Wireless Sensor Node Localization Algorithm Based on Particle Swarm Optimization and Quantum Neural Network

https://doi.org/10.3991/ijoe.v14i10.9314

Yulong Liu(✉), Xiaoming Yu, Yuhua Hao
Yancheng Institute of Technology, Yancheng, China
yulongliu@ycit.cn

Abstract—Aiming at the problem of node localization in wireless sensor networks, a location algorithm for optimizing distance vector hopping (DV-hop) by constructing a quantum neural network model based on particle swarm optimization (PSO) is proposed. According to the average distance obtained by the traditional DV-HOP and the distance from the measured nodes, the quantum neural network model is constructed, and the average distance is trained by the particle swarm optimization algorithm which would shorten the training time of the traditional artificial neural network and accelerate the convergence speed. By using the proposed model, the optimal mean value is obtained, and the optimization of the DV-HOP algorithm is realized. The simulation results show that compared with the traditional DV-HOP algorithm, the proposed algorithm can reduce the positioning error by about 20%, and the positioning accuracy is significantly improved.

Keywords—wireless sensor networks, quantum neural network, particle swarm optimization, DV-HOP algorithm

1 Introduction

In recent years, the research results of wireless sensor networks (WSNs) have been widely used in environmental detection, target tracking, military and other fields [1]. With the deep integration of WSNs theory research and practical application [2-4], the current research on WSNs is mainly in three aspects: the energy consumption, node deployment and node location of network nodes. The research of node location is the premise of sensor networks in many fields [5].

In the positioning process of WSNs, the localization algorithm is divided into range-based and range-free according to whether the node position needs to be measured [6]. Actually, there are many typical range-free localization algorithms, such as Centroid algorithm, approximate triangle interior point-in-triangulation test algorithm [7], distance vector-hop (DV-HOP) algorithm [8], Amorphous algorithm and so on. Their common advantages are simple implementation, no additional hardware, strong viability of network and low cost. So they are often used in the positioning of WSNs.

What is worth mentioning here is that we mainly focus on range-free localization algorithm, especially DV-HOP. The traditional DV-HOP algorithm mainly relies on the estimation of the average hop distance in the process of WSNs node localization. The misbehavior nodes in the network will result in the unreasonable hops information obtained, so that there is a large error in the average single-hop distance obtained from the hop information. As a result, the accuracy and efficiency of WSNs node localization decrease sharply, which makes the positioning process is fuzzy and uncertain.

Therefore, we use particle swarm optimization (PSO) algorithm as the learning mechanism of quantum neural network (QNN) [9], and propose a PSO-QNN localization algorithm for WSNs. By combining the two algorithms, a node localization model with short learning time, fast convergence speed and high precision is constructed, and after training, quantum neural network obtains the best initialization weights and thresholds. As a result, when the model is applied to DV-HOP algorithm, a better average hop distance is obtained, thus improving the positioning accuracy of nodes. Besides, simulation results show that the algorithm has greatly improved the positioning accuracy and efficiency.

2 DV-HOP Algorithm

Figure 1 is a concrete indication of the principle of hop distance.

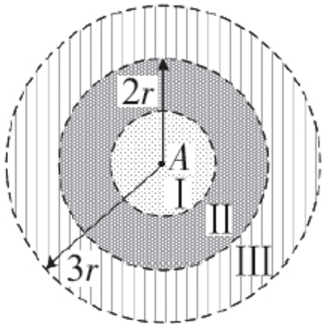

Fig. 1. The principle of hop distance

In Figure 1, *A* is the anchor node, and the communication radius is *r*, the I, II and III regions are the first, second and third hop restricted regions of anchor node *A*.

Effectively, the DV-HOP algorithm [10] can be divided into three stages: 1) The anchor node broadcasts the location information and hops in the network to the whole WSNs. The initial value of hops is 0, and the minimum hop value of each anchor node is recorded by the location node, and the hop value is increased by 1 and forwarded to the neighbor node. At the moment, all nodes get the minimum hops of their own to the anchor nodes through the distance vector exchange protocol; 2) The actual average distance between each anchor node and the other anchor nodes is calculated by each anchor node's position information and the distance with the other anchor nodes, which are recorded by the stage *A*.

$$\overline{d_{DV-HOP}} = \frac{\sum_{i \neq j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum_{i \neq j} h_{ij}}, i \neq j \tag{1}$$

Where $(x_i, y_i)(x_j, y_j)$ are the position coordinates of node i and node j respectively. h_{ij} is the number of hops between anchor node i and anchor node j. Then, the average hop distance as the correction value is broadcast throughout the whole WSNs. When the first correction value is received, the pending node will save it and then no longer receive other correction values. According to the correction value, convert the minimum hop count between the node and anchor node to the distance of them; 3) When the location node gets three or more than three anchor nodes, the nodes to be located can be obtained by three or multilateral measurements.

3 DV-HOP Algorithm Based on PSO-QNN

3.1 Quantum neural network

The back propagation (BP) neural network consists of three layers: input layer, hidden layer and output layer. In BP model, input layer LA, output layer LC and hidden layer LB are composed of m, n and p nodes respectively. Moreover, the nodes of the same layer are not connected, and the adjacent layer nodes are all interconnected.

The output of the hidden layer LB of traditional BP neural network can be expressed as follows:

$$b_r = f(W^T X - \theta), r = 1,2,3,\cdots, p \tag{2}$$

The output of the output layer LC can be expressed as follows:

$$c_j = f(V^T B - \varphi), j = 1,2,3,\cdots, n \tag{3}$$

where the corresponding relationship f is function $Sigmoid$ (it is also called S growth curve), that is, $f(x) = (1 + e^{-x})$; X is the input of the neural network, the neuron of the input layer is represented by a_i, in the same way, the hidden layer is b_r, and the output layer is c_j. Besides, W_{ir} is the connection weight between the input layer neuron and the hidden layer neuron; V_{rj} is the connection weight between the hidden layer neuron and the output layer neuron; θ is the threshold of the hidden layer, and φ is the threshold of the output layer.

For quantum neural network model [11-14], the multiple Sigmoid functions are superimposed, and a hidden layer neuron can represent a number of orders of magnitude, thus constructing an incentive function of a sandwich-like structure, it makes the fuzziness of the network increase substantially. Now, the hidden layer output can be expressed as follows:

$$b_r = \frac{1}{n_s}\sum_{r=1}^{n_s} f[\alpha_m(W^T X - \theta^\gamma)], r = 1,2,3,\cdots, p \tag{4}$$

Where n_s is the number of quantum intervals, which is the same as the number of faults. Meanwhile, α_m is a steepness factor, θ^r is a quantum interval.

The gradient neural network algorithm is still used in the training algorithm of layered excitation function. In the training process, the updated contents include the connection weights between different layers of neurons and the quantum spacing between neurons in the hidden layer.

Here, the updating of connection weights is no different from that of BP neural network. And the quantum interval algorithm for hidden layer neurons is updated.

For the pattern class vector C_m (m is the number of pattern class, that is, the number of neurons in the input layer), so the output variables of the j neurons in the hidden layer can be expressed as follows:

$$\sigma_{j,m}{}^2 = \sum_{x_k \in C_m} (\langle \mathrm{O}_{j,m} \rangle - O_{j,k})^2 \tag{5}$$

$$\langle \mathrm{O}_{j,m} \rangle = \frac{1}{|\mathrm{C}_m|}\sum_{x_k \in C_m} O_{j,k} \tag{6}$$

where $O_{j,k}$ is the output of the j neurons in the hidden layer when the input vector is x_k.

By minimizing the total class variance δ_j^2, the renewal equation of the hidden layer quantum interval θ_j^r can be obtained, and the γ quantum interval of the j neurons in the hidden layer can be expressed as follows:

$$\Delta\theta_j^\gamma = \eta \frac{\alpha_m}{n_s}\sum_{m=1}^{n_s}\sum_{x_k \in C_m} (\bar{h}_{jm} - \bar{h}_{jk})(\bar{v}_{jm}^{\gamma} - \bar{v}_{jk}^{\gamma}) \tag{7}$$

Where η is the learning rate, and α_m is the steepness factor.

3.2 Particle swarm optimization algorithm

For particle swarm optimization algorithm, suppose that there is a n dimension target search space composed of m particles, then the position of the p particle in d dimension space is

$$\mathrm{L}_p = \lfloor l_{p1}, l_{p2}, l_{p3}, \cdots, l_{pd} \rfloor, p = 1,2,\cdots, m \tag{8}$$

Flight speed can be expressed as follows:

$$\mathrm{V}_p = \lfloor v_{p1}, v_{p2}, v_{p3}, \cdots, v_{pd} \rfloor, p = 1,2,\cdots, m \tag{9}$$

The fitness value is expressed as follows:

$$\text{fitness}_p = f(L_p) \tag{10}$$

The algorithm has two extremes: the best solution $pbest$ found by each particle itself; the best location $gbest$ found by the whole particle swarm. The position and velocity of particle p in the d dimensional space are updated according to the following iterative equation:

$$\mathrm{V}_{\mathrm{pd}}^{\mathrm{k+1}} = V_{pd}^{k} + c_1 \times rand() \times (pbest_{pd}^{k} - L_{pd}^{k}) + c_2 \times \mathrm{rand}() \times (gbest_{pd}^{k} - L_{pd}^{k}) \tag{11}$$

$$L_{pd}^{k+1} = L_{pd}^{k} + V_{pd}^{k+1} \tag{12}$$

where V_{pd}^{k} represents the flight speed of the first k iteration of the p particle in the d dimension space; then L_{pd}^{k} is the position of the p particle in the d dimensional space for the k iteration; $rand()$ is a random number between [0, 1]; c_1 and c_2 are both learning factors. The particle search will stop at the best location or the maximum number of iterations. The PSO algorithm flow is shown in Figure 2.

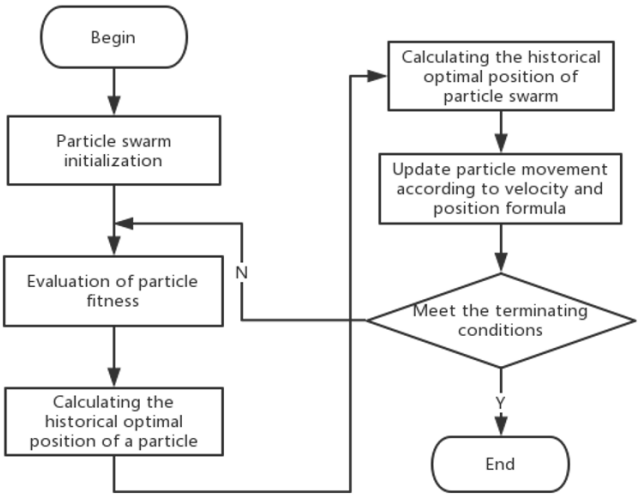

Fig. 2. Flow chart of particle swarm optimization algorithm

3.3 Algorithm implementation steps

Assuming that the number of anchor nodes is *P*, and the number of nodes to be positioned is *M*. Then the whole WSNs node localization steps are as follows:

1. Establish the PSO-QNN empirical model and use the prearranged anchor node to perceive the environment. Through two sets of data (the actual distance $D\{d_{ij}, \mathrm{i} \neq \mathrm{j}\}$ between the anchor node and other anchor nodes and the average distance $D^*\{d_{ij}^*, \mathrm{i} \neq \mathrm{j}\}$ between the anchor node and other anchor nodes), PSO can train QNN and optimize the initial weight and quantum interval of QNN.

2. PSO algorithm has a very strong searching ability, which can obtain the global optimal solution for each particle.
3. After establishing the model, the anchor nodes and the nodes to be located are distributed in WSNs. Each anchor node regularly broadcasts its own related parameters in the WSNs, including the node coordinates and the parameters of the quantum neural network. All nodes get the minimum hop count to the anchor node through the distance vector exchange protocol.
4. According to the location information and hop count of each anchor node recorded to other anchor nodes, the average distance per hop is calculated by Eq. (1).
5. The average hop distance $\overline{d_{DV-HOP}}$ obtained in step (4) is used as the input of the PSO-QNN model, then the optimized distance value $\overline{d_{PSO-QNN}}, \mathrm{i} = 1,2,3, \ldots q$ can be obtained at the output of the model. q is the number of selected anchor nodes.
6. According to the average hop distance $d_{PSO-QNN}$, it will be broadcast as a correction value in the whole WSNs.
7. When the first correction value is received and saved by the location node, it is forwarded to the adjacent node, and the location node calculates the hop distance to each anchor node according to the corrected value and the recorded hops information.
8. When the location node gets three or more than three anchor nodes, the nodes to be located can be obtained by three or multilateral measurements.

3.4 Algorithm specific description

The initialization of PSO algorithm is the process of finding the optimal solution through a continuous population iteration of a group of randomly distributed particles. By comparing the optimal solution of the new generation with the previous generation, the optimal value is updated, such as Eq. (11) and Eq. (12). Matlab is used to build the simulation environment of the logarithmic - constant distribution model. According to the measured node distance $D\{d_{ij}, \mathrm{i} \neq \mathrm{j}\}$ and the average distance $D^*\{d_{ij}^*, \mathrm{i} \neq \mathrm{j}\}$, the quantum neural network model is constructed based on the traditional DV-HOP algorithm. In this way, the average distance D^* is trained by the PSO algorithm, thus the better average value is obtained.

First of all, we should select corresponding data to learn and train QNN, so as to form a practical model applied to WSNs positioning. We trained the parameters of QNN through PSO. The parameters for QNN to be trained are the $m \times p$ parameter θ and the p parameter λ contained in hidden layer nodes, then the $n \times p$ parameter θ and the n parameter λ of the output node nodes, where m, p and n are the number of nodes in input layer, hidden layer and output layer, respectively. Besides, θ is the weight between the neurons, and the threshold is λ. The connection weights between neurons are mapped to the positions of the particles in the PSO algorithm, then the dimension of the particle swarm is $m \times p + n \times p + p + n$. After initializing the particle swarm, the iterative process is carried out according to the algorithm steps, until the position of the particle with the minimum mean square error (MSE) is produced. At this time, the dimension values of the particle position are the optimal initial weights and thresholds.

Assuming that the number of training samples is *N*, and the MSE of the network is expressed as follows:

$$MSE = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{p}(O_{ij}(X_i)-t_{ij})^2 \tag{13}$$

Where the t_{ij} in the formula is the expected output value of sample i on the j output terminal.

The coordinates of the nodes to be positioned in the three side or the multilateral measurement method are (x, y) . The coordinates of the anchor nodes are $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_k, y_k)$, in this way, the distance from the location node to the anchor node can be expressed as $d_1, d_2, d_3, \ldots, d_k$, respectively.

$$\begin{cases} d_1 = \sqrt{(x_1-x)^2+(y_1-y)^2} \\ d_2 = \sqrt{(x_2-x)^2+(y_2-y)^2} \\ \vdots \\ d_k = \sqrt{(x_k-x)^2+(y_k-y)^2} \end{cases} \tag{14}$$

Especially, let

$$A = -2\begin{pmatrix} (x_1-x_k) & (y_1-y_k) \\ (x_2-x_k) & (y_2-y_k) \\ \vdots & \vdots \\ (x_{k-1}-x_k) & (y_{k-1}-y_k) \end{pmatrix}$$

$$B = \begin{pmatrix} d_1^2-d_k^2-x_1^2+x_k^2-y_1^2+y_k^2 \\ d_2^2-d_k^2-x_2^2+x_k^2-y_2^2+y_k^2 \\ \vdots \\ d_{k-1}^2-d_k^2-x_{k-1}^2+x_k^2-y_{k-1}^2+y_k^2 \end{pmatrix}$$

Here, $X = \begin{pmatrix} x \\ y \end{pmatrix}$, according to Eq. (14), it satisfies the relationship $AX = B$. By simplification, the following relationship can be obtained

$$X = (A^T A)^{-1} A^T B \tag{15}$$

Accordingly, the position coordinates of the nodes to be located can be obtained by the Eq. (15).

4 Simulation Results

The experiment is carried out in Matlab7.0 simulation environment, which is specifically configured as Intel (R) Core (TM) i3-2370M CPU, 2.40GHz, and 4GB memory

platform. In the simulation experiment, a node location detection area of $100m \times 100m$ is constructed. More specifically, the communication radius of the node r is set to 25m, which contains 150 sensor nodes and is deployed randomly. In the PSO algorithm, the number of initialization of the particle swarm m is set to 40, the maximum iteration number Q is set to100, c_1 and c_2 are the learning factor, their value both are 1.8. Additionally, the anchor node and the location node are distributed as Figure 3.

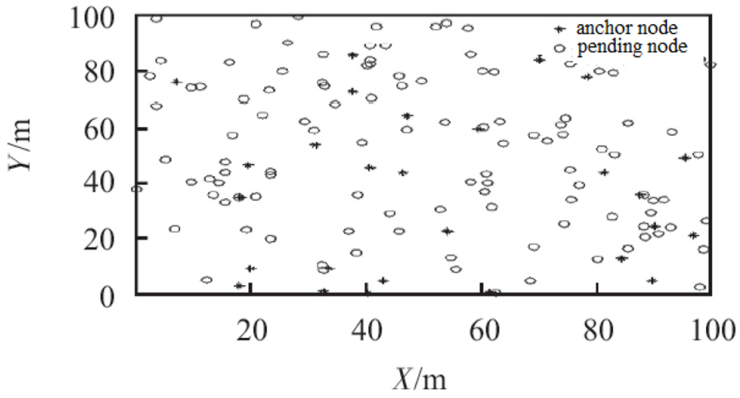

Fig. 3. Node distribution

4.1 The number of anchor nodes' influence on positioning error

The nodes are located according to the implementation steps of the proposed algorithm, and then the average positioning error is obtained based on the simulation results [15-16]. The average location error is defined as

$$e = \frac{\sum_{i=1}^{M} \sqrt{(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2}}{Mr} \times 100\% \tag{16}$$

From Figure 4, we can see that in the same environment, the location error of the algorithm is much lower than that of the traditional DV-HOP algorithm. Additionally, the weighted centroid algorithm is selected as the contrast algorithm. It can be seen that the average positioning error of the three algorithms decreases with the increase of anchor nodes. As a consequence, the average location error of the proposed algorithm is about 26%, in comparison, the weighted centroid algorithm is about 38% and the traditional DV-HOP algorithm is about 48%. Thus this algorithm reduces the average positioning error by about 20% compared with the traditional DV-HOP algorithm.

Due to the large error in the process of hopping to distance, the algorithm requires the nodes to be positioned should have a high density in the implementation process. The simulation results are shown in Figure 5, with the increase of the density of the nodes to be located, the duration of the 3 algorithms is reduced. Accordingly, the time required in this algorithm is reduced by about 100ms compared with the traditional DV-HOP algorithm.

Information can be transmitted directly between nodes by using DV-HOP algorithm. Actually, connectivity mainly reflects the relationship between the number of anchor nodes and the radius of communication. In view of that, the experiment compares the positioning error by analyzing the change of connectivity. From Figure 6, we can see that with the increase of connectivity, the average positioning error of the three algorithms is decreasing in general. Therefore, under the same connectivity, the positioning error of the proposed algorithm is the smallest.

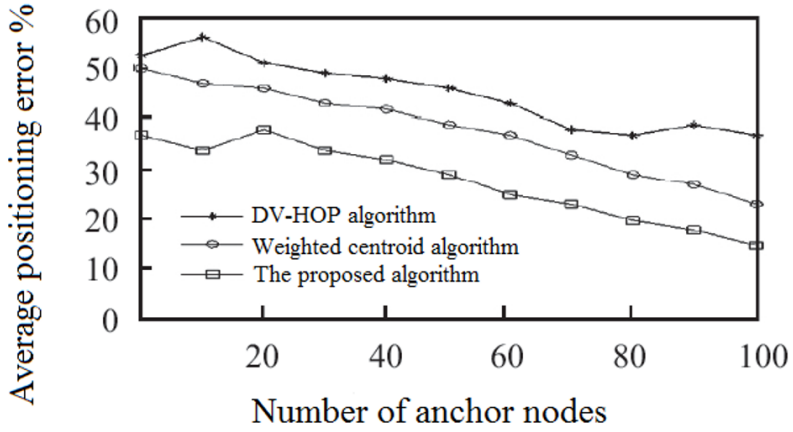

Fig. 4. Comparison of positioning error

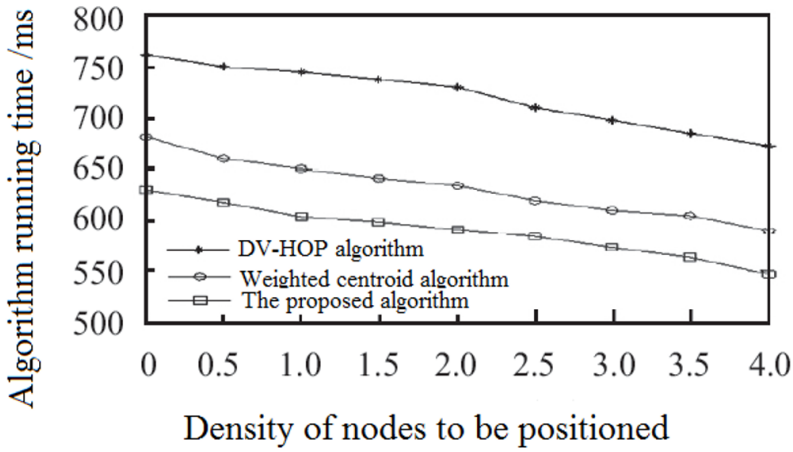

Fig. 5. Influence of node density on algorithm running time

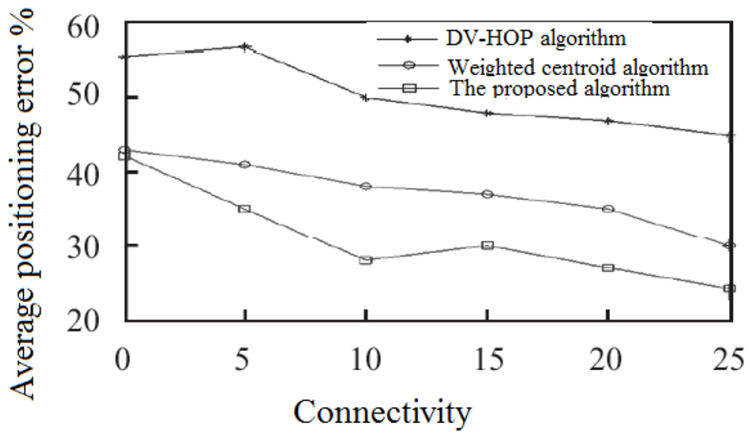

Fig. 6. The influence of different connectivity on the average positioning error

5 Conclusions

In view of the problem that DV-HOP algorithm is easily affected by the external environment factors during the location of WSNs nodes, PSO-QNN algorithm is proposed. Compared with the traditional BP neural network [17], QNN accelerates the convergence speed of the algorithm, and more importantly, it reduces its fuzziness. Furthermore, by using the PSO algorithm as the learning algorithm of QNN, we constructed the PSO-QNN network model, and the DV-HOP algorithm is further improved. The data obtained are analyzed through simulation experiments from three aspects, and the results show that the positioning accuracy of this algorithm has been greatly improved, up to about 20%, which proves the effectiveness of the algorithm in the node location.

6 References

[1] Fu, L.H. (2010). Method to Decrease Noise in Speech Based on Improved Quantum Neural Networks. Information and Control, 39(4): 466-471.

[2] Gai, H.C., Zhang, X.F., Jiang, Z.T. (2010). Face recognition technology based On Quantum Neural Networks. Computer Engineering and Applications 2010; 46(8):187-189.

[3] Li, P., Liao, B., Luo, J., Wu, D. (2008). Localization Method in Mobile Wireless Sen-sor Networks. Journal of Chinese Computer Systems, 29(11): 2051-2054.

[4] Litvintseva, L.V., Ul'Yanov, S.V. (2009). Intelligent control systems. I. Quantum computing and self-organization algorithm. Journal of Computer & Systems Sciences International, 48(6):946-984. https://doi.org/10.1134/S1064230709060112

[5] Ma, Z., Sun, Y., Mei, T. (2004). Survey On wireless sensors network. Journal of China Institute of Communication, 25(4): 114-124.

[6] Paula, T., Bernardos, A.M., Casar, J.R. (2011). Weighted Least Squares Techniques for Improved Received Signal Strength Based Localization. Sensors, 11(9): 8569-8592. https://doi.org/10.3390/s110908569

[7] Shi, H., Wang, W. (2010). Game theory for wireless sensor networks: A survey. Sen-sors, 12(7): 9055-9097. https://doi.org/10.3390/s120709055
[8] Shi, W.R., Jia, C.J., Liang, H.H. (2011). An Improved DV-Hop Localization Algorithm for Wireless Sensor Networks. Chinese Journal of Sensors and Actuators, 24(1): 83-87.
[9] Shi, X.W., Zhang, H.Q., Deng, G.H. (2012). Research on Indoor Location Algorithm based on Distance-loss Model Using Back Propagation Neural Network. Computer Measurement & Control, 20(7): 1944-1947.
[10] Sun, G., Bin, S. (2017). Router-level internet topology evolution model based on mul-ti-subnet composited complex network model. Journal of Internet Technology, 18(6): 1275-1283.
[11] Sun, G., Bin, S. (2018). A new opinion leaders detecting algorithm in multi-relationship online social networks. Multimedia Tools & Applications, 77(4): 4295–4307. https://doi.org/10.1007/s11042-017-4766-y
[12] Tang, W., Zhou, L. (2015). An improved APIT localization algorithm based on trian-gle-circumcircle cover. Chinese Journal of Sensors & Actuators, 28 (1): 121-125.
[13] Tao, Z.Y., Wei, Q., Liu, Y. (2014). Improved DV-Hop localization algorithm based on more power auxiliary anchor nodes. Computer Engineering and Applications, 50(21): 121-124.
[14] Wang, Y., Shi, H. (2012). Improved DV-Hop Localization Algorithm for Wireless Sensor Network. Computer Engineering, 38(7):66-69.
[15] Zhang, P., Mao, W., Xu, J. (2015). A secure ire localization algorithm for WSNs based on iterative vote. Transducer and Microsystem Technology, 34(12):118-120.
[16] Zhang, Y.P., Chen, L., Hao, H. (2013). An Improved Training Algorithm for Quantum Neural Networks. Journal of Electronics & Information technology, 35(7):1630-1635. https://doi.org/10.3724/SP.J.1146.2012.01417
[17] Wang, T.C., Xie, Y.Z., Yan, H. (2016). Research of multi sensor information fusion technology based on extension neural network. Mathematical Modelling of Engineer-ing Problems, 3(3): 129-134. https://doi.org/10.18280/mmep.030303

7 Authors

Yulong Liu has a Master's degree and is an associate professor at the School of Mathematics and Physics, Yancheng Institute of Technology, Yancheng 224051, China, he is mainly engaged in teaching basic physics and researching the properties of nanomaterials.

Xiaoming Yu is a PhD student and is an associate professor at the School of Mathematics and Physics, Yancheng Institute of Technology, Yancheng 224051, China, he is mainly engaged in basic physics teaching and computational physics and other research work.

Yuhua Hao is an associate professor at the School of Mathematics and Physics, Yancheng Institute of Technology, Yancheng 224051, China, he is mainly engaged in basic physics teaching and applied physics research.

Article submitted 29 July 2018. Resubmitted 07 September 2018. Final acceptance 10 October 2018. Final version published as submitted by the authors.