Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 5720659, 13 pages
https://doi.org/10.1155/2017/5720659

Research Article

An Energy Balancing Strategy Based on Hilbert Curve and Genetic Algorithm for Wireless Sensor Networks

Lingping Kong,[1] Jeng-Shyang Pan,[1,2] Tien-Wen Sung,[2] Pei-Wei Tsai,[3] and Václav Snášel[4]

[1] *Innovative Information Industry Research Center, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China*
[2] *Fujian Provincial Key Lab of Big Data Mining and Applications, Fujian University of Technology, Fuzhou, China*
[3] *Swinburne University of Technology, Melbourne, VIC, Australia*
[4] *Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava, Czech Republic*

Correspondence should be addressed to Jeng-Shyang Pan; jspan@cc.kuas.edu.tw

Received 20 January 2017; Accepted 24 April 2017; Published 9 July 2017

Academic Editor: Gianluca De Marco

A wireless sensor network is a sensing system composed of a few or thousands of sensor nodes. These nodes, however, are powered by internal batteries, which cannot be recharged or replaced, and have a limited lifespan. Traditional two-tier networks with one sink node are thus vulnerable to communication gaps caused by nodes dying when their battery power is depleted. In such cases, some nodes are disconnected with the sink node because intermediary nodes on the transmission path are dead. Energy load balancing is a technique for extending the lifespan of node batteries, thus preventing communication gaps and extending the network lifespan. However, while energy conservation is important, strategies that make the best use of available energy are also important. To decrease transmission energy cost and prolong network lifespan, a three-tier wireless sensor network is proposed, in which the first level is the sink node and the third-level nodes communicate with the sink node via the service sites on the second level. Moreover, this study aims to minimize the number of service sites to decrease the construction cost. Statistical evaluation criteria are used as benchmarks to compare traditional methods and the proposed method in the simulations.

1. Introduction

Wireless sensor networks (WSNs) are spatially distributed autonomous sensors used to monitor physical or environmental conditions, such as pressure, sound, and temperature. WSNs are composed of common sensor nodes and sink nodes [1, 2]; the common sensor nodes cooperatively pass their data through the network to a sink node. The development of wireless sensor networks was originally motivated by military applications such as remote sensing or data collection in dangerous or remote environments [3]. Today, these networks are used in many industrial and consumer applications and have become part of daily life. WSNs are built of a few to several hundreds or even thousands of nodes, where each node can connect with one or more sensors. Each sensor node is equipped with several parts, namely, a transceiver, a sensing device, and an energy source. These sensor nodes differ in size and cost, which results in corresponding constraints on resources such as energy, memory, and computational speed [4–7]. Their energy source is usually a battery, which is undesirable and infeasible to replace or recharge [8–10]. Therefore, network lifespan becomes a vital concern in the construction of a WSN [11]. However, unbalanced energy consumption between inner nodes (the nodes close to the sink node) and outer nodes (the node far away from sink node) always occurs and is uncontrolled in two-tier network structures. Sink nodes, the only nodes that control and operate as processing centers, collect all the valuable packages from the sensor nodes via a predefined routing path. The inner nodes not only transfer their own sensed data, but also pass on data from outer nodes. Thus, inner nodes have greater energy consumption than that of outer nodes. The more energy one node uses, the earlier it depletes its battery. The worst case scenario resulting from this is if the depleted node is the only communication line between outer nodes and the sink node. In this network structure, if even a few inner nodes die, many outer nodes will be affected. In this situation, several service sites which have part of the functions of a

sink node become necessary, and the sensor nodes then send their data to the nearest service site instead of the sink node. This also decreases the workload on inner nodes and extends the lifespan of the overall network. This paper focuses on developing a method to determine the optimal number of service sites for a given network. The cost of deployment and construction of a service site is much greater than that of a common sensor; thus, there should be a minimum necessary number of service sites in the network to satisfy full coverage demand.

Given m nodes with specified distances, k centers must be constructed for groups of nodes in such a way as to minimize the maximum distance between nodes and their centers. This is the k-center problem. The goal of this paper is to minimize the number of service sites in a wireless sensor network, thus reducing the construction cost of a three-tier network caused by service sites. More importantly, this three-tier network must satisfy the full coverage requirement. The number of service sites is considered k in the k-center problem. However, k is not yet known. One of the most popular methods for resolving the k-center problem is the farthest first method [12]; although this method satisfies a 2-approximation solution, it is not perfect. This paper proposes a new scheme, HHSG, to solve the service site problem. The name of "HHSG" was given by an integrated abbreviation of "Huffman coding," "Hilbert curve," "Sudoku puzzle," and "genetic algorithm" because the concepts of these four classical terms were utilized in our proposed scheme. Furthermore, several other methods are simulated and applied to wireless sensor networks.

The remainder of this paper is structured as follows: Section 2 reviews background work on Hilbert curves, the k-center problem, and wireless sensor networks and will also describe related work on basic genetic algorithms and Sudoku and Huffman codes. Section 3 describes the HHSG process in detail. Experimental results and some analysis with other methods are given in Section 4. Conclusion is offered in Section 5.

2. Related Works

Wireless sensor networks have been widely used in vast variety of different fields. Driven by microelectromechanical systems technology advances in low-cost networking, there have been rapid development and use of wireless sensor networks in recent years [13, 14]. These sensor networks carry the promise of significantly improving and expanding the quality of care across a wide range of applications, which include air pollution monitoring, medicine and public health, and natural disaster prevention. Although a general two-tier network is considered to be a flat network and has a very simple structure, it has an inherent disadvantage in terms of balancing the workload of its sensor nodes. When inner nodes deplete their batteries, they die and disconnect from their outer nodes, interrupting the routing path from the outer nodes to a sink node. As a result, many nodes that still have sufficient energy to function will be removed from the network, and their information will no longer be forwarded to a sink node. Alternatively, a hierarchical network is a network in which all sensor nodes are clustered through some specific technique according to given protocols [15, 16]. Hierarchical networks facilitate equalized power consumption.

2.1. Genetic Algorithms. Genetic algorithms are a family of computational models inspired by natural evolution [17–20]. In a genetic algorithm, a population of candidate solutions to a problem is evolved toward better solutions. Each candidate solution, which is expressed in binary string of 0 or 1, is a chromosome with a set of attributes which can be mutated and modified. The basic genetic algorithm usually starts by generating several random chromosome solutions, then evaluating each chromosome, and storing the ones with better fitness values as the algorithm approaches an optimal solution by randomly mutating and altering the predefined number of genes to generate a new solution. This new solution will be used in the next iteration. Commonly, the algorithm stops when it reaches a predefined number of iterations or time limit or when there is one solution that is satisfied. Genetic algorithm is widely used in many applications and is also combined with other methods to generate new optimal solutions [21, 22].

2.2. Space-Filling Curve. A space-filling curve is a single one-dimensional curve that tours around an entire 2 or more dimensional space and recursively fills up all points when the number of iterations approaches infinity [23, 24]. Because Giuseppe Peano (1858–1932) was the first to discover one of the filling curve constructions, space-filling curves in 2-dimensional planes are sometimes called *Peano curves*. Some of the most celebrated are the Hilbert curve and the *Sierpiński curve* [23]. Space-filling curves are used in many fields. In 2014, Yan and Mostofi [24] scheduled a data collection path for mobile robots using space-filling curves; his goal was to minimize the total energy consumption, including the communication cost between the robot and sensors and the motion cost of the robot. In this study [25], the problem of how mobile sinks should move is addressed. A good strategy for a moving trajectory for mobile sinks can reduce data loss and delivery delay, increase network lifetime, and enable better handling of sparse networks. A dynamic Hilbert curve is used to design a trajectory for a mobile sink while achieving efficient network coverage. The dynamic curve order varies with node densities in a network. Simulation results show the effectiveness of network coverage and scalability.

For Hilbert curves, if there is a point within the unit square, with coordinates (x, y), m is the distance along the curve from the start till it reaches that point. Points from the curve that have nearby m' values will also have nearby coordinate (x', y') values. The basic level one (also called first order) Hilbert trajectory is a 2×2 grid. The method of recursively constructing a Hilbert filling curve is described as follows: dividing the network field into 4 small grid cells, the one-level Hilbert curve will be the line passing through the centers of those four-grid cells in a specific order of points. To derive a two-level curve, it simply replaces each small grid cell with a one-level curve which may be appropriately rotated and reflected. And M-level curve is derived from

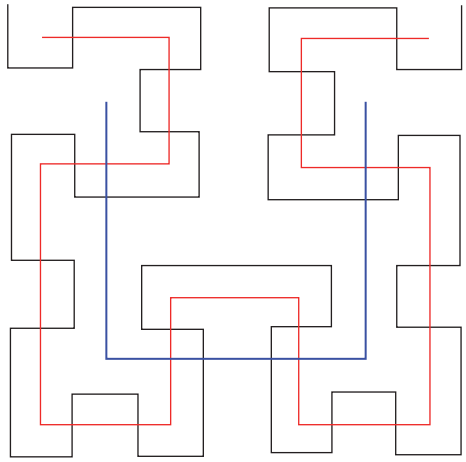


Figure 1: First- to third-order Hilbert curves.

an $M-1$-level curve. Intuitively, the higher the level of the curve is, the more accurate its localization precision will be. However, this means that more space is needed for recording the positions, at greater cost. Figure 1 shows one-, two-, and three-order Hilbert curves. There are 4 points in a one-level Hilbert, 16 points in a two-level Hilbert, and 64 points in a three-level Hilbert. $f_n = 4^l$ is the equation used to compute the relationship between the level of Hilbert curves and the number of points, where l is Hilbert level and f_n is number of points.

2.3. Sudoku. Sudoku is a logic-based, number-placement puzzle which consists of $M \times M$ grid of blocks, where M smaller cells of each M element are partitioned. The numeric values 1 to M appear uniquely in each row and column of the grid and in each block [26, 27]. Given a 9×9 grid, the goal is to fill this grid with digits from one to nine only. The rule is that each row and each column, even the nine 3×3 subgrids which compose the big grid, should contain all of the digits of one to nine. Although the 9×9 grid is by far the most commonly used, many other variations exist. Number placement could be 4×4 with 2×2 regions or 16×16 with 4×4 regions.

2.4. The k-Center Problem. One of the well-known fundamental facility location strategies [28] is the solution of k-center problem, and this problem is known to be NP-hard [29]. The basic k-center problem starts from a given graph with n vertexes, where it is required to put k facilities into the graph, so as to narrow down the maximum distance from any vertex to the facility to which it is assigned. Several optimal algorithms that can achieve a factor of 2-approximation performance have been proposed for it. An algorithm could be called ε-approximation algorithm which means that the algorithm can always output a value in polynomial times, where the value is no more than ε times the optimal for a minimization problem. With the widely used k-center problem, some variant versions of it have also been massively explored. For example, some special constraints on the centers positions were added to the problem. In 2015, Du et al. [30] explored the incremental one that all the centers should lie on the boundary of a convex polygon. In the same year, Liang et al. [31] addressed the constraint of vertexes with internal connectedness, where it is guaranteed that any two nodes in one set should be lined by an internal path. This is actually a classic k-center problem, which is called connected k-center (CkC) problem. In [32], the authors presented a solution for the k-center problem and did some research about its generalizations. He also noted that dominating set problem is another specific form of k-center problem. The authors Chechik and Peleg [33] studied the other constrained version of capacitated k-center problem and examined the fault tolerance in failures of one or more centers simultaneously and then proposed methods to address the problem.

3. HHSG Scheme Implementation

The flowchart of the HHSG scheme is shown in Figure 2. The process of this scheme focuses on selecting k service sites out of n sensor nodes. Every sensor node will be assigned two-kinds of serial number. One is the node numbering (NID), which is nonrepeatable. The NID ranges from 1 to n. The other number is a Huffman code (h_c). It is reasonable that there can be more than one node with the same Huffman code.

The process runs as follows. First, encode sensor nodes using Huffman code and define the level of Hilbert filling curve used. Second, pick the appropriate size and order of Sudoku according to the communication radius and network scale, randomly select a digit from the Sudoku grid, and record and encode the positions (t_d) of the digit into h_c. Third, mark p_s that have Huffman codes that are the same as or similar to t_d, and initialize a chromosome using p_s. Fourth, repeat steps 2 to 3 until chromosome initializing is complete. Fifth, find the best solution by executing the mutation or crossover operation to output the outcome.

3.1. Encoding and Defining Curve Order. Huffman code uses a prefix-free code that is a bit string representing some particular points but is never a prefix of any other points. As shown in Figure 3, each position with specific distance d or multiple times d distances starts from the red point (Figure 3(b)), signifying a Huffman code, and this code expresses one or more real sensor nodes in wireless sensor network. By utilizing its locality property, it needs a six-bit string to represent a three-level Hilbert curve. The encoding process is given below.

Divide this field into four small grid cells, and set a two-bit binary number to it; its order is from upper left to lower left and then lower right to upper right, with values of 00, 01, 10, and 11, respectively. This coding order also strictly applies to the inner subgrid cell. As showed in Figure 3(a), binary 10 is in red on the lower right, and all the sensor nodes located in this quarter of the area will be prefixed with a two-bit code 10. Then, this quarter area is also divided into four small grid cells, and the binary number order is exactly the same as that of the bigger area. Binary 10 is in blue and is 1/16th of the total area. Any node in this area will have 10 in the second part of its Huffman code. As shown in Figure 3(b), the red point starts from 0 in the Huffman code to the blue point

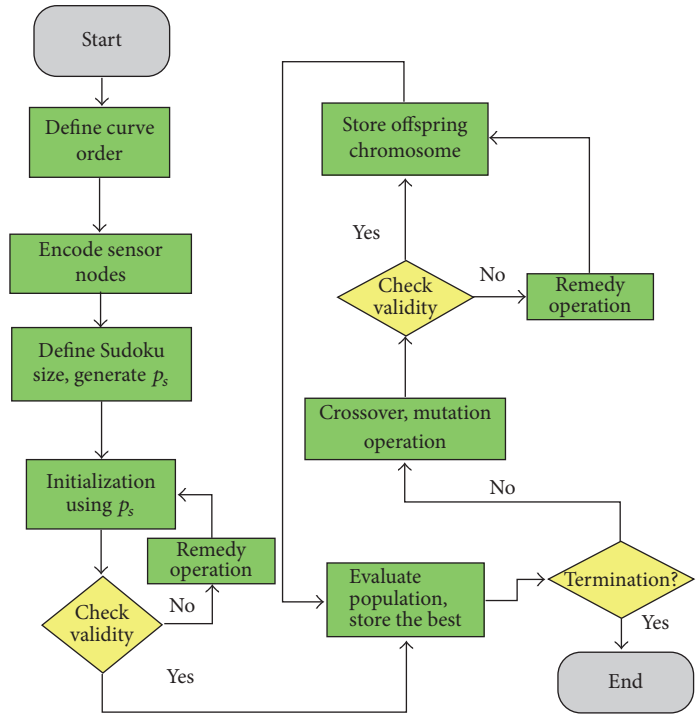


Figure 2: Flowchart of HHSG scheme.

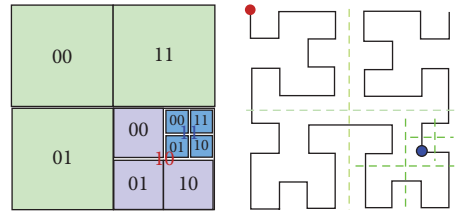


Figure 3: Huffman coding model in a 3-order Hilbert curve.

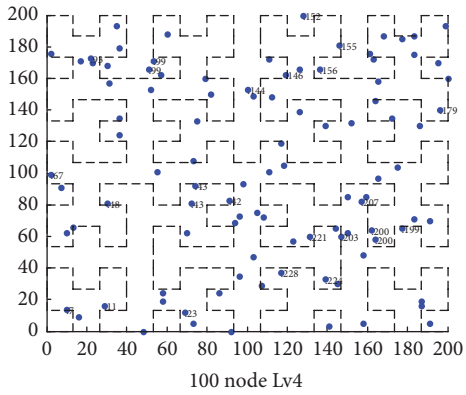
47. The binary string for 0 of the red starting point is "00 00 00." The encoding process for the blue point is described here in detail. The first 2-bit binary string is 10 as it appears in the lower-right quadrant. Then the second 2-bit binary string relies on the upper-right 1/16th area, which is 11. The point located in lower-left 1/64th area results in a 01 suffix. Assemble the binary string 10 11 01, which is 47 in decimal numerals. Finally, the Hilbert curve gives every node an h_c [34–36].

h_c of one node varies according to the order of the Hilbert curve. The largest code number is 63 for a three-order curve and 1023 for a five-order curve. Assume a 100-node network with a five-order Hilbert curve. Almost less than ten percent of codes are truly used in sensors, which is a great waste. Similarly, for a 600-node network in a three-order curve, more than ten sensor nodes have the same h_c. Figure 4 shows the configurations of nodes with h_c in varied order of Hilbert curve.

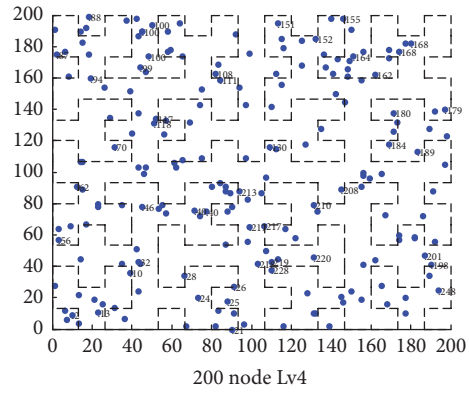
100 to 400 nodes randomly scattered in a 200 × 200 unit area, and nodes are coded in Huffman code with four- or five-order Hilbert curves.

A filling curve has a locality property which means that any two close points in one-dimensional space are mapped to two points that are close in the original 2 (or more) dimensional space, but the converse cannot always be true. There are points where the coordinates are close but their d values are far apart, which means that two close nodes may not close in the curve. Thus, if one selects the nodes from this curve directly to initialize chromosomes, there may be neighbor nodes in one chromosome. This is why the Sudoku is needed.

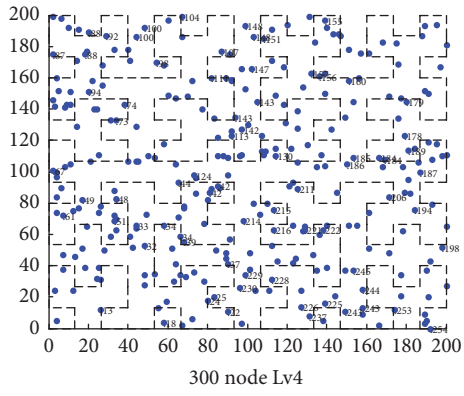
3.2. Sudoku Size and Order. This section will demonstrate how size and order of Sudoku are chosen. The size of Sudoku expresses the M value of grids in one row or one column. The order of Sudoku gives the level value of a block constructed by several Sudoku of the same size. A single- or multiple-order Sudoku that can sketch the network appropriately is desired, whereby each grid may cover the right amount of sensor nodes. It is not appropriate to use a single-order 9 × 9 grid Sudoku in a 600-node network with a ten-unit sensing radius



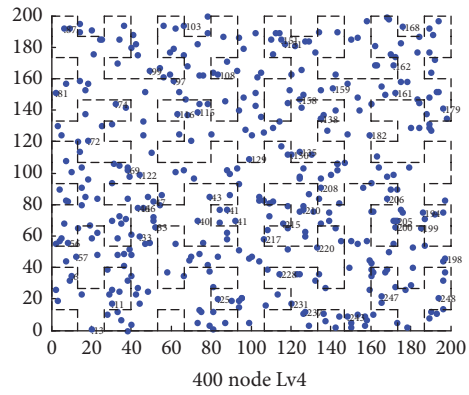
(a) 100 nodes in 4-order Hilbert curve



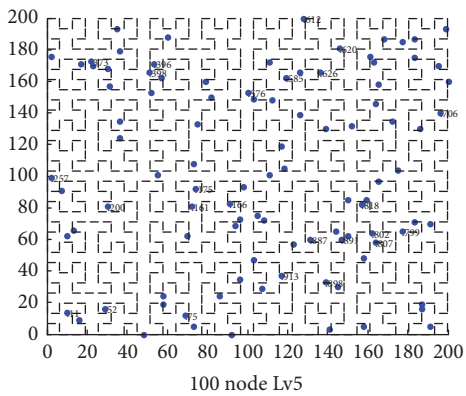
(b) 200 nodes in 4-order Hilbert curve



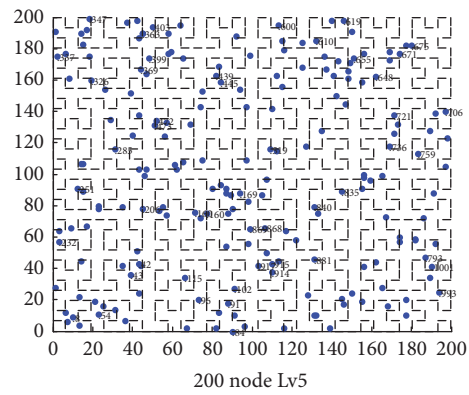
(c) 300 nodes in 4-order Hilbert curve



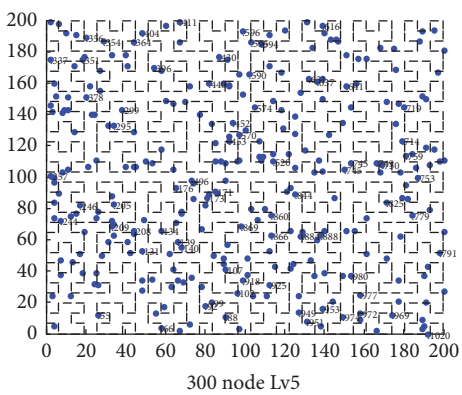
(d) 400 nodes in 4-order Hilbert curve



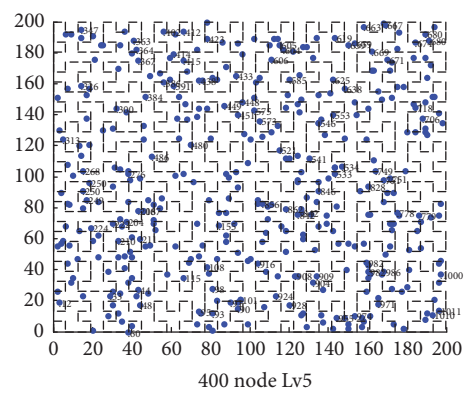
(e) 100 nodes in 5-order Hilbert curve



(f) 200 nodes in 5-order Hilbert curve



(g) 300 nodes in 5-order Hilbert curve



(h) 400 nodes in 5-order Hilbert curve

Figure 4: Sensor node deployment in Hilbert curve.

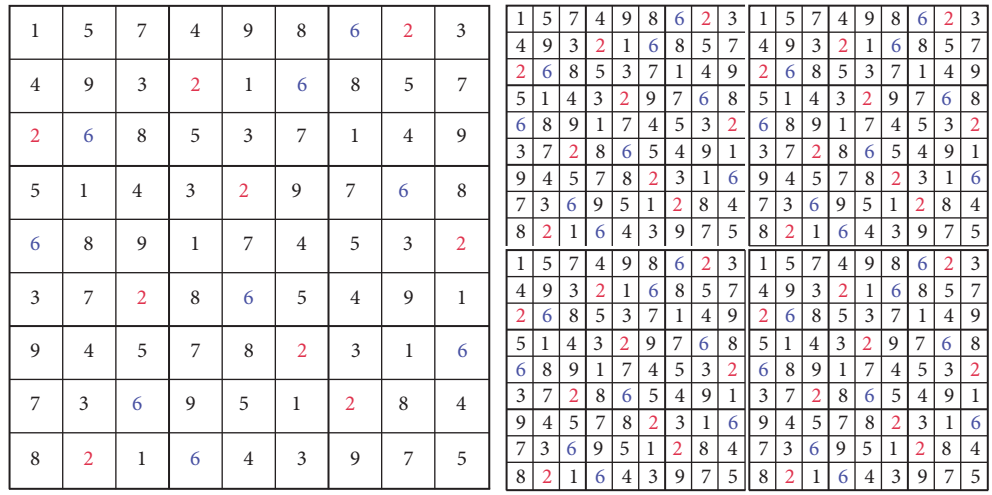


Figure 5: Grids of a 9 × 9 Sudoku and a 2-order 9 × 9 Sudoku.

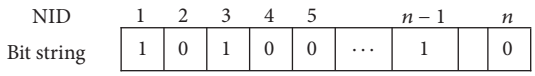


Figure 6: An individual chromosome.

of 200 × 200 unit network. If the sensor nodes are deployed randomly and uniquely, the practical number of service sites used will be more than 100. However, only nine positions can be chosen at one time to generate the service site candidates. So before choosing the size and order of Sudoku, the number of service sites must be calculated. A one-order and two-order 9 × 9 grid Sudoku resolution are shown in Figure 5.

Each digit randomly chosen from the Sudoku is labeled as a target digit (t_d). Those nodes near the t_d position are potential sites (p_s). One t_d in a 9 × 9 grid Sudoku generates a solution set with nine p_s. In the experiment, more than one t_d are usually used, and a two-order 9 × 9 grid Sudoku generates $36p_s$. In the same way, one t_d generates $64p_s$ in a two-order 16 × 16 grid Sudoku. p_s will be used in a genetic algorithm initialization, but not all the chromosomes are generated from p_s directly.

3.3. Initialization and Evaluation. An n-bit binary string with binary values 0 and 1 represents the structure of a chromosome. The order corresponds exactly to the NID order of sensor nodes, and n is the number of sensor nodes in network. This n-dimension string exactly expresses the relationship between service sites and common sensor nodes. In this string, value 1 represents a service site, and value 0 represents common nodes. As shown in Figure 6, NID ranges from 1 to n, and some of those value 1 bits come from p_s. There also are some extra value 1 bits randomly added. Each chromosome obtains a fitness value based on its fitness function. The best chromosome with the best fitness value is stored as C_{best}.

3.4. Fitness Function. A well-constructed fitness function may substantially increase the chance of finding a solution. This section presents a new fitness function which includes four parameters. k is the number of service sites in the field, which is also the amount of 1 values in one chromosome. The field is divided into [width/radius × length/radius] cells, and the number of cells in which those service sites are located is the n_c value; here width and length are the size of the field, and radius is the communication range of the sensor nodes. The fitness function is shown as (1). The function $d(x_i, s_{\text{sink}})$ indicates the distance between the node x_i and the sink node. The function $\rho(x_i, C)$ means the distance between the node x_i and its closest service site. S and C are the sets of sensor nodes (size is n) and service sites (size is n_c), respectively. Coefficients α_1, α_2, and α_3 are constants.

$$f_{\text{fit}} = \alpha_1 \cdot (n-k) + \alpha_2 \cdot \frac{n_c}{k} + \alpha_3 \cdot \left(\frac{\sum_{1 \le i \le n} d(x_i, s_{\text{sink}})}{\sum_{1 \le i \le |S-C|} \rho(x_i, C)} \right). \tag{1}$$

3.5. Crossover and Mutation Process. Let $C_1 = C_{1,1}, \ldots, C_{1,n}$ (mother) and $C_2 = C_{2,1}, \ldots, C_{2,n}$ (father) be the parents; after crossover operation, the child C_3 is

$$C_3 := C_{1,1}, C_{1,2}, \ldots, C_{1,\lceil n/2 \rceil}, C_{2,\lceil n/2 \rceil + 1}, C_{2,\lceil n/2 \rceil + 2}, \ldots, C_{2,n}. \tag{2}$$

The mutation process works by inverting a bit value in the chromosome with a small probability. Here, the mutation rate is set as constant 0.02. The crossover and mutation processes are shown in Figures 7 and 8, respectively.

3.6. Remedy Process. As mentioned above, each chromosome represents a candidate solution for service sites versus common nodes in this model. This model should satisfy validity and feasibility demands. In other words, those 1-bit values representing service sites should be able to cover all 0-bit values representing common nodes. If not, this individual must be repaired. The following method is used in this

Table 1: The values for the parameters.

Number of nodes	Parameter				
	Hilbert order	Sudoku grid number	Sudoku order	Transmit radius (number of units)	Initial number
100	4	9	2	40	8
200	4	16	1	30	8
300	4	16	1	25	10
400	4	16	1	20	8

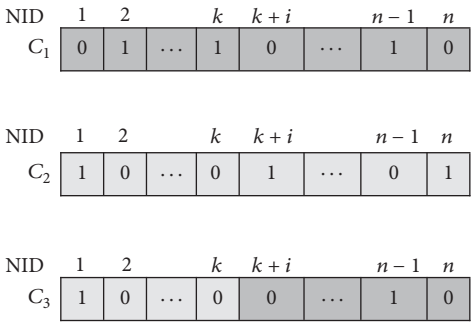


Figure 7: Crossover process.

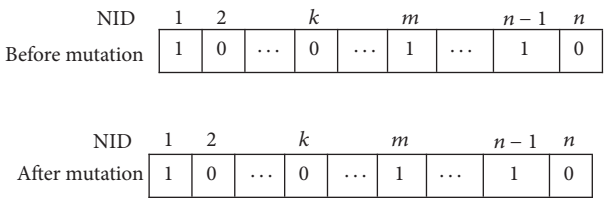


Figure 8: Mutation process.

experiment to revise incorrect chromosomes. First, generate the chromosome again if it happens in the initialization step. Second, change those uncovered bits with value 1. Third, list those uncovered value 0 bits, and change one of them to 1 each time, and remove all other bits dominated by it in the list. Repeat the process until the list is empty.

4. Experiment Results

The simulation environment is a 200 × 200 *(unit)* area, with 100, 200, 300, and 400 sensor nodes scattered randomly with a communication radius of 40, 30, 25, and 20 units, respectively, in the network. Six methods are implemented, including the FF, HL, HD, DO, GA, and HHSG scheme, where FF is the farthest first traversal, HL and HD are the Harel and Koren [37] methods, DO is a heuristic algorithm solving the minimum dominating set problem [38], GA is the original genetic algorithm itself, and HHSG is the proposed scheme.

4.1. Parameters. The order and size of *Hilbert* and *Sudoku* values play an important role in generating p_s. Therefore, different parameters used in the experiment produce diverse results. Table 1 shows the final values for the parameters used in this experiment.

Table 2: DCT values (number of times for distance computation) with different methods.

Methods	Number of nodes			
	100	200	300	400
FF	1499100	16709000	87236700	288422400
HL	129500	969800	3119100	8184000
HD	6531400	63311200	243945000	626614400

DCT records the number of times for distance computation used in the methods' operation process. The distance could be node to node or node to service site. Table 2 lists the DCT values for FF, HL, and HD, three nonevolutionary algorithms. The values are computed by simple equation, and the practical values may be smaller due to some pruning strategies used in the methods. As shown in Table 2, HL has the lowest DCT value. Here, the times of HL, FF, and HD are set as three baselines labeled L_{HL}, L_{FF}, and L_{HD}, to be used later.

4.2. Influence of the Hilbert and Sudoku. In the experiment, HHSG was executed on a 200-node network with different *Hilbert* curve and Sudoku parameters. In Table 3, the consecutive numbers show the Hilbert curve order, Sudoku size, and Sudoku order parameters used in the experiment. For example, 5-16-1 represents that the test operates on a 5-order Hilbert curve with a one-order 16 × 16 grid Sudoku. Column one lists the DCT level. HHSG may produce different results with different parameters. With a five-order *Hilbert* curve, the results are clearly better than the other two cases.

4.3. Comparison of GA and HHSG. GA and HHSG belong to a larger class of evolutionary algorithms. They may generate high quality solutions by operating endless iterations for optimization. For further comparison of the rate of evolution, the following tests were made. In Table 4, GA and HHSG were run in 100-node to 400-node networks with the same number of iterations. In the 100-node network, HHSG only used 13 service sites to cover the field, two less than GA, and its superiority is obvious in a 400-node network.

In order to test the stability of the HHSG and GA methods, the two methods were run sixteen times in the same situation, with the exception of the random number used.

Table 3: The effects of Hilbert order, Sudoku size, and Sudoku order to HHSG.

HHSG	4-9-2		4-16-1		5-16-1	
	Value of fitness function	Number of service sites	Value of fitness function	Number of service sites	Value of fitness function	Number of service sites
L_{HL}	174.805	32	177.240	29	178.551	28
L_{FF}	177.562	29	178.205	28	179.327	27
L_{HD}	—	—	—	—	—	—
Final result	181.322	25	180.904	25	180.257	26

Table 4: Comparison of DCT and number of service sites.

Number of nodes	Method			
	GA		HHSG	
	DCT (number of times)	Number of service sites	DCT (number of times)	Number of service sites
100	17508996	15	13706692	13
200	95466454	33	56255739	25
300	253569455	55	138944694	36
400	543754047	91	308883333	54

Table 5: Stability of HHSG.

Number of nodes	Method	Number of service sites			
		Best	AG	Worst	SD
100	GA	15	15.9	17	0.7378
	HHSG	13	13.9	14	0.3162
200	GA	33	36.3	39	1.9465
	HHSG	25	25.3	25	0.4830
300	GA	55	67.0	73	5.8878
	HHSG	36	36.1	26	0.3162
400	GA	91	99.5	104	5.7397
	HHSG	54	54.4	55	0.5164

Table 6: The final results (value of fitness function) in six methods.

Number of nodes	Value of fitness function					
	FF	HL	HD	DO	GA	HHSG
100	86.61	90.02	87.95	87.55	90.15	91.75
200	175.88	179.55	176.05	175.78	174.05	180.90
300	260.90	269.41	262.14	263.97	253.18	271.25
400	345.81	353.10	341.78	345.11	320.24	355.16

This study lists the standard deviation (SD), best value (best), average (AG), and the worst value (worst) for comparison in Table 5. The standard deviation values stay below one for the HHSG method, where the GA method reaches seven in a 400-node network, which fully illustrates the stability of the HHSG scheme. From the table, it can be seen that the worst result of HHSG is still better than the GA method in 200-node, 300-node, and 400-node networks.

4.4. Overall Evaluation. Tables 6 and 7 list results for the number of service sites and fitness values obtained by the six different methods. In the experiment, the larger the fitness value the better, and the lower the number of service sites the better. Although the number of service sites result by HL is equal to HHSG, the HL fitness value in a 400-node network is lower. The HHSG scheme is better than HL in other networks sizes overall. As for the other methods, they yield lower results overall than those of HHSG in both number of service sites and fitness values. In Figure 9, HHSG1 plots the fitness value evolution process for a 100-node network using the HHSG scheme, HHSG represents the same for a 200-node network, and the processes for the other schemes are labeled accordingly. The superiority of HHSG is clear.

Figures 10–13 show the simulated network after service sites are added. Sensor nodes are randomly scattered in the

Table 7: The final results (number of times for distance computation) in six methods.

Number of nodes	Number of times for distance computation					
	FF	HL	HD	DO	GA	HHSG
100	19	15	17	18	15	13
200	31	27	31	31	33	25
300	47	38	46	44	55	36
400	64	56	68	65	91	54

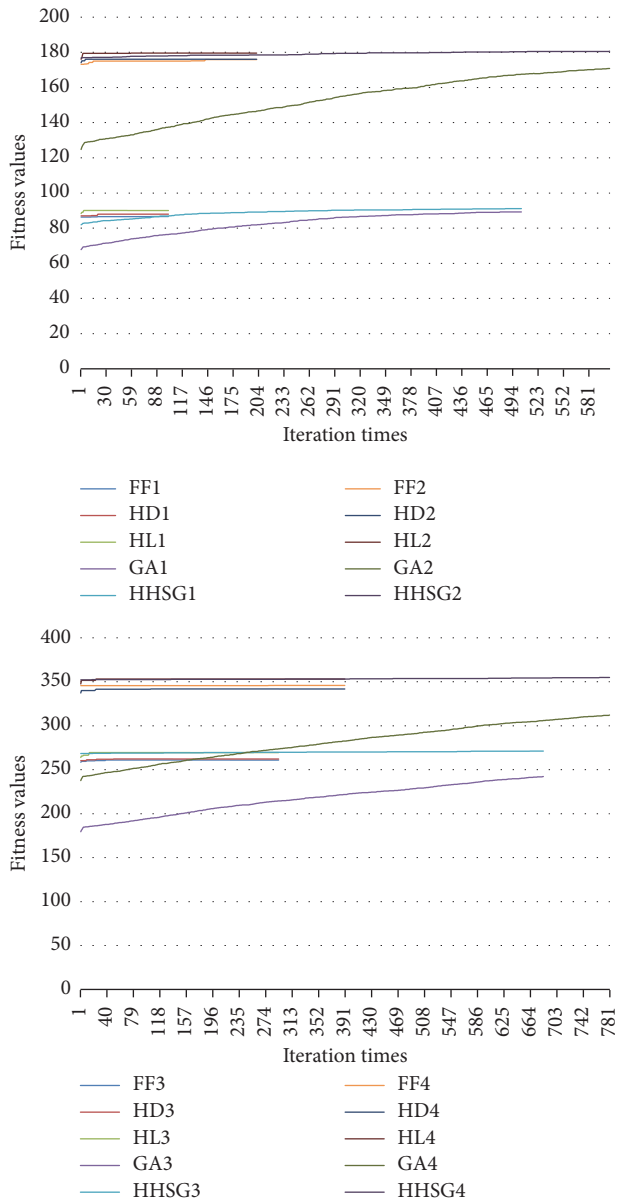


Figure 9: Fitness evolving curves for 100–400 nodes.

field, with the sink node deployed in the center of area. The sink node is shown in the 100-node and 200-node networks, but it is too complex to plot all the lines between service sites to sink nodes in the 300-node and 400-node networks.

5. Conclusion

Energy load balancing is critical to extending the lifespan of wireless sensor networks, in addition to ensuring continued functionality and avoiding communication interruptions caused by dead nodes. Wireless sensor networks usually consist of hundreds or even thousands of sensor nodes scattered randomly in adverse, remote, or dangerous environments, with only nonrechargeable, nonreplaceable batteries to power each node. Thus, energy conservation for individual nodes is important, but equally important is the energy efficiency of the overall network. Traditional two-tier networks with one sink are vulnerable to energy holes, which cut off many nodes from the sink when one inner node dies.

This paper therefore proposes a solution, HHSG, to minimize the construction cost of a three-tier network and take full advantage of node energy. A Hilbert curve is scheduled for different sized networks, Huffman codes are assigned to nodes, and chromosomes for a genetic algorithm are initialized using a Sudoku puzzle. Furthermore, five other methods are tested in the experiment for performance comparison with the proposed method. The experiment lists the relationship between Hilbert order and sensor nodes' Huffman codes and the convergence of results due to the varied Hilbert and Sudoku order. It also compares the service site performance of the other five methods with that of the HHSG algorithm. Importantly, this paper lists the standard deviation (SD), best value (best), average (AG), and the worst value (worst) of each method in order to compare the stability and benefits of each method. The standard deviation values stay below one for the HHSG method, which fully illustrates the stability of the algorithm. Simulation results show the superior performance of the proposed method, which builds a stable three-tier network using fewer service sites than other methods. In terms of the costs of the proposed scheme, because HHSG is a centralized algorithm, the cost could be a communication overhead for collecting global information before executing the algorithm; however, this is also the common cost in all of the centralized algorithms to solve the problem. Other possible costs could be the computation time and memory space. However, centralized algorithms are usually executed in a resource-rich machine, and computing power and memory space are not the most important considerations to solve the problem by a centralized algorithm.

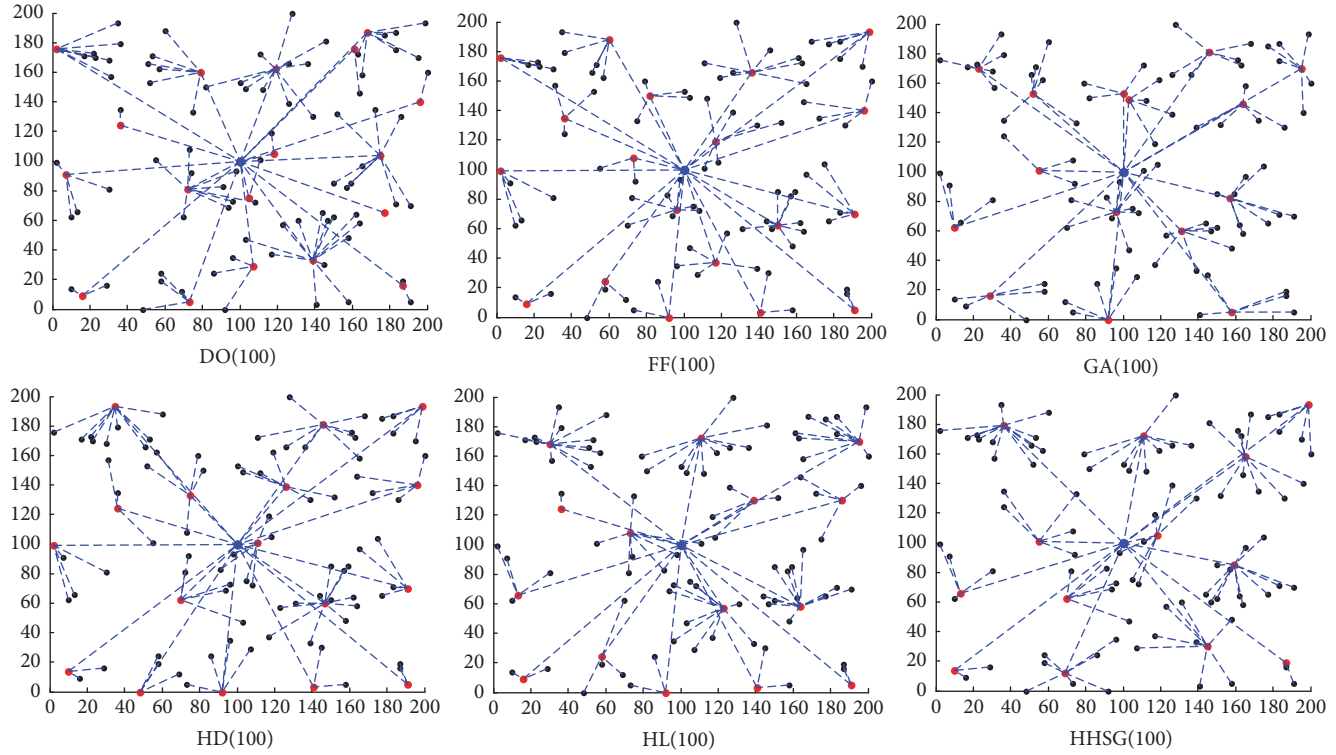


Figure 10: 100-node three-tier network in six methods.

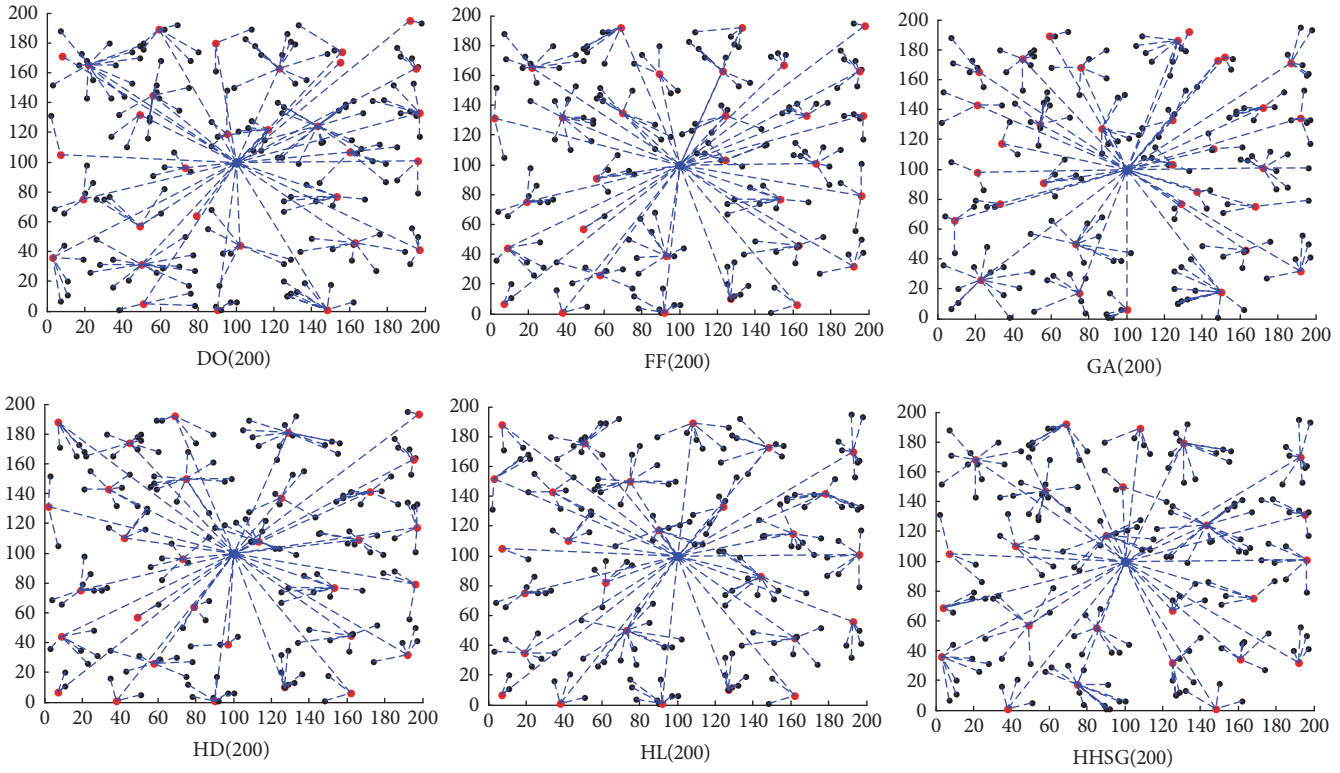


Figure 11: 200-node three-tier network in six methods.

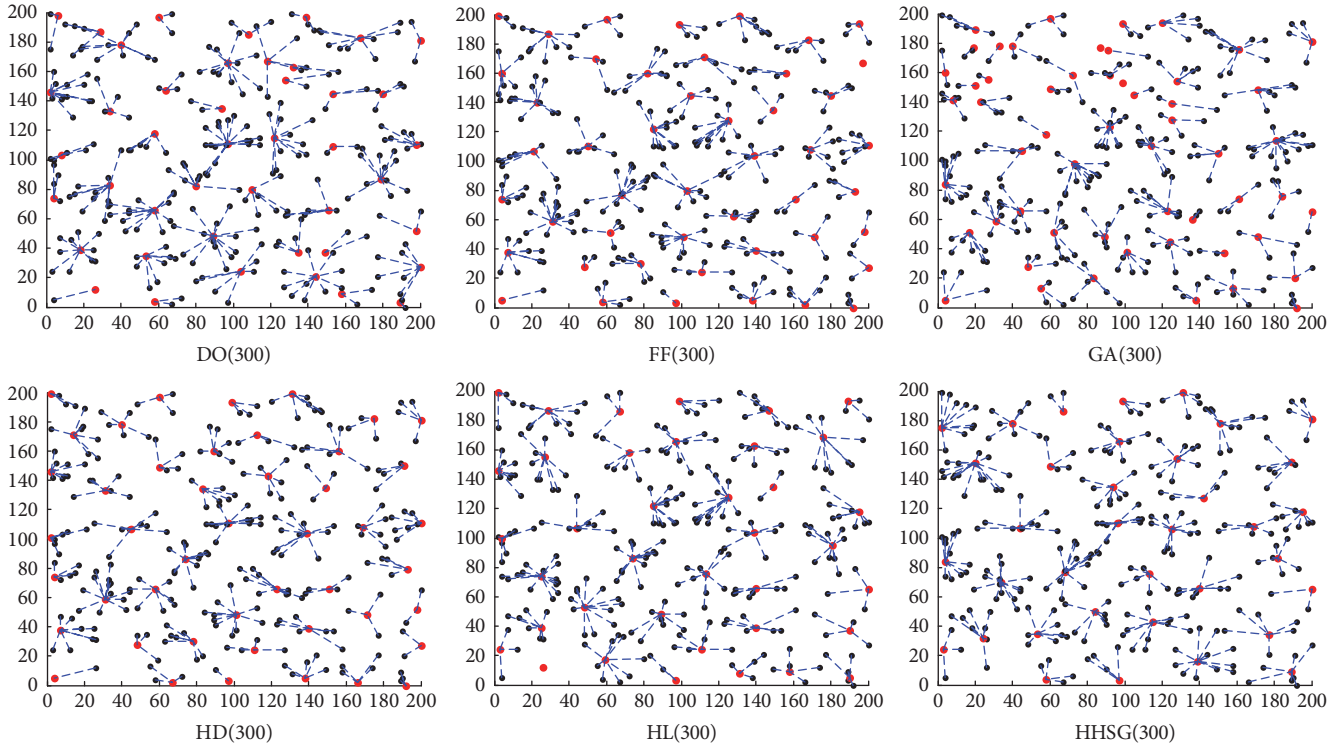


Figure 12: 300-node three-tier network in six methods.

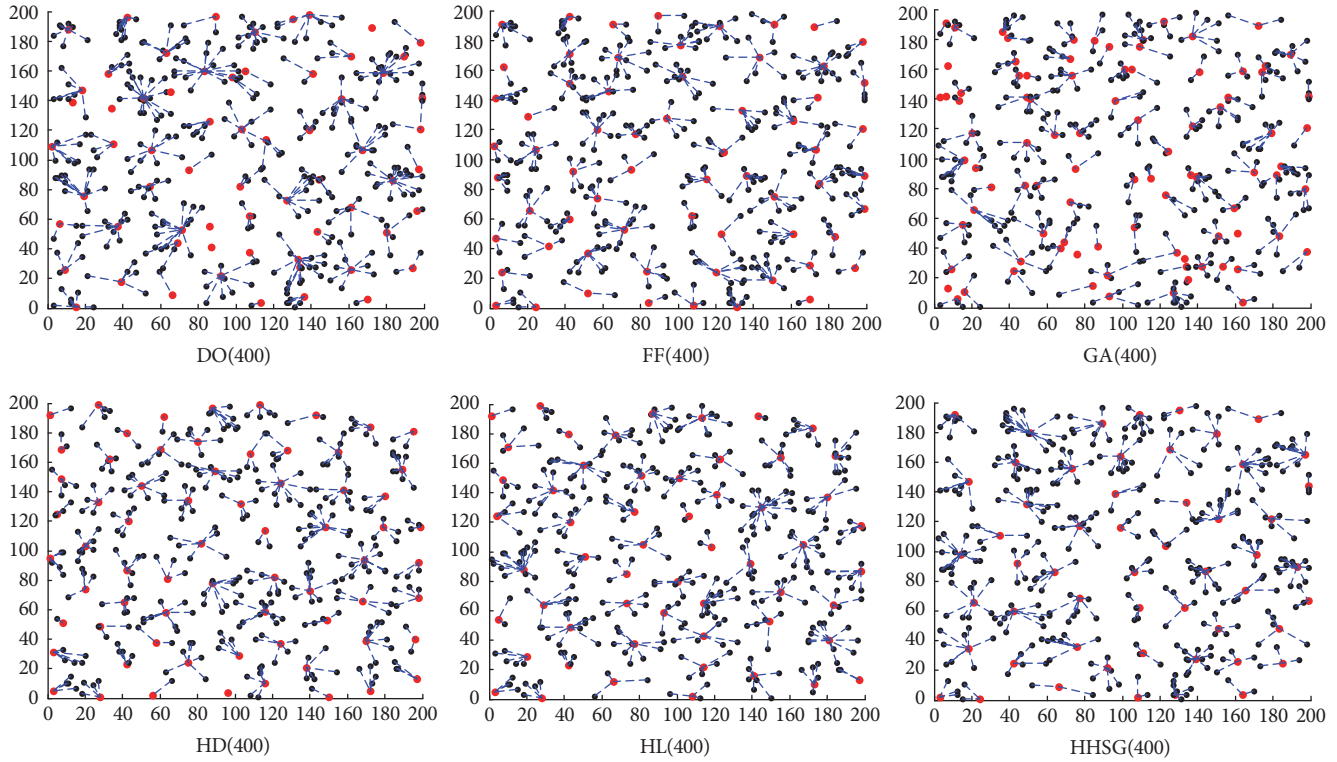


Figure 13: 400-node three-tier network in six methods.

Symbols

n: The number of sensor nodes in a network
c_p: The number of chromosomes in a population
t_n: The number of chromosomes generated by p_s
NID: Node numbering, which is nonrepeatable for each node from 1 to n
h_c: Huffman code for one sensor node
t_d: The positions of one digit in a Sudoku grid
p_s: Those nodes near to t_d position which are potential sites.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the Fujian Provincial Natural Science Foundation, China (2017J01730), the Key Project of Fujian Education Department Funds, China (JA15323), and Shenzhen Innovation and Entrepreneurship Project no. GRCK20160826105935160.

References

[1] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: early architecture and research perspectives," *IEEE Network*, vol. 29, no. 3, pp. 104–112, 2015.

[2] J. J. Loverich, J. W. Stephen, and D. B. Scott, "Wireless sensor network," Tech. Rep. U.S. Patent No. 20,160,077,501, 2016.

[3] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[4] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, 2011.

[5] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, vol. 2, 2001.

[6] T. Anker et al., "Efficient clustering for improving network performance in wireless sensor networks," in *Wireless Sensor Networks*, pp. 221–236, Springer, Berlin, Germany, 2008.

[7] J.-S. Leu, T.-H. Chiang, M.-C. Yu, and K.-W. Su, "Energy efficient clustering scheme for prolonging the lifetime of wireless sensor network with isolated nodes," *IEEE Communications Letters*, vol. 19, no. 2, pp. 259–262, 2015.

[8] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," *IEEE Communications Letters*, vol. 9, no. 11, pp. 976–978, 2005.

[9] L. Tian-Hua, Y. Si-Chao, and W. Xiao-Wei, "A fault management protocol for low-energy and efficient wireless sensor networks," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 1, pp. 34–45, 2013.

[10] M. Magno, T. Polonelli, L. Benini, and E. Popovici, "A low cost, highly scalable wireless sensor network solution to achieve smart LED light control for green buildings," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2963–2973, 2015.

[11] S. Tien-Wen, L. Chao-Yang, L. You-Te, S. Liang-Cheng, and L. Fu-Tian, "Energy-aware topology construction and maintenance for energy harvesting sensor networks," *Advances in Intelligent Systems and Computing*, vol. 533, pp. 723–731, 2016.

[12] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2-3, pp. 293–306, 1985.

[13] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1947–1960, 2010.

[14] T.-W. Sung and C.-S. Yang, "A Voronoi-based sensor handover protocol for target tracking in distributed visual sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 586210, 14 pages, 2014.

[15] Y.-T. Chen, M.-F. Horng, C.-C. Lo, S.-C. Chu, J.-S. Pan, and B.-Y. Liao, "A transmission power optimization with a minimum node degree for energy-efficient wireless sensor networks with full-reachability," *Sensors (Switzerland)*, vol. 13, no. 3, pp. 3951–3974, 2013.

[16] N. Trong-The, D. Thi-Kien, H. Mong-Fong, and S. Chin-Shiuh, "An energy-based cluster head selection algorithm to support long-lifetime in wireless sensor networks," *Journal of Network Intelligence*, vol. 1, no. 1, pp. 23–37, 2016.

[17] S. Majumdar Suprotim, M. Pattanaik, and J. V. Alamelu, "Energy efficient wireless sensor network for polyhouse monitoring," *European Journal of Advances in Engineering and Technology*, vol. 2, no. 6, pp. 77–82, 2015.

[18] A. Al-Radaideh, A. R. Al-Ali, S. Bheiry, and S. Alawnah, "A wireless sensor network monitoring system for highway bridges," in *Proceedings of the 1st International Conference on Electrical and Information Technologies, ICEIT 2015*, pp. 119–124, mar, March 2015.

[19] W.-L. Chang, D. Zeng, R.-C. Chen, and S. Guo, "An artificial bee colony algorithm for data collection path planning in sparse wireless sensor networks," *International Journal of Machine Learning and Cybernetics*, 2013.

[20] T.-S. Pan, T.-T. Nguyen, T.-K. Dao, and S.-C. Chu, "An Optimal Clustering Formation for Wireless Sensor Network Based on Compact Genetic Algorithm," in *Proceedings of the 3rd International Conference on Robot, Vision and Signal Processing, RVSP 2015*, pp. 294–299, Kaohsiung, Taiwan, November 2015.

[21] M. S. Moradi and M. Abedini, "A combination of genetic algorithm and particle swarm optimization for optimal DG location and sizing in distribution systems," *International Journal of Electrical Power & Energy Systems*, vol. 34, no. 1, pp. 66–74, 2012.

[22] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[23] M. Bader, *Space-Filling Curves: An Introduction with Applications in Scientific Computing*, vol. 9, Springer Science & Business Media, 2012.

[24] Y. Yan and Y. Mostofi, "An efficient clustering and path planning strategy for data collection in sensor networks based on space-filling curves," in *Proceedings of the 2014 53rd IEEE Annual Conference on Decision and Control, CDC 2014*, pp. 6895–6901, Los Angeles, CA, USA, December 2014.

[25] S. Ghafoor et al., "An efficient trajectory design for mobile sink in a wireless sensor network," *Computers & Electrical Engineering*, vol. 40, no. 7, pp. 2089–2100, 2014.

[26] A. Alsaadi Shanon, W. Tat-Chee, and A. Munther, "Application of harmony search optimization algorithm to improve connectivity in wireless sensor network with non-uniform density," *Journal of Information Science and Engineering*, vol. 31, no. 4, pp. 1475–1489, 2015.

[27] K. Konrad-Felix and G. Wunder, "6doku: towards secure over-the-air preloading of 6LoWPAN nodes using PHY key generation," in *Proceedings of Smart SysTech 2015; European Conference on Smart Objects, Systems and Technologies (VDE)*, 2015.

[28] S. Khuller, R. Pless, and Y. J. Sussmann, "Fault tolerant K-center problems," *Theoretical Computer Science*, vol. 242, no. 1-2, pp. 237–245, 2000.

[29] J. Hartmanis, M. R. Garey, and D. S. Johnson, "Computers and Intractability: a guide to the theory of np-completeness," *Siam Review*, vol. 24, article 90, no. 1, 1982.

[30] H. Du, Y. Xu, and B. Zhu, "An incremental version of the k-center problem on boundary of a convex polygon," *Journal of Combinatorial Optimization*, vol. 30, no. 4, pp. 1219–1227, 2015.

[31] D. Liang, L. Mei, J. Willson, and W. Wang, "A simple greedy approximation algorithm for the minimum connected k-Center problem," *Journal of Combinatorial Optimization*, vol. 31, no. 4, pp. 1417–1429, 2016.

[32] A. E. Feldmann, "Fixed Parameter Approximations for k-Center Problems in Low Highway Dimension Graphs," in *Automata, Languages, and Programming*, Lecture Notes in Computer Science, pp. 588–600, Springer, Heidelberg, Germany, 2015.

[33] S. Chechik and D. Peleg, "The fault-tolerant capacitated K-center problem," *Theoretical Computer Science*, vol. 566, pp. 12–25, 2015.

[34] http://chnglv.lofter.com/?page=4&t=1425308233546.

[35] K.-L. Chung, Y.-L. Huang, and Y.-W. Liu, "Efficient algorithms for coding Hilbert curve of arbitrary-sized image and application to window query," *Information Sciences. An International Journal*, vol. 177, no. 10, pp. 2130–2151, 2007.

[36] N. Chen, N. Wang, and B. Shi, "A new algorithm for encoding and decoding the Hilbert order," *Software - Practice and Experience*, vol. 37, no. 8, pp. 897–908, 2007.

[37] D. Harel and Y. Koren, "Graph drawing by high-dimensional embedding," in *Graph drawing*, vol. 2528 of *Lecture Notes in Comput. Sci.*, pp. 207–219, Springer, Berlin, Germany, 2002.

[38] B. Robič and J. Mihelič, "Solving the k-center problem efficiently with a dominating set algorithm," *Journal of Computing And Information Technology*, vol. 13, no. 3, pp. 225–234, 2005.