International Journal of Future Generation Communication and Networking
Vol. 9, No. 4 (2016), pp. 179-190
http://dx.doi.org/10.14257/ijfgcn.2016.9.4.15

A Node Positioning Algorithm in Wireless Sensor Networks Based on Improved Particle Swarm Optimization

Sun Shunyuan[1a,b], Yu Quan[1a,b] and Xu Baoguo[1a,b]

[1.a]School of Internet of Things Engineering
[b]Key Laboratory of Advanced Process Control for Light Industry, Jiangnan University, Wuxi Jiangsu 214122, China
yantaissy@gmail.com, 54robin@163.com, Bg.xu@gmail.com

Abstract

The estimation error of the least square method in traditional Distance Vector-Hop (DV-Hop) algorithm is too large and the Particle Swarm Optimization (PSO) algorithm is easy to trap into local optimum. In order to overcome the problems, a fusion algorithm of improved particle swarm algorithm and DV-Hop algorithm was presented. Firstly, PSO algorithm was improved from aspects of particle velocity, inertia weight, learning strategy and variation, which enhanced the ability to jump out of local optimum of the algorithm and increased the search speed of the algorithm in later iterative stage. Then, the node localization result was optimized by using the improved PSO algorithm in the third stage of the DV-Hop algorithm. The simulation results show that compared with the traditional DV-Hop algorithm, the improved DV-Hop based on chaotic PSO algorithm and the DV-Hop algorithm based on improved PSO, the proposed algorithm has higher positioning accuracy and better stability, which is suitable for high positioning accuracy and stability requirements scenes.

Key words: *Wireless Sensor Network (WSN); Particle Swarm Optimization (PSO) algorithm; Distance Vector-Hop (DV-Hop) algorithm; inertia weight; variation*

1. Introduction

Location is important support technology to wireless sensor networks, and is widely applied [1]. Domestic and foreign scholars have conducted a lot of research on node localization issues of wireless sensor network, and it mainly includes range-based and range-free localization algorithm [2]. Range-free localization algorithm includes centroid algorithm [3], DV-Hop algorithm [4], Approximate Point-In-Ttriangulation test algorithm (APIT) [5] and Amorphous algorithm [6]. Due to its advantages of the hardware requirements, network deployment cost, energy consumption and other aspects, range-free localization algorithm is more suitable for Wireless Sensor Networks [7]. The Vector-Hop Distance (DV-Hop) algorithm, which transforms distance measurement between nodes to the product of hop count and average hop distance, is one of the most widely studied algorithms [8].

DV-Hop algorithm can be divided into three stages, in the first and second stage of DV-Hop algorithm, the distance between the unknown node and the anchor node is obtained by the hop count and the average hop distance between nodes, then in the third stage the location of the unknown node is estimated by the distance between nodes. There are two ideas to improve the traditional DV-Hop algorithm: the first one is to make more accurate estimation of the distance between nodes in the first and second stage; The second is to make node position estimation error smaller in the third stage. This paper adopt the second ideal, which means paying attention to the large error problem of the least square method in the third stage. To solve this problem, many scholars have proposed to use particle swarm algorithm (Particle Swarm Optimization, PSO) to replace the least squares method to estimate the node location. For example: document [9] proposed DV-Hop algorithm improved by particle swarm optimization, which replace the least squares method with the standard particle swarm optimization, although positioning accuracy was improved, but it was not significantly; Document [10] proposed a node self-positioning with improved particle swarm algorithm based on chaos disturbance, which use the Tent map chaotic to search, it reduced the probability of falling into local optimum and improved search algorithm accuracy, but still showed low positioning accuracy; Document [11] proposed a self-localization method based on improved particle swarm optimization, in which the inertia weight and learning factor is set to linear decline, and using the method of component variation to escape from local optima when encountering search stagnation, also this method improved Positioning performance, but still not ideal and added a certain computational complexity; Document [12] proposed node localization algorithm based on an improved particle swarm optimization, the historical global best position was mutated one by one component when encountering searching stagnation, and unknown nodes that has been positioned was taken as the anchor node to do iterative search, although positioning accuracy was improved, but there is still some room for improvement, and also added a certain communication overhead and computational complexity. This paper presents an improved particle swarm algorithm and DV-Hop fusion algorithm, an effective solution to the big least squares estimation error problem in the third stage of DV-Hop algorithm, compared with the Above-mentioned algorithms, this algorithm shows higher positioning accuracy and better stability; What's more, this algorithm only improves arithmetic operations in the third stage of DV-Hop algorithm, and the first and second stage are the same with the conventional DV-Hop algorithm, therefore no communication overhead and hardware costs are added.

2. Theory

2. 1. Description of Poisition Problem

DV-Hop algorithm can be divided into three stages[13], in the first and second stages of DV-Hop algorithm, the distance $d_1,d_2,\ldots,d_n$ between the unknown node o(x,y) and the anchor node $A_1(x_1,y_1),A_2(x_2,y_2),\ldots,A_n(x_n,y_n)$ is obtained by the hop count and the average hop distance between nodes, the ranging error is$\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_n$, the estimated coordinates (x,y) satisfies the following inequalities:

$$\begin{cases} d_1^2 - \varepsilon_1^2 \le (x - x_1)^2 + (y - y_1)^2 \le d_1^2 + \varepsilon_1^2 \\ d_2^2 - \varepsilon_2^2 \le (x - x_2)^2 + (y - y_2)^2 \le d_2^2 + \varepsilon_2^2 \\ \vdots \\ d_n^2 - \varepsilon_n^2 \le (x - x_n)^2 + (y - y_n)^2 \le d_n^2 + \varepsilon_n^2 \end{cases} \tag{1}$$

Positioning problem is transformed into finding coordinates (x,y) which minimize $f(x,y)$ of formula (2), and minimum $f(x,y)$ guarantees minimum total error. Therefore, the third stage of DV-Hop is converted into solving constrained optimization problem, finding $f(x,y)$ is a nonlinear optimization problem [14].

$$f(x,y) = \sum_{i=1}^{M} \left| \sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i \right| \tag{2}$$

Where M is the number of anchor nodes.

2.2. Design of Fitness Function

Fitness function is used to evaluate the merits of the particle position and guide the direction of the search algorithm, the calculation is as follows:

$$fitness_i = \frac{1}{M}\sum_{j=1}^{M}\left|\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - d_j\right| \tag{3}$$

Where: *fitness*$_i$ is the fitness value of particle *i*, (x_i,y_i) is the position coordinates of the particles *i*, (x_i,y_i) is the location coordinates of the anchor node *j*, d_j is the distance between unknown node to the anchor node *j*.

3. Improved Particle Swarm Optimization

3.1. Standard Particle Swarm Optimization

PSO algorithm is a global optimization algorithm (PSO) based on natural population search strategy [11], it has advantages of being easy to carry out, less parameters, fast search speed, etc. and is widely used in scientific research and practical engineering. Particle Swarm can be specifically described as follows: Suppose the number of the particle population is N, the dimension of search space is D, the *i*-th particle position is $x_i=(x_{i1},x_{i2},\ldots,x_{iD})$, velocity is v $v_i=(v_{i1},v_{i2},\ldots,v_{iD})$, best historical position of particle *i* is $pbest_i=(p_{i1},p_{i2},\ldots,p_{iD})$, global best historical position is $gbest=(p_{g1},,p_{g2},\ldots,p_{gD})$. Velocity and position update formulas of particle *i* are as follows:

$$\boldsymbol{v}_i(t+1) = \omega \boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{pbest}_i - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{gbest} - \boldsymbol{x}_i(t)) \tag{4}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{5}$$

where: $i=1,2,\ldots,N$, *w* is the inertia weight; *t* is the current iteration time; c_1,c_2 are learning factors; r_1,r_2 are random numbers uniformly distributed in [0,1].

In PSO, the particles tend to aggregate to the global best position and itself's best position, then the population will have convergence effect and premature convergence, then search stagnates in later iterative stage, it is difficult to escape from local optimum, resulting in low search accuracy of PSO algorithm. To optimize the performance of particle swarm algorithm, this paper respectively improve 4 aspects, speed, inertia weight, learning strategy and variability, to improve search precision and stability of the algorithm.

3.2. Improvement of Speed

In earlier iterative stage, the particles can be easily guided by the global best position to premature aggregation; Search speed becomes slow in later iterative stage and the algorithm can not jump out of local optimum. To solve this problem, this paper uses a high-quality solution, which is beneficial to global search of the initial iteration and local convergence of the later iteration, to replace the global best position, it can be described as the following: randomly select *k* particles from the particle population and compare the best positions of *k* particles, then select the its own best position instead of the global best position to guide the particle motion. The value of *k* increases linearly with iteration: smaller value of *k* makes high-quality solution worse, which does well to global search and increases the diversity of the population; the greater value of *k* makes better high-quality, which does well to local convergence and improve the convergence rate. Improved speed update formula is as follows:

$$\boldsymbol{v}_i(t+1) = \omega \boldsymbol{v}_i(t) + c_1 r_1 (\boldsymbol{pbest}_i - \boldsymbol{x}_i(t)) + c_2 r_2 (\boldsymbol{pbest}_m - \boldsymbol{x}_i(t)) \tag{6}$$

$$k = k_{\min} + (k_{\max} - k_{\min}) \frac{t}{T} \tag{7}$$

where: *pbest*$_m$ is the best of the *k* particles' own best history positions; *t* is the current iteration time; *T* is the maximum time of iteration; $k_{\max}$ and $k_{\min}$ are the maximum and minimum values of *k*, $k_{\max}$ value takes population of particles , $k_{\min}$ value takes 1.

3.3. Inertia Weight Improvements

In order to balance global search and local development capabilities of Particle Swarm Optimization, inertia weight takes a larger value in initial iteration to benefit global search, and it takes a larger value in later iterative stage to benefit local development, so Shi *et al.*, [15] proposed a method of linearly decreasing inertia weight, The formulas are calculated as follows:

$$w = w_{\max} - (w_{\max} - w_{\min}) \frac{t}{T} \tag{8}$$

where: t is the current iteration time; T is the maximum time of iterations; wmax and wmin are the maximum and minimum of the inertia weight w.

Larger inertia weight in the initial iteration does good to global search, but high speed is easy to exceed the optimal position, as a result produces shock, increases communication overhead and decreases search efficiency; In later iterative stage, Smaller inertia weight does good to local development, but it is not easy to escape from local optimum algorithms and lower the search precision. To solve this problem, a linearly decreasing stochastic volatility strategy, whose volatility decreases with iterations, is proposed: In the early iteration, highly fluctuate inertia weight improves search efficiency of the algorithm; in later iterative stage, low fluctuate inertia weight enhances the ability to escape from local optimum. w is calculated as follows:

$$w = w_{\max} - (w_{\max} - w_{\min}) \frac{t}{T} + \sigma \times randn \tag{9}$$

$$\sigma = \frac{1}{2} \times e^{(-\frac{t}{T})} \tag{10}$$

where: *randn* is a random number of Gaussian distribution with mean 0 and variance 1; *t* is the current iteration time; *T* is the maximum time of iteration; $w_{\max}$ and $w_{\min}$ are the maximum and minimum values of inertia weight *w*, After a lot of simulation experiments, $w_{\max}$ taking 0. 9 and $w_{\min}$ taking 0. 4 guarantee better location.

3.4. Learning Strategies

In order to increase the diversity of particle population and enhance ability to escape from local optimum, learning strategy which select by probability is proposed: In the early iterations, learning strategy helps to improve the search efficiency and speed of convergence of the algorithm; In later iterations, learning strategy is conducive to escape from local optimum and improve the quality of the candidates. Due to the different merits of particle position vectors' each dimension the learning strategy to be selected is different, so each dimension uses independent learning methods.

Particle *i* learning process pseudo-code is as follows:

// xi is the position vector of the particle *i*, *xx* is an intermediate vector, *D* is the dimension of the search space, $f(x)$ is the fitness function, $\max(x_{:,j})$ and $\min(x_{:,j})$ are the the maximum and minimum of particle population's j-dimensional component, $\text{mean}(x_{:,j})$ is the average of *j*-th dimensional component of particle population, and the average takes reciprocal of the fitness value as weight.

```
For $j=1:D$
    $xx=x_i$;
     If $rand<P_{1,j}$
       $xx_j=2\times gbest_j - xx_j$;            // learning strategy 1
     Else If $rand<P_{1,j}+P_{2,j}$
           $xx_j=2\times pbest_{i,j} - xx_j$;    // learning strategy 2
        Else If $rand< P_{1,j}+P_{2,j}+P_{3,j}$
              $xx_j = \max(x_{:,j})+\min(x_{:,j}) - xx_j$;
                                                   // learning strategy 3
              else
                 $xx_j=2\times \text{mean}(x_{:,j}) - xx_j$;
                                                   // learning strategy 4
     End If
     If $f(xx)< f(x_i)$
       $x_i=xx$;
       $f(x_i)=f(xx)$;
    End If
End For
```

Formulas of $\text{mean}(x_{:,j})$, $P_{1,j}$~$P_{4,j}$ in the Pseudo-code are as follows:

$$\boldsymbol{mean}\left(x_{:,j}\right)=\frac{\sum_{i=1}^{N}\frac{1}{f(\boldsymbol{x}_i)}\times \boldsymbol{x}_{i,j}}{\sum_{i=1}^{N}\frac{1}{f(\boldsymbol{x}_i)}} \tag{11}$$

Where: $f(x_i)$ is the fitness value of particle *i*; $x_{i,j}$ is the *j*-dimensional component in position vector of particle *i*; *N* is the number of particles in the population.

$$P_{k,j}^{t}=\begin{cases}0.25 & ,t\le 10\\ \dfrac{u_j(t-10:t-1)}{n_j(t-10:t-1)} & ,t>10\end{cases} \tag{12}$$

where: $k=1,2,3,4$, t is the current iteration number; $P_{k,j}^{t}$ is the using probability of learning strategy *k* for *j*-dimensional particle component of particle *i* in the *t*-th iteration; $n_j(t\text{-}10:t\text{-}1)$ is the number of times of *j*-dimensional particle component of particle *i* using learning strategy *k* in *t*-10 to *t*-1 iteration; $u_j(t\text{-}10:t\text{-}1)$ is the number of times of *j*-dimensional component of particle i finding a more optimal solution by learning strategy *k* in *t*-10 to *t*-1 iteration.

3.5. Mutation

To overcome the PSO algorithm's easily to fall into local optimum, low capacity of local development and other shortcomings in the later stage, disturbances to current position can both speed up the convergence and improve the accuracy of the optimal solution. Each dimension of the current position is not the best, so any dimension can be chosen to be mutated. Lévy distribution has infinite variance [16], which consists of frequent small value and aleatory large value, frequent small value is conducive to the latter part of local development, and aleatory large value can help to escape from local optima, so a Lévy distribution step is used to mutate the current best value, the mutation formula is as follows:

$$\boldsymbol{gbest}_m = \boldsymbol{gbest}_m + a \times s \tag{13}$$

Where: m is a random number in 1~D; D is the dimension of the search space; a is the step size factor, whose value is 0.1; s is step that is subject to Lévy distribution.

4. Algorithmic Process

This algorithm process is as follows:

Step 1 Randomly deploy a number of sensor nodes in the target area, then obtain the distance between unknown node to the anchor node by the first and second stage of DV-Hop algorithm.

Step 2 set the parameters. Including the maximum number of times of iterations $T_{\max}$, learning factor c_1,c_2, maximum $w_{\max}$ and minimum $w_{\min}$ of inertia weight, the maximum $k_{\max}$ and minimum $k_{\min}$ number of particles.

Step 3 Initialize population. The initial position of particle i is $x_i=(x_{i1},x_{i2}...,x_{iD})$, speed is $v_i=(v_{i1},v_{i2},…,v_{iD})$, its own best position is $pbest_i=x_i$, Evaluate swarm fitness value of particle according to equation (3), the global best historical position $gbest$ is best initial position of particle population, so that t=0.

Step 4 Set t=t+1, update particle position x and speed v according to formula (5) to (10), and carry out cross-border treatment on x, v.

Step 5 Implement learning strategies. Implement learning strategies for particles in accordance with pseudo-code in section 2.4 and equation (11), (12).

Step 6 Update particles' own best position and the global best position.

Step 7 Mutation. Mutate the optimal location $gbest$ according to equation (13), update $gbest$ if the mutated position is better.

Step 8 Put out global History optimal solution if the iteration termination condition is meet, then skip to step 9; otherwise, return to step 4 and continue the iterative optimization.

Step 9 Repeat step 3-8 until the estimated location of all the unknown nodes are put out.

5. Simulation and Analysis

5.1. Simulation Environment and Parameters

Simulate in Matlab R2010b platform in order to demonstrate the positioning performance of the improved particle swarm algorithm. Referred to the improved particle swarm optimization algorithm which replaces the least squares method in document [10] and [11] as MPSO1, MPSO2, referred to DV-Hop algorithm in the third stage of MPSO1 algorithm, MPSO2 algorithm as MPSO1-DV-Hop, MPSO2-DV-Hop. Compare the traditional DV-Hop algorithm, MPSO1-DV-Hop algorithm, MPSO2-DV-Hop algorithm with the proposed algorithm, to avoid different design on fitness function affecting positioning, fitness functions of MPSO1, MPSO2 algorithm and the proposed algorithm are designed by the formula (3). Set the parameters of the algorithm: Learning factor $c1$=$c2$=1.4945, the number of particle population was 30, the maximum number of iterations is 100 and the maximum speed of the particle is 10 m/s, then adjust the ratio of anchor nodes, total number of nodes, communication radius respectively to compare the performance of four location algorithms. In order to reduce the interference of random error in experiments, all the experimental results are average of 30 experiments. Use uniform localization error [17] to evaluate algorithm performance, uniform localization error is:

$$error = \frac{\sum_{k=1}^{K}\sum_{i=1}^{N}\sqrt{(x_i - x_{ki}^*)^2 + (y_i - y_{ki}^*)^2}}{N \times R \times K} \tag{14}$$

Where: (x_i,y_i) is the actual coordinates of the unknown node i; (x_{ki}^*,y_{ki}^*) coordinates of the k-th estimation of the unknown node i; N is the number of unknown nodes; R is the communication radius; K is the number of experiments.

5.2. Anchor Node Ratio's Effect on the Positioning Accuracy

Randomly distribute 100 sensor nodes in the 100m×100m square area, with communication radius 30m. By adjusting the ratio of anchor nodes, compare and analyze the proposed algorithm with the traditional DV-Hop count, MPSO1-DV-Hop algorithm and MPSO2-DV-Hop algorithm. As shown in Figure 1, with the increase in the proportion of anchor nodes, four algorithms positioning error Deviations are decrease, and the proposed algorithm shows biggest drop in positioning errors. Compared with DV-Hop algorithm, MPSO1-DV-Hop algorithm and MPSO2-DV-Hop algorithm, the positioning accuracy of the proposed algorithm averagely improves 15. 75%, 8. 43% and 5. 84%.

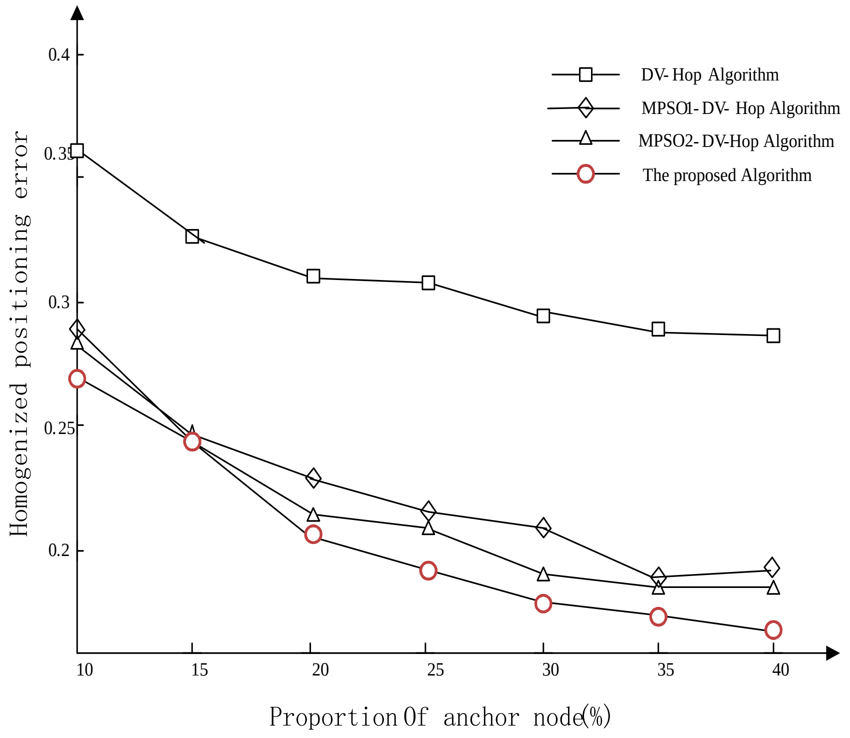

Figure 1. Anchor Node Ratio's Effect on the Positioning Accuracy

5.3. Number of Nodes' Effect on the Positioning Accuracy

Randomly distribute 100 sensor nodes in the 100m×100m area, with communication radius 30m and the anchor node ratio of 30%, by adjusting the number of nodes, compare the positioning performance of four algorithms, as shown in Figure 2. The positioning error of the proposed algorithm is always less than the traditional DV-Hop algorithm, MPSO1-DV-Hop algorithm and MPSO2-DV-Hop algorithm. In comparison, the average positioning accuracy of the proposed method increases by 14. 44 %, 13. 65% and 8.02%.

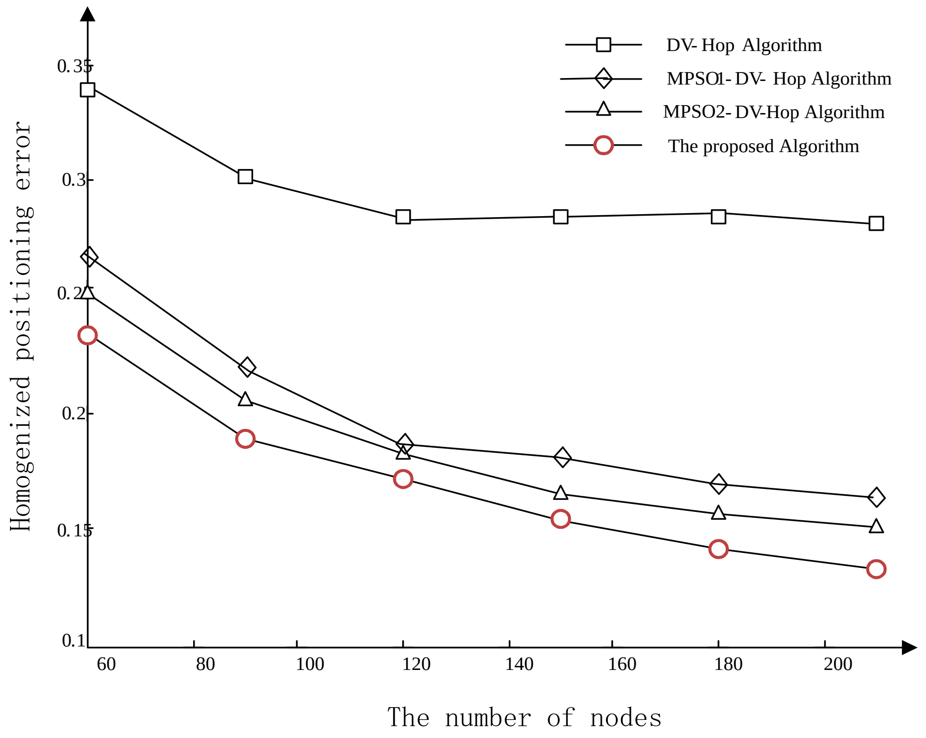

Figure 2. Number of Nodes' Effect on the Positioning Accuracy

5.4. Communication Radius' Effect on the Positioning Accuracy

Randomly distribute 100 sensor nodes in the 100m×100m area, wherein the number of anchor nodes is 30, with communication radius of 10m~45m, compare the Location performance of four algorithms by adjusting the communication radius, as shown in "Figure 3". With the increase of communication radius, network connectivity increases and ranging error reduces, thereby the positioning accuracy of the algorithm improved; When the communication radius is greater than 45m, with the increase of communication radius, the average jump distance error and the distance error increase, so the curve shows an upward trend.

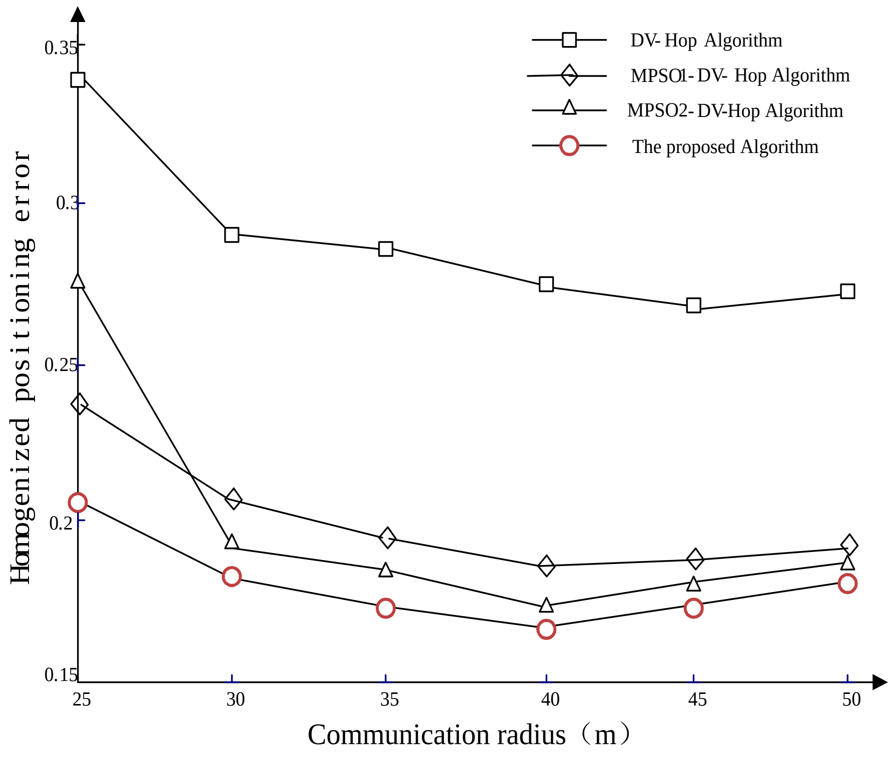

Figure 3. Communication Radius' Effect on the Positioning Accuracy

Randomly distribute 100 sensor nodes in the 100m*100m area, wherein the number of anchor nodes is 30, with communication radius of 30m. Respectively implement 30 simulation tests on traditional DV-Hop algorithm, MPSO1-Hop algorithm, MPSO2-DV-Hop algorithm and the proposed algorithm, to obtain the mean and standard deviation of the average positioning difference of 30 experiments, and the average running time of locating each unknown node, as shown in "Table 1". The mean and standard deviation of the uniform localization error of 30 experiments are shown in Table 1.

Table 1. Stability Analysis

Algorithm	Average	Standard deviation
DV-Hop Algorithm	0.2934	0.0250
MPSO1-DV-Hop Algorithm	0.2041	0.0291
MPSO2-DV-Hop Algorithm	0.1938	0.0151
The Proposed Algorithm	0.1818	0.0136

As can be seen from Table 1, compared to the traditional DV-Hop algorithm, MPSO1-DVHop algorithm and MPSO2-DV-Hop algorithm, this algorithm always maintains high-precision positioning and better stability. global search and local development of particle swarm algorithm hnve different efficiency to find a high quality solution for each iteration, and location efficiency and population-related particles and iterations, The efficiency is related to particle location and the number of iterations, but it is not a strictly linear relationship, learning strategies selection based on probability is to select a global search or local development by the feedback information prior to the current iteration, so the algorithm has stronger adaptive ability, thus has higher search efficiency and better stability.

6. Conclusion

In order to improve the performance of traditional DV-Hop localization algorithm, PSO will replace the least squares method to estimate the node location, but the positioning effect is poor. This paper proposes a fusion algorithm of improved particle swarm algorithm and DV-Hop. In the case of maintaining hardware costs and constant communication overhead, it has better positioning accuracy and stability, to some extent, the problem of unsatisfactory performance of PSO's DV-Hop algorithm is solved. The improved algorithm has improved speed, inertia weight, learning strategies and calculate variation, although the the computational load increases a little, the operation did not increase the number of cycles, the time complexity is in the same order of magnitude with the original algorithm, thus the improved algorithm has the computation time with conventional DV-Hop algorithm and does not affect its application in WSN positioning. The latter studies should focus on reducing the amount of calculation and energy consumption.

Acknowledgment

The authors thank the support from the Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education) and the Fundamental Research Funds for the Central Universities, No. JUSRP11560.

References

[1] X. Li, Y. Xu and F. Ren, "Techniques for wireless sensor network", Beijing: Beijing Institute of Technology Press, **(2007)**, pp. 191-192.
[2] J. A. Costa, N. Patwari and A. O. Hero, III, "Distributed weighted multidimensional scaling for node localization in sensor networks", ACM Transactions on Sensor Networks, vol. 2, no. 1, **(2006)**, pp. 39-64.
[3] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less low cost outdoor localization for very small devices", IEEE Personal Communications Magazine, vol. 7, no. 5, **(2008)**, pp. 28-34.
[4] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks", Journal of Telecommunication Systems, vol. 22, no. 1-4, **(2003)**, pp. 267-280.
[5] R. Nagpal, H. Shrobe and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network", The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)[C]. Palo Alto., **(2003)**, pp. 1-16.
[6] Z. Ma, Y. Liu and B. Shen, "Distributed locating algorithm for wireless sensor networks-MDS-MAP(D)", Journal on Communications, vol. 29, no. 6, **(2008)**, pp. 58-62.
[7] W. Yang and W. Pan, "DV-Hop localization algorithm based on RSSI ratio correction in wireless sensor network", Transducer and Microsystem Technologies, vol. 32, no. 7, **(2013)**, pp. 26-135.
[8] J. Wen, X. Fan and X. Wu, "DV-Hop localization algorithm based on the RSSI hop correction", Chinese Journal Of Sensors And Actuators, vol. 27, no. 1, **(2014)**, pp. 113-117.
[9] X. Chen, M. Liao M and J. Lin, "Improvement of node localization in wireless sensor network based on particle swarm optimization", Journal of Computer Applications, vol. 30, no. 7, **(2010)**, pp. 1736-1738.
[10] Z. Liu, Z. Liu and X. Tang, "Node self-localization algorithm based on modified particle swarm optimization", Journal of Central South University (Science and Technology), vol. 43, no. 4, **(2012)**, pp. 1371-1275.
[11] L. Li and Y. Du, "Application of Modified Particle Swarm Optimization in Node Locating of Wireless Sensors Networks", Computer Applications and Software, vol. 31, no. 4, **(2014)**, pp. 69-72.
[12] Y. Wang and J. Yang, "Localization in wireless sensor network based on improved particle swarm optimization algorithm", Computer Engineering and Applications, vol. 50, no. 18, **(2014)**, pp. 99-102.
[13] S.-S. Wang, K.-P. Shih and C.-Y. Chang, "Distributed direction-based localization in wireless sensor networks", Computer Communications, vol. 30, no. 6, **(2007)**, pp. 1424-1439.
[14] Y. Liu, X. LV and X. Wang, "Application of hybrid genetic algorithm in WSNs localization", Transducer and Microsystem Technologies, vol. 33, no. 2, pp. 150-153.
[15] Y. Shi and R. Eberhart, "A modified particle swarm optimizer", Proceedings of IEEE International Conference on Evolutionary Computation, **(1998)**, pp. 69-73.
[16] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing", Journal of Computational Physics, vol. 226, no. 2, **(2007)**, pp. 1830-1844.
[17] A. Zhang, X. Ye and H. Hu, "Improved DV-Hop localization algorithm based on RSSI of every jump and jump distance correction", Journal of Instrument, vol. 33, no. 11, **(2012)**, pp. 2552-2559.

Authors

Shunyuan Sun, author is an associate professor of Jiangnan University, Wuxi Jiangsu 214122, China. (E-mail: yantaissy@gmail.com). His main research interests include localization, channel estimation and equalization, detection, cross-layer design and throughput analysis for wireless sensor networks.