An Improved Quantum Particle Swarm Optimization Algorithm for Target Tracking Deployment in Spatial Sensor Networks

Lisha Liu, Xincan Fan[*]

Department of Integrated Circuits, Shenzhen Polytechnic University, China
liulisha@szpu.edu.cn, horsefxc@szpu.edu.cn

Abstract

This paper presents a comprehensive review of the current research on spatial sensor networks and the node deployment methods employed for mobile target tracking. The study introduces particle swarm optimization (PSO) and its quantum behavior extension, detailing concepts such as the quantum state wave function and particle position representation. Subsequently, an improved Quantum Particle Swarm Optimization (QPSO) algorithm is proposed. This enhanced algorithm increases population diversity by incorporating quantum rotation gates and quantum mutation mechanisms, expands the search space through the superposition state and interference principles of quantum mechanics, and dynamically adjusts algorithm parameters to balance global exploration and local search. These modifications aim to improve both the convergence speed and accuracy of the algorithm. Simulation results demonstrate that the improved QPSO algorithm surpasses traditional mobile tracking deployment algorithms and the standard quantum behavior particle swarm optimization algorithm in terms of target tracking deployment within spatial sensor networks. Notably, it significantly enhances the tracking success rate and reduces tracking errors.

Keywords: Spatial sensor network, Target tracking, Node deployment, Quantum particle swarm optimization algorithm

1 Introduction

Spatial sensor networks consist of numerous micro-sized, low-power-consumption, low-cost sensor nodes deployed within a monitored environment. These nodes form a dynamic distributed network through wireless communication in an ad hoc manner, enabling environmental monitoring and data acquisition. Each node possesses capabilities in sensing, processing, and communication, allowing them to autonomously and cooperatively complete monitoring tasks. Multi-target tracking technology within spatial sensor networks aims to monitor and track the position, state, and trajectory of multiple target objects in real-time with high accuracy through the collaboration of multiple sensor nodes.

Significant progress has been made in multi-target tracking, data association, and fusion within spatial sensor networks. Target detection and tracking leverage the sensors within nodes to detect the appearance and movement of targets, and to track real-time positional changes through target prediction and path planning. Commonly used tracking algorithms include particle filtering, Kalman filtering, and data association. However, challenges remain, such as the complexity of data association, communication costs, resource limitations, the adaptability of limited fields of view, and the robustness and accuracy of algorithms in practical applications. Current research focuses on cooperative sensing and data fusion, utilizing multi-node cooperation to enhance monitoring range and accuracy. Data fusion algorithms, such as Bayesian estimation and neural networks, are employed to combine data collected by each node, reduce noise, and improve signal quality.

Numerous universities and research institutions have conducted studies on the deployment of spatial sensor nodes and target tracking technology, proposing various improved methods such as computational geometry, heuristic algorithms, and grid deployment methods.

Recent studies have explored various algorithms and techniques for improving target tracking in spatial sensor networks. For instance, Ramadevi et al. proposed a meta-heuristic aided target movement prediction scheme using adaptive distributed extended Kalman filtering to enhance mobility target tracking in Wireless Sensor Networks (WSNs) [1]. Feng et al. proposed a PSO algorithm based on modified crowding distance for multimodal multi-objective problems, demonstrating the versatility of PSO in different application domains [2]. Ye and Dong presented an ensemble algorithm based on adaptive chaotic quantum-behaved particle swarm optimization with Weibull distribution and hunger games search, highlighting the potential of QPSO in financial applications [3]. Similarly, Hu et al. proposed an energy-efficient clustering and routing protocol based on QPSO and fuzzy logic for WSNs, emphasizing the algorithm's efficacy in network optimization [4].

This research contributes to the growing field of optimization algorithms inspired by quantum mechanics and swarm intelligence. It builds on previous work, such as the application of QPSO in forest cover prediction [5]. Furthermore, it complements research on other swarm optimization algorithms, such as the chicken swarm optimization algorithm [6]. To strike a balance between global exploration and local search, the algorithm

*Corresponding Author: Xincan Fan; E-mail: horsefxc@szpu.edu.cn

DOI: https://doi.org/10.70003/160792642024092505006

dynamically adjusts its parameters. This adaptability ensures that the search process is both efficient and effective, leading to improved convergence speed and accuracy and various other domains where QPSO has shown promise and the glowworm swarm optimization algorithm, by offering a novel approach to target tracking deployment in spatial sensor networks.

2 Target Tracking in Space Sensor Networks

2.1 Current Research on Space Sensor Networks

Current research in space sensor networks, particularly regarding target tracking, builds upon advancements in wireless sensor networks and their applications in various domains.

One study, conducted by a team including Tao, Luo Jianpeng, Xie, Bai Tao, Zhang, Yao, Zhang, Chaoqun, and Li, focuses on an improved energy-efficient clustering routing scheme for industrial wireless sensor networks. This research employs a Levy chaotic particle swarm optimization algorithm to enhance the performance of the sensor network [7]. Another significant contribution to the field is a study by Xiao Jiang Du and Xiao Hua Chen, which explores wireless sensor network security [8]. Published in IEEE Wireless Communications, this research delves into the security aspects of wireless sensor networks, highlighting potential vulnerabilities and proposing solutions to mitigate security risks. Ensuring the security of space sensor networks is essential for accurate and reliable target tracking, as any compromise in security could lead to inaccurate data or even system failures [9].

The research on target tracking within space sensor networks encompasses a wide range of topics, including algorithm optimization, data association and fusion, resource management, multi-target tracking, and three-dimensional (3D) space tracking. Current efforts in target tracking algorithms are primarily focused on enhancing tracking efficiency and accuracy. Researchers are continuously seeking algorithms capable of processing large volumes of sensor data swiftly while accurately predicting target locations. In dynamic environments, these algorithms must be both real-time and robust to meet the application requirements of complex scenarios [10].

2.2 Node Deployment Based on Moving Target Tracking

Node deployment algorithms for mobile target tracking represent a complex problem that spans multiple fields and technologies. The primary objective is to optimize the layout and performance of a sensor network or tracking system to ensure accurate tracking of a moving target. Moving target tracking is key to ensuring efficient network monitoring and response. This process begins by establishing a network model, which defines the network topology, including node locations, communication ranges, and other relevant parameters [11]. Additionally, a motion model of the target is established, describing its trajectory and speed, along with a sensing model for the nodes, which includes factors such as detection range and accuracy [12].

In order to effectively track mobile targets, sensor nodes need to be deployed reasonably so that relevant data can be obtained in a timely manner when the target moves. In target tracking applications, once a target is detected, the focus shifts from the coverage quality of the entire detection area to the coverage quality of the path the moving target is expected to traverse. This involves calculating the probability that a moving target will be detected as it moves through the node deployment area along any potential path. A critical challenge is how to maximize this coverage by establishing constraints that ensure every point within the monitoring area is covered by at least one sensor node [13]. This seamless monitoring also requires that sensor nodes maintain effective communication with each other, enabling the transmission of collected data to the aggregation node or processing center, which is a key focus of this paper.

Assume the target location is $\vec{V}(t)=(x_T(t), y_T(t), z_T(t))$, Speed is $\vec{V}(t)=(v_x(t), v_y(t), v_z(t))$, The position of the target in the future can be predicted using the following formula:

$$\vec{T}(t+\Delta t)=\vec{T}(t)+\vec{V}(t)\times\Delta t \quad \textbf{(1)}$$

Through this formula, the future position of the target can be predicted based on its current position and speed, which is of great significance for achieving effective target tracking.

For the optimization of node deployment, a fitness function can be defined to evaluate the advantages and disadvantages of node deployment schemes, as shown in formula:

$$F = w_1\times Cov + w_2\times EC^{-1} + w_3\times L^{-1} \quad \textbf{(2)}$$

Among them:

- Cov represents the coverage of sensor nodes, reflecting the sensor network's ability to perceive the target.
- EC represents the energy consumption of sensor nodes, reflecting the operating efficiency of the sensor network.
- L represents the average distance between sensor nodes, reflecting the topological structure and communication efficiency of the sensor network.
- w_1, w_2, w_3 are weight coefficients used to adjust the degree of influence of different factors on the fitness function.

By comprehensively considering these factors, the optimal node deployment scheme can be selected to improve the performance and efficiency of the spatial sensor network [14].

2.3 Autonomous Deployment Based on Complex Paths

In spatial sensor networks, the movement of targets along complex paths poses great challenges for tracking deployment. Complex paths may have varying directions, different speeds, and irregular shapes.

For this situation, accurate modeling and feature extraction of complex paths are first required. Mathematical models such as polynomial curve fitting and spline curves can be used to approximate the motion path of the target. At the same time, based on the historical movement data of the target and current environmental factors, predict its possible future direction. To effectively describe complex paths, the following models can be used:

Polynomial model: Use polynomial functions to fit the motion trajectory of the target, for example:

$$P(t) = a_n t^n + a_{n-1} t^{n-1} + ... + a_1 t + a_0 \quad (3)$$

Among them, a_i is the coefficient of the polynomial, and t is the time variable.

Bezier curve: it is a curve representation method widely used in computer graphics and related fields suitable for describing smooth and complex paths, with the following formula:

$$B(t) = \sum_{i=0}^{n} P_i B_{i,n}(t) \quad (4)$$

Among them, $B_{i,n}(t)$ is the Bessel function and P_i is the control point. These control points determine the shape and direction of the curve. The advantage of the Bezier curve is that it can flexibly change the shape of the curve by adjusting the control points, thereby achieving an accurate description of various smooth paths. By reasonably selecting and setting the control points, a smooth curve that meets specific needs can be generated, enabling it to accurately simulate and represent the motion trajectory or shape change of an actual object [15].

Random process model: Considering the uncertainty of the target's motion, a random walk model can be used to describe the target's movement:

$$X(t+1) = X(t) + \in_t \quad (5)$$

In this model, represents the position of the target at time t, represents the position of the target at time t. is a random walk, which represents the uncertainty factor of the target's motion. This means that the movement of the target at each time step is based on its current position plus a random perturbation.

By adopting the random walk model, we can better capture the uncertainty of the target's motion, thereby more realistically simulating the movement of the target in space. This model has applications in many fields, such as target tracking in wireless sensor networks, robot navigation, etc.

However, it should be noted that the random walk model is only a simplified description method. In actual situations, the movement of the target may be affected by many factors, such as the environment, the characteristics of the target itself, etc. Therefore, in specific applications, it may be necessary to combine more information and more complex models to improve the accuracy of target tracking.

Assuming the position of the node is $\vec{x}_i = (x_i, y_i, z_i)$ and the position of the target is $\vec{t}_i = (t_x, t_y, t_z)$.

The fitness function can be:

$$f = w_1 \times d\left(\vec{x}_i, \vec{t}_i\right) + w_2 \times E\left(\vec{x}_i\right) \quad (6)$$

3 Quantum Particle Swarm Algorithm Applications

3.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a nature-inspired metaheuristic computational method that draws its principles from the foraging patterns of avian species in their natural habitats. By simulating the collaborative behavior observed in bird flocks during foraging, PSO solves optimization problems through the interaction and movement of individual particles within a search space. The algorithm's key operation entails refining and modulating each particle's position and velocity, enabling an exhaustive search of the solution space and aiding in the determination of the global optimum. When applied to spatial sensor network deployment, this global search capability ensures comprehensive coverage of the target area by the sensor network [16].

In PSO, candidate solutions to the optimization problem are abstracted as "particles" within the solution search space. These particles possess two pivotal attributes: velocity and position. Velocity dictates the speed and direction of a particle's movement, while position indicates the particle's current location within the search space. Particles navigate the solution space by following the best-performing particles (both individually and globally) in an effort to identify the optimal solution [17].

Specifically, each particle's movement is influenced by two factors: p*best*, the best solution found by the particle itself, and g*best*, the best solution identified by the entire swarm.

Envision a swarm of m particles positioned within an S-dimensional objective search space, wherein the location of the *i*-th particle is denoted by an S-dimensional vector. $x_i = (x_{i1}, x_{i2}, x_{is})$, $i = 1, 2, \ldots, m$. The position of each particle serves as a potential solution, and its fitness value is calculated by substituting x_i into an objective function. The quality of the solution is determined by the fitness value. The velocity of the *i*-th particle is also represented as an S-dimensional vector $v_i = (v_{i1}, v_{i2}, v_{is})$, while the best position discovered by the *i*-th particle is denoted as $P_i = (p_{i1}, p_{i2}, p_{is})$. The best position found by the entire swarm is denoted as $P_{gs} = (p_{g1}, p_{g2}, p_{gs})$. Let $f(x)$ represent the objective function to be minimized. The current best position of particle i is updated using Equation (6):

$$p_i = (t+1) = \begin{cases} p_i(t), & f(x_i(t+1)) \geq f(p_i(t)) \\ X_i(t+1), & f(x_i(t+1)) < f(p_i(t)) \end{cases} \quad (7)$$

The process of the PSO algorithm is outlined below:

1. Initialize the particle swarm: randomly set the initial position and speed of each particle.

2. Calculate the fitness value: substitute the position of each particle into the objective function and calculate its fitness value.

3. Update individual extreme values and global extreme

values: for each particle, the current fitness value is juxtaposed with its historical optimal fitness value. Should the current value prove more advantageous, the individual extremum is updated; concomitantly, the individual extrema of all particles are compared to ascertain the global extremum.

4. Update particle speed and position: update the speed and position of the particle according to the following formula:

Speed update formula:

$$v_i(t+1) = w \times v_i(t) + c_1 \times r_1 \times (p_i(t) - x_i(t) + c_2 \times r_2 \times (p_g(t) - x_i(t) \tag{8}$$

Among them, w is the inertia weight, c_1 and c_2 are the learning factors, r_1 and r_2 are random numbers between 0 and 1.

Position update formula:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{9}$$

5. Determine whether the termination condition is met: If the termination condition is met (such as reaching the maximum number of iterations or the fitness value meets the requirements), the algorithm ends; otherwise, return to step 2 to continue iterating.

Through ongoing iterative refinement of the particles' velocity and position, the particle swarm explores the solution domain in pursuit of the optimal solution, ceasing only upon satisfaction of the termination criterion. The PSO algorithm is characterized by its straightforward implementation and rapid convergence, attributes that have facilitated its extensive application across diverse fields.

The search process in the particle swarm algorithm is dependent on the velocity of particles. As particles continuously evolve their trajectory and speed, their search is confined to fixed trajectories, causing the search space to become progressively limited. This limitation restricts the algorithm's global convergence capabilities, often leading to a scenario where particles are confined to searching within a limited area, thereby impeding the discovery of the global optimal solution [18].

3.2 Introduction of Quantum Behavior into the Particle Swarm Algorithm

While the PSO algorithm exhibits strong global search capabilities, it is still susceptible to becoming trapped in local optima when faced with complex optimization problems. This limitation arises primarily because particles in the PSO algorithm rely too heavily on the information provided by the current best solution and the global best solution during the search process. This reliance results in a unidirectional search, preventing particles from escaping local optimal regions. In the context of spatial sensor network deployment, if the initial positions of the sensor nodes are not well-chosen or if the target motion patterns are complex and variable, the PSO algorithm may fail to identify the globally optimal deployment scheme. The PSO algorithm's search process is heavily dependent on the velocity and position update formulas for the particles, but it lacks an effective mechanism for maintaining diversity within the swarm. This deficiency can lead to premature convergence in the later stages of the search, where the particle swarm loses its ability to explore new solutions. In spatial sensor network deployment, if the sensor nodes are deployed too homogeneously, the network may not be able to adapt to the dynamic and complex requirements of target tracking [19].

Moreover, the performance of the PSO algorithm is highly sensitive to parameter settings. For instance, the inertia weight magnitude influences the balance between global and local search capabilities, while the learning factors determine the degree to which particles rely on their own experiences versus the experiences of the swarm. Incorrect parameter settings can result in slow convergence rates or poor convergence accuracy.

Quantum-behaved Particle Swarm Optimization (QPSO) introduces quantum computing principles into the traditional PSO algorithm. In QPSO, the motion of particles is described as a quantum state, and their movement is viewed as an evolution process within this quantum state. This quantum approach significantly enhances the algorithm's global search ability and convergence stability.

The introduction of quantum behavior transforms the classical search space into a quantum space. According to the principle of uncertainty, the position and velocity of particles cannot be simultaneously determined. Instead, the position of particles is determined by a wave function, and their quantum state is updated according to a probability distribution. This probabilistic update mechanism endows QPSO with superior global search capabilities.

In QPSO, the particle position update formula is:

$$x_i(t+1) = p_i \pm \alpha \cdot |C - x_i(t)| \cdot \ln\left(\frac{1}{u}\right) \tag{10}$$

Where, $x_i(t+1)$ represents the position of the i particle at the $t+1$ iteration, is the individual optimal position of the particle p_i, α is the control parameter, C is the center of the individual optimal positions of all particles, and u is a random number from the interval [0,1].

The procedure for the QPSO algorithm is as follows:

1. Particle swarm initialization: randomly set the initial and individual optimal positions for each particle, evaluate the initial fitness values, and identify the global optimal position.

2. Calculate the center of the particles' individual optimal positions:

$$C = \frac{1}{m} \sum_{i=1}^{m} p_i \tag{11}$$

Where, m is the number of particle swarms.

3. Update the position of the particle: calculate the new position of the particle according to the position update formula.

4. Calculate the fitness value of the particle: substitute the updated particle position into the fitness function and calculate the fitness value.

5. Update the individual optimal position and the global

optimal position: if the current fitness value of the particle is better than its individual optimal fitness value, update the individual optimal position; if the current fitness value of the particle is better than the global optimal fitness value, update the global optimal position.

6. Assess if the termination criterion is satisfied: should the termination criterion be met (for instance, attaining the maximum iteration count or achieving a fitness value that satisfies predefined standards), the algorithmic process concludes; if not, revert to step 2 to persist in iteration.

By introducing quantum behavior, the QPSO algorithm can overcome the limitations of the PSO algorithm to a certain extent, improve the performance of the algorithm in dealing with complex optimization problems, and is more suitable for problems such as spatial sensor network deployment.

3.3 Quantum State Wave Function

The wavefunction associated with a quantum state is a mathematical construct that characterizes the condition of a particle within a quantum system. This function encapsulates details pertaining to the particle's location, momentum, energy, and additional physical attributes.

Wave functions have the following properties:

1. Complex function: A wave function is usually a complex function, which means that it has a real part and an imaginary part.

2. Normalization: A wave function must be normalized, which means that the integral of its squared modulus over the entire space is equal to 1. This means that the probability of a particle being somewhere in space is 1.

3. Linear superposition: A quantum state can be a linear superposition of multiple wave functions. This means that a particle can be in a superposition of multiple states.

4. Time evolution: A wave function evolves over time, and its evolution is described by the Schrödinger equation.

In contrast to the position and velocity of particles in classical PSO, the state of particles in QPSO is described using a quantum state wave function. This wave function quantitatively characterizes the state of microscopic particles and is denoted by $\Psi(x, t)$, where x represents the position of the particle. The probability density function of the particle is defined as the square of the absolute value of the wave function, i.e., $|\Psi(x, t)|^2$. In three dimensions, $|\Psi(x, t)|^2 dxdydz$ represents the probability density of finding the particle within the volume element $dxdydz$ at time t, and this satisfies the normalization condition shown in Equation (12):

$$\int_{-\infty}^{+\infty} |\Psi(x,t)|^2 dxdydz = \int_{-\infty}^{+\infty} \Psi(y) dxdydz = 1 \tag{12}$$

In the QPSO algorithm, determining the state of a particle first requires obtaining the particle's wave function. This is done by solving the Schrödinger equation, as shown in Equation (13):

$$\begin{cases} V(X) = -\gamma\delta(X-p) = -\gamma(y) \\ \hat{H} = -\dfrac{h^2}{2m}\nabla^2 + V(X) - i\hbar\dfrac{\partial}{\partial t}\Psi(x,t) = H(x,t) \end{cases} \tag{13}$$

Solving this equation yields the wave function, which is expressed in Equation (14):

$$\Psi(y) = \frac{1}{\sqrt{L}} e^{-|y|/L} \tag{14}$$

Here, $L = \frac{h^2}{m\gamma}$, and $y = X - p$, where $V(x)$ is the potential well, i.e. the potential function, $\hat{H}$ is the Hamiltonian operator, and ∇^2 is the Laplace operator. This gives the probability density function, as shown in Equation (15):

$$Q(y) = |\Psi(y)|^2 = \frac{1}{L} e \tag{15}$$

3.4 Position of Particles

In the quantum-behavior particle swarm optimization algorithm,The positions of particles in the QPSO algorithm are determined using Monte Carlo stochastic simulation. The equations for updating particle positions are given by Equations (16) through (20):

$$p(t) = \theta \cdot p_b(t) + (1-\theta) p_g(t) \tag{16}$$

$$m(t) = \frac{1}{N} \sum_{i-1}^{N} p_{bi}(t) \tag{17}$$

$$L(t+1) = 2\alpha \cdot |m(t) - X(t)| \tag{18}$$

$$\alpha = \alpha - (a-b) \cdot \frac{t}{G_{\max}} \tag{19}$$

$$X(t) = p(t) \pm \frac{L}{2} \ln(\frac{1}{u}) \tag{20}$$

In formula (16), $p(t)$ represents the position of the particle at the t iteration, which is jointly determined by the individual optimal position $p_b(t)$ of the particle and the global optimal position $p_g(t)$ of the population, where θ is a random number that follows a uniform distribution on [0,1].

In formula (17), $m(t)$ represents the average value of the individual optimal positions of all particles in the population at the tth iteration.

In formula (18), $L(t+1)$ represents the weighted distance between the particle and the average optimal position of the population, α is the contraction-expansion coefficient, which is used to control the convergence speed of the particle. It changes linearly from α to b as the iteration proceeds. Usually $\alpha = 1$, $b = 0.5$.

In formula (19), $G_{\max}$ represents the maximum number of iterations.

In formula (20), $X(t)$ represents the position of the particle at the tth iteration, which $p(t)$ is determined by adding or subtracting a term related L to u and random number, where u is a random number that follows a uniform distribution on [0,1].

By synthesizing the above equations, we obtain the position update equation (21):

$$X(t+1) = p(t) \pm \alpha \left| m(t) - X(t) \right| \cdot \ln(1/u) \tag{21}$$

4 Algorithm Based on Improved QPSO Algorithm in Moving Target Tracking

4.1 Algorithm Improvement Ideas

The QPSO algorithm exhibits strong global search capability and convergence speed; however, it may face limitations in scenarios involving moving target tracking. Enhancing the algorithm can further improve its performance and adaptability. The following are key directions for improvement:

Develop an adaptation function that more accurately reflects the effectiveness of moving target tracking. This may include factors such as target position prediction accuracy, real-time tracking, and other relevant metrics. Adjust key algorithm parameters dynamically to adapt to various moving target characteristics and tracking environments. Combine QPSO with other algorithms or techniques, such as Kalman filtering and particle filtering, to leverage their strengths and improve tracking accuracy and stability. Consider optimizing multiple objectives simultaneously, such as energy consumption, tracking accuracy, and response time, to achieve a balanced performance in moving target tracking scenarios.

The core idea behind the improved QPSO algorithm (IQPSO) involves introducing quantum behavior to increase population diversity through quantum rotation gates and quantum mutation. By leveraging the principles of superposition states and quantum interference, particles can concurrently occupy multiple positions, thereby amplifying the search space and enhancing the probability of identifying the global optimal solution. Additionally, dynamic adjustments to inertia weights and learning factors help balance global exploration with local exploitation, thereby improving the convergence speed and accuracy of the algorithm.The improved algorithm also optimizes objectives related to tracking accuracy, energy consumption, and network coverage.

In the IQPSO algorithm, the quantum rotation gate plays a key role in adjusting particle positions and velocities, enhancing population diversity, and preventing premature convergence to suboptimal solutions. This optimization process can be represented by the unitary matrix shown in Equation (22):

$$U(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{22}$$

Here, θ is the rotation angle, which determines the balance between exploration and development in the algorithm. By adjusting this angle, the magnitude and direction of particle movement can be finely controlled, enabling precise management of the search process.

Quantum mutation refers to the phenomenon where a quantum system transitions between energy levels, typically associated with the collapse of a particle's wavefunction. In the context of IQPSO, quantum mutation is used to maintain population diversity, thereby enhancing the algorithm's ability to solve complex optimization problems. Quantum mutation in QPSO can be implemented as follows: for each particle, a quantum mutation operation is applied with a certain probability, as described by Equation (23):

$$x_i = x_i + \Delta x \cdot U(-1,1) \tag{23}$$

In this equation, x_i represents the position of the i-th particle, Δx is the mutation amplitude, and $U(-1,1)$ is a uniformly distributed random number generator that produces values between -1 and 1. This operation introduces random perturbations to the particle positions, enabling the exploration of new regions in the search space.

Algorithm improvement strives to improve the global search capability of the algorithm and avoid the risk of falling into a local optimal solution. Improve the convergence speed and accuracy of the algorithm and find better solutions faster. Improving the practicability of the algorithm can more comprehensively meet the actual needs of moving target tracking. Algorithm improvements innovate in the following aspects:

1. Simultaneously considering tracking accuracy, energy consumption and network coverage as optimization goals, instead of just focusing on a single indicator, enables the algorithm to find the optimal balance between multiple interrelated factors, improving the comprehensiveness of the algorithm in practical applications performance.

2. Increase population diversity through quantum revolving doors and quantum mutations, preventing the algorithm from prematurely converging to suboptimal solutions, expanding the search space, and elevating the probability of ascertaining the optimal solution, and significantly enhancing the global search capability of the algorithm. The superposition state and interference principle of quantum mechanics are used to allow particles to be in multiple positions at the same time. This unique method is very different from the search strategy of traditional algorithms and provides new ideas for solving complex optimization problems.

3. Dynamically adjust inertial weights and learning factors to balance global exploration and local development. It can adaptively adjust the search strategy according to the operation of the algorithm, improve the convergence speed

and accuracy, and enable the algorithm to deal with different problem scenarios and complexities more effectively. degree.

4. Take tracking accuracy, energy consumption and network coverage as optimization goals, consider the actual needs of moving target tracking more comprehensively, and improve the practicality of the algorithm.

4.2 Target Tracking Algorithm Model

To develop the target tracking algorithm, the first step is to establish the sensor network topology model. This involves defining the moving target motion model, analyzing the characteristics of the sensing data, and assessing the noise characteristics. The moving target tracking problem within the spatial sensor network is then transformed into a multidimensional optimization problem. In this context, the target's position becomes the variable to be optimized, while the measurement data and communication capabilities of the sensor network serve as constraints. A fitness function is defined to guide the optimization process, such as minimizing the distance between the actual target position and its estimated position. The algorithm calculates the position of each sensor node, providing the determined coordinate information. The node model is represented as $[O(x, y, z), R_m, R_k, R_c, F_{\max}]$, where $O(x, y, z)$ denotes the three-dimensional coordinates of the node within the space, with x, y, and z corresponding to the X-axis, Y-axis, and Z-axis values in a Cartesian coordinate system. The parameters R_m, R_k, R_c represent the minimum mutual collision radius between nodes, the equilibrium radius, and the communication radius, respectively. $F_{\max}$ is the maximum repulsive force exerted when the distance between nodes falls below the minimum radius.

In this model, the set of nodes in the spatial sensor network is treated as a virtual physical system, with parameters such as force F and mass m. The equations governing the induced and repulsive forces between nodes are provided in Equations (23) and (24):

$$F_a(i,j)=\begin{cases}-\dfrac{k_1 m_i m_j}{d(i,j)^{\alpha^1}} & 0<d_{ij}<R_c\\ 0 & d_{ij}\ge R_c\end{cases} \tag{24}$$

$$F_\tau(i,j)=\begin{cases}\dfrac{k_2 m_i m_j}{d(i,j)^{\alpha^2}} & 0<d_{ij}<R_k\\ 0 & d_{ij}\ge R_k\end{cases} \tag{25}$$

In these equations, k_1 and k_2 are gain coefficients, $\alpha 1$ and $\alpha 2$ are exponents that determine the strength of the forces, m_i and m_j are the mass-like quality factors of nodes i and j, and $d(i, j)$ represents the distance between nodes i and j.

To achieve effective tracking and monitoring of the moving target, it is assumed that the tracked target exerts a stronger attractive force on the sensor nodes than the forces between the nodes themselves. When the tracked target appears, the sensor nodes can quickly respond by being attracted toward it, ensuring that the target remains within the communication range for timely tracking. The magnitude of the attractive force that a node receives from the tracked target T is given by Equation (26):

$$F_\tau(i,T)=\begin{cases}-\dfrac{k_5 m_i m_\tau}{d(i,j)^{\alpha 5}} & d_{i\tau}<r\\ 0 & d_{i\tau}\ge r\end{cases} \tag{26}$$

4.3 Improved Algorithm Flow

The algorithm flow is illustrated in Figure 1 and is divided into four stages, comprising ten steps as outlined below.

4.3.1 Algorithm Initialization

1. Define the Problem Space: Determine the scope and dimensionality of the search space, which typically depends on the geographical area covered by the sensor network and the specific requirements for target tracking.

2. Initialize the Particle Swarm: Generate a set of random particles within the deployment area range $\{x_{\max}\ x_{\min}\ y_{\max}\ y_{\min}\ z_{\max}\ z_{\min}\}$ for node i. Each particle represents a possible deployment scenario for the sensor network. Set the particle swarm size N and particle dimension D. Initialize the moving distance step Δd, time step Δt, and determine the global optimal position g*best* and individual optimal position p*best* based on the initial positions of the particles.

3. Set Parameters: Configure parameters such as particle swarm size N, maximum number of iterations $G_{\max}$, inertia weight w, and learning factors $c1$ and $c2$. For the quantum particle swarm algorithm, also set the relevant parameters for the quantum bit probability density function.

4.3.2 Adaptation Evaluation

4. Calculate the Fitness Value: For each particle, calculate the fitness value based on its current position (which represents a sensor deployment scheme). The fitness function is typically designed to account for factors such as target tracking accuracy, sensor network coverage area, and energy consumption.

5. Update Individual and Global Bests: Compare each particle's fitness value with the fitness value of its historical best position (pbest) and update the pbest accordingly. Simultaneously, compare the fitness values of all particles to update the global best (representing the current optimal deployment scheme).

4.3.3 Iterative Optimization

6. Quantum Behavioral Update: Utilize the quantum bit probability density function, unique to the quantum particle swarm algorithm, to update the positions of the particles. This approach simulates the quantum behavior of particles using the concept of a quantum potential well, thereby enhancing search diversity and global search capability.

7. Velocity Update: Determine the revised velocity for each particle by applying the standard velocity update equation of the PSO algorithm. This equation routinely encompasses terms representing inertia, self-awareness, and social cognition, and integrates these dimensions with positional data acquired subsequent to quantum behavioral updates.

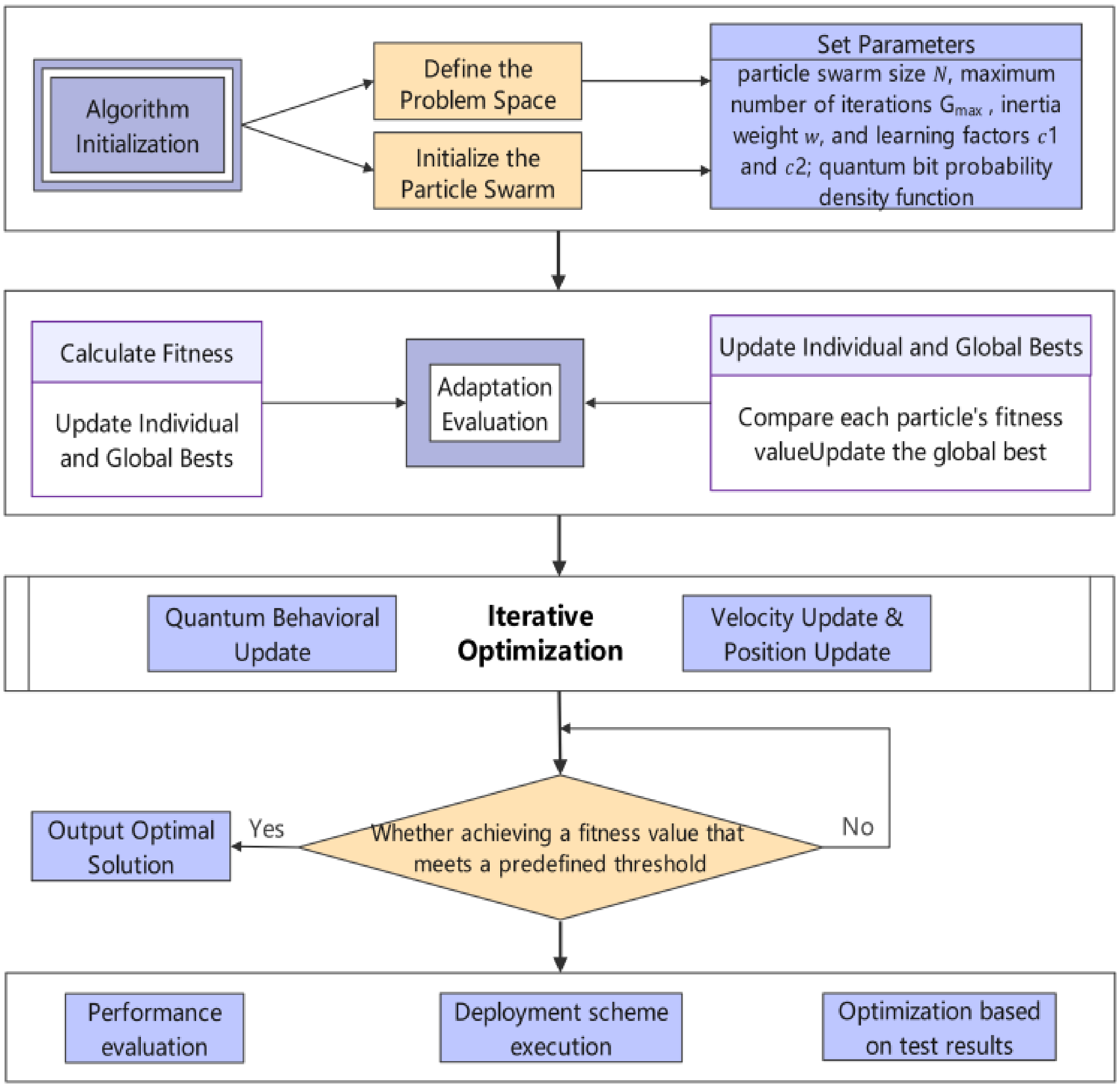
8. Position Update: Adjust the positions of the particles using the recently computed velocity. If any particle exceeds the perimeter of the search area, adjust its position to remain within the defined boundaries.

4.3.4 Iterative Termination Condition Checking

9. Check Termination Conditions: After each iteration, verify whether the termination conditions have been met, such as reaching the maximum number of iterations G_{max} or achieving a fitness value that meets a predefined threshold.

10. Output Optimal Solution: If the termination condition is satisfied, output the current global best value as the optimal deployment scheme. If not, continue with the next iteration.

Following this process, the actual deployment of the sensor network is adjusted according to the identified optimal deployment scheme. The performance of this deployment is then tested in a real-world environment, evaluating metrics such as target tracking accuracy, network coverage, and energy consumption. Based on the test results, the algorithm parameters and fitness function may be further adjusted to optimize the algorithm's performance.



Algorithm Initialization
Define the Problem Space
Initialize the Particle Swarm
Set Parameters
particle swarm size N, maximum number of iterations Gmax , inertia weight w, and learning factors c1 and c2; quantum bit probability density function
Calculate Fitness
Update Individual and Global Bests
Adaptation Evaluation
Update Individual and Global Bests
Compare each particle's fitness valueUpdate the global best
Quantum Behavioral Update
Iterative Optimization
Velocity Update & Position Update
Output Optimal Solution
Yes
Whether achieving a fitness value that meets a predefined threshold
No
Performance evaluation
Deployment scheme execution
Optimization based on test results

Figure 1. Improved algorithm flow

5 Experimental Simulation and Result Analysis

5.1 Experimental Simulator Statistics

To evaluate the performance of the improved QPSO algorithm, a simulation program was developed using MATLAB. This program generates a target trajectory with random motion and initializes the positions of sensor nodes. During each iteration, the node deployment strategy is optimized using the improved QPSO algorithm, with node positions being calculated and updated until the termination conditions are met (such as achieving the predetermined tracking accuracy or reaching the maximum number of iterations).

Scenario Description: In the simulation, 100 sensor nodes are randomly deployed within a 3D surveillance area of size [0,100]×[0,100]×[0,100] cubic meters. The surveillance area is divided into several smaller compartments.

Target Trajectory: The simulated target moves according to a random wandering model, with its speed varying randomly within a specified range.

Evaluation Metrics: Tracking Accuracy (Measured as the average position error between the target's actual position and the estimated position), Network Energy Consumption (Measured as the percentage of remaining energy in the sensor network after the tracking process), Coverage Efficiency (Measured by the number of sensors that successfully detect and cover a new target).

Sensor Type: The r sensor has a spherical sensing range with a radius r, and the detection probability is inversely related to the distance from the target.

Algorithm Parameter Settings: Population Size N, Maximum Number of Iterations T_{max}, and other parameters are adjusted based on experimental requirements.

During each time step, the state of the sensors, energy consumption, and target detection data are recorded. At the end of the simulation, the tracking accuracy, energy consumption, and coverage efficiency are calculated. The performance of the improved QPSO algorithm is then compared to the traditional PSO algorithm under the same spatial sensor network (SSN) configuration. This comparison includes an analysis of energy consumption during the tracking process and an evaluation of the impact of the improved QPSO algorithm on sensor network coverage efficiency while maintaining tracking accuracy.

Sample Simulation Experiment Code:

```
%% Parameter Initialization
num_particles = 50; % Number of particles
max_iterations=100; %Maxim umnumber of  iterations
inertia_weight = 0.9; % Inertia weight
cognitive_factor = 2.0; % Cognitive factor
social_factor = 2.0; % Social factor

%% Objective Function
fitness_function=@(particle)evaluate_fitness(particle);
//fitness_function=@(particle)evaluate_fitness(particle): Define the objective function to evaluate the fitness of each particle. The evaluate_fitness function should be defined according to the specific problem. For example, in spatial sensor network target tracking, it may calculate the target position estimation error.

%% QPSO Algorithm Simulation
qpso=QPSO(fitness_function,num_particles, max_iterations,inertia_weight,cognitive_factor, social_factor);
//Create a QPSO algorithm object, pass in parameters and objective function.
best_params=zeros(max_iterations, size_of_parameters);
//Define a matrix to store the best parameters for each iteration.
best_fitness = zeros(max_iterations, 1);
//Define a vector to store the best fitness of each iteration.

for i = 1:max_iterations
[best_params(i,:), best_fitness(i)] = qpso.optimize();
end
//Loop through the QPSO optimization process, call the qpso.optimize() method for each iteration, obtain the best parameters and fitness, and store them in the corresponding matrix and vector.

%% Performance Evaluation
mean_fitness = mean(best_fitness);
//Calculate the average fitness of all iterations.
std_fitness = std(best_fitness);
//Calculate the standard deviation of fitness for all iterations.

%% Plot the Results
figure;
plot(1:max_iterations, best_fitness, 'b-');
//Draw a curve chart showing the change of fitness with the number of iterations.
title('Improved Performance of QPSO Algorithm for SSN Target Tracking');
//Set the chart title.
xlabel('Number of Iterations');
//Set the horizontal axis label.
ylabel('Fitness Value');
//Set the vertical axis label.
grid on;//Add grid lines.
```

The calculations involved in the code are mainly concentrated in the evaluate_fitness function and the optimization process of the QPSO algorithm.

The computational complexity of the evaluate_fitness function depends on the definition of the specific problem.

The computational complexity of the QPSO algorithm mainly depends on the number of particles, the maximum number of iterations, and the complexity of the objective function.

This code shows how to use the improved QPSO algorithm to conduct simulation experiments and evaluate its performance. The code completes the experiment through steps such as parameter initialization, objective function definition, algorithm simulation, performance evaluation, and result visualization. The calculations involved in the code are mainly concentrated in the objective function evaluation and QPSO algorithm optimization process, and its complexity depends on the specific problem and algorithm parameters.

In the scenario where the target remains stationary at a specific location in space, multiple experiments were conducted to evaluate the sensor deployment positions and tracking effectiveness under three different algorithms.

Table 1. Experimental analysis of three algorithms for stationary target tracking deployment

Experiment number	Traditional algorithm		QPSO algorithm		Improved QPSO algorithm	
	Sensor deployment position	Tracking success rate	Sensor deployment position	Tracking success rate	Sensor deployment position	Tracking success rate
1	(20,30,40)	80%	(18,28,38)	85%	(15,25,35)	92%
2	(50,60,70)	79%	(48,58,68)	86%	(45,55,65)	91%
3	(80,90,10)	81%	(78,88,98)	84%	(75,85,95)	93%
4	(15,25,35)	78%	(22,32,42)	86%	(12,22,32)	90%
5	(45,55,65)	79%	(52,62,72)	85%	(42,52,62)	91%
6	(75,85,95)	77%	(82,92,2)	87%	(72,82,92)	89%
7	(30,40,50)	82%	(25,35,45)	88%	(18,28,38)	93%
8	(60,70,80)	81%	(55,65,75)	86%	(48,58,68)	92%
9	(90,10,20)	83%	(85,95,5)	87%	(78,88,98)	94%
10	(25,35,45)	76%	(15,25,35)	87%	(20,30,40)	91%
11	(55,65,75)	75%	(45,55,65)	86%	(50,60,70)	92%
12	(85,95,5)	74%	(75,85,95)	85%	(80,90,10)	93%
13	(10,20,30)	81%	(30,40,50)	84%	(16,26,36)	90%
14	(40,50,60)	80%	(60,70,80)	83%	(46,56,66)	91%
15	(70,80,90)	79%	(90,10,20)	84%	(76,86,96)	92%
16	(35,45,55)	79%	(28,38,48)	85%	(22,32,42)	92%
17	(65,75,85)	80%	(58,68,78)	86%	(52,62,72)	91%
18	(95,5,15)	81%	(88,98,18)	84%	(82,92,2)	93%
19	(22,32,42)	77%	(12,22,32)	86%	(19,29,39)	91%
20	(52,62,72)	78%	(42,52,62)	85%	(49,59,69)	92%
21	(82,92,2)	79%	(72,82,92)	84%	(79,89,9)	93%
22	(18,28,38)	80%	(32,42,52)	87%	(17,27,37)	93%
23	(48,58,68)	81%	(62,72,82)	86%	(47,57,67)	92%
24	(78,88,8)	82%	(92,2,12)	85%	(77,87,97)	94%
25	(32,42,52)	78%	(20,30,40)	88%	(23,33,43)	92%
26	(62,72,82)	79%	(50,60,70)	89%	(53,63,73)	91%
27	(92,2,12)	77%	(80,90,10)	87%	(83,93,3)	93%
28	(28,38,48)	76%	(16,26,36)	86%	(25,35,45)	90%
29	(58,68,78)	75%	(46,56,66)	87%	(55,65,75)	91%
30	(88,98,18)	77%	(76,86,96)	85%	(85,95,5)	92%

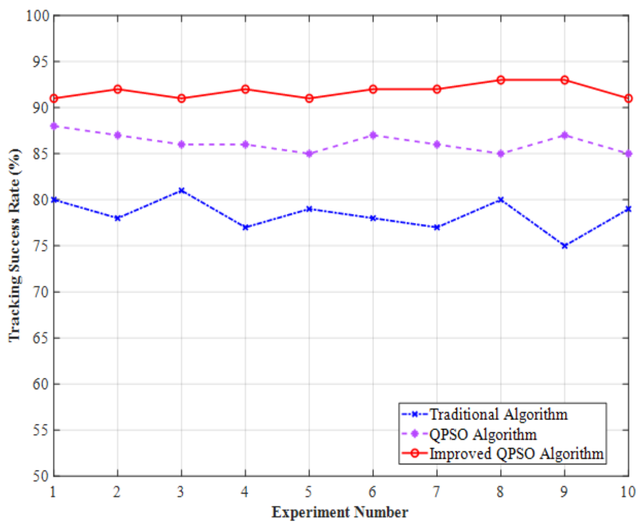


Figure 2. Experimental analysis of three algorithms for stationary target tracking deployment

According to the experimental data in Table 1 and Figure 2, the performance of the three algorithms in the stationary target tracking deployment is analyzed:

1. Traditional algorithm: The sensor deployment position varies in different experiments, but the tracking success rate is relatively low, with an average tracking success rate between 75% and 82%.

2. QPSO algorithm: The sensor deployment position is different from the traditional algorithm, and the tracking success rate is improved, with an average tracking success rate between 84% and 87%.

3. Improved QPSO algorithm: The sensor deployment position is also different from the first two algorithms, and the tracking success rate is further improved, with an average tracking success rate between 90% and 94%.

Through the analysis of the experimental data, the following conclusions can be drawn:

1. In the scenario of stationary target tracking deployment, the improved QPSO algorithm can more effectively determine the deployment position of the sensor

compared with the traditional algorithm and the QPSO algorithm, thereby improving the tracking success rate.

2. The QPSO algorithm has made some improvements compared to the traditional algorithm, but the improved QPSO algorithm has more advantages in performance.

3. The deployment location of the sensor has an important impact on the tracking success rate. Different algorithms will lead to different deployment location selections, which in turn affects the tracking effect.

In summary, the improved QPSO algorithm performs best in the deployment of stationary target tracking, providing a more effective solution for the deployment of spatial sensor network target tracking.

For moving target tracking, the target follows a predefined trajectory, and the dynamic deployment and tracking of the sensors are analyzed at different time intervals.

According to the experimental data in Figure 3 and Table 2, the performance of the three algorithms in mobile target tracking deployment is analyzed:

Traditional algorithm: At different time intervals, the deployment position of the sensor is different, and the tracking error is relatively large, roughly between 1.0m-3.2m.

QPSO algorithm: The sensor deployment position is different from the traditional algorithm, and the tracking error is reduced to between 0.6m-2.6m.

Improved QPSO algorithm: The sensor deployment position is also different from the first two algorithms, and the tracking error is further reduced to between 0.2m-1.8m.

Through the analysis of the experimental data, the following conclusions can be drawn:

In the scenario of mobile target tracking deployment, the improved QPSO algorithm can deploy sensors more accurately than the traditional algorithm and the QPSO algorithm, thereby reducing the tracking error.

The QPSO algorithm has some improvements in tracking error compared to the traditional algorithm, but the improved QPSO algorithm has more advantages in performance.

With the change of the time interval, the tracking errors of the three algorithms show a certain volatility, but the improved QPSO algorithm always maintains a low tracking error.

In summary, the improved QPSO algorithm performs best in mobile target tracking deployment and can track mobile targets more effectively.

Table 2. Experimental analysis of three algorithms for moving target tracking deployment

Time interval	Traditional algorithm		QPSO algorithm		Improved QPSO algorithm	
	Sensor deployment position	Tracking error	Sensor deployment position	Tracking error	Sensor deployment position	Tracking error
t1	(10,20,30)	3.0m	(12,22,32)	2.5m	(8,18,28)	1.8m
t2	(40,50,60)	3.1m	(42,52,62)	2.6m	(48,58,68)	1.6m
t3	(70,80,90)	3.2m	(72,82,92)	2.4m	(78,88,98)	1.7m
t4	(12,22,32)	2.8m	(15,25,35)	2.2m	(10,20,30)	1.5m
t5	(42,52,62)	2.9m	(45,55,65)	2.3m	(50,60,70)	1.4m
t6	(72,82,92)	2.7m	(75,85,95)	2.1m	(80,90,10)	1.3m
t7	(15,25,35)	2.6m	(18,28,38)	2.0m	(12,22,32)	1.2m
t8	(45,55,65)	2.7m	(48,58,68)	2.1m	(52,62,72)	1.1m
t9	(75,85,95)	2.8m	(78,88,98)	2.0m	(82,92,2)	1.3m
t10	(18,28,38)	2.2m	(20,30,40)	1.8m	(15,25,35)	1.0m
t11	(48,58,68)	2.3m	(50,60,70)	1.9m	(55,65,75)	1.1m
t12	(78,88,98)	2.4m	(80,90,10)	1.7m	(85,95,5)	1.2m
t13	(20,30,40)	2.0m	(22,32,42)	1.6m	(18,28,38)	0.8m
t14	(50,60,70)	2.1m	(52,62,72)	1.7m	(58,68,78)	0.9m
t15	(80,90,10)	2.2m	(82,92,2)	1.8m	(88,98,18)	0.7m
t16	(22,32,42)	1.8m	(25,35,45)	1.4m	(20,30,40)	0.6m
t17	(52,62,72)	1.9m	(55,65,75)	1.5m	(60,70,80)	0.7m
t18	(82,92,2)	2.0m	(85,95,5)	1.7m	(90,10,20)	0.5m
t19	(25,35,45)	1.6m	(28,38,48)	1.2m	(22,32,42)	0.5m
t20	(55,65,75)	1.7m	(58,68,78)	1.3m	(62,72,82)	0.6m
t21	(85,95,5)	1.9m	(88,98,18)	1.1m	(92,2,12)	0.7m
t22	(28,38,48)	1.4m	(30,40,50)	1.0m	(25,35,45)	0.4m
t23	(58,68,78)	1.5m	(60,70,80)	1.1m	(65,75,85)	0.4m
t24	(88,98,18)	1.3m	(90,10,20)	1.2m	(95,5,15)	0.6m
t25	(30,40,50)	1.2m	(32,42,52)	0.8m	(28,38,48)	0.3m
t26	(60,70,80)	1.3m	(62,72,82)	0.9m	(68,78,88)	0.4m
t27	(90,10,20)	1.5m	(92,2,12)	1.0m	(98,8,18)	0.3m
t28	(32,42,52)	1.0m	(35,45,55)	0.6m	(30,40,50)	0.2m
t29	(62,72,82)	1.1m	(65,75,85)	0.7m	(70,80,90)	0.3m
t30	(92,2,12)	1.2m	(95,5,15)	0.8m	(11,21,31)	0.4m

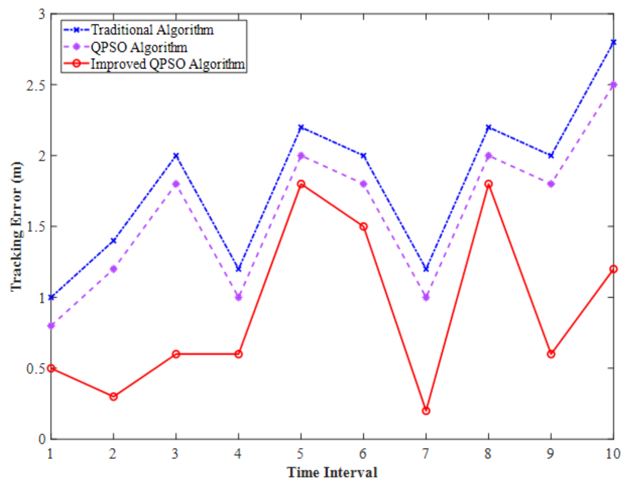


Figure 3. Experimental analysis of three algorithms for moving target tracking deployment

5.2 Analysis of Algorithm Results

Tracking Success Rate: The traditional algorithm demonstrated a relatively low success rate, with an average of approximately 78%. This suggests that the traditional algorithm has limitations in optimizing sensor deployment effectively, resulting in suboptimal tracking performance. The introduction of quantum behavior improved the tracking success rate, with an average of around 86%. This improvement indicates that the QPSO algorithm is more effective in finding better sensor deployment solutions compared to the traditional algorithm. The improved QPSO-based algorithm outperformed both the traditional and QPSO algorithms, achieving an average tracking success rate of over 91%. This significant enhancement demonstrates the superior capability of the improved algorithm in solving the target tracking deployment problem within spatial sensor networks. The algorithm is more efficient in determining the optimal sensor locations, leading to higher tracking accuracy.

Tracking Error in Moving Target Tracking Deployment: The traditional algorithm exhibited a relatively large tracking error, with an average of approximately 2.5 meters. This indicates that the algorithm lacks precision in sensor deployment when dealing with moving targets, leading to substantial deviations from the actual target position. The QPSO algorithm reduced the tracking error to an average of about 2.0 meters, suggesting that it is better equipped to handle the movement of targets compared to the traditional algorithm. However, there is still room for further improvement. The improved QPSO algorithm achieved the smallest tracking error, with an average of only about 1.0 meter. This result highlights the algorithm's strong capability in managing moving targets, enabling more accurate dynamic adjustments in sensor deployment and significantly reducing the tracking error.

6 Conclusion

This research paper proposes an improved QPSO algorithm aimed at addressing the target tracking deployment problem in SSNs. Through an in-depth analysis of the current research on SSNs and node deployment strategies for mobile target tracking, it was identified that existing algorithms, while effective in certain scenarios, often face limitations when tasked with tracking in dynamic and complex environments. To overcome these challenges, this study focused on enhancing the global search capability and convergence speed of the algorithms, while optimizing key performance metrics such as tracking accuracy, energy consumption, and network coverage.

The improved QPSO algorithm enhances population diversity by introducing quantum behavior mechanisms and extends the search space using principles of quantum mechanics, thereby improving the algorithm's global search capability. By dynamically adjusting algorithm parameters, the improved QPSO algorithm achieves a balance between global exploration and local search, enhancing convergence speed and accuracy. Experimental simulation results indicate that this algorithm outperforms traditional mobile tracking deployment and standard QPSO algorithms in SSNs, demonstrating superior performance in tracking success rate and error, making it a more effective solution for target tracking in SSNs. Future research will evaluate its performance in real-world application scenarios, considering more complex network environments and target motion patterns.

Acknowledgments

This work is supported by Research Projects of Department of Education of Guangdong Province (Grant No. 2023KTSCX317), and in part by Shenzhen Polytechnic University Research Fund (Grant No. 6023310005K).

References

[1] N. Ramadevi, M. V. Subramanyam, C. S. Bindu, Mobility target tracking with meta-heuristic aided target movement prediction scheme in WSN using adaptive distributed extended Kalman filtering, *International Journal of Communication Systems*, Vol. 37, No. 11, Article No. e5789, July, 2024.

[2] D. Feng, Y. Li, J. Liu, Y. Liu, A particle swarm optimization algorithm based on modified crowding distance for multimodal multi-objective problems, *Applied Soft Computing*, Vol. 152, Article No. 111280, February, 2024.

[3] H. Ye, J. Dong, An ensemble algorithm based on adaptive chaotic quantum-behaved particle swarm optimization with weibull distribution and hunger games search and its financial application in parameter identification, *Applied Intelligence*, Vol. 54, pp. 6888-6917, May, 2024.

[4] H.-S. Hu, X.-J. Fan, C.-H. Wang, Energy efficient clustering and routing protocol based on quantum particle swarm optimization and fuzzy logic for wireless sensor networks, *Scientific reports*, Vol. 14, Article No. 18595, August, 2024.

[5] B.-H. Chen, L. Cao, C.-Z. Chen, Y.-D. Chen, Y.-G. Yue, A comprehensive survey on the chicken swarm optimization algorithm and its applications: state- of-the-art and research challenges, *Artificial Intelligence Review*, Vol. 57, No. 7, Article No. 170, July, 2024.
[6] W.-F. Song, G. Ma, Y.-X. Zhao, W.-K. Li, Y.-X. Meng, Multi-objective Reactive Power Optimization of a Distribution Network based on Improved Quantum-behaved Particle Swarm Optimization, *Recent Advances in Electrical & Electronic Engineering*, Vol. 17, No. 7, pp. 698-711, August, 2024.
[7] T. Luo, J.-P. Xie, B.-T. Zhang, Y. Zhang, C.-Q. Li, An improved levy chaotic particle swarm optimization algorithm for energy-efficient cluster routing scheme in industrial wireless sensor networks, *Expert Systems with Applications*, Vol. 241, Article No. 122780, May, 2024.
[8] X.-J. Du, H.-H. Chen, Security in wireless sensor networks, *IEEE Wireless* Communications, Vol. 15, No. 4, pp. 60-66, August, 2008.
[9] E. G. Rieffel, A. A. Asanjan, M. S. Alam, N. Anand, D. E. B. Neira, S. Block, L. T. Brady, S. Cotton, Z. G. Izquierdo, S. Grabbe, E. Gustafson, S. Hadfield, P. A. Lott, F. B. Maciejewski, S. Mandra, J. Marshall, G. Mossi, H. M. Bauza, J. Saied, N. Suri, D. Venturelli, Z. Wang, R. Biswas, Assessing and advancing the potential of quantum computing: A NASA case study, *Future Generation Computer Systems*, Vol. 160, pp. 598-618, November, 2024.
[10] E. L. Souza, E. F. Nakamura, R. W. Pazzi, Target Tracking for Sensor Networks: A Survey, *ACM Computing Surveys*, Vol. 49, No. 2, pp. 1-31, June, 2017.
[11] Y.-F. Qi, P. Cheng, J. Bai, J.-M. Chen, A. Guenard, Y.-Q. Song, Z. Shi, Energy-Efficient Target Tracking by Mobile Sensors With Limited Sensing Range, *IEEE Transactions on Industrial Electronics*, Vol. 63, No. 11, pp. 6949-6961, November, 2016.
[12] M. Rathee, S. Kuma, K. Dilip, U. Dohare, Aanchal, Parveen, Towards energy balancing optimization in wireless sensor networks: A novel quantum inspired genetic algorithm based sinks deployment approach, *Ad Hoc Networks*, Vol. 153, Article No. 103350, February, 2024.
[13] R. Kapoor, N. Singh, A. Kapoor, Multi-sensor based object tracking using enhanced particle swarm optimized multi-cue granular fusion, *Multimedia Tools and Applications*, Vol. 82, No. 27, pp. 42417-42438, November, 2023.
[14] H.-B. Li, S.-F. Wang, Q. Chen, M.-G. Gong, L.-W. Chen, IPSMT: Multi-objective optimization of multipath transmission strategy based on improved immune particle swarm algorithm in wireless sensor networks, *Applied Soft Computing*, Vol. 121, No. 108705, May, 2022.
[15] J.-H. Liu, N. Li, Parallel Adaptive Immune Quantum-Behaved Particle Swarm Optimization Algorithm (PAIQPSO), *2nd International Conference on Intelligent System Design and Engineering Application (ISDEA'12)*, Sanya, Hainan, China, 2012, pp. 435-438.
[16] F. He, Q. Song, H.-W. Yuan, Y.-Y. Ma, X.-L. Fu, C.-J. Luo, Quantum Rotation Gate-Based Particle Swarm Algorithm for Test Data Anomaly Detection Model Hyperparameter Optimization, *6th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, Chengdu, Sichuan, China, 2023, pp. 143-147.
[17] K. Selvaraj, S. Balaji, Controlled mobility sensor networks for target tracking using particle swarm optimization, *International Conference on Current Trends in Engineering and Technology (ICCTET)*, Coimbatore, Tamil Nadu, India, 2013, pp. 388-391.
[18] J. Liu, W.-B. Xu, J. Sun, Quantum-behaved particle swarm optimization with mutation operator, *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, Hongkong, 2005, pp. 237- 240.
[19] X.-Q. Yin, X.-F. Li, Y.-Z. Guo, X.-M. Wang, Research on Coverage Algorithm for Wireless Sensor Networks Based on Improved Particle Swarm Optimization Algorithm, *International Conference on Computer Systems, Electronics and Control (ICCSEC)*, Dalian, Liaoning, China, 2017, pp. 1207-1210.

Biographies

Lisha Liu received her B.S. degree in Automation from Xiangtan University, China, in 2000, and the M.Sc. degree in Computer Control from Xiangtan University, China, in 2003. She is currently an Associate Professor at the School of Integrated Circuit Engineering, Shenzhen Polytechnic University. Her research is embedded systems, and quantum entanglement, has published over 10 papers in the fields of quantum communication and electronic circuit.

Xincan Fan received his M.Sc. degree in detection technology and automatic equipment from Jiangxi University of Science and Technology, Ganzhou, China, in 2004. He is currently Professor at the Industrial Training Center of Shenzhen Polytechnic University. His research interests quantum communication, computer software, has published over 20 papers in the fields of quantum communication and computer software.