Hindawi
Journal of Sensors
Volume 2019, Article ID 2986954, 9 pages
https://doi.org/10.1155/2019/2986954

Research Article

DV-Hop Node Location Algorithm Based on GSO in Wireless Sensor Networks

Ling Song[1,2], **Liqin Zhao**,[1,2] **and Jin Ye**[1,2]

[1]*School of Computer & Electronic Information, Guangxi University, Nanning 530004, China*
[2]*Guangxi Key Laboratory of Multimedia Communications and Network Technology, Nanning 530004, China*

Correspondence should be addressed to Ling Song; jqian@gxu.edu.cn

Received 2 August 2018; Accepted 16 December 2018; Published 7 February 2019

Academic Editor: Stephen James

Copyright © 2019 Ling Song et al. 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.

Node location is one of the most important problems to be solved in practical application of WSN. As a typical location algorithm without ranging, DV-Hop is widely used in node localization of wireless sensor networks. However, in the third phase of DV-Hop, a least square method is used to solve the nonlinear equations. Using this method to locate the unknown nodes will produce large coordinate errors, poor stability of positioning accuracy, low location coverage, and high energy consumption. An improved localization algorithm based on hybrid chaotic strategy (MGDV-Hop) is proposed in this paper. Firstly, a glowworm swarm optimization of hybrid chaotic strategy based on chaotic mutation and chaotic inertial weight updating (MC-GSO) is proposed. The MC-GSO algorithm is used to control the moving distance of each firefly by chaos mutation and chaotic inertial weight when the firefly falls into a local optimum. The experimental results show that MC-GSO has better convergence and higher accuracy and avoids the premature convergence. Then, MC-GSO is used to replace the least square method in estimating node coordinates to solve the problem that the localization accuracy of the DV-Hop algorithm is not high. By establishing the error fitness function, the linear solution of coordinates is transformed into a two-dimensional combinatorial optimization problem. The simulation results and analysis confirm that the improved algorithm (MGDV-Hop) reduces the average location error, increases the location coverage, and decreases and balances the energy consumption as compared to DV-Hop and the location algorithm based on classical GSO (GSDV-Hop).

1. Introduction

A wireless sensor network (WSN) is a network formed by a large number of sensor nodes through wireless communication, which is used to perceive and transmit all kinds of information. There is a wide demand for location service information in real life and work. The location technology of WSNs is the base of location service.

Currently, the positioning technology of WSNs can be divided into two categories: range-based and range-free. Although range-based positioning technology has high positioning accuracy, it is not suitable for the applications which require low power consumption and low cost [1–3]. DV-Hop as a range-free algorithm has been widely used because of its simplicity, efficiency, and low cost. However, a least square method used in the third stage of DV-Hop makes large node positioning error. Essentially, DV-Hop is an optimization problem based on the measured values of different distances or paths. Applying intelligent optimization algorithms to the localization technology is a new attempt to solve the problems of the DV-Hop localization algorithm. Its basic idea is to establish the mathematical theory system of intelligent location by observing the biological model of nature and carry out iterative optimization of the corresponding objective function. At present, many intelligent algorithms have been applied to node localization, such as the ant colony algorithm, genetic algorithm, particle swarm optimization (PSO), and glowworm swarm optimization (GSO) or firefly swarm optimization. An improved DV-Hop algorithm based on mixed chaos strategy (MGDV-Hop) is proposed in this paper, in which the GSO of hybrid chaotic strategy (MC-GSO) is based on chaos mutation and chaotic inertia weight. MC-GSO increases the diversity of GSO algorithm population, enhances the

ability of local and global search, avoids the algorithm falling into the local optimal state prematurely, and improves the robustness of the algorithm. MGDV-Hop based on MC-GOS can improve the location coverage and reduce the node energy consumption while improving the accuracy of node location.

The rest of the paper is organized as follows. Section 2 contains the related works. Section 3 contains the introduction of the DV-Hop algorithm. Section 4 describes the MC-GSO and MGDV-Hop algorithms proposed in this paper. Section 5 contains the simulation experiments and analysis. Section 6 gives the conclusions.

2. Related Works

Many scholars proposed the following improvements on the low accuracy of DV-Hop localization: the concept of minimum deviation degree was proposed, although the positioning error was reduced, the energy consumption was high, and when the number of nodes is large, only the unknown nodes close to the beacon have higher positioning accuracy [4]. A localization algorithm combining a centroid with DV-Hop was proposed, which reduced the energy consumption of nodes but still had a high localization error [5]. The communication radius of nodes and the number of hops between nodes were corrected, and the average distance of per hop was corrected. The improved algorithm reduced the average localization error. However, the power consumption of the nodes is increased due to the increase of calculation for the multiple corrections [6]. In recent years, some intelligent algorithms have been used to improve DV-Hop; for example, the particle swarm optimization (PSO) is one of the most popular algorithms. The PSO was used to improve the localization accuracy of DV-Hop [7–10]. However, the PSO algorithm is weak in solving discrete and combinatorial optimization problems, especially for some nonrectangular coordinate systems, and the parameter control of the PSO is not flexible. This research on DV-Hop localization algorithms mentioned above barely mentioned the experimental data of energy consumption.

The concepts of success and failure of search in glowworm swarm optimization were introduced, and the GSO was optimized by changing the step size and establishing multimodal functions [11]. The range of fluorescein changes in GSO during the process of fluorescein renewal was limited [12]. The population of fireflies was divided into several subgroups of different sizes, and each subgroup was optimized separately, improving the convergence rate [13]. The GSO was combined with integrated learning, using the greedy algorithm to select some fireflies, and constructed the integrated firefly [14]. The local search ability of GSO was improved by introducing chaotic local search operator [15]. Chaotic perturbation term was introduced in an iterative process, and the optical absorption coefficient was linearly correlated with an iteration number [16]. A strategy based on Gao Si mutation was proposed, in which the mutation factor was added to avoid the algorithm falling into a local optimum [17]. A GSO of variable step size was proposed and applied to an MLP equalizer [18]. For the convenience of comparison, Table 1 gives some main performances of these improved GSO algorithms in [11-18] with respect to the average number of iterations, the average convergence time, and the error of optimal value.

In order to obtain better performances in all three of the above, in this paper, a GSO of hybrid chaotic strategy (MC-GSO) is proposed based on chaos mutation and chaotic inertia weight. MC-GSO increases the diversity of GSO algorithm population, enhances the ability of local and global search, avoids the algorithm falling into the local optimal state prematurely, and improves the robustness of the algorithm. Then, MC-GSO is used to replace the least square method in the third stage of DV-Hop for solving the defect of low positioning accuracy of DV-Hop, improving location coverage and the power consumption of nodes in WSNs.

3. DV-Hop Algorithm

3.1. DV-Hop Process. DV-Hop is a no distance measurement location algorithm based on distance vector routing; it is proposed to avoid the direct measurement of distance between nodes. The basic idea of DV-Hop is that the distance between the unknown node and the reference node is represented by the product of the average distance of per hop and the least hop number between the two nodes. Furthermore, the location of nodes is determined by triangulation or maximum likelihood estimation. The work flow of the DV-Hop algorithm is as follows.

3.1.1. Distance Vector Switching. In this stage, the minimum number of hops between the unknown node and the beacon node is obtained. The data storage unit of each node maintains a data table $<\mathrm{id}_i, x_i, y_i, h_i>$, in which id_i is the identifier of the beacon node i that is the ID number and (x_i, y_i) is the location of the beacon node i, and h_i is the number of hops from unknown node to beacon node i.

When the node detects that the packet information it receives comes from the same ID number, it invokes the information in its own data table. If the number of hops in the data table is higher than the number of hops currently received, the information of hops in the data table will be updated; otherwise, it will be ignored.

3.1.2. Mean Hop Distance and Flooding Broadcast. After the first stage, the minimum hop number between the unknown node and the beacon node, as well as the coordinates of the other beacons deployed in WSNs, has been recorded in the data table of each beacon node. Then we can calculate the average distance of each hop ($\mathrm{AvgHopDistance}_i$) in the network, and the average hop distance is broadcast to all nodes in the WSNs as a correction value. The calculation method is given by formula (1), and then the distances between the unknown node and other beacon nodes are obtained.

$$\mathrm{AvgHopDistance}_i = \frac{\sum_{i\neq j}\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}}{\sum_{i\neq j} h_{ij}} \quad (i \neq j, \forall j). \tag{1}$$

Table 1: Comparison of references [11–18].

References	Average number of iterations	Average convergence time	Error of optimal value
Zheng-xin and Yong-quan [11]	Fewer	Longer	Bigger
Jia-kun and Yong-quan [12]	More	Shorter	Bigger
Qiang et al. [13]	Fewer	Longer	Bigger
Wang et al. [14]	More	Longer	Smaller
Nan et al. [15]	More	Longer	Smaller
Hua-li et al. [16]	More	Shorter	Bigger
Pan and Xu [17]	More	Longer	Smaller
Sarangi et al. [18]	More	Longer	Bigger

In formula (1), h_{ij} represents the minimum number of hops between the i_{th} beacon node $i(x_i, y_i)$ and the j_{th} beacon node (x_i, y_i) .

3.1.3. Estimation of the Location of Unknown Nodes. Assuming that $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ are the position coordinates of N beacons, according to the second stage, the distance between the unknown node and the beacon node is $d_1, d_2, \ldots d_N$, respectively, and the coordinates of the unknown node are (x_L, y_L). A linear equation group is established to solve the coordinates of unknown node L.

3.2. Shortcomings of DV-Hop. Using the location information of different beacon nodes, the mean hop distance between nodes (AvgHopDistance$_i$) in the WSNs is calculated. If the distribution of nodes in WSNs is not uniform, the average hop distance can not represent the hop distance between nodes in the whole network. Therefore, there is a great error in estimating the coordinates of an unknown node in this case.

When the number of hops from an unknown node to a beacon node increases to 2, the error will accumulate because the error is proportional to the number of hops.

When determining the location of an unknown node, the iterative location of a beacon node will cause error to accumulate. When the beacon node rebroadcasts its own data packet, it may add a new error to the previous positioning error, increasing the final error. When the WSN is deployed in a wide range, it will result in a greater positioning error [19].

4. DV-Hop Based on GSO of Mixed Chaos Strategy(MGDV-Hop)

4.1. Basic Idea of MGDV-Hop. In order to solve the problems of DV-Hop, we consider improving this algorithm with GSO.

Firstly, a hybrid chaotic strategy (MC-GSO) based on chaos variation and position update of inertial weight is proposed. MC-GSO is used to avoid the firefly individuals falling into the local optimal state prematurely during the evolution of the population. Then, the MC-GSO algorithm is used to replace the least square method used by DV-Hop in estimating node coordinates. According to the calculation formulas of an unknown node, the error between the estimated coordinates and the real coordinates is obtained, and the corresponding error fitness function is derived. The minimum value of fitness function is solved by the MC-GSO algorithm, reducing the mean positioning error.

4.2. GSO Based on Mixed Chaos Strategy (MC-GSO)

4.2.1. Position Update Based on Chaotic Inertial Weight. As the population is evolving, the differences among individuals gradually reduce in the later iteration period of the standard GSO. Inspired by the inertia weight of the particle swarm optimization, the position updating formula of the MC-GSO is as follows:

$$X_i(t+1) = w_t * X_i(t) + s * \frac{X_j(t) - X_i(t)}{\left\| X_j(t) - X_i(t) \right\|}. \quad (2)$$

In formula (2), the first part on the right is the position of the firefly individual before moving, which ensures the global convergence of the algorithm; the second part is the moving step size and the probability factor of the firefly individual choosing the moving direction. In order to enhance the global and local search ability of the algorithm, the inertia weight is introduced to ensure that the position of the firefly after moving is superior to the previous position.

The chaotic inertial weight is using logistic mapping, such as

$$w_t + 1 = 4w_t(1 - w_t). \quad (3)$$

In formula (3), t is the number of iterations, $w \in (0, 1)$, $w_1 \neq \{0.25, 0.5, 0.75\}$, and w_t is the inertial weight of the t_{th} iteration.

4.2.2. Chaotic Variation of Firefly Individual. When the convergence rate of the GSO algorithm is slow, chaotic variables are used to replace some fireflies and make the algorithm jump out of the local optimum quickly. The strategy is defined as

$$X_{\text{in}}(t) = X_i(t) \times [1 + k \times M(n)], \quad n = 1, 2, \ldots, Z, \quad (4)$$

where t denotes the number of iterations of the algorithm, $M(n)$ is a chaotic sequence, z denotes the number of random vectors generated by chaos, and k is the influence factor of chaotic mutation operators; its expression are as follows:

$$k = 1 - \frac{t-1}{T_{\max}}, \quad t = 1, 2, \ldots, T_{\max}, \tag{5}$$

where $T_{\max}$ represents the maximum iteration number of the algorithm.

The chaotic sequence produced by chaos has an iterative initial value in the range of [0,1], and the iterative initial value $w_1 \neq 0$. Firstly, a vector in the interval of [0,1] is generated randomly in the solution space of d dimensional. $M(1)$ is shown as formula (6), and chaotic sequence is as formula (7):

$$M(1) = 1 - 2\text{rand}(1, d), \tag{6}$$

$$\begin{aligned} M(n+1) &= 4M(n)^3 - 3M(n), \\ 0 \leq M(n) &\leq 1, \quad n = 1, 2, \ldots, Z. \end{aligned} \tag{7}$$

Z chaotic sequences $\{M(1), M(2), \ldots, M(Z)\}$ are obtained by using formula (7).

By using the generated Z chaotic sequences, the chaotic variation of the t_{th} generation firefly i is carried out according to formula (4), and the sequence $X_{i,(1,2,\ldots,Z)}(t) = X_i \times [1 + k \times \{M(1), M(2), \ldots, M(Z)\}]$ is obtained. Comparing the fitness of the target function before and after the mutation, if the optimal solution after chaotic mutation is superior to the original solution, the individual of firefly i will be updated.

4.2.3. Algorithm Flow of MC-GSO. The MC-GSO algorithm is a multichaos strategy which combines chaotic inertial weight and chaos mutation. In the initial iteration of the algorithm, a weight is added to the first part of the position update formula to control the influence of the previous generation of firefly location information on the current firefly location information. The weight determines the moving distance of the firefly and strengthens the local search ability of the population. With the iterative evolving of the population, the individual trapped in the local optimum is mutated by chaotic perturbation mechanism. The states of individual fitness before and after mutation are compared, and the best state is selected to make the algorithm jump out of the local optimum and continue to iterate; the iteration stops until a global optimum is found.

The specific steps of MC-GSO are as follows:

Step 1. Initiate the relevant parameters. The size of the firefly population is n, and the search dimension is d. The maximum range of the individual decision domain of the firefly is Rs, and its coefficient of variation is β. The maximum iteration number is $T_{\max}$. The volatilization coefficient of fluorescein and its renewal rate are ρ and γ, respectively. The random position of firefly individual i is $x_i(0)$. The initial value of the fluorescein is I_0, its perceptual range is r_0, and the moving step size is s when the position is updated. The population is initialized at random, and the range of activity of fireflies is set.

Step 2. Update fluorescein. Calculate all fireflies' objective function values (fitness values), and initialize bulletin boards, and convert them to the corresponding fluorescein value $I_i(t)$. Meanwhile, firefly selects individuals with larger fluorescein than its own to form neighbor set $N_i(t)$.

Step 3. The probability P_{ij} of individual i moving to individual j is calculated, and the roulette method is used to select individual j.

Step 4. Update the position of fireflies according to formulas (2) and (3).

Step 5. Update the firefly's perceptual radius.

Step 6. Update the fitness value. Calculate the fluorescein and fitness value of the firefly individuals in the current population. If the current fitness value of the firefly individuals is better than the original state, the original state and the fitness value will be changed.

Step 7. If the optimal fitness of the firefly population does not change or change very little after three successive iterations, the algorithm is in the state of local extremum, and Step 8 is executed; otherwise, Step 9 is executed.

Step 8. The chaotic variation of fireflies in the local optimal state is carried out, comparing the fitness of the target function before and after the chaos mutation. If the value after chaos mutation is better than the original value, the mutated firefly individual will replace the premutation firefly individual i. If the firefly individual that has been replaced is superior to the optimal in the bulletin board, the individual in the bulletin board is replaced by the former.

Step 9. After the end of an iteration, if the algorithm satisfies the termination condition, it exits the iteration; otherwise, it turns to Step 2 and continues to iterate until the optimal solution is output.

4.3. The Flow of MGDV-Hop

4.3.1. Fitness Function of Error. Let the estimation distance between the beacon node i and the unknown node is $\bar{d}_i$, which is obtained in the second stage of DV-Hop based on the AvgHopDistance_i and the number of hops. Let the true distance between the unknown node and the beacon node be d_i; the error of the localization is shown as

$$f(x, y) = \bar{d}_i - d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i. \tag{8}$$

Fitness function Fitness(x, y) is used to denote the sum of errors in positioning, such as

$$\text{Fitness}(x, y) = \sum_{i=1}^{N} f(x, y)_i, \tag{9}$$

where N is the total number of beacons in the WSNs. MC-GSO calculates the minimum value of Fitness(x, y) by iterating. Finally, the unknown node coordinates with the smallest error are output.

4.3.2. The Steps of MGDV-Hop

Step 1. When $t = 0$, the network topology of WSNs and related parameters is initiated, including the size of the network, the number of nodes in the network, the number of beacon nodes, the number of unknown nodes, and the number of hops between nodes.

Step 2. The beacon nodes broadcast their own information to the whole network by flooding mechanism. According to the position information obtained by the beacon nodes, the average hop distance AvgHopDistance$_i$ is calculated by using the minimum hop value according to formula (1).

Step 3. Each unknown node obtains AvgHopDistance$_i$ from the nearest beacon, and the distance between the unknown node and the beacon node is obtained.

Step 4. Calculate the value of Fitness() according to the formula (4).

Step 5. Initialize the firefly population, and use the fitness function Fitness() as a firefly individual; the fluorescein of the firefly can be obtained by the fitness function which is constructed by the location error $\bar{d}_i - d_i$(the larger the fluorescein intensity, the smaller the location error). The dimension of the population is D, and the scale is N, setting the relevant parameters.

Step 6. Carry out Step 3 to Step 9 of MC-GSO.

Step 7. Through MC-GSO, the minimum value of the fitness function is obtained, which is the optimal solution, and output it. Finally, the minimum error point is obtained, and the coordinates of the unknown node can be obtained.

5. Simulation Experiment

5.1. Experiment of MC-GSO

5.1.1. Experimental Parameters and Environment. The experimental parameters are as follows:

The step size s was selected according to experience, and different test functions were chosen with different step sizes. Other experimental parameters are set according to Table 2.

The experimental environment is as follows:

Table 2: Reference values of parameters of GSO.

ρ	γ	β	I_0	n_t	N
0.4	0.6	0.08	5	5	100

The processor is 2.7 GHz Intel Core i5, the display is Intel Iris Graphics 6100 1536 MB, the memory is 8 GB 1867 MHz DDR3, the operating system is Macos Sierra, and the integrated development environment is MATLAB R2015a.

5.1.2. Experimental Test Function. The MC-GSO is compared with the classical GSO and the GS-GSO which is based on Gaussian mutation, and the performances are compared by using six standard test functions.

The parameter setting of the test functions and the search range are shown in Table 3.

5.1.3. Experimental Results and Analysis. GSO, GS-GSO, and MC-GSO were used to carry out 25 independent experiments on the six test functions above. The optimal solution of the minimum value of the six functions is solved. We can obtain the following performances: the minimum iteration times of convergence ($T_{\min}$), the average iteration times (T_{average}), the maximum iteration times ($T_{\max}$), the average convergence time (AverageTime), convergence rate (T_{rate}), the optimal value of the function (fitness$_{\text{best}}$), the worst value (finness$_{\text{bad}}$), and the average value (fitness$_{\text{average}}$). The algorithm converges when the fitness function satisfies $|\text{fitness}_{\text{best}}() - \text{fitness}| < 10^{-6}$; otherwise, it is considered that the algorithm is not convergent.

5.1.4. Convergence Rate and Iteration Times. Table 4 shows that the GSO algorithm does not converge to the functions F_1 to F_5, while the convergence rate of the GS-GSO and the MC-GSO to the function F_1 to F_6 reaches 100%. The convergence accuracy of the MC-GSO is the highest, followed by that of the GS-GSO, and the convergence accuracy of the GSO is the least.

5.1.5. Global Optimization Ability and Solution Accuracy. From the data in Table 5, we can see that the convergent iterative of MC-GSO can reach a relatively stable state. With the increasing of iteration times, the optimal function value of MC-GSO can achieve convergence accuracy when the number of iterations is less than 40 times. For the same iteration of the same function, the convergence time of MC-GSO is much lower than that of GS-GSO, and the accuracy of MC-GSO is higher than that of GSO and GS-GSO.

5.1.6. Analysis. In MC-GSO, aiming at the shortcomings of the classical GSO algorithm, in the early stage of iteration, chaotic inertia weight is introduced into the position update formula in order to avoid individual missing of the optimal solution and falling into local optimum. In the later stage of iteration, in order to improve the convergence rate, when the firefly falls into the local optimal solution, the algorithm makes chaos mutation on each individual firefly for jumping out of the local optimum.

Table 3: Six standard test functions.

Function	Range	Dimension
$F_1 = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right] + 10$	$[-5.12,5.12]^n$	10
$F_2 = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	10
$F_3 = \sum_{i=1}^{n}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$[-50,50]^n$	10
$F_4 = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-100,100]^n$	10
$F_5 = \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2} + 0.5$	$[-100,100]^n$	10
$F_6 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	$[-50,50]^n$	2

Table 4: Convergence comparison of three algorithms.

Function	Algorithms	T_{max}	$T_{average}$	T_{min}	AverageTime	T_{rate}
	GSO	—	—	—	—	0/25
F_1	GS-GSO	251	113	76	18.4361	25/25
	MC-GSO	28	14	10	6.3827	25/25
	GSO	—	—	—	—	0/25
F_2	GS-GSO	75	53	38	9.0037	25/25
	MC-GSO	40	23	10	6.1043	25/25
	GSO	—	—	—	—	0/25
F_3	GS-GSO	152	85	58	14.3502	25/25
	MC-GSO	9	15	28	8.6545	25/25
	GSO	—	—	—	—	0/25
F_4	GS-GSO	45	35	25	11.4311	25/25
	MC-GSO	6	9	14	6.0548	25/25
	GSO	—	—	—	—	0/25
F_5	GS-GSO	55	42	25	7.1680	25/25
	MC-GSO	15	13	8	3.4345	25/25
	GSO	110	74	50	13.6288	0/25
F_6	GS-GSO	50	35	20	4.4947	25/25
	MC-GSO	25	6	3	2.7042	25/25

5.2. Experiments for MGDV-Hop

5.2.1. Location Error Calculation Model. The calculation for the location error of the node is shown as formula (10), and the calculation for the mean positioning error is shown as formula (11):

$$\text{AvgError} = \frac{\sum_{i=1}^{n}\sqrt{(x - x_i)^2 + (y - y_i)^2}}{n \times r}, \tag{10}$$

$$\overline{\text{AvgError}} = \frac{\sum_{i=1}^{n}\text{AvgError}}{n}, \tag{11}$$

where n is the number of unknown nodes in the region, (x_i, y_i) is the coordinates of an unknown node obtained by MGDV-Hop, (x, y) is the actual coordinates of the node, and r is the communication radius of the node.

5.2.2. Location Coverage Calculation. The formula for calculating the location coverage is shown as

$$C = \frac{m}{n}, \tag{12}$$

where C is location coverage, m is the number of the nodes which have been located, and n is the total number of unknown nodes.

5.2.3. Energy Consumption Calculation Model. The energy consumption model is shown as formulas (13) and (14):

$$\begin{cases} E_{\text{Tx}}(k, d) = k \times E_{\text{elec}} + k \times \varepsilon_{\text{fs}} \times d^2, & d < d_0, \\ E_{\text{Tx}}(k, d) = k \times E_{\text{elec}} + k \times \varepsilon_{\text{mp}} \times d^4, & d \geq d_0, \\ d_0 = \sqrt{\dfrac{\varepsilon_{\text{fs}}}{\varepsilon_{\text{mp}}}}, \end{cases} \tag{13}$$

$$E_{\text{Rx}}(k) = k \times E_{\text{elec}}, \tag{14}$$

where $E_{\text{Tx}}(k, d)$ is the energy consumption that a node sends k bit data to another node which is at a distance d, $E_{\text{Rx}}(k)$ is the energy consumption of a node for receiving k bit data, E_{elec} is the energy consumption of the circuit for sending or receiving 1 bit data, ε_{fs} is free-space model amplifier multiple, and ε_{mp} is multipath fading model amplifier multiple.

In this paper, the average residual energy of all nodes E_{aver} and the variance of residual energy V are used to

Table 5: Optimum comparison of three algorithms.

Function	Algorithms	$\text{fitness}_{\text{best}}$	$\text{fitness}_{\text{bad}}$	$\text{fitness}_{\text{average}}$
	GSO	9.8194	34.6484	19.7816
F_1	GS-GSO	$4.6581e-29$	$9.4376e-10$	$6.3299e-11$
	MC-GSO	$1.863e-76$	$2.0307e-06$	$1.3539e-07$
	GSO	$5.2807e+03$	$5.7545e-67$	$4.1631e-68$
F_2	GS-GSO	$2.0007e-22$	$1.8004e-48$	$1.2179e-49$
	MC-GSO	$1.5763e-65$	$8.6243e+03$	$6.3502e+03$
	GSO	0.5717	3.1014	1.7556
F_3	GS-GSO	$7.7053e-15$	$2.7370e-07$	$4.7492e-08$
	MC-GSO	$6.2529e-42$	$5.4045e-32$	$3.6030e-33$
	GSO	1.9315	3.0087	2.6411
F_4	GS-GSO	$2.1675e-60$	$1.0421e-31$	$6.9475e-33$
	MC-GSO	$8.8282e-85$	$1.8735e-77$	$1.4173e-78$
	GSO	0.0038	0.9338	0.2944
F_5	GS-GSO	$1.1695e-32$	$1.5392e-10$	$1.2418e-11$
	MC-GSO	$7.7504e-83$	$1.4879e-73$	$1.2517e-74$
	GSO	$7.0542e-06$	$1.0968e-04$	$7.2056e-05$
F_6	GS-GSO	$1.5213e-59$	$3.3184e-39$	$6.2815e-46$
	MC-GSO	$5.8094e-111$	$1.3939e-98$	$1.1314e-88$

measure the energy consumption performance of the location algorithms. Their calculation is shown in (15), respectively:

$$E_{\text{aver}} = \frac{1}{n}\sum_{i=1}^{n} E_i,$$
$$V = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(E_i - E_{\text{aver}})^2}, \tag{15}$$

where E_i is the residual energy of node i after location and transmission and n is the total number of the nodes.

5.2.4. Experimental Parameters and Environment. In the environment of MATLAB 2015a, 100 sensor nodes are randomly deployed in a WSNs of 100 m × 100 m. The simulation experiment is performed 100 times under the condition of different communication radii and different numbers of beacon nodes. The experimental results are compared with the classical DV-Hop and the location algorithm based on classical GSO named the GSDV-Hop algorithm in terms of the average location error, node location coverage, and network energy consumption. The parameters of the following simulation experiments are shown in Table 6.

5.2.5. Experimental Results and Analysis

5.2.5.1. Influence of Communication Radius on Mean Positioning Error. In the case of changing the communication radius, the number of beacon nodes is set to 30, and the experimental results are shown in Figure 1. When the communication radius is from 15 m to 50 m, the average location error of the node with a larger communication radius is smaller. When the communication radius continues to increase, the average positioning error tends to be stable. From Figure 1, it can be concluded that the MGDV-Hop is superior to the other two algorithms in terms of location error.

Table 6: Parameters of simulation experiments.

Parameter	Value
Network size	100 m*100 m
Number of nodes	100
Number of beacons	5~50
Communication radius (r)	15 m~50 m
ε_{fs}	10 pJ/bit/m^2
ε_{mp}	0.0013 pJ/bit/m^4
E_{elec}	50 J/bit
Initial energy of a node	100 J

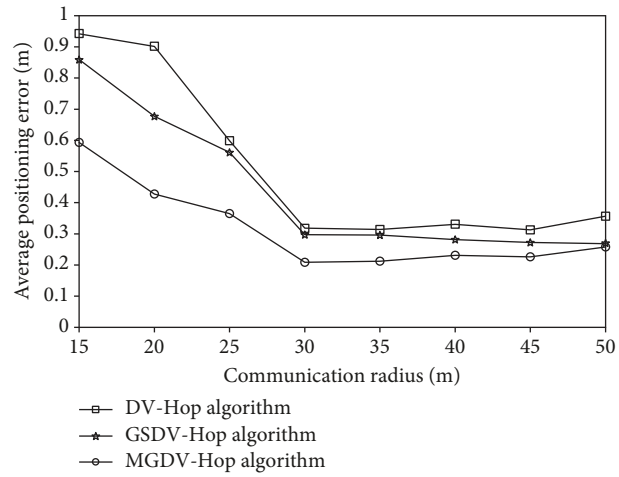


Figure 1: Relationship between the location error and communication radius.

5.2.5.2. Influence of the Number of Beacons on Positioning Error. From the experiment of the influence of the communication radius on the mean positioning error, we realize that the average location error is the smallest when the communication radius is 30 m; in this case, by changing the number of beacon nodes, the effect of the number of the beacon node number on the location error is determined. The simulation results are shown in Figure 2. When the number of beacons increases from 15 to 50, the average positioning error of the three algorithms shows a certain fluctuation, but it can be seen that the average positioning error of the MGDV-Hop is the lowest and the error curve is smoother than the other two algorithms, indicating that its stability is better.

5.2.5.3. Influence of the Number of Beacons on Positioning Coverage. Set $r = 30$ m; the MGDV-Hop, classical DV-Hop, and GSDV-Hop are simulated when the numbers of beacons

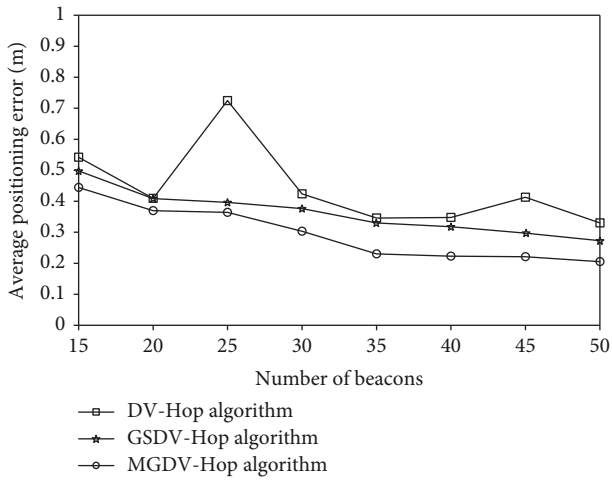


Figure 2: Relationship between the location error and number of beacons.

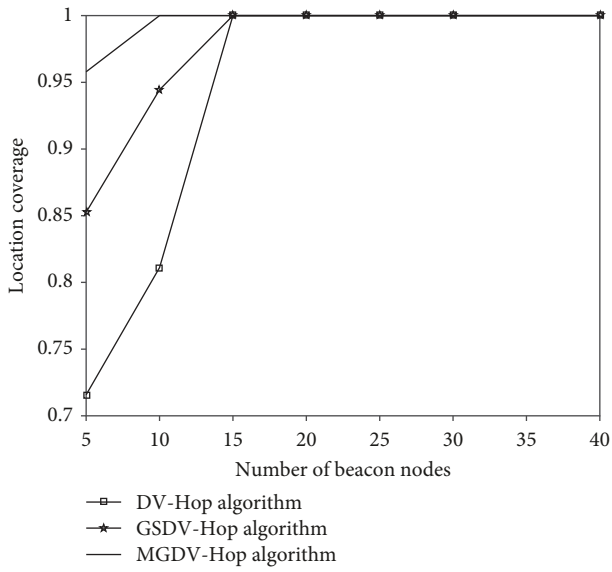


Figure 3: Relationship between the number of beacons and location coverage.

are 5, 10, 15, 20, 25, 30, and 40, respectively. Each simulation experiment is performed 100 times, and the average value of the simulation results is taken.

As shown in Figure 3, with the increasing of the number of beacon nodes, the positioning coverage of these three algorithms increases gradually, and when the number of beacons is small, the location coverage of MGDV-Hop is higher than that of GSDV-Hop and DV-Hop.

5.2.5.4. Comparison of Node Energy Consumption. Set $r = 30$ m; the number of beacon nodes is 35. The simulation experiment is performed 100 times, and the average value of the simulation results is taken.

In the process of location, the unknown node can establish a nonlinear system of equations by only knowing the distance to the beacon and then can estimate its coordinates by using the firefly optimization algorithm.

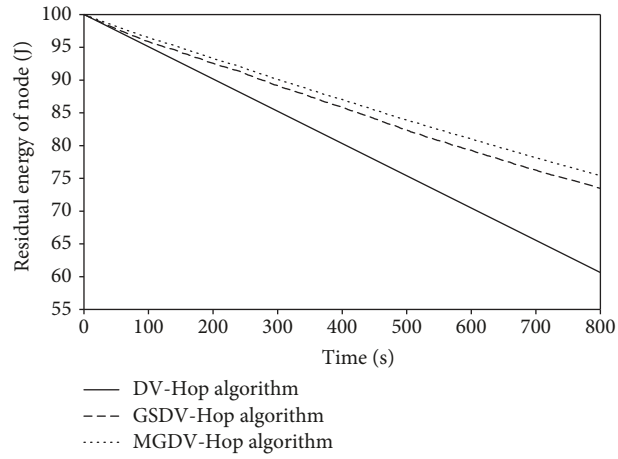


Figure 4: Relationship between the residual energy and time.

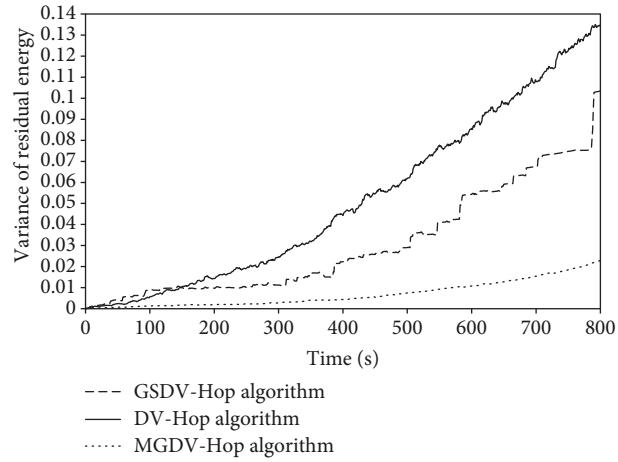


Figure 5: Relationship between the variance of residual energy and time.

The MC-GSO algorithm and the least square algorithm use the same method to calculate the distance between the unknown node and the beacon. The difference is that the MC-GSO is based on iterative optimization of fitness function, and no extra calculation is required. The least square method needs to solve the nonlinear equations. As a result, MGDV-Hop consumes less energy of nodes. Figure 4 shows the relationship between the average residual energy of nodes and time, and the relationship between the variance of residual energy of nodes and the time is shown in Figure 5. These compared results confirm that MGDV-Hop can reduce the energy consumption of nodes and make it more balanced.

5.2.6. Analysis. In practical applications, the distribution of nodes in WSNs is random. The DV-Hop will result in a high location error when the nodes are not evenly distributed. Aiming at the error caused by the random distribution, in MGDV-Hop, the MC-GSO proposed in the third section is used to reduce the error caused by randomness. By analyzing and estimating the coordinate errors, the error fitness function is established, and the MC-GSO algorithm is used

to approximate the fitness function and solve the coordinates of the unknown nodes when the error is minimum.

6. Conclusions

Because the classical DV-Hop produces a large localization error in estimating the position of the unknown node, a firefly algorithm based on a mixed chaos strategy is proposed to improve the DV-Hop.

In MC-GSO, chaotic mutation and chaotic inertial weight are used to avoid firefly individuals falling into a local optimal state prematurely during population evolution, and the diversity of the GSO algorithm population is increased. The ability of local search and global search can also be enhanced.

In MGDV-Hop, the least square method used by DV-Hop in estimating node coordinates is replaced by MC-GSO. According to the calculation formula of unknown nodes, the error between the estimated coordinates and the real coordinates is obtained, and the corresponding error fitness function is derived. The MC-GSO is used to solve the minimum value of fitness function, which reduces the average positioning error, increases the location coverage of unknown nodes, and reduces the energy consumption.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61762030).

References

[1] Y. Huang, J. Zheng, Y. Xiao, and M. Peng, "Robust localization algorithm based on the RSSI ranging scope," *International Journal of Distributed Sensor Networks*, vol. 11, no. 2, Article ID 587318, 2015.

[2] J. Zheng, Y. E. Sun, Y. Huang, Y. Wang, and Y. Xiao, "Error analysis of range-based localisation algorithms in wireless sensor networks," *International Journal of Sensor Networks*, vol. 12, no. 2, pp. 78–88, 2012.

[3] J. Zheng, Y. Huang, Y. Wang, Y. Xiao, and C. L. P. Chen, "Range-based localisation algorithms integrated with the probability of ranging error in wireless sensor networks," *International Journal of Sensor Networks*, vol. 15, no. 1, pp. 23–31, 2014.

[4] S. Xue-li and C. Guang, "An improved DV-Hop localization algorithm for wireless sensor networks," *Computer Engineering*, vol. 41, no. 7, pp. 115–119, 2015.

[5] Z. Zi-wei and Z. Chao, "A WSN node location algorithm based on centroid and DV-Hop algorithm," *Computer Applications and Software*, vol. 31, no. 11, pp. 130–133, 2015.

[6] X. Shao-bo, Z. Jian-mei, and Z. Xiao-li, "Improvement on DV-Hop localization algorithm in wireless sensors networks," *Journal of Computer Applications*, vol. 35, no. 2, pp. 340–344, 2015.

[7] Y. Gao, W. S. Zhao, C. Jing, and W. Z. Ren, "WSN node localization algorithm based on adaptive particle swarm optimization," *Applied Mechanics and Materials*, vol. 143-144, pp. 302–306, 2011.

[8] X. Chen and B. Zhang, "3D DV-hop localisation scheme based on particle swarm optimisation in wireless sensor networks," *International Journal of Sensor Networks*, vol. 16, no. 2, pp. 100–105, 2014.

[9] S. P. Singh and S. C. Sharma, "Implementation of a PSO based improved localization algorithm for wireless sensor networks," *IETE Journal of Research*, pp. 1–13, 2018.

[10] S. P. Singh and S. C. Sharma, "A modified DV-Hop algorithm for wireless sensor network localisation," *International Journal of Communication Networks and Distributed Systems*, vol. 21, no. 2, p. 187, 2018.

[11] H. Zheng-xin and Z. Yong-quan, "Adaptive glowworm swarm optimization algorithm with changing step for optimizing multimodal functions," *Computer Engineering and Applications*, vol. 48, no. 8, pp. 43–47, 2012.

[12] L. Jia-kun and Z. Yong-quan, "Glowworm swarm optimization algorithm based on max-min luciferin," *Application Research of Computers*, vol. 28, no. 10, pp. 3662–3664, 2011.

[13] F. Qiang, T. Nan, and Z. Yi-ming, "Firefly algorithm based on multi-group learning mechanism," *Application Research of Computers*, vol. 30, no. 12, pp. 3600–3602, 2013.

[14] Q. Wang, Y. Shi, G. Zeng, and X. Tu, "Improving global optimization ability of GSO using ensemble learning," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, pp. 118–121, Hangzhou, China, November 2013.

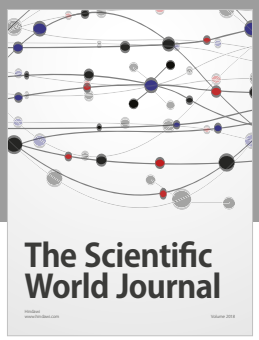
[15] Z. Nan, W. Xiang, and Y. Hao-jie, "An improved chaotic firefly algorithm," *Computer Simulations*, vol. 31, no. 10, pp. 306–312, 2014.

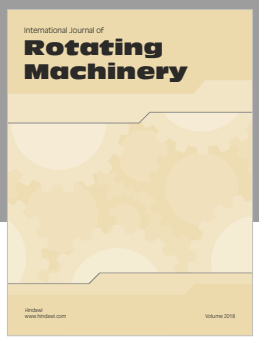
[16] X. Hua-li, S. Shou-bao, and C. Jia-jun, "Firefly algorithm with scale chaotic light absorption coefficient," *Application Research of Computers*, vol. 51, no. 2, pp. 368–371, 2015.

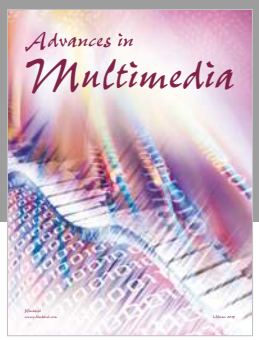
[17] G. Pan and Y. Xu, "Chaotic glowworm swarm optimization algorithm based on Gauss mutation," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 205–210, Changsha, China, August 2016.

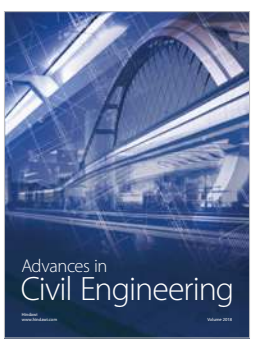
[18] A. Sarangi, S. Priyadarshini, and S. K. Sarangi, "A MLP equalizer trained by variable step size firefly algorithm for channel equalization," in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pp. 1–5, Delhi, India, July 2017.

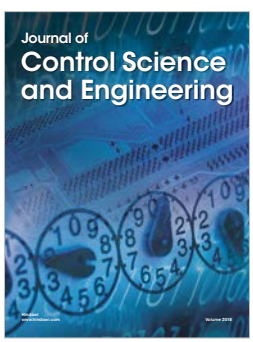
[19] K. N. Kaipa and D. Ghose, "Glowworm swarm optimization: algorithm development," in *Glowworm Swarm Optimization*, pp. 21–56, Springer, 2017.

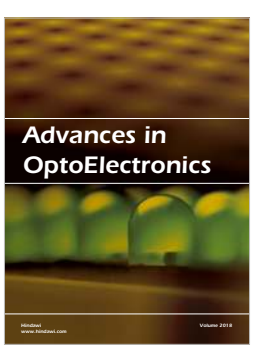


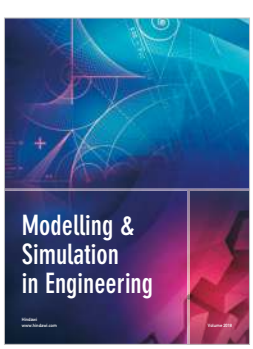


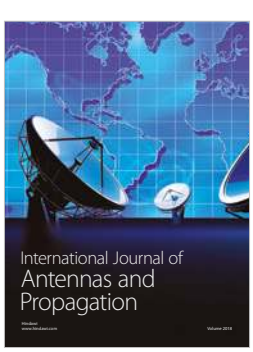


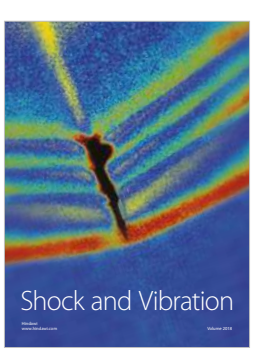


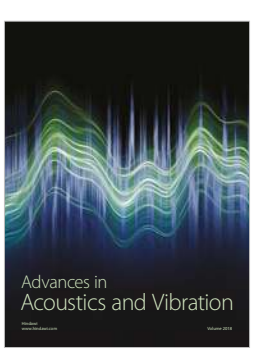












Hindawi

Submit your manuscripts at
www.hindawi.com