Research Article

Design of Randomly Deployed Heterogeneous Wireless Sensor Networks by Algorithms Based on Swarm Intelligence

Joon-Woo Lee and Won Kim

1Department of Robotics and Mechatronics, Advanced Manufacturing Systems Research Division, Korea Institute of Machinery and Materials (KIMM), 156 Gajeongbuk-Ro, Yuseong-Gu, Daejeon 305-343, Republic of Korea
2Department of Computer Science, Woosong University, 171 Dongdaejeon-Ro, Dong-Gu, Daejeon 300-719, Republic of Korea

Correspondence should be addressed to Won Kim; kimwon@wsu.ac.kr

Received 9 January 2015; Revised 23 April 2015; Accepted 23 April 2015

Copyright © 2015 J.-W. Lee and W. Kim. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper reports the design of a randomly deployed heterogeneous wireless sensor network (HWSN) with two types of nodes: a powerful node and an ordinary node. Powerful nodes, such as Cluster Heads (CHs), communicate directly to the data sink of the network, and ordinary nodes sense the desired information and transmit the processed data to powerful nodes. The heterogeneity of HWSNs improves the networks lifetime and coverage. This paper focuses on the design of a random network among HWSNs. In the design of a random HWSN, this paper uses algorithms based on the binary-valued versions of swarm intelligence, such as Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO). The design is then considered to be an optimization problem of how many powerful and ordinary nodes will combine to minimize the network cost, while guaranteeing a desired coverage during a given period. Simulation results show the performance of each algorithm for solving the defined optimization problem.

1. Introduction

A heterogeneous wireless sensor network (HWSN) consists of more than two types of devices (or sensor nodes) to improve the network’s lifetime, coverage, and scalability. The HWSNs lifetime can be effectively prolonged [1], especially for cases when powerful nodes that play a role as Cluster Heads (CHs) exist in a relatively large HWSN.

This paper focuses on the effective design of WSNs constructed by randomly deploying sensors, thus solving the optimization problem of how many powerful and ordinary nodes will combine to minimize the network cost while guaranteeing a desired coverage during a given period. Almost all of the sensors used in randomly deployed WSNs are constrained by their energy use because they have an embedded battery. Therefore, in studies on randomly deployed WSNs, the issue of how to effectively establish and efficiently manage the WSNs is very important and has been actively studied [2–13]. In particular, the researches of approaching this problem with metaheuristics have increased in recent years [2, 3, 5–14].

In previous research [4], the authors had simplified this design problem by assuming that only ordinary nodes sense the area to solve the problem through the mathematical analysis. Our previous paper [7] designed the HWSN with nodes and CHs that are able to sense information by defining the constrained integer optimization problem, converting it to the unconstrained problem and solving this problem using the discrete binary version of the PSO algorithm known as the Discrete PSO (DPSO) algorithm [15]. In this study, we solve this optimization problem with more realistic and efficient approaches at the expense of an increase in complexity by applying other metaheuristic methods based on swarm intelligence, such as binary Artificial Bee Colony (binABC) [16] and binary Ant Colony Optimization (binACO$_2$), which are the binary-valued versions corresponding to their original algorithms as well as the DPSO algorithm (hereafter renamed...
as binary PSO (binPSO)). Notably, the binACO$_R$ is first proposed in this paper. This algorithm is implemented by combining the binary concepts from the binPSO and the binABC with the ACO algorithm for continuous domains, ACO$_R$ [17]. The details of the binACO$_R$ are introduced in Section 3.3.

The remainder of this paper is organized as follows. In Section 2, we define the sensor system model and provide its coverage analysis. This section also defines the integer optimization problem with some constraints for the design of HWSN and briefly introduces Deb’s method that converts a constrained optimization problem to an unconstrained one and converts the defined problem to the unconstrained integer optimization as in [7]. In Section 3, the binary-valued optimization algorithms based on swarm intelligence are introduced. In Section 4, we show the comparative results between algorithms obtained from the simulation, designing several HWSNs. We then conclude this paper in Section 5.

2. Randomly Deployed Heterogeneous WSN Design

2.1. System Model. The proposed heterogeneous wireless sensor network (HWSN) is constructed by uniformly and randomly deploying $M$ powerful nodes (hereafter CHs) and $N$ ordinary nodes (hereafter nodes) over a large area $\mathbb{R}$ with a large size $|\mathbb{R}|$ because sensor clustering is usually used in relatively large WSNs. In the HWSN, the CHs communicate directly with the data sink of the network, and the nodes sense the desired information and transmit the processed data to powerful nodes. After each node collects the desired information, it directly transmits the data to its closest CH every $T$ hours (i.e., single-hop network). CHs then forward all data received to the sink of the HWSN. At this time, all CHs and nodes consume energy for the transmission of a data packet, and this is modified as follows [1]:

$$e(d) = kd^\gamma + c,$$

(1)

where $k$ is the loss coefficient and $c$ represents the overhead energy for a packet transmission. Additionally, $d$ and $\gamma$ denote the transmission distance and the path loss exponent, respectively.

For nodes and CHs, we assume that nodes have an initial energy of $E_i$ Joules and a sensing range of $R_s$ meters. CHs are not restricted by energy limitations. They also may or may not carry out a sensing task, and they have a sensing range of $R_s$.

One objective of this paper is to find proper $M$ and $N$ so that any arbitrary point in area $\mathbb{R}$ can be covered with at least probability $C_g$ (i.e., guaranteed coverage) during time $T_f$ (the network lifetime).

2.2. Coverage Analysis. Like in [4], we first find an initial coverage in terms of the number of nodes in a HWSN deployed randomly over the area from coverage theory [18]. Then, we transform the survival and lifetime of nodes during $T_f$ into a coverage problem in terms of the density of the CHs affecting survival nodes after $T_f$. From the two results above, we are able to lead the final coverage $C_f$ of the HWSN after $T_f$. This procedure is possible because we can state that a node has covered an area only when the transmission of the data to its closest CH after performing a sensing task is completed.

We will consider two cases: in the first case, only nodes sense the area; and in the second case, CHs also sense the area. The latter case is more realistic and effective than the former. Finally, with these results, we establish the network design in Section 2.3 as an integer program to minimize the network cost while keeping the desired network coverage over a desired period.

2.2.1. Case Sensed by Only Nodes. For the first case above, from the results of coverage process theory [18], the mean and variance of the probability $C_n$ of covering an arbitrary point in $\mathbb{R}$ by $N$ nodes are as follows:

$$\mu_{C_n} = 1 - e^{-\pi \lambda_1},$$

$$\sigma_{C_n}^2 = \frac{8\pi R_s^2}{|\mathbb{R}|} e^{-2N\lambda_1} \int_0^1 \left(e^{\lambda_1 B(x)} - 1\right) x \, dx - \pi \lambda_1,$$

(2)

where

$$B(x) = \begin{cases} \pi - 2x\sqrt{1 - x^2} - 2 \arcsin(x), & 0 \leq x \leq 1 \\ 0, & x \geq 1 \end{cases}$$

and $\lambda_1 = N R_s^2 / |\mathbb{R}|$. In the case that $|\mathbb{R}| \to \infty$, then $\sigma_{C_n}^2 \to 0$. Therefore, the initial coverage $C_i$ of $N$ nodes deployed randomly over a large area is the following deterministic value because its variance tends to 0:

$$C_i = C_n = 1 - e^{-\pi \lambda_1}.$$  

(4)

To determine the final coverage $C_f$, we have to find the number of dead nodes after $T_f$. Suppose $E_c$ represents the energy consumed by a node during $T_f$. This is then equal to the consumed energy per packet of transmission multiplied by the number of packets sent to CH. If the sensor has performed its tasks of sensing and transmitting, then $E_c < E_i$. From (1), we can yield the maximum distance between a node and a CH, allowing the node to perform its tasks up to $T_f$, as shown in Figure 1, which is determined as follows:

$$D_m = \left(\frac{E_i T}{kT_f} - \frac{c}{k}\right)^{1/\gamma}.$$  

(5)

Thus, if a node has no CH closer than $D_m$, the node will die before reaching $T_f$. The number of operating and survival nodes after $T_f$, therefore, is related to the area covered by $M$ disks with the radius $D_m$ and the center at CHs positions. Like finding the initial coverage of nodes from coverage process theory, we can find the probability that a certain node is close enough to a CH and survives until $T_f$. It is represented as follows:

$$C_f' = 1 - e^{-\pi \lambda'},$$  

(6)

where $\lambda' = MD_m^2 / |\mathbb{R}|$. 
The boundary effect is negligible because the sensing range of a node is much smaller than \( \|R\| \), and the nodes and CHs are placed in \( \mathbb{R} \) independently. Therefore, the final coverage of network is as follows:

\[
C_f = C_k \cdot C' = \left(1 - e^{-\pi \lambda_1}\right) \left(1 - e^{-\pi \lambda'}\right).
\]

### 2.2.2. Case Sensed by CHs and Nodes
In the more practical case where CHs also sense the area, we can determine the probability of covering an arbitrary point by CHs in a similar fashion to the first case as follows:

\[
C_{CH} = 1 - e^{-\pi \lambda_2},
\]

where \( \lambda_2 = M R^2_{S_2}/\|R\| \). At the beginning, an arbitrary point in \( \mathbb{R} \) is either covered by CHs or nodes, and at this time, the initial coverage probability is as follows:

\[
C_i = 1 - e^{-\pi (\lambda_1 + \lambda_2)}.
\]

We can assume \( R_{S_2} < D_n \) because the sensors range for communication is usually greater than that its range for sensing [19]. When the end of the desired period \( T_f \) is reached, a point in \( \mathbb{R} \) will either be covered by survival nodes, CHs, or both. The final coverage probability is as follows:

\[
C_f = C_k C' + C_{CH} - C_k C_{CH}.
\]

The first and second terms are the coverage probabilities of only survival nodes and CH, respectively. The third term represents coverage by nodes and CHs simultaneously, which is allowed because given that \( R_{S_2} < D_n \), all nodes that are located in the range covered by CHs have survived until \( T_f \). Therefore,

\[
C_f = 1 - e^{-\pi \lambda_1} + \left(e^{-\pi \lambda_2} - e^{-\pi \lambda'}\right) \left(1 - e^{-\pi \lambda_1}\right).
\]

### 2.3. Heterogeneous WSN Design
This section provides a non-linear integer optimization problem derived from the above coverage analysis with which we can predict the final network coverage with known parameters as a method to design a randomly deployed HWSN.

In previous research [4], the authors had simplified this design problem by assuming that only nodes sense the area to solve it through the mathematical analysis. This paper, however, is going to solve this optimization problem with more realistic approaches at the expense of increased complexity by applying optimization algorithms based on swarm intelligence. As in our previous work [7], we also consider the second case above because the case where CHs with rich energy resources also sense the area is efficient and effective in practical use.

We formulate the optimization problem to minimize the implementation cost while keeping at least guaranteed coverage \( C_g \) during the desired network lifetime \( T_f \) by adjusting the number of nodes and CHs. For this purpose, we assume that each node costs \( H_n \) and each CH costs \( H_{CH} \) for their implementation, such that \( H_n < H_{CH} \). Hence, the problem minimizing the network cost is as follows:

\[
\min_{N,M} f(N,M) = N H_n + M H_{CH}
\]

subject to \( C_f(N,M) \geq C_g, \quad N > 0, \quad M > 0 \).

In (13), \( C_f \) can be defined according to (7) or (11) according to the cases mentioned above.

As in [7], we can apply the binPSO algorithm to the above problem by first converting this constrained optimization problem into an unconstrained problem using the method of handling constraints. There are many methods for accomplishing this task: the penalty function method, the Lagrange multiplier method, the gradient projection method, and so forth. However, these methods are unfavorable because they require proper values for their control parameters to achieve satisfactory performance. So, like in [7], we apply Deb’s method [20] based on the penalty function approach, which does not require any penalty parameter. The details about this method can be found in [20].

By Deb’s method, the former problem is converted as follows:

\[
F(N,M)
\]

\[
= \begin{cases} 
  f(N,M), & \text{if } g(N,M) \geq 0, \quad N > 0, \quad M > 0 \\
  f_{\max} + |g(N,M)|, & \text{otherwise},
\end{cases}
\]

where \( F(N,M) \) is the fitness function and the value \( f_{\max} \) is the objective function value of the worst feasible solution of the solution candidates found by an algorithm. Additionally, \( g(N,M) = C_f(N,M)/C_g - 1 \), which is a normalized constraint.

### 3. Binary Optimization Algorithms Based on Swarm Intelligence

As defined in Section 2.3, the HWSN design becomes an integer optimization problem with an integer-valued solution. To
solve this problem, we use several optimization algorithms based on swarm intelligence, such as PSO, ABC, and ACO, because the more practical problem is more difficult to solve via mathematical analysis. However, the original forms of these algorithms are ill-suited to the integer optimization problem because these algorithms were first proposed to solve continuous-valued problems (by PSO [21] or ABC [22]) or combinatorial problems (by ACO [23]). Hence, we use the binary versions of each algorithm, that is, binPSO [15], binABC [16], and binACO [17]. In other words, we first find binary-valued solutions using binary algorithms and then convert these to integer-valued solutions. Then, we evaluate these converted solutions with the objective function of the above integer optimization problem. Notably, the binACO is first proposed in this paper. This algorithm is implemented by combining the binary concepts from the binPSO and the binABC with the ACO algorithm for continuous domains, ACO [17]. The details of the binACO are introduced in Section 3.3.

3.1. Binary Particle Swarm Optimization (binPSO) Algorithm. The Particle Swarm Optimization (PSO) algorithm is an optimization algorithm inspired by the movement of schools of fish and flocks of birds. In the PSO algorithm, a particle's position is a candidate solution for the given optimization problem and the particles, or swarm, are iteratively moved in the search space of an optimization problem according to information about each particle's previous best position, $p_{best}$, and the best position of the particles so far, $g_{best}$. Most variants based on the first PSO [21] have operated with continuous values in continuous search space, where the particles’ trajectories are determined by changes in position across some number of dimensions of the problem. In the binary version of the PSO algorithm (i.e., binPSO [15]), trajectories are defined as changes with the probability of whether the particle’s coordinate will take on a zero or one value. In the binPSO, although the velocity $v_{i,d}$ represents the probability of the coordinate $x_{i,d}$ taking the value 1 (unlike in the continuous version of the PSO), the formula remains unchanged as follows:

$$ v_{i,d} (k + 1) = w v_{i,d} (k) + \varphi_p r_p (p_{i,d} (k) - x_{i,d} (k)) + \varphi_g r_g (g_{i,d} (k) - x_{i,d} (k)) , $$

where the parameters $r_p$ and $r_g$ are determined using a uniformly distributed random number within the range $[0, 1]$, that is, $r_p, r_g \sim U(0, 1)$. In the first part of (15), the inertia weight $w$ represents the degree of the momentum of the particles. The second term is the “cognition component,” which represents the independent behavior of the particle itself. The final term is the “social component,” which represents the interaction between the particles. The constants $\varphi_p$ and $\varphi_g$ determine the relative influence of the cognition and social components. In (15), unlike the continuous version, the values of $x_{i,d}$, $p_{i,d}$, and $g_{i,d}$ are only integers in $\{0, 1\}$. From the probability $v_{i,d}$, the next coordinate of $x_{i,d}$ is determined as follows:

$$ x_{i,d} (k + 1) = \begin{cases} 1, & \text{if } rand() < S(v_{i,d} (k + 1)) \\ 0, & \text{otherwise}, \end{cases} $$

where the function $S(\cdot)$ is a sigmoid limiting transformation and $\text{rand()}$ is a quasi-random number selected from a uniform distribution with a range $[0, 1]$. Likewise, the continuous version of the PSO limited $v_{i,d}$ by the user parameter $V_{max}$; also in the binPSO, $V_{max}$ is retained; that is, $|v_{i,d}| < V_{max}$, and the smaller value allows a higher mutation rate.

3.2. Binary Artificial Bee Colony (binABC) Algorithm. The Artificial Bee Colony (ABC) algorithm is a population-based and stochastic optimization algorithm inspired by the intelligent foraging behavior of honeybees. In the ABC algorithm, artificial bees search for rich artificial food sources, that is, good solutions for a given optimization problem. The artificial bee colony consists of three groups of bees: employed bees associated with specific food sources that exploit their food sources, onlooker bees that watch the dance of employed bees within the hive to select a food source and help the employed bees find richer food sources, and scout bees that search food sources randomly. Likewise, their cooperative and iterative behaviors find the optimal solution to a given problem.

Most of the variants based on the original ABC [22] operate with continuous values in the continuous search space like the PSO. In contrast, the binary ABC algorithm (binABC [16]) finds a binary-valued solution in the binary-valued search space. The binABC borrows concepts from the binPSO. As mentioned above, in the binPSO, each component of the velocity vector, $v_{i,d}$, is used to compute the probability that the corresponding component of the solution vector is bit 0 or bit 1. In a similar way, the binABC uses each component of the neighbor food source position, $v_{m,i}^j$, to calculate the probability of the coordinate $x_{m,i}^j$ taking the value 1. Unlike in the continuous version of the ABC, the formula remains unchanged as follows:

$$ v_{m,i}^j = x_{m,i}^j + \phi_m^j (x_{m,i}^j - x_k^j) , $$

where the parameter $\phi_m^j$ is a uniform random number within the range $[-a, a]$, that is, $\phi_m^j \sim U(0, 1)$, where $a = 1$ in general. From the value $v_{m,i}^j$ calculated from (17), the next coordinate $x_{m,i}^j$ is determined as follows:

$$ x_{m,i}^j = \begin{cases} 1, & \text{if } \text{rand()} < S(v_{m,i}^j) \\ 0, & \text{otherwise}, \end{cases} $$

where the function $S(\cdot)$ is a sigmoid limiting transformation and $\text{rand()}$ is a quasi-random number selected from a uniform distribution with a range $[0, 1]$. Likewise, the binPSO
limited $v_{i,d}$ by the user parameter $V_{\text{max}}$; also in the binABC, $V_{\text{max}}$ is used; that is, $|v_{i,d}^t| < V_{\text{max}}$.

3.3. Binary Ant Colony Optimization (binACO$_R$) Algorithm.
The Ant Colony Optimization (ACO) algorithm is a meta-heuristic based on the foraging behavior of real ants. While moving, real ants that find food deposit pheromones on the way to their nests. The other ants then follow these deposited pheromones. Although pheromones evaporate as time passes, they open up new possibilities as ants can choose to follow a path heavily laden with pheromones. In this way, ants can search for the shortest path from their nest to a food source with only pheromone information. In ACO algorithm, a set of virtual ants searches for optimal solutions to a given optimization problem in the search space.

The first ACO algorithm [23] was proposed to solve a Traveling Salesman Problem (TSP) that is a representative combinatorial problem of finding the shortest path that traverses all cities exactly once and then returns to the starting city. Since then, the ACO for continuous domains, called ACO$_R$ [17], was proposed without any major conceptual change to the original ACO. In this paper, a binary version of the ACO$_R$ algorithm, binACO$_R$, is proposed by combining the above ACO$_R$ for continuous domains with the binary concepts from the binPSO, as in the binABC. In other words, each component $x_i^t$ is used to compute the probability of the next coordinate taking the value 1. In the ACO$_R$, each component $x_i^t$ is determined by sampling from the following Gaussian kernel for ith dimension:

$$G_i^t(x) = \sum_{k=1}^{N_t} \omega_i g_i^k(x)$$

$$= \sum_{k=1}^{N_t} \omega_i \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x - \mu_i)^2}{2\sigma_i^2}\right),$$

(19)

where

$$\omega_i = \frac{1}{q \sqrt{N_T} \sqrt{2\pi}} \exp\left(-\frac{(l-1)^2}{2(\gamma N_T)^2}\right),$$

$$\mu_i = x_i^t,$$

$$\sigma_i = \xi \sum_{k=1}^{N_t} \frac{|x_k - x_i^t|}{N_T - 1},$$

(20)

and $N_T$ is the size of the archive $T$ that stores and keeps the best $N_T$ solutions found by previous moving ants in the ACO$_R$. From each component $x_i^t$ selected from (19), the next coordinate of $x_i^t$ is determined as follows:

$$x_i^t = \begin{cases} 1, & \text{if rand()} < S(x_i^t) \\ 0, & \text{otherwise}, \end{cases}$$

(21)

where the function $S(\cdot)$ is a sigmoid limiting transformation and rand() is a quasi-random number selected from a uniform distribution with a range $[0,1]$. Likewise, the binPSO limited $v_{i,d}$ by the user parameter $V_{\text{max}}$; also in the binACO$_R$, $V_{\text{max}}$ is used; that is, $|x_i^t| < V_{\text{max}}$.

4. Simulation Results and Discussions

In this section, we solve several HWSN design problems with the second sensing condition from Section 2.2.2 using the above three binary optimization algorithms and compare their performances. The simulation was implemented in MATLAB on a PC running Windows 8.1, with an Intel Core i3 CPU 540 operating at 3.07 GHz and 4 GB of RAM. The parameters used for each algorithm are shown in Table 1. The termination criterion of all algorithms in all of the scenarios is reaching the maximum number of the evaluation, $\text{MaxNFEs} = 10^6$. The parameters related to the consumed energy for the transmission of a data packet were set for all simulations as follows: $k = 10^{-7}$, $c = 5 \times 10^{-5}$, $\gamma = 2$, $T = 1$, and $E_t = 3$. For each scenario in the simulation, the parameters for the environment and the desired HWSN were set as shown in Table 2. All scenarios were performed 30 times for statistics.

Table 1: Parameter setting for each algorithm in the simulation.

| Algorithm   | Parameter                   | Value |
|-------------|-----------------------------|-------|
| binPSO [15] | Number of particles ($N_p$) | 20    |
|             | Weight for previous velocity ($u_r$) | 0.729 |
|             | Coefficient of cognition term ($\varphi_p$) | 1.49445 |
|             | Coefficient of social term ($\varphi_s$) | 1.49445 |
|             | Maximum velocity ($V_{\text{max}}$) | 1     |
| binABC [16] | Number of employed bees ($N_b$) | 50    |
|             | limit or abandonment criteria ($N_c$) | 100   |
|             | Maximum velocity ($V_{\text{max}}$) | 1     |
|             | Archive size ($N_T$) | 10    |
|               | Number of ants used in an iteration ($N_s$) | 50    |
| binACOR     | Speed of convergence ($\xi$) | 0.85 |
|             | Locality of search process ($\gamma$) | 0.1   |
|             | Maximum velocity ($V_{\text{max}}$) | 1     |

Table 3 shows the statistical results of the simulations with CH and node sensing in five different environments, and Figure 2 is the boxplot of three algorithms for each scenario. In all scenarios, the binPSO algorithm shows the best performance among the three binary optimization algorithms. The binPSO obtains near-optimal solutions with similar costs in all scenarios. As a statistical significance test, the two-sample t-test was conducted between the binPSO and the others. The significance sign “(+)” in the table means that the binPSO is significantly better, the sign “(−)” means no significant difference, and the sign “(−−)” means significantly worse. The results of the t-test also showed that the binPSO is significantly better than the other algorithms. The binACOR first proposed in this paper shows the second best performance, demonstrating an improvement in
Table 2: Parameter setting for the environment and the desired HWSN in each scenario: least guaranteed coverage ($C_g$), desired network lifetime ($T_f$), sensing range of node ($R_{S_1}$) and CH ($R_{S_2}$), and cost for each node ($H_n$) and each CH ($H_{CH}$).

| Parameters          | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 |
|---------------------|------------|------------|------------|------------|------------|
| Area size $||R||$ [m²] | 1000 × 1000 (square) | 2.5 × 10⁵ (circle) | 500 × 500 (square) | 5 × 10⁵ (circle) | 600 × 600 (square) |
| $C_g$               | 0.80       | 0.85       | 0.90       | 0.80       | 0.85       |
| $T_f$ [hour]        | 6000       | 4000       | 3000       | 4000       | 5000       |
| $R_{S_1}$ [m]       | 10         | 7          | 3          | 5          | 6          |
| $R_{S_2}$ [m]       | 15         | 10         | 6          | 20         | 15         |
| $H_n$               | 0.1        | 0.5        | 0.1        | 0.3        | 0.2        |
| $H_{CH}$            | 1          | 2          | 1.5        | 5          | 1.5        |

Figure 2: Boxplot of three algorithms for each scenario.

performance over the binABC algorithm. Additionally, the binACO$_R$ finds near-optimal solutions that are the same as those found by the binPSO. However, the binACO$_R$ requires more exploration to achieve the same performance as the binPSO.

5. Conclusions

In this paper, we designed a heterogeneous wireless sensor network (HWSN) with nodes and CHs that are all able to sense information. We accomplished this by defining the constrained integer optimization problem, converting it to the unconstrained problem and solving this problem using the binary optimization algorithms based on swarm intelligence, namely, binPSO, binABC, and binACO$_R$. We then compared the performance of each algorithm by applying them to real problems. From the simulation results, the binPSO algorithm outperformed the others based on swarm intelligence, showing relatively consistent solutions every time for each test case. The binACO$_R$, which was first proposed in this paper, needed to improve the exploration ability, though its performance was better than that of the binABC algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Table 3: Simulation results with sensing of CHs and nodes in five different environments: the listed values are the statistics of the costs of five scenarios for each algorithm for the 30 runs. Q1 and Q3 are the 25th and 75th percentiles, respectively. For median and mean values, the numbers in square brackets indicate the rankings of the three algorithms. The \( p \) values are the results of the two-sample \( t \)-test between the binPSO which has the best performance among three algorithms and the others for each scenario. The significance sign "(+)" means that the binPSO is significantly better, the sign "(−)" means no significant difference, and the sign "(−)" means significantly worse.

| Scenarios | binPSO | binABC | binACO | \( p \) value |
|-----------|--------|--------|--------|-------------|
| Cost \( N \) | Cost \( N \) | Cost \( N \) | \( p \) value |
| Best | 743.9 | 5089 | 235 | — |
| Q1 | 743.9 | 5049 | 239 | — |
| Median | 744 [1] | 5080 | 236 | — |
| Q3 | 744 | 5060 | 238 | — |
| Worst | 744.3 | 5123 | 232 | — |
| Mean | 744.0 [1] | — | — | — |

\( p \) value: 4.51\( E \)-11(+) — — —

| Cost \( N \) | Cost \( N \) | Cost \( N \) | \( p \) value |
| Best | 1619 | 2974 | 66 | — |
| Q1 | 1619 | 2970 | 67 | — |
| Median | 1619 [1] | 2966 | 68 | — |
| Q3 | 1619 | 2958 | 70 | — |
| Worst | 1619.5 | 2951 | 72 | — |
| Mean | 1619 [1] | — | — | — |

\( p \) value: 6.83\( E \)-11(+) — — —

| Cost \( N \) | Cost \( N \) | Cost \( N \) | \( p \) value |
| Best | 3130.5 | 16380 | 995 | — |
| Q1 | 3130.5 | 16380 | 995 | — |
| Median | 3130.5 [1] | 16380 | 995 | — |
| Q3 | 3130.8 | 16383 | 995 | — |
| Worst | 3133 | 16375 | 997 | — |
| Mean | 3130.9 [1] | — | — | — |

\( p \) value: 3.00\( E \)-11(+) — — —

| Cost \( N \) | Cost \( N \) | Cost \( N \) | \( p \) value |
| Best | 3111.5 | 7605 | 166 | — |
| Q1 | 3111.6 | 7672 | 162 | — |
| Median | 3111.6 [1] | 7522 | 171 | — |
| Q3 | 3111.7 | 7489 | 173 | — |
| Worst | 3112.1 | 7757 | 157 | — |
| Mean | 3111.7 [1] | — | — | — |

\( p \) value: 1.24\( E \)-08(+) — — —

| Cost \( N \) | Cost \( N \) | Cost \( N \) | \( p \) value |
| Best | 1246.8 | 5229 | 134 | — |
| Q1 | 1246.8 | 5229 | 134 | — |
| Median | 1246.8 [1] | 5199 | 138 | — |
| Q3 | 1247 | 5260 | 130 | — |
| Worst | 1247.3 | 5284 | 127 | — |
| Mean | 1246.9 [1] | — | — | — |

\( p \) value: 1.70\( E \)-13(+) — — —
Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning (Grant no. 2010-0023035).

References

[1] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 660–670, 2002.

[2] J.-W. Lee, B.-S. Choi, and J.-J. Lee, “Energy-efficient coverage of wireless sensor networks using ant colony optimization with three types of pheromones,” IEEE Transactions on Industrial Informatics, vol. 7, no. 3, pp. 419–427, 2011.

[3] J.-W. Lee and J.-J. Lee, “Ant-colony-based scheduling algorithm for energy-efficient coverage of WSN,” IEEE Sensors Journal, vol. 12, no. 10, pp. 3036–3046, 2012.

[4] M. Noori and M. Ardakani, “Design of heterogeneous sensor networks with lifetime and coverage considerations,” IEEE Wireless Communications Letters, vol. 1, no. 3, pp. 193–196, 2012.

[5] M. Saleem, I. Ullah, and M. Farooq, “BeeSensor: an energy-efficient and scalable routing protocol for wireless sensor networks,” Information Sciences, vol. 200, pp. 38–56, 2012.

[6] J.-W. Lee, J.-Y. Lee, and J.-J. Lee, “Jenga-inspired optimization algorithm for energy-efficient coverage of unstructured WSNs,” IEEE Wireless Communications Letters, vol. 2, no. 1, pp. 34–37, 2013.

[7] J.-W. Lee and J.-J. Lee, “Design of efficient Heterogeneous Wireless Sensor Networks optimized by discrete Particle Swarm Optimization,” in Proceedings of the 44th IEEE International Symposium on Robotics (ISR ’13), pp. 1–5, Seoul, Republic of Korea, October 2013.

[8] R. R. Sahoo, M. Singh, B. M. Sahoo, K. Majumder, S. Ray, and S. K. Sarkar, “A light weight trust based secure and energy efficient clustering in wireless sensor network: honey bee mating intelligence approach,” Procedia Technology, vol. 10, pp. 515–523, 2013.

[9] P. Kanda, S. K. Gupta, and P. K. Jana, “A novel evolutionary approach for load balanced clustering problem for wireless sensor networks,” Swarm and Evolutionary Computation, vol. 12, pp. 48–56, 2013.

[10] S. Halder and S. D. Bit, “Enhancement of wireless sensor network lifetime by deploying heterogeneous nodes,” Journal of Network and Computer Applications, vol. 38, no. 1, pp. 106–124, 2014.

[11] X. Liu and D. He, “Ant colony optimization with greedy migration mechanism for node deployment in wireless sensor networks,” Journal of Network and Computer Applications, vol. 39, no. 1, pp. 310–318, 2014.

[12] P. Kanda and P. K. Jana, “Energy efficient clustering and routing algorithms for wireless sensor networks: particle swarm optimization approach,” Engineering Applications of Artificial Intelligence, vol. 33, pp. 127–140, 2014.

[13] M. Tao, S. Huang, Y. Li, M. Yan, and Y. Zhou, “SA-PSO based optimizing reader deployment in large-scale RFID Systems,” Journal of Network and Computer Applications, vol. 52, pp. 90–100, 2015.

[14] Y.-H. Zhu, W.-D. Wu, J. Pan, and Y.-P. Tang, “An energy-efficient data gathering algorithm to prolong lifetime of wireless sensor networks,” Computer Communications, vol. 33, no. 5, pp. 639–647, 2010.
