Hindawi
Journal of Sensors
Volume 2021, Article ID 6670061, 12 pages
https://doi.org/10.1155/2021/6670061

Research Article

Monte Carlo Node Localization Based on Improved QUARTE Optimization

Ling Song,[1,2] **Xiaoyu Jiang**,[1] **Liying Wang**,[1] **and Xiaochun Hu**[3]

[1]*School of Computer and Electronic Information, Guangxi University, Nanning 530004, China*
[2]*Guangxi Key Laboratory of Multimedia Communications and Network Technology, Nanning 530004, China*
[3]*School of Information and Statistics, Guangxi University of Finance and Economics, Nanning 530007, China*

Correspondence should be addressed to Ling Song; 731486203@qq.com

Received 17 December 2020; Revised 14 March 2021; Accepted 18 March 2021; Published 7 April 2021

Academic Editor: Xavier Vilanova

Wireless sensor network (WSN) is a research hot spot of scholars in recent years, in which node localization technology is one of the key technologies in the field of wireless sensor network. At present, there are more researches on static node localization, but relatively few on mobile node localization. The Monte Carlo mobile node localization algorithm utilizes the mobility of nodes to overcome the impact of node velocity on positioning accuracy. However, there are still several problems: first, the demand for anchor nodes is large, which makes the positioning cost too high; second, the sampling efficiency is low, and it is easy to fall into the infinite loop of sampling and filtering; and third, the positioning accuracy and positioning coverage are not high. In order to solve the above three problems, this paper proposes a Monte Carlo node location algorithm based on improved QUasi-Affine TRansformation Evolutionary (QUATRE) optimization. The algorithm firstly selects the high-quality common nodes in the range of one hop of unknown nodes as temporary anchor nodes, and takes the temporary anchor nodes and anchor nodes as the reference nodes for positioning, so as to construct a more accurate sampling area; then, the improved QUATRE optimization algorithm is used to obtain the estimated location of unknown nodes in the sampling area. Simulation experiments show that the Monte Carlo node positioning algorithm based on the improved QUATRE optimization has higher positioning accuracy and positioning coverage, especially when the number of anchor nodes is relatively small.

1. Introduction

Wireless sensor network (WSN) has become a cutting-edge technology in many fields of today's life, such as smart home [1], health care [2], precision agriculture [3], animal tracking, forest fire detection, early landslide detection, and earthquake detection. WSN is a multihop self-organizing wireless communication system composed of multiple sensor nodes. At present, the research on wireless sensor network mainly focuses on the optimization of wireless sensor network [4, 5] and the node localization of wireless sensor. The sensor node localization is a key technology in WSN. The location information of nodes in WSN is very important. For example, in precision agriculture, the information collected by sensors includes temperature, humidity, and ultraviolet intensity; if the location information of nodes is not clear, the collected data is not meaningful. In order to determine the location of sensor nodes, many positioning algorithms have been proposed, including the centroid positioning algorithm, the DV-HOP positioning algorithm and its improved algorithm, and the MDS-MAP positioning algorithm and its improved algorithm. For example, in Reference [6], an improved localization algorithm named SIC-DV-Hop is proposed. The algorithm considers reduction of average distance error, selection of anchors, and solving inconsistency issue in centroid methods to enhance accuracy. In Reference [7], the authors propose a cluster-based MDS localization algorithm called MDS-MAP (CH) for WSN with energy holes. Simulation results show that the improved MDS-MAP (CH) localization algorithm has achieved higher localization accuracy, better balanced energy consumption, and stronger network robustness. These algorithms are for static wireless sensor network node localization scheme and do not consider the mobility of sensor nodes. However, in many cases, sensor

nodes are mobile, such as wildlife tracking and underground miners positioning. In the above various application requirements, mobile WSN positioning is becoming more and more important and valuable.

The early mobile WSN node localization is realized by the periodic calculation of the static node localization algorithm, but this kind of algorithm does not make use of the mobility of nodes and has a large amount of calculation and high communication cost [8–10]. The Monte Carlo node localization algorithm uses the mobility of nodes, combined with particle filter method, can effectively locate mobile WSN nodes. In a certain range, the positioning accuracy will improve with the increase of speed. However, the localization algorithm also has the problems of too many anchor nodes, low sampling efficiency, low positioning accuracy, and location coverage. To solve the above problems, this paper proposes an improved Monte Carlo node localization algorithm based on temporary anchor nodes.

The main contributions of this work are summarized as follows:

(1) We propose a temporary anchor node, which uses both the temporary anchor node and the anchor node as a positioning reference node to construct the sampling area, which reduces the demand for anchor nodes in the entire network and reduces the cost of the entire network

(2) When the anchor node and the temporary anchor node are used to construct the sampling area, the correction radius is used to correct the position of the temporary anchor node, so that the position of the temporary anchor node is more accurate, and the sampling area is more accurate

(3) Aiming at the shortcomings of the QUasi-Affine TRansformation Evolutionary (QUATRE) optimization algorithm that the search speed is slow and easy to fall into the local optimum, an improved QUATRE (IQUATRE) optimization algorithm is proposed. This algorithm combines reverse learning to generate more uniform initial particles, and then divides the particles into two groups and uses different evolution strategies to find the optimal particle faster and more accurately

(4) After constructing the sampling area, use the IQUATRE optimization algorithm to find the optimal particles. This algorithm greatly improves the positioning accuracy and positioning coverage of nodes

The rest of the paper is organized as follows: In Section 2, the current research status of the Monte Carlo node positioning algorithm and the basic principles of the Monte Carlo algorithm and the QUATRE algorithm are presented; in Section 3, the Monte Carlo based on the improved QUATRE optimization proposed in this paper is introduced; in Section 4, the simulation results are given and compared with other similar algorithms; finally, Section 5 gives the conclusion.

2. Related Work

2.1. Research Status of Monte Carlo Node Localization Algorithm. For the situation where unknown nodes move and anchor nodes are not restricted, there are mainly two positioning methods, one is to apply the static positioning algorithm to dynamic network, periodically calculate the location of unknown node, such algorithm has high communication cost, large amount of calculation, and the positioning accuracy will decrease with the increase of the maximum speed of node movement; the other is using the Monte Carlo method. The location of unknown nodes is calculated periodically. This kind of algorithm makes use of the mobility of nodes. In a certain range, the positioning accuracy will be improved with the increase of the maximum speed of nodes. Therefore, this paper takes this kind of algorithm as the research focus. Hu and Evans [11] proposed a localization algorithm for mobile sensor networks, which is called the Monte Carlo node localization (MCL) algorithm. The core point of MCL is to use N discrete samples to estimate the posterior probability and iteratively update using importance sampling. The algorithm makes full use of the mobility of nodes to improve the accuracy, and can achieve better results in the case of limited storage space, but it requires a large number of samples and a longer positioning time, and the positioning accuracy is limited. Baggio and Langendoen proposed the Monte Carlo box (MCB) location algorithm [12] in 2006. The biggest difference between it and the MCL algorithm is the introduction of the concept of anchor box and sample box, which improves the sampling efficiency of the algorithm. Therefore, the algorithm improves the efficiency and energy consumption of positioning calculation. But it does not improve the positioning accuracy in essence, because the filtering conditions of MCB and MCL are the same. The MSL$*$ algorithm proposed in Reference [13] introduces common one-hop and two-hop nodes and anchor nodes to colocate. This algorithm is suitable for static and dynamic networks, and also improves the positioning accuracy. However, the disadvantages are obvious, the communication consumption is very large, and it cannot be used in actual positioning effectively. Li et al. [14] proposed the MCL based on fuzzy theory in 2013. The fuzzy processing of the signal energy value makes the filtering conditions accurate, makes up for the shortcomings of the traditional MCL, and shortens the positioning time. Juqin et al. [15] introduced dead reckoning method on the basis of MCL and proposed an improved adaptive Monte Carlo method. Compared with the MCL algorithm, the improved MCL algorithm can still ensure higher positioning accuracy when the number of sampling particles is small. But in the process of positioning, additional hardware is needed to provide data such as heading angle, which increases additional expenses. In Reference [16], Haoshan et al. proposed a Monte Carlo node localization algorithm based on time sequence. The algorithm uses the sequence of feedback signals from neighbor anchor nodes within one-hop range to form the sampling area. The sampling area is further reduced, and the positioning accuracy is improved. However, this algorithm only aims at the situation that anchor nodes are very dense, and it is

difficult to achieve the positioning conditions when the anchor nodes are sparse. The combination of the intelligent algorithm with less parameters and simple calculation with the Monte Carlo node localization algorithm has also become a popular improvement method. Lu and Wang [17] proposed a Monte Carlo mobile node localization algorithm based on Newton interpolation in the literature, which uses inherited Newton interpolation. The mobile node inherits the historical trajectory prediction mechanism to estimate the current motion speed and direction of the mobile node, and uses the particle filter optimized by weight to prevent particle collection loss. The algorithm effectively avoids the degradation of particles and improves the positioning accuracy. However, the localization coverage of the algorithm needs to be improved. In Reference [18], the differential evolution algorithm and the Monte Carlo node localization algorithm are combined to improve the positioning accuracy. However, the use of hardware equipment to measure the distance leads to large energy consumption of the whole network and reduces the life cycle of the network. Although there are many algorithms to improve the MCL algorithm, there are still some problems in this algorithm, which has a certain improvement space.

2.2. Monte Carlo Node Localization Algorithm. The Monte Carlo localization (MCL) algorithm was first used in robot localization [19]. Then in 2004, it was first used in wireless sensor networks by Hu et al. The core of MCL is to use N discrete samples to estimate posterior probability, and importance sampling is used to update iteratively. The algorithm makes full use of the mobility of the nodes to improve the positioning accuracy. In a certain range, the greater the speed of the nodes, the higher the positioning accuracy.

2.2.1. Main Steps of MCL Algorithm. We assume that time is divided into discrete time units. Because the node is mobile, it needs to be relocated in each time unit. Mobile location problem can be expressed in the form of state space: Let t be the discrete time, L_t be the position distribution of nodes at time t, and O_t be the observation value of anchor nodes received from $t-1$ to t time. The transition equation $P(l_t \mid l_{t-1})$ describes that the prediction of the current position of the node is based on the previous position, and the observation equation $P(l_t \mid O_t)$ describes the observation of the anchor node based on the node position. N samples L_t are used to represent the distribution of possible positions of nodes. Our algorithm calculates the sample set of each time step recursively. Since L_{t-1} reflects all previous observations, we use L_{t-1} and O_t to calculate L_t. The MCL algorithm is mainly divided into the following steps:

(1) Initialization

Since the initial node does not have the location information of the previous time, N samples $L_0 = \{l_0^1, l_0^2, l_0^3 \cdots\cdots\}$ are obtained in the whole deployment area.

(2) Prediction

In the prediction step, the node starts from the set of possible positions calculated in the previous step L_{t-1} and applies the movement model to each sample to obtain a set of new samples L_t. Let us assume that a node does not know its speed and direction, except that it knows that its speed is less than $v_{\max}$. Therefore, if the previous step l_{t-1}^i is a possible position of the node, then the current possible position is contained in the circular region where l_{t-1}^i is the original center and $v_{\max}$ is the radius. We use $d(l_1, l_2)$ to represent the Euclidean distance between two points l_1 and l_2. If the velocity is uniformly distributed in the interval $[0, v_{\max}]$, the current position probability estimated based on the previous position is uniformly distributed, as shown in the following equation:

$$P(l_t, l_{t-1}) = \begin{cases} \dfrac{1}{\pi v_{\max}^2}, & d(l_t, l_{t-1}) < v_{\max}, \\ 0, & d(l_t, l_{t-1}) \geq v_{\max}. \end{cases} \quad (1)$$

Therefore, the set R calculated in the prediction stage contains a location randomly selected from a circle with a radius $v_{\max}$ around each possible location point in L_{t-1}.

(3) Filtration

Filter out the impossible samples according to the new observation data. Suppose that the communication radius of the node is r, and the common nodes within one hop can forward the information of anchor nodes. Let S be the set of one-hop anchor nodes of unknown node n and T the set of two-hop anchor nodes of n; then, the filtering formula for samples is shown in the following equation:

$$\text{filter}(l) = \forall s \in S, d(l, s) \leq r \wedge \forall s \in T, d(l, s) \leq 2r. \quad (2)$$

The sample filtering formula shows that when the sample point should meet the following conditions, the distance from the one-hop anchor node to the unknown node should be less than r, and the distance from the two-hop anchor node to the unknown node should be less than $2r$. If the condition is satisfied, the sample weight is set to 1; if not, the sample weight is set to 0.

(4) Resampling

After filtering, the unqualified samples can be removed, but the number of samples will be insufficient. Therefore, it is necessary to repeat the prediction and filtering steps until enough samples are obtained or the maximum number of samples is reached. The idea of resampling is to remove samples with small weight and increase samples with heavy weight.

The MCL algorithm uses the importance sampling method, but the conditional variance of importance weight will increase, which will lead to algorithm degradation. Therefore, the algorithm needs to set a reasonable number of effective samples N_{eff}; when the number of effective samples is lower than the fixed threshold $N_{\text{threshold}}$, resampling is necessary.

(5) Position calculation

After obtaining enough samples, the weighted average value of all samples is obtained, which is the estimated position of unknown nodes.

2.2.2. MCL Algorithm Analysis. The MCL algorithm makes use of the mobility of nodes and provides a new idea for mobile node location in WSN. It is advanced and defective.

The advantages of the MCL algorithm are as follows:

(1) The algorithm does not need ranging, there is no additional hardware and no complex calculation, and the algorithm process is simple and easy to implement

(2) The algorithm makes use of the mobility of the nodes and overcomes the obstruction of the node's velocity on the positioning accuracy in the traditional mobile node localization

The disadvantages of the MCL algorithm are as follows:

(1) Because the sampling area is too large, the sampling efficiency is very low and may even fall into the dead cycle of sampling and filtering

(2) Sampling and filtering process is a time-consuming and energy-consuming process, which may not obtain enough samples many times

(3) The localization accuracy of the MCL algorithm depends on the density of anchor nodes. When the density of anchor nodes in WSN is small, the positioning error will be larger

2.3. QUATRE Optimization Algorithm. In recent years, optimization problem has been favored by many scholars. Many optimization algorithms are inspired by nature, such as biology, population, and physics, for example, the genetic algorithm (GA) [20], particle swarm optimization (PSO) [21], differential evolution (DE) algorithm [22], ant colony (ACO) algorithm [23], artificial bee colony (ABC) optimization [24], firefly algorithm (FA) [25], bat algorithm (BA) [26], cat colony optimization (CSO) algorithm [27], and QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm [28]. The optimization process of the natural inspired intelligent algorithm usually generates a group of randomly initialized particles, which are combined, migrated, or evolved in a predefined algorithm. These algorithms are effective methods to get acceptable solutions to complex problems by trial and error in a reasonable time. The QUATRE (QUasi-Affine TRansformation Evolutionary) algorithm is a new optimization algorithm proposed by Meng et al. In 2016, compared with other intelligent algorithms, the QUATRE algorithm has simpler structure and fewer parameters. Moreover, the QUATRE algorithm enhances the cooperation among particles and has strong global search ability. In Reference [28], the QUATRE algorithm is compared with several improved particle swarm optimization algorithms and improved differential evolution algorithm. The results show that the QUATRE algorithm performs well in both single-mode and multimode functional models, and has better global search ability.

2.3.1. QUATRE Algorithm Theory. Since the evolution formula of the QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm is similar to affine transformation in geometry, its name is obtained. Its evolution formula is shown as follows:

$$\widehat{X} = \bar{M} \otimes \widehat{X} + M \otimes B. \tag{3}$$

In Equation (3), $\widehat{X}$ is the individual population matrix, which contains multiple individuals. $\widehat{X} = (X_1, X_2, \cdots, X_{\text{ps}})^{\text{T}}$, where ps is the individual size in the population, M is the coevolutionary matrix, $\bar{M}$ is the incidence matrix of M, B is the evolution guidance matrix, and $\otimes$ is the bitwise multiplication operation of the matrix. If G is the population evolution algebra, then the i-th individual in the G generation population can be expressed as X_{iG}, and the corresponding G generation population can be expressed as $\widehat{X}_G = (X_{1,G}, X_{2,G}, \cdots, X_{\text{ps},G})^{\text{T}}$.

$M$is an automatically generated matrix transformed from the initial matrix M_{init}. For simplicity, M_{init} is a lower triangular matrix with a value of 1. Equation (4) gives the transformation of M matrix from M_{init}.

$$M_{\text{init}} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & & \cdots & \\ 1 & 1 & \cdots & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & & \\ & \cdots & & \\ 1 & 1 & \cdots & 11 \\ & & 1 & \end{bmatrix} = M. \tag{4}$$

Equation (4) is divided into two steps. Firstly, all elements in each row vector are randomly arranged, and then, all the row vectors in the matrix are randomly arranged. Among them, the number of rows ps in the matrix represents the number of individuals in the population, and the dimension D of the row vector represents the dimension of the individual. In general, the population size ps in the optimization algorithm is larger than the individual dimension D. At this time, the line vector needs to be expanded from D to ps. Equation (5) introduces the examples when the population size ps meets the requirements of ps/$d = s$ and ps%$d = k$:

$$M_{\text{init}} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & & \cdots & \\ 1 & 1 & \cdots & 1 \\ 1 & & & \\ 1 & 1 & & \\ & & \cdots & \\ 1 & 1 & \cdots & 1 \\ & & \vdots & \\ 1 & & & \\ 1 & 1 & & \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & & \\ & \cdots & & \\ 1 & 1 & \cdots & 1 \\ & & 1 & \\ & \cdots & & \\ 1 & \cdots & 1 & \\ & 1 & & \\ 1 & & \cdots & 1 \\ & & \vdots & \\ & & 1 & 1 \\ 1 & & \cdots & 1 \end{bmatrix} = M. \tag{5}$$

Table 1: Six kinds of matrix B_{iG} generation strategies in the QUATRE algorithm.

No.	QUATRE/x/y	Equation
1	QUATRE/target/2	$B = X_{i,G} + F \cdot (X_{r1,G} - X_{r2,G}) + F \cdot (X_{r3,G} - X_{r4,G})$
2	QUATRE/best/1	$B_{i,G} = X_{g\text{best},G} + F \cdot (X_{r1,G} - X_{r2,G})$
3	QUATRE/target/1	$B_{i,G} = X_{i,G} + F \cdot (X_{r1,G} - X_{r2,G})$
4	QUATRE/rand/1	$B_{i,G} = X_{r0,G} + F \cdot (X_{r1,G} - X_{r2,G})$
5	QUATRE/target-to-best/2	$B_{i,G} = X_{i,G} + F \cdot \left(X_{g\text{best},G} - X_{i,G}\right) + F \cdot (X_{r1,G} - X_{r2,G})$
6	QUATRE/target-to-rand/1	$B_{i,G} = X_{i,G} + F \cdot (X_{r0,G} - X_{i,G}) + F \cdot (X_{r1,G} - X_{r2,G})$

Initialization: Initially the node has no knowledge of its location N is a constant that denotes the number of samples to maintain L_{t-1}
$A_0 = \{(x, y) | x_{\min} < x < x_{\max}, y_{\min} < y < y_{\max}\}$
$L_0 = \{l_0^i | l_0^i = (x_0^i, y_0^i), 0 \le i \le N\}$
Step 1: Construct sampling area;
$A_t^i = \{(x_{\min}, x_{\max}) ; (y_{\min}, y_{\max})\}$
Step 2: Sampling and Filtering;
while (size(L_t)<N)
$R = \{l_t^i | l_t^i = (x_t^i, y_t^i)\}$
Done

Algorithm 1: Sampling area construction and filtering based on temporary anchor nodes.

In Equation (5), the first $s * D$ is the concatenation of s lower triangular matrices of $D * D$, and the last k is the first k rows of the lower triangular matrix. All the elements in the incidence matrix $\bar{M}$ of M are obtained by the inversion of the elements in M, that is, 1 is replaced by 0 and 0 is replaced by 1.

Each individual has an evolutionary guidance vector B_{iG}. There are several different ways to generate B_{iG}, which are represented by QUATRE/x/y. Different evolution guidance vectors are suitable for different scenarios. Table 1 shows several different strategies for calculating B_{iG}. $x_{g\text{best},G}$ represents the optimal individual of G generation; r_1, r_2, r_3, and r_4 represent the number of individuals in the population.

2.3.2. Basic Flow of QUATRE Algorithm. The basic flow of the QUATRE algorithm is as follows:

(1) Initialize the particle population, in the search space $[X_{\min}, X_{\max}]$, randomly initialize the particle swarm of population size, calculate the fitness value fitness($X_{i,G}$) of the evaluation function of the particles in the population, and mark the optimal solution $X_{g\text{best}}$ in the population and the particle history optimal solution $X_{p\text{best}}$

(2) Update the particle population according to Equation (3)

(3) Calculate the fitness value fitness($X_{i,G}$) of the evaluation function of each particle

(4) Compare the fitness value of the evaluation function of each particle with the fitness value of the historical optimal solution fitness($X_{i,G}$). If it is better, update the particle swarm and set the current particle as the historical optimal solution

(5) Compare the fitness value of the evaluation function of each particle with the global optimal particle fitness value, and if it is better, set the current position as the global optimal

(6) If it is less than the iteration accuracy or reaches the maximum number of iterations, the calculation ends and the result is output. Otherwise, go to step (2)

3. Monte Carlo Node Location Based on Improved QUATRE Optimization

3.1. Sampling Area Construction and Filtering Based on Temporary Anchor Nodes

3.1.1. Algorithm Design. The basic idea is as follows: the algorithm based on the sampling area construction and filtering of temporary anchor nodes proposed in this section is different from the sampling filtering in the RMCL algorithm. It is to select ordinary nodes as temporary anchor nodes and anchor nodes as reference nodes for unknown nodes. The biggest difference between the sampling region construction and filtering algorithm proposed in this section and the RMCL algorithm is that ordinary nodes are selected as temporary anchor nodes and temporary anchor nodes and anchor nodes are used as reference nodes of unknown nodes to calculate the location of unknown nodes. When the number of anchor nodes is relatively small, the precise sampling area can be reduced by using the combined positioning method of temporary anchor nodes and anchor nodes.

The following sections are a detailed description of several important parts of the Algorithm 1, such as the selection of temporary anchor nodes and the construction of sampling areas.

3.1.2. Selection of Temporary Anchor Nodes. Given an unknown node i, the common nodes in the communication range are k, the distance between them is r_{ik}, and n_s is the number of anchor nodes within one hop. Determine the number of temporary anchor nodes according to the number of anchor nodes. According to the distance between the common node and the unknown node, one or several common nodes closest to the unknown node are selected as temporary anchor nodes.

As shown in Figure 1, when $n_s \geq 3$, there is no need to select temporary anchor nodes, and only anchor nodes are used for positioning.

As shown in Figure 2, when $n_s = 2$, at most one common node k within one-hop range closest to the unknown node is selected as the temporary anchor node for positioning.

As shown in Figure 3, when $n_s = 1$, at most two common nodes k within the range of one hop that are closest to the unknown node are selected as temporary anchor nodes for positioning.

As shown in Figure 4, when $n_s = 0$, at most three common nodes k within the one-hop range that are closest to the unknown node are selected as temporary anchor nodes for positioning.

3.1.3. Sampling Area Construction. Use $x_{\min}$; $x_{\max}$; $y_{\min}$; $y_{\max}$ to constrain the sampling area; (x_j, y_j) are the coordinates of anchor node j; (x_k, y_k) are the coordinates of temporary anchor node k; n is the number of anchor nodes that can be heard; r_{ij} is the distance between the unknown node i and the anchor node j; r_{ik} is the distance between the unknown node i and the temporary anchor node k; TAN (temporary anchor node) is the temporary anchor node. First, use anchor nodes and temporary anchor nodes to construct a distance box. On this basis, combine the particle swarm at the previous moment to construct the sampling area. The specific method is as follows: If the number of anchor nodes n is greater than 3, then the anchor node observation value is used for reference positioning. Do not select available temporary anchor nodes. Then, the distance box is shown in the following equations:

$$x_{\min} = \max\left(x_j - r_{ij}\right), \tag{6}$$

$$x_{\max} = \min\left(x_j + r_{ij}\right), \tag{7}$$

$$y_{\min} = \max\left(y_j - r_{ij}\right), \tag{8}$$

$$y_{\max} = \min\left(y_j + r_{ij}\right). \tag{9}$$

If the number of anchor nodes is n less than 3, appropriate temporary anchor nodes and anchor nodes should be selected as reference nodes. Then, the distance box is shown in the following equations:

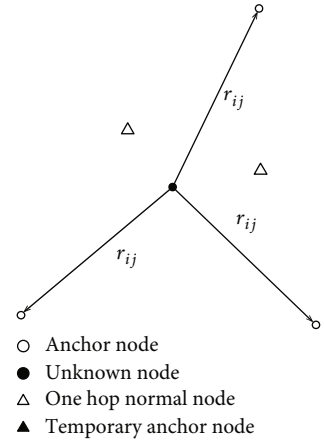

FIGURE 1: The selection of temporary anchor nodes when the number of anchor nodes is $n_s \geq 3$.

$$x_{\min} = \max\left(x_{\min}, x_k - r_{ik}\right), \tag{10}$$

$$x_{\max} = \min\left(x_{\max}, x_k + r_{ik}\right), \tag{11}$$

$$y_{\min} = \max\left(y_{\min}, y_k - r_{ik}\right), \tag{12}$$

$$y_{\max} = \min\left(y_{\max}, y_k + r_{ik}\right). \tag{13}$$

If the sample set at the previous time is empty, the distance box is the sampling area. If the sample set at the previous time is not empty, the sampling area is the movement information of the node added to the distance box, and the sampling area is shown as follows:

$$x_{\min} = \max\left(x_{\min}, x_{t-1}^i - v_{\max}\right), \tag{14}$$

$$x_{\max} = \min\left(x_{\max}, x_{t-1}^i + v_{\max}\right), \tag{15}$$

$$y_{\min} = \max\left(y_{\min}, y_{t-1}^i - v_{\max}\right), \tag{16}$$

$$y_{\max} = \min\left(y_{\max}, y_{t-1}^i + v_{\max}\right). \tag{17}$$

3.1.4. Correction Radius. The coordinates of the temporary anchor node are the position coordinates of the node at the previous moment. If the last time coordinate position information is used without correction, a relatively large error will occur. These errors will continue to accumulate with the iteration of the algorithm. Therefore, we use the correction radius to reduce the adverse effects of the error. The correction radius is set to the maximum movement speed $v_{\max}$ of the node.

As shown in Figure 5, if we do not consider the correction radius, a part of the effective sampling area will be lost. Figure 6 shows the sampling area after increasing the correction radius.

3.2. Improved QUATRE Optimization Algorithm. Although the QUATRE algorithm is advanced in the model, it has better search capabilities than other algorithms, but like other

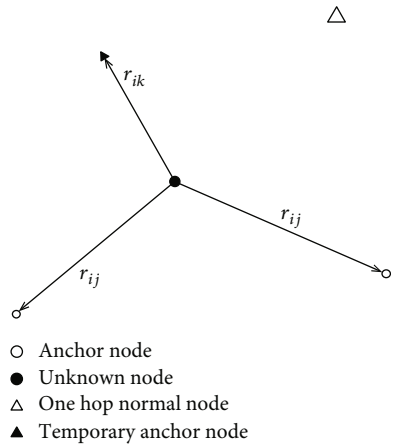

Figure 2: The selection of temporary anchor nodes when the number of anchor nodes is $n_s = 2$.

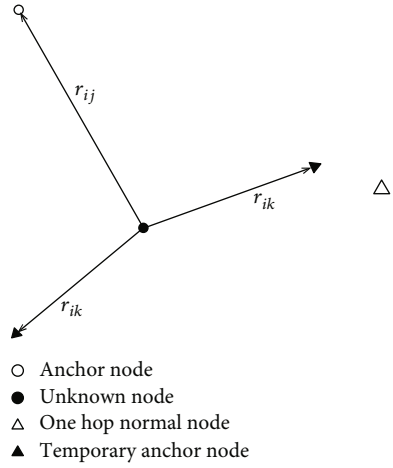

Figure 3: The selection of temporary anchor nodes when the number of anchor nodes is $n_s = 1$.

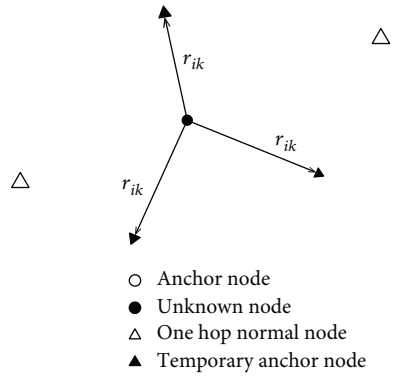

Figure 4: The selection of temporary anchor nodes when the number of anchor nodes is $n_s = 0$.

optimization algorithms, the QUATRE algorithm also has the same defects, such as premature convergence and search stagnation. In response to the above problems, a variant algorithm of the QUATRE algorithm has also been proposed. The C-QUATRE [29] algorithm divides the entire population into two populations at random to improve the performance of the population, so that the inferior population is selected by pairwise competition between the two groups and evolved. For S-QUATRE [30], it divides the entire population into two groups, namely, the superior group and the inferior group using the sorting strategy, and its individual evolution only changes in the inferior group. Both of these two variants use an evolutionary guidance vector to enhance the performance of the QUATRE algorithm, which improves the performance of the algorithm to a certain extent, but there are still certain defects. The initial particles of the QUATRE algorithm and its variants are all randomly generated by sampling, which cannot guarantee the diversity of the initial particles; a single evolutionary guide vector and a fixed scale factor cannot satisfy the entire evolution process of the entire population of particles. Therefore, an improved QUATRE algorithm (IQUATRE) is proposed to improve the QUATRE algorithm from the following three aspects. The basic idea of the improvement is first use the reverse learning strategy to increase the initial population diversity, then select different mutation methods for different populations, and finally propose an adaptive scale factor.

3.2.1. Reverse Learning Strategy Generates Initial Population. The initial population with good diversity can have a good effect on the convergence speed and solution accuracy of the algorithm. In the traditional QUATRE algorithm, the initial population is randomly generated, that is, the range of values is known, and values are randomly selected in the area according to the upper and lower limits of the value range. However, the population diversity of randomly generated methods cannot be guaranteed. In order to improve the diversity of the initial population, this algorithm introduces a reverse learning strategy to generate the initial population. First generate N initial populations, and then generate the reverse population of the initial population according to the reverse strategy, and then calculate the fitness values of the initial population and the reverse population, respectively, and retain individuals with better fitness values into the final initial population. The calculation formula of reverse individual is shown as follows:

$$R_i = A_i + B_i - r_i, \tag{18}$$

where R represents the reverse individual, A and B, respectively, represent the upper and lower limits of the individual, r represents the initial individual, and i represents the dimension of the vector.

3.2.2. Evolutionary Guidance Vector Generation Strategy for Dual Groups. The evolution guide vector B_{iG} of the QUATRE algorithm adopts different mutation schemes and often shows different performance when solving different optimization problems, because different mutation schemes have different evolution capabilities. The "QUATRE/best/1" mutation scheme uses the best individuals discovered so far to guide the search direction, so it has a very fast convergence speed and excellent local search capabilities. But it is easy to fall into the local optimum, leading to premature

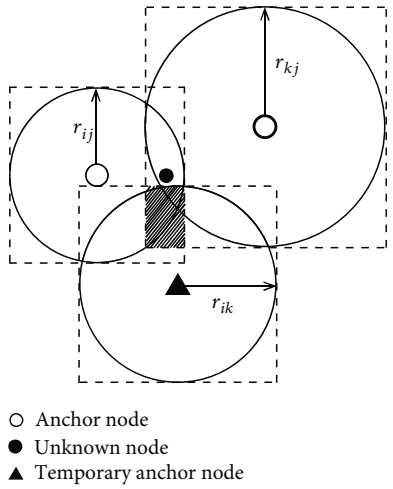

Figure 5: Sampling area constructed without correction radius.

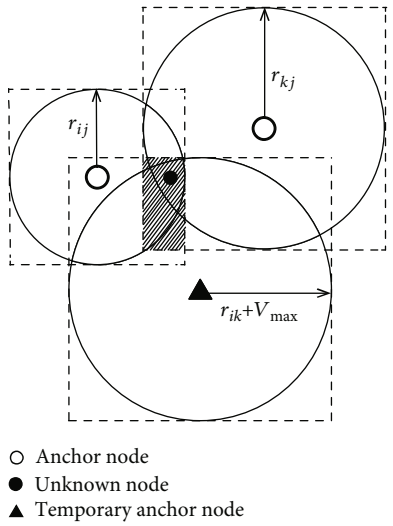

Figure 6: The sampling area constructed by adding a modified radius.

convergence. The mutation scheme "QUATRE/rand/1" is the most commonly used, powerful, and robust mutation scheme. Although the local search capability is insufficient, the exploration capability is relatively strong and the population diversity is maintained. In order to combine the advantages of the two mutation schemes, this chapter uses two different mutation schemes based on the QUATRE algorithm. The details are as follows: First, the initial population is randomly divided into two small populations, the fitness values of the individuals in the two small populations are compared in pairs, and the winners are reclassified into the superior population, and the inferior population is divided into the inferior population. The winning population adopts the mutation method of "QUATRE/best/1" and has relatively strong local search ability. The inferior population adopts the mutation method of "QUATRE/rand/1", which has strong global search ability. Figure 7 shows the basic ideas of the two mutation schemes.

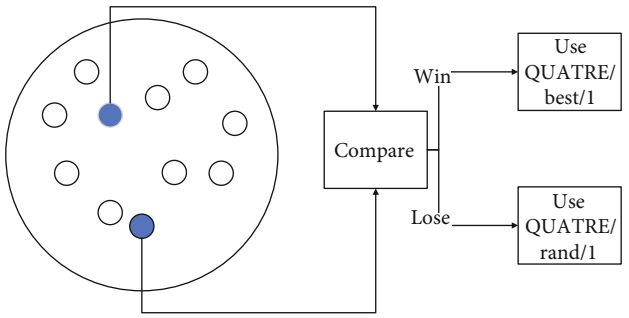

Figure 7: Selection of mutation methods.

3.2.3. Adaptive Scale Factor. Throughout the evolution stage, the scale factor has a great influence on the search ability of the algorithm. When solving problems of different scales, it is difficult to use the same scale factor to adapt to different evolutions. Therefore, this chapter proposes an adaptive scale factor.

In the early stage of the algorithm, in order to avoid the algorithm falling into the local optimum, a larger scale factor is used to make the algorithm perform a comprehensive global search in the solution space. In the later stage, in order to retain the elite solution, the scale factor is reduced to make the algorithm have strong local search capabilities. Improve the optimization accuracy of the algorithm. The update strategy of the g-th generation scale factor is shown in the following equation:

$$F = F_l + (F_u - F_l) \times \frac{f_{\text{bad}}^g - f_{\text{best}}^g}{f_{\text{bad}}^1 - f_{\text{best}}^1}, \tag{19}$$

where F represents the scale factor of the g-th generation, F_l represents the lower limit of the scale factor, F_u represents the upper limit of the scale factor, f_{bad}^g represents the worst value of the fitness value of the g-th generation particle evaluation function, and f_{best}^g represents the g-th generation particle evaluation function. The optimal value of the fitness value f_{bad}^1 represents the worst value of the fitness value of the initialized particle evaluation function, and f_{best}^1 represents the optimal value of the fitness value of the initialized particle evaluation function.

3.2.4. IQUATRE Algorithm Flow. The basic flow of the IQUATRE algorithm is as follows:

(1) Use the reverse learning strategy to generate the initial particle population, calculate the fitness value $\text{fitness}(X_{i,G})$ of the evaluation function of the particles in the population, compare the fitness value of the evaluation function of the particles in pairs, and divide them into superior and inferior populations, and mark the optimal solution $X_{g\text{best}}$ in the population and the particle historical optimal solution $X_{p\text{best}}$

(2) Use different evolution strategies and scale factors for different populations

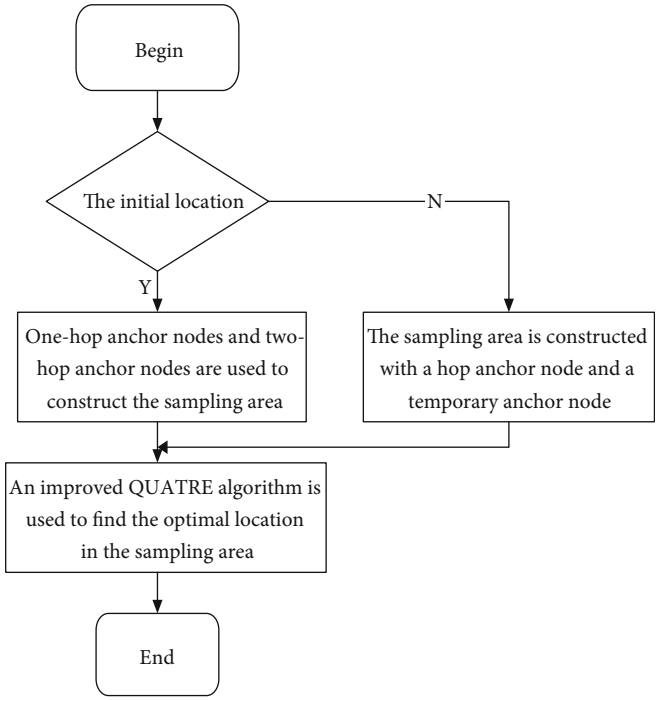

Figure 8: The IQUATRE_RMCL algorithm process diagram.

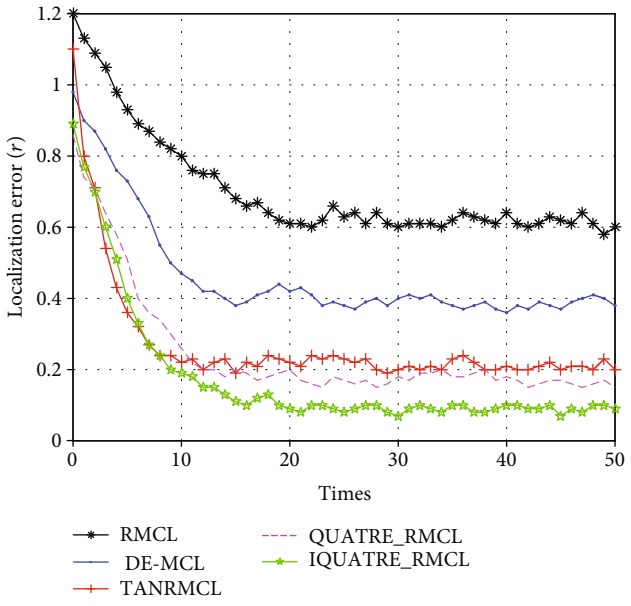

Figure 9: The localization algorithm of different algorithms.

(3) Update the particle population using Equation (3)

(4) Calculate the fitness value fitness$(X_{i,G})$ of the evaluation function of each particle

(5) Compare the fitness value of the evaluation function of each particle with the fitness value of the historical optimal solution fitness$(X_{i,G})$. If it is better, update the particle swarm and set the current particle as the historical optimal solution

(6) Compare the fitness value of the evaluation function of each particle with the global optimal particle fitness. If it is better, set the current position as the global optimal

(7) If it is less than the iteration accuracy or reaches the maximum number of iterations, the calculation ends and the result is output. Otherwise, go to step (2)

3.3. Monte Carlo Node Location Algorithm Based on Improved QUATRE Optimization. The Monte Carlo node location algorithm based on improved QUATRE evolution (IQUATRE_RMCL) uses the IQUATRE evolution algorithm to replace the prediction and filtering steps in the TANRMCL algorithm proposed in the previous chapter, and no longer uses the weighted average of the sampled particles to calculate the position of the unknown node. It uses the IQUATRE algorithm to find the optimal node position in the sampling area.

3.3.1. Fitness Function. The goal of the IQUATRE_RMCL algorithm optimization is to minimize the distance between the predicted position and the anchor node and temporary anchor node within one hop, and the difference between the actual position and the distance between the anchor node and temporary anchor node within one hop. In the iterative process of the IQUATRE algorithm, the individuals with the least fitness are retained. Assuming that d_{ij} is the distance between the unknown node i and the anchor node or temporary anchor node j within one-hop range measured according to RSSI, the coordinates of the unknown node are (x_i, y_i), and the coordinates of the anchor node or temporary anchor node are (x_j, y_j), M represents the total number of

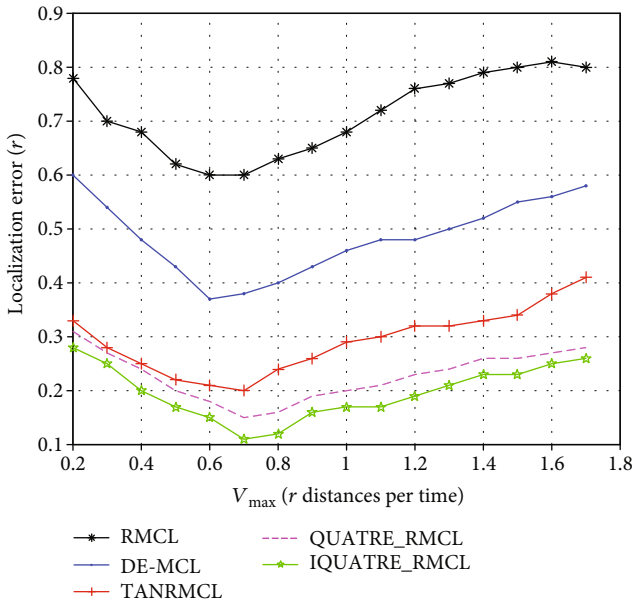

Figure 10: Influence of maximum velocity v_{max} on positioning error.

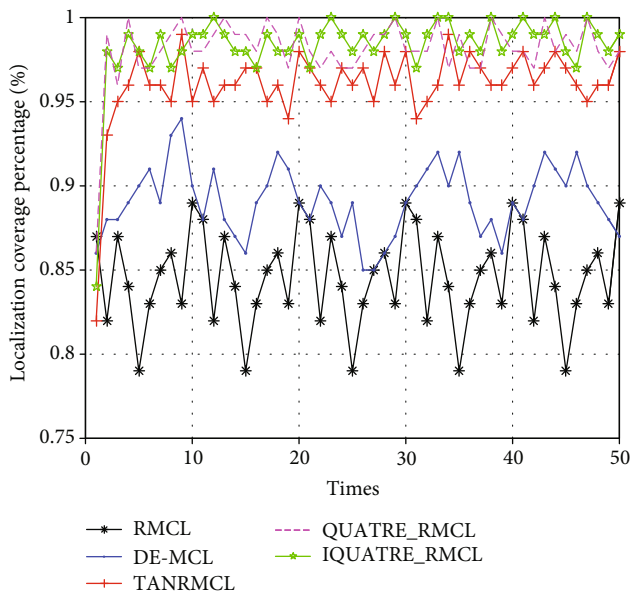

Figure 11: Localization coverage of different algorithms.

temporary anchor nodes and anchor nodes within one hop of the unknown node, and the fitness function is shown in the following equation:

$$\text{fitness}_i = \sum_{j=1}^{M} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} - d_{ij}. \quad (20)$$

3.3.2. IQUATRE_RMCL Algorithm Process. The algorithm process of IQUATRE_RMCL is shown in Figure 8.

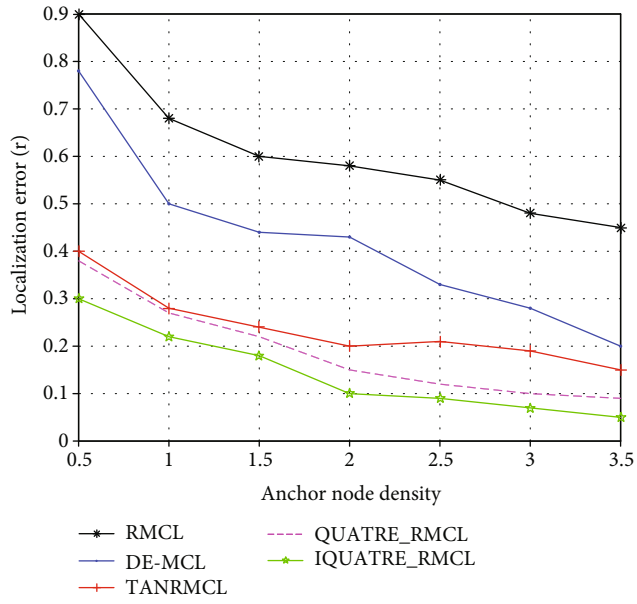

Figure 12: Influence of anchor node density on localization error.

4. Simulation and Analysis of the Algorithm

The simulation experiment in this chapter is carried out in the Intel(R) Core (TM) i5-2450M CPU, clocked at 1.60 GHz, 1.80 GHz, and 8 GB memory platform, and MATLAB 2015b simulation environment.

Experiments will compare the algorithm proposed in this chapter with the classical method and the algorithm proposed in the previous chapter, such as RMCL, TANRMCL, QUATRE_RMCL, QUATRE_RMCL, and DE-MCL in terms of positioning accuracy, positioning coverage, maximum speed, anchor node density, etc. In the simulation experiment, 320 nodes are randomly deployed in a 500 ∗ 500 rectangular area, among which there are 30 anchor nodes with known locations, and the remaining nodes are unknown nodes. We assume that all sensors have the same communication distance of 50 meters. A node can hear another node in r radio range and directly communicate with the node in r radio range.

In order to simulate the mobility of nodes, it is assumed that all nodes follow the random waypoint (RWP) model. RWP is a commonly used model in mobile ad hoc networks. The speed of each node in the network can be randomly changed between 0 and v_{max} in each time step.

Figure 9 shows the IQUATRE-RMCL algorithm, RMCL algorithm, TANRMCL algorithm, QUATRE_RMCL algorithm, and DE-MCL algorithm after 50 time steps, the change in positioning error, and the comparison of the positioning error of each algorithm in each time step. At the same time, it can be seen from Figure 10 that the positioning error of the algorithm proposed in this chapter is smaller than that of the other algorithms most of the time, which is 83% smaller than the average positioning error of the RMCL algorithm and 75% smaller than that of the DE-MCL algorithm. The positioning error is 50% smaller than the TANRMCL

algorithm and 50% smaller than the QUATRE_RMCL algorithm.

Figure 10 shows that the change of the maximum speed has different effects on the positioning errors of several algorithms. When the moving speed of the node is relatively slow, the sampling area based on the moving speed will be relatively small, and the actual position of the node may fall outside the sampling area; and when the moving speed of the node is too large, the sampling area will come from the movement speed constraint will be smaller, and the sampling area will be relatively larger, thereby reducing the positioning accuracy of the node. Therefore, when the maximum speed increases within a certain range, the positioning accuracy of the node will increase with the increase of the maximum speed, but beyond this range, the positioning accuracy will decrease instead. Among them, the RMCL algorithm, the DE-MCL algorithm, the TANRMCL algorithm, the QUATRE_RMCL algorithm, and the IQUATRE-RMCL algorithm all achieve the maximum positioning accuracy when $v_{\max}$ is around $0.7r$.

Figure 11 shows the changes in location coverage of several algorithms. Location coverage refers to the ratio of the number of successfully located nodes to all unknown nodes in a time step. The RMCL algorithm and DE-MCL algorithm are sampling anchor node reference positioning, while the TANRMCL algorithm, QUATRE_RMCL algorithm, and IQUATRE-RMCL algorithm not only use anchor nodes but also select temporary anchor nodes to locate together. In contrast, the latter three algorithms are easier to achieve positioning conditions. In theory, the location coverage of nodes will be higher. Among them, the RMCL algorithm has the lowest location coverage rate, which is around 80%. The location coverage rate of the DE-MCL algorithm is relatively improved, floating around 85%~90%, while the TANRMCL algorithm, QUATRE_RMCL algorithm, and IQUATRE-RMCL algorithm are relatively large. The improved location coverage rate fluctuates around 95%~98%, 97%~100%, and 97%~100%, respectively.

The anchor node density greatly affects the cost of the entire mobile WSN, so maintaining high positioning accuracy under the condition of low anchor node density is an important basis for our research. As shown in Figure 12, it shows the change in the positioning accuracy of the node when the anchor node density changes from 0.5 to 2.5. Among them, the TANRMCL algorithm, the QUATRE_RMCL algorithm, and the IQUATRE-RMCL algorithm can still maintain relatively accurate positioning under the condition of low anchor node density, and the positioning accuracy of the IQUATRE-RMCL algorithm is more accurate.

5. Conclusion

In summary, this article first selects the temporary anchor node, locates other nodes together with the temporary anchor node, and indirectly increases the number of reference nodes, then uses the correction radius to correct the error of the temporary anchor node, and finally optimizes the IQUATRE algorithm used to find the location of unknown nodes. Especially in the sparse anchor node environment, the IQUATRE-RMCL algorithm greatly improves the positioning accuracy and positioning coverage. Although the IQUATRE-RMCL algorithm is a positioning algorithm based on ranging, its hardware requirements are not high, and it can meet the low-cost and high-precision mobile WSN positioning requirements. Since the movement of sensor nodes is random, the next step will be to study the algorithm in this paper under other node movement models.

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 (grant number 61762030), the Innovation-driven Development Special Fund of Guangxi (grant number AA1720417), the Scientific Research and Technology Development Program of Guangxi (grant numbers AB19110050 and AB18126094), and the Guangxi Natural Science Foundation of China (grant number 2018GXNSFAA138170).

References

[1] Z. A. Almusaylim and N. Zaman, "A review on smart home present state and challenges: linked to context-awareness internet of things (IoT)," *Wireless Networks*, vol. 25, no. 6, pp. 3193–3204, 2019.

[2] S. Selvaraj and S. Sundaravaradhan, "Challenges and opportunities in IoT healthcare systems: a systematic review," *SN Applied Sciences*, vol. 2, no. 1, 2020.

[3] N. Gupta, M. Khosravy, N. Patel et al., "Economic data analytic AI technique on IoT edge devices for health monitoring of agriculture machines," *Applied Intelligence*, vol. 50, no. 11, pp. 3990–4016, 2020.

[4] W. Yu, X. Li, H. Yang, and B. Huang, "A multi-objective metaheuristics study on solving constrained relay node deployment problem in WSNS," *Intelligent Automation and Soft Computing*, vol. 24, no. 2, pp. 367–376, 2018.

[5] D. Gao, S. Zhang, F. Zhang, X. Fan, and J. Zhang, "Maximum data generation rate routing protocol based on data flow controlling technology for rechargeable wireless sensor networks," *Computers, Materials & Continua*, vol. 59, no. 2, pp. 649–667, 2019.

[6] J. Wang, A. Hou, Y. Tu, and H. Yu, "An improved DV-hop localization algorithm based on selected anchors," *Computers, Materials & Continua*, vol. 65, no. 1, pp. 977–991, 2020.

[7] J. Wang, X. Qiu, and Y. Tu, "An improved MDS-MAP localization algorithm based on weighted clustering and heuristic merging for anisotropic wireless networks with energy holes," *Computers, Materials & Continua*, vol. 60, no. 1, pp. 227–244, 2019.

[8] J. Xing, Y. Ren, L. Yang, J. Sha, and J. Sun, *Distributed Grid-Based Localization Algorithm for Mobile Wireless Sensor Networks*, Springer, Berlin Heidelberg, 2012.

[9] H. Akcan, V. Kriakov, H. Brönnimann, and A. Delis, "Managing cohort movement of mobile sensors via GPS-free and compass-free node localization," *Journal of Parallel & Distributed Computing*, vol. 70, no. 7, pp. 743–757, 2010.

[10] S. Fujii, A. Uchiyama, T. Umedu, H. Yamaguchi, and T. Higashino, "Trajectory estimation algorithm for mobile nodes using encounter information and geographical information," *Pervasive and Mobile Computing*, vol. 8, no. 2, pp. 249–270, 2012.

[11] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking - MobiCom '04*, Philadelphia, PA, USA, 2004.

[12] A. Baggio and K. Langendoen, "Monte Carlo localization for mobile wireless sensor networks," *Ad Hoc Networks*, vol. 6, no. 5, pp. 718–733, 2008.

[13] M. Rudafshani and S. Datta, "Localization in wireless sensor networks," in *2007 6th International Symposium on Information Processing in Sensor Networks*, Cambridge, MA, USA, 2007.

[14] L. Jianpo, S. Ming, Y. Xie, and S. Jisheng, "A MCL mobile node localization algorithm based on fuzzy theory," *Computer Applications and Software*, vol. 30, no. 12, pp. 147–150, 2013.

[15] Z. Juqin, C. Yangyan, S. Xiaoping, and X. Guo, "Research on node localization algorithm based on adaptive Monte Carlo algorithm for dynamic sensor networks," *Applied Laser*, vol. 36, no. 4, pp. 446–450, 2016.

[16] T. I. A. N. Haoshan, L. I. Cuiran, X. I. E. Jianli, and L. I. A. N. G. Yingxin, "Node localization algorithm for WSN based on time sequence Monte Carlo," *Chinese Journal of Sensors and Actuators*, vol. 29, no. 11, pp. 1724–1730, 2016.

[17] J. Y. Lu and C. Wang, "A new Monte Carlo mobile node localization algorithm based on Newton interpolation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, Article ID 161, 2018.

[18] M. Qin and R. Zhu, "A Monte Carlo localization method based on differential evolution optimization applied into economic forecasting in mobile wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, Article ID 32, 2018.

[19] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, Detroit, MI, USA, 1999.

[20] D. E. Goldberg, "Genetic algorithm in search optimization and machine learning," *Addison Wesley*, vol. xiii, no. 7, pp. 2104–2116, 1989.

[21] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings*no. 4-9, pp. 69–73, Anchorage, AK, USA, 1998.

[22] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[23] J. Hong, A. Diabat, V. V. Panicker, and S. Rajagopalan, "A two-stage supply chain problem with fixed costs: an ant colony optimization approach," *International Journal of Production Economics*, vol. 204, pp. 214–226, 2018.

[24] X. S. Yang, "Firefly algorithm, Levy flight sand global optimization," in *Research and Development in Intelligent Systems XXVI*, pp. 209–218, Springer, London, UK, 2010.

[25] Z. C. Li, X. Zhou, Z. Dai, and X.-Y. Zou, "Identification of protein methylation sites by coupling improved ant colony optimization algorithm and support vector machine," *Analytica Chimica Acta*, vol. 703, no. 2, pp. 163–171, 2011.

[26] X.-S. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations: International Journal for Computer-Aided Engineering and Software*, vol. 29, no. 5, pp. 464–483, 2012.

[27] S.-C. Chu, P. Tsai, and J.-S. Pan, "Cat swarm optimization," in *PRICAI 2006: Trends in Artificial Intelligence*, Q. Yang and G. Webb, Eds., pp. 854–858, Springer: Berlin/Heidelberg, Germany, Berlin/Heidelberg, Germany, 2006.

[28] Z. Meng, J. S. Pan, and H. Xu, "QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: a cooperative swarm based algorithm for global optimization," *Knowledge-Based Systems*, vol. 109, pp. 104–121, 2016.

[29] J. S. Pan, Z. Meng, S.-C. Chu, and J. F. Roddick, "QUATRE algorithm with sort strategy for global optimization in comparison with DE and PSO variants," *Proceedings of the Fourth Euro-China Conference on Intelligent Data Analysis and Applications*, Springer, Cham, 2018.

[30] Z. Meng and J. Pan, "A competitive QUasi-Affine TRansformation Evolutionary (C-QUATRE) algorithm for global optimization," in *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1644–1649, Budapest, Hungary, October 2016.