of that agent.
- *Task energy*: The task energy is the total communication energy consumption, which includes transmitting, receiving, retransmissions, overhearing and collision, to obtain sensory data from all the target sources.
- *Energy-Delay Product (EDP)*: For time-sensitive applications over energy constrained WSNs, EDP (calculated by $EDP = energy \times delay$) gives us a unified view. The smaller the value of EDP is, the better the unified performance will be.

[2]OPNET, http://www.opnet.com/.

5.3 Parameters selection for genetic algorithm

Some parameters are critical to the performance of the genetic algorithm, such as the iteration times, the size of search space, the ratio of crossover and mutations for *Source-Ordering-Code* and *Source-Grouping-Code*. In this section, we evaluate the impacts of these parameters on the performance of the GA evolution, which is the estimated energy cost $E_{\text{migration}}$.

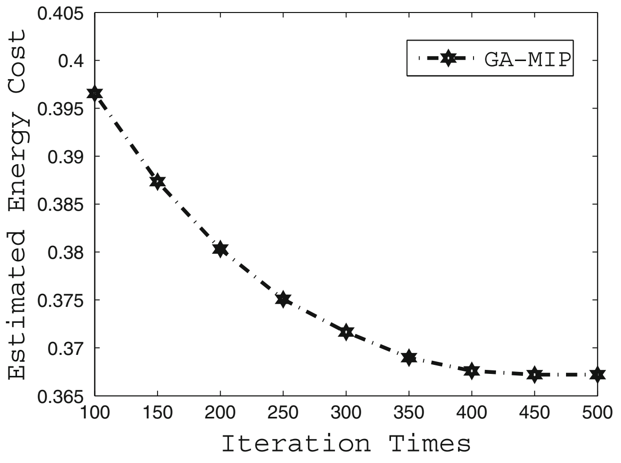
Figure 5 illustrates the impacts of the number of iterations (I) and the size of search space (P) on the estimated cost. obviously, the more iterations and the larger search space in GA evolution will lead to lower estimated cost. Thus, regarding to the enhancement of GA-MIP, we should set their values as large as possible. However, the larger iteration times and search space will incur the more computational overhead. There is a trade-off between the algorithm performance and the computational complexity. We observe that the estimated cost almost converges after 450 iterations, meanwhile, 400 is an acceptable value for the size of search space.

In addition, we study the optimal ratio for performing crossover and mutation during the gene evolution. As shown in Fig. 6a–c, the GA estimated energy cost reaches the minimum when the values for the ordering crossover ratio, ordering mutation ratio and grouping mutation ratio are around 0.8, 0.5, and 0.4, respectively. We adopt these values to set p_{OC}, p_{OM} and p_{GM} in evaluating GA-MIP.

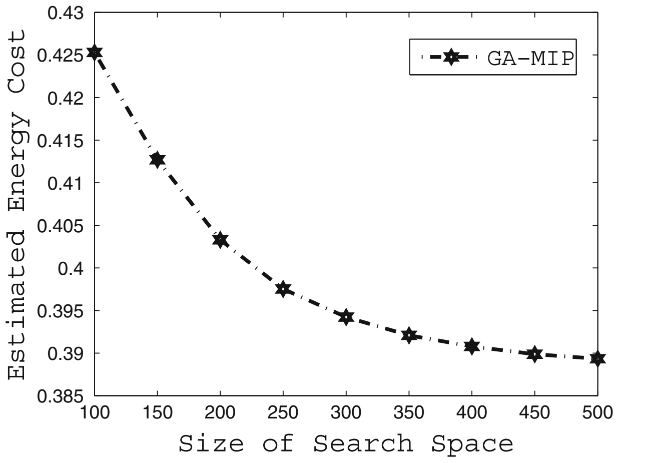
Table 2 lists the GA parameters adopted in evaluation.

5.4 Visualization of search result computed by GA-MIP and LCF

Figures 7 and 8 visualize different outcomes of the MIP solution (GA-MIP) and SIP solution (LCF). In the 20 source node scenario, Fig. 7 shows the result of the itinerary searching of GA-MIP, in which three MAs are sent by the sink node to collect the sensory data in source nodes simultaneously. In contrast, for the same network topology, Fig. 8 shows the result of the itinerary planning by LCF, in which a single MA travels

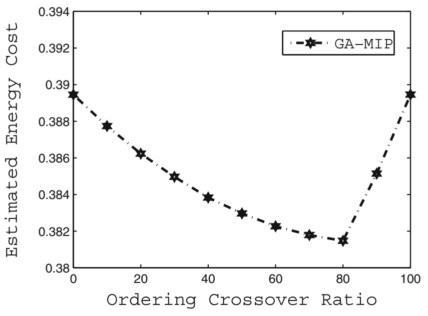


(a) More iteration leads to lower estimated energy cost

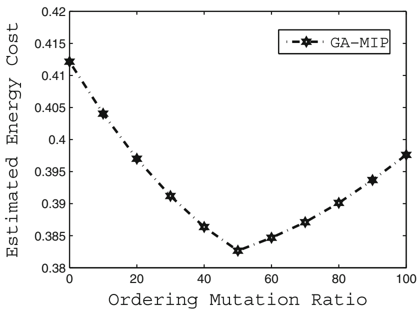


(b) Larger search space leads to lower estimated energy cost

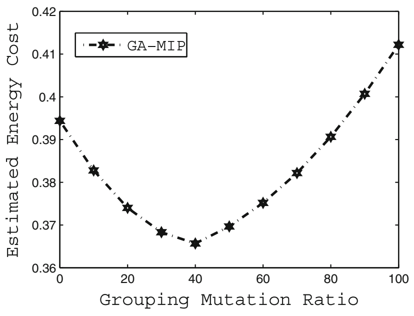
Fig. 5 The larger iterations and search space lead to the lower estimated energy cost



(a) Approximate Optimal Ratio: 0.8



(b) Approximate Optimal Ratio: 0.5



(c) Approximate Optimal Ratio: 0.4

Fig. 6 The impacts of the changing crossover and mutation ratios

Table 2 Simulation parameters for GA-MIP

GA iterations (I)	450
GA search space (P)	400
Ordering crossover ratio	0.8
Ordering mutation ratio	0.5
Grouping mutation ratio	0.4
Population expanding factor α	1

a long route in the network. Apparently, for GA-MIP, the task duration is significantly reduced during the data collection by delivering multiple MAs in parallel.

One of the salient features of GA-MIP is not to split the grouping and source visiting order into two problems. Thus, it can dynamically adapt to the deployment of source nodes by choosing some appropriate number of MAs. We provide another two snapshots for readers to see the outcomes of GA-MIP scheme with varying source node deployments. Note that the number of source nodes is identically 20 in these two tasks, however, the deployment of source nodes are different due to the random seed. The number of MAs resulted from the GA-MIP scheme is different. For instance, in Fig. 9, only two MAs are dispatched into the network, but in Fig. 10 four MAs are delivered.

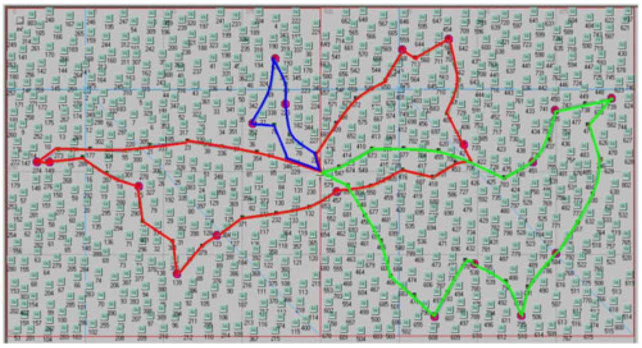


Fig. 7 Visualization of GA-MIP with 3 MAs

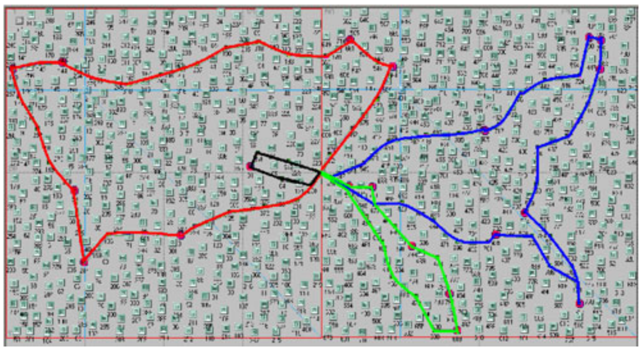


Fig. 10 Visualization of GA-MIP with 4 MAs

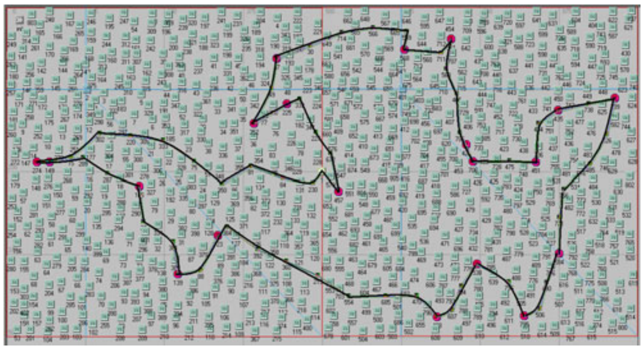


Fig. 8 Visualization of LCF with 1 MA

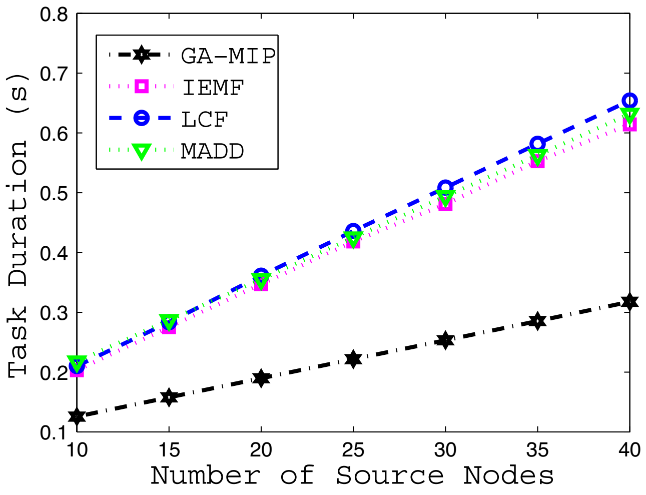


Fig. 11 The impact of number of source nodes on task duration

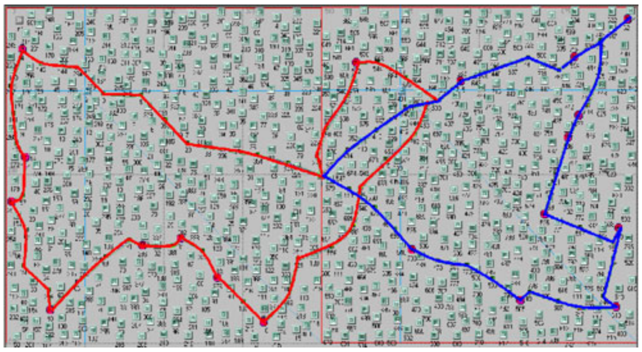


Fig. 9 Visualization of GA-MIP with 2 MAs

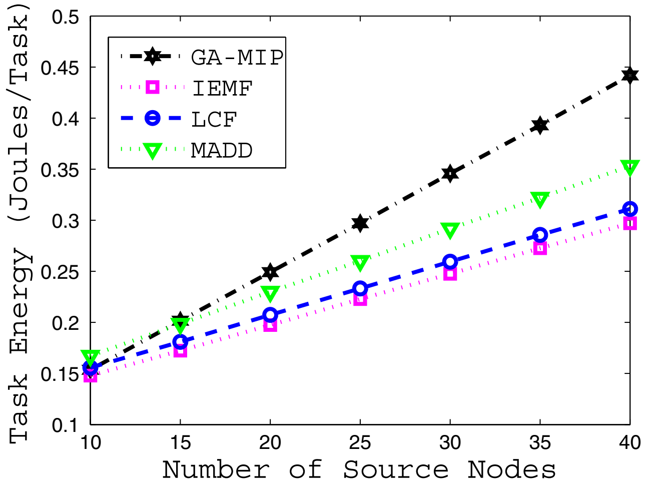


Fig. 12 The impact of number of source nodes on energy cost

5.5 Comparison of GA-MIP, LCF, MADD and IEMF with varying source number

In this section, we compare the GA-MIP scheme with the three SIP algorithms, i.e., LCF [7], MADD [6] and IEMF[8]. During the simulation, the number of source nodes is changed for 10–80 by the step size of 10. For each scenario of the particular number of source nodes, extensive simulations are performed with different random seeds. The distribution of source nodes is randomly selected by the recalculation of the seeds.

As shown in Fig. 11, the GA-MIP algorithm achieves a significant gain in terms of task duration, which is less than half of those of LCF, MADD and IEMF. The reason for this phenomenon is that in single MA systems, one MA should travel along the whole network to collect information in all sensor nodes. This procedure will cost a large latency since the sensor nodes may be distributed all over the network. Multi-agent can speed up the task because more than one itineraries are applied simultaneously.

In Fig. 12, the energy consumption of the GA-MIP algorithm is comparable to those of SIP algorithms when the number of source nodes is low. However GA-MIP's energy consumption becomes higher than the others as the number of sources increases. It is because the three following reasons:

- In a GA-MIP solution, multiple MAs are sent, thus more energy is consumed for delivering the larger size of processing codes while they are traveling in the WSN. By comparison, only one process code is carried by the single MA;
- Since multiple MAs migrate in parallel, contention may exist when two itineraries overlaps each other. Especially, the number of MAs increases when the number of the source nodes is large, which may cause a higher possibility for the contentions among multiple itineraries;
- During MA migration, data aggregation between source nodes leads to less accumulate data size from source nodes to the sink, thus, reduce the energy consumption. A SIP solution dispatches a single MA into the surveillance area, which implies maximum data aggregation. In contrast, dispatching more than one MA will potentially reduce the data aggregation, and thus, lead to more energy consumption.

However, from the visualization in Figs. 7 and 8, we can realize that the MIP solutions cover more intermediate nodes in the dissemination procedure. It implies that MIP can achieve better global load balancing than SIP solutions, and thus a longer network lifetime.

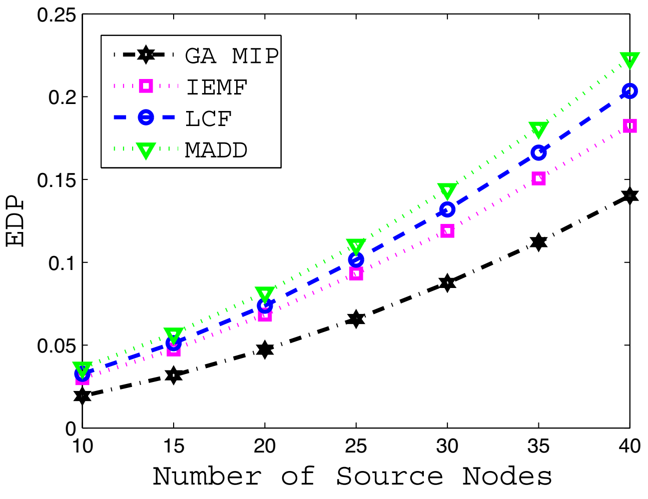


Fig. 13 The impact of number of source nodes on EDP

Instead of considering energy and delay separately, it is also important to test the combined performance of delay and energy (i.e., EDP), especially for delay constraint traffic in wireless sensor network. For example, in wireless multimedia sensor network and video sensor networks, EDP can be a viable criterion to evaluate the QoS provided by the WSN to support real-time video or multimedia services. In Fig. 13, we compare the EDP values of GA-MIP and other SIP algorithms. It shows that the GA-MIP algorithm achieves the better performance than SIP solutions overall, which verifies the effectiveness of the proposed algorithm.

6 Conclusion and future work

Applying mobile agents (MAs) in WSNs can facilitate a wide spectrum of dynamic applications that require flexible software (re)installation or adaptive data acquisition. In this paper, we first investigate the shortcoming of using a single MA in the literature such as large latency of data collection and global unbalance in energy consumption. Thus, we proposed a genetic algorithm based multi-agent itinerary planning (GA-MIP) algorithm to address the problems. We substantiate the proposed GA approach by encoding how many MAs are dispatched and which sensors are covered by individual MAs. Extensive simulations have been carried out to show the better performance of GA-MIP in terms of the product of delay and energy consumption. For the future work, we focus on the design of a more complicated fitness function. Currently, we simply model the energy consumption in the fitness function. However, depending on application requirements, we need to design different fitness functions targeting at various objectives, such as lowest task duration, longest lifetime, etc. Furthermore, the MIP problem would be investigated in the following promising and challenging scenarios:

- *Mobile sink node* [15]: If the sink node can be mobile, there is a good opportunity that we can further minimize the task duration and total communication cost by coordinating the multiple MAs and the mobile sink node.
- *Duty cycled WSNs* [16]: Currently, we assume that all the sensor nodes keep on operating, which is often not feasible in the real WSN applications. We need to investigate the MIP problem when some of sensor nodes are turned into sleep mode.

Acknowledgements This work was supported by the IT R&D program of MKE/KEIT. [KI001862, MKE/KEIT], and was partially supported by Grant-in-Aid for Scientific Research (S)(21220002) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Romer K, Mattern F (2004) The design space of wireless sensor networks. IEEE Wirel Commun 11(6):14–21
2. Tong L, Zhao Q, Adireddy S (2003) Sensor networks with mobile agents. In: Proceedings of the 2003 IEEE international conference on military communications (MILCOM 2003). Boston, Massachusetts
3. Beigl M, Krohn A, Zimmer T, Decker C, Robinson P (2003) AwareCon: situation aware context communication. In: Proceedings of the 5th IEEE international conference on ubiquitous computing (UbiComp 2003). Seattle, Washington, pp 132–139
4. Chen M, Gonzalez S, Leung V (2007) Applications and design issues for mobile agents in wireless sensor networks. Wirel Commun IEEE 14(6):20–26
5. Chen M, Kwon T, Yuan Y, Leung VC (2006) Mobile agent based wireless sensor networks. Journal of Computers 1(1):14–21
6. Chen M, Kwon T, Yuan Y, Choi Y, Leung VCM (2007) Mobile agent-based directed diffusion in wireless sensor networks. EURASIP J Appl Signal Process 2007(1):219–219
7. Qi H, Wang F (2001) Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks. In: Proceedings of the IEEE 2001 international conference on communications (ICC 2001). Helsinki, Finland
8. Chen M, Leung V, Mao S, Kwon T, Li M (2009) Energy-efficient itinerary planning for mobile agents in wireless sensor networks. In: Proceedings of the IEEE 2009 international conference on communications (ICC 2009). Bresden, Germany, pp 1–5
9. Wu Q, Rao NSV, Barhen J, Iyengar SS, Vaishnavi VK, Qi H, Chakrabarty K, Member S, Member S (2004) On computing mobile agent routes for data fusion in distributed sensor networks. IEEE Trans Knowl Data Eng 16:740–753
10. Chen M, Gonzlez S, Zhang Y, Leung VC (2009) Multi-agent itinerary planning for sensor networks. In: Proceedings of the IEEE 2009 international conference on heterogeneous networking for quality, reliability, security and robustness (QShine 2009). Las Palmas de Gran Canaria, Spain
11. Szewczyk R, Mainwaring A, Polastre J, Anderson J, Culler D (2004) An analysis of a large scale habitat monitoring application. In: Proceedings of the ACM 2004 2nd international conference on embedded networked sensor systems (SenSys 2004). Boston, MA, pp 214–226
12. Mitchell M (1998) An introduction to genetic algorithms. MIT
13. Poli R, Langdon WB (1998) Schema theory for genetic programming with One-Point crossover and point mutation. Evol Comput 6(3):231–252
14. Jong KAD, Spears WM (1992) A formal analysis of the role of multi-point crossover in genetic algorithms. Ann Math Artif Intell 5(1):1–26
15. Zhao Y, Wang Q, Jiang D, Wu W, Hao L, Wang K (2008) An agent-based routing protocol with mobile sink for wsn in coal mine. In: Proceedings of the 3rd international conference on pervasive computing and applications (ICPCA 2008). Alexandria, Egypt
16. Cheng L, Chen C, Ma J, Shu L, Chen H, Yang LT (2009) Residual time aware forwarding for randomly duty-cycled wireless sensor networks. In: Proceedings of the 7th IEEE/IFIP international conference on embedded and ubiqutious computing (EUC 2009). Vancouver, Canada

Wei Cai is currently a master candidate in School of Computer Science and Engineering at Seoul National University (SNU), Korea. He is working as an Research Assistant in Multimedia and Mobile Communications Laboratory under the supervision of Prof. Taekyoung Kwon. He received Bachelor of Engineering in Software Engineering from Xiamen University (XMU), China in 2008. During his undergraduate study, he served as the president of student union in software school and participated a one-year exchange program at department of computer science and engineering of Inha University, Korea. Before joining SNU, he was recommended to study in computer software and theory in Graduate School at Xiamen University. His research interests include interactive media transport over wireless networks, intelligent mobile agent system, wireless networking and ubiquitous network.

Min Chen is an assistant professor in School of Computer Science and Engineering at Seoul National University (SNU). Before joining SNU, he was a Post-Doctoral Fellow and Research Associate at University of British Columbia for three and half years. He received the Best Paper Runner-up Award from QShine 2008. He was interviewed by Chinese Canadian Times where he appeared on the celebrity column in 2007. He has published more than 70 technical papers. Dr. Chen is the author of OPNET Network Simulation (Tsinghua University Press, 2004). He serves as TPC co-chair and web chair for BodyNets-2010, workshop co-chair for CHINACOM 2010. He is the co-chair of MMASN-09 and general co-chair of UBSN-10. He was the TPC chair of ASIT-09, TPC co-chair of PCSI-09, publicity co-chair of PICom-09. He is workshop co-chair for EMC 2010. He is co-chair for ASIT 2010. He served as guest editors for several journals, such as ACM MONET, IJCS, IJSNET. He is a managing editor for IJAACS, and an editor for TIIS. He is an IEEE Senior Member.

Takahiro Hara received his BE and ME degrees in information systems engineering from Osaka University in Japan in 1995 and 1997, respectively. Currently, he is a research associate in the Department of Information Systems Engineering at Osaka University. His research interests include distributed database systems in advanced computer networks such as high-speed ATM networks and mobile computing environments. He is a member of the IEEE.

Lei Shu is a specially assigned Research Fellow in the Department of Multimedia Engineering of the Graduate School of Information Science and Technology at Osaka University in Japan. He performed the work described in this article while working as a research scientist in the Digital Enterprise Research Institute at the National University of Ireland, Galway. His research interests include wireless sensor networks, context-aware middleware, and sensor-network middleware and security. He has a PhD in digital enterprises from the National University of Ireland, Galway. He is a member of IEEE and the ACM. Contact him at lei.shu@ieee.org.

Taekyoung Kwon has been an Associate Professor in the School of Computer Science and Engineering, Seoul National University (SNU) since 2008. Before joining SNU, he was a Postdoctoral Research Associate at UCLA and at City University New York (CUNY). He obtained B.S., M.S., and Ph.D. degrees from the Department of Computer Engineering, SNU, in 1993, 1995, 2000, respectively. During his graduate program, he was a visiting student at IBM T. J. Watson Research Center and at University of North Texas. His research interest lies in sensor networks, wireless networks, IP mobility, and ubiquitous computing.