Postprint of WSN Coverage Optimization Based on Particle Swarm Algorithm

Authors: Guo Chao, Yang Yuxuan, Ronglei Hu, Xiao Chao' en, Wang Jianxin, Xu Fenglin

Date: 2019-01-28T00:00:00+00:00

Abstract

Particle swarm optimization-based network coverage algorithms are widely employed owing to their advantages of few parameters, fast computational speed, and relatively simple implementation; nevertheless, they still exhibit defects such as slow convergence speed and susceptibility to local optima, resulting in "premature convergence." To enhance wireless sensor network performance, this study investigates node distribution and coverage schemes, integrating artificial gravitational force and artificial Coulomb force from artificial physics optimization algorithms with particle swarm optimization to propose an artificial physics particle swarm optimization algorithm based on inertia weight. This approach strengthens the algorithm' s global search capability, accelerates convergence to the global optimum, and reduces algorithmic time consumption and redundant coverage. Simulation results substantiate that the proposed algorithm achieves superior global convergence speed, higher coverage rate, and lower redundant coverage ratio compared with both the basic particle swarm optimization and the standard particle swarm optimization algorithm based on inertia weight.

Full Text

Preamble

Vol. 37 No. 4
Application Research of Computers
ChinaXiv Partner Journal

WSN Coverage Optimization Based on Particle Swarm Optimization

Guo Chao, Yang Yuxuan, Hu Ronglei, Xiao Chaoen, Wang Jianxin, Xu Fenglin

(Department of Electronics & Communication Engineering, Beijing Electronic Science & Technology Institute, Beijing 100070, China)

Abstract: The particle swarm optimization (PSO) coverage algorithm is widely applied due to its advantages of fewer parameters, faster computation speed, and relatively simple implementation. However, it still suffers from defects such as slow convergence speed and susceptibility to falling into local optima, leading to "premature" convergence. To improve wireless sensor network performance, this paper investigates node distribution and coverage schemes, combining quasi-gravitational and quasi-Coulomb forces from quasi-physical algorithms with particle swarm optimization to propose a quasi-physical particle swarm algorithm based on inertia weight. This approach enhances global search capability, enables faster convergence to the global optimum, and reduces algorithmic time consumption and redundant coverage. Simulation results demonstrate that the new algorithm achieves faster global convergence, higher coverage rates, and lower redundant coverage ratios compared to both basic PSO and standard PSO with inertia weight.

Keywords: wireless sensor network; particle swarm optimization; quasi-physical algorithm; node coverage

Classification: TN915.06
doi: 10.19734/j.issn.1001-3695.2018.10.0767

0 Introduction

Wireless sensor networks (WSNs) are wireless self-organizing networks composed of distributed sensors, designed for multi-node, multi-task operations [1]. The objective of WSN coverage optimization is to achieve seamless, complete coverage of designated areas [2]. In most scenarios, the physical environment of target regions is relatively harsh, making deterministic deployment through manual means difficult to implement. Consequently, only dynamic nodes can satisfy the complex environmental requirements of test areas. However, dynamic node deployment exhibits high randomness, cannot guarantee full coverage of target regions, and struggles to achieve expected coverage quality. While particle swarm algorithms can improve coverage rates in dynamic networks [3], they remain prone to premature convergence and repeated coverage, leading to local optima problems [4]. This paper introduces quasi-physical algorithms into the PSO-based network coverage model and proposes a quasi-physical optimization-based PSO coverage algorithm.

1.1 Basic Particle Swarm Algorithm

Particle Swarm Optimization (PSO) [5], proposed by Kennedy and Eberhart, resembles genetic algorithms [6] in computational methods but differs by eschewing genetic operators such as crossover and mutation. Instead, PSO simulates the behavior of herds, flocks, schools of fish, or other biological group behaviors for searching [7]. The algorithm offers conceptual simplicity, fewer control parameters, easy implementation, and good parallelism [8].

Similar to other evolutionary algorithms, PSO employs concepts of "population" and "evolution," achieving optimal solution searches in complex spaces through individual cooperation and competition [9]. PSO first generates an initial population by randomly initializing a swarm of particles in the feasible solution space, where each particle represents a feasible solution to the optimization problem and receives a fitness value determined by the objective function. Distinguishing itself from other evolutionary algorithms, PSO does not apply evolutionary operators to individuals but treats each individual as a volumeless, weightless particle in an n-dimensional search space. Each particle moves in the solution space with a velocity determining its direction and distance. Particles follow the current optimal particle while moving, conducting searches across generations to ultimately obtain the optimal solution.

In each generation, particles track two extrema: one is the optimal solution found by the particle itself thus far, and the other is the optimal solution found by the entire population thus far. Recognizing the broad application prospects of PSO in function optimization and other fields, numerous scholars have conducted research following Kennedy and Eberhart' s work, resulting in various improved PSO algorithms widely applied across many domains [10].

The PSO algorithm iterates particle velocity and position using the following formulas:

$$v_{ij}(n+1) = v_{ij}(n) + c_1 r_1[p_{id} - x_{ij}(n)] + c_2 r_2[p_{gd} - x_{ij}(n)] \tag{1}$$

$$x_{ij}(n+1) = x_{ij}(n) + v_{ij}(n+1) \tag{2}$$

where $v_{ij}(n)$ represents the particle' s previous velocity, which exhibits a self-development trend capable of expanding the search space and exploring new domains. This provides the algorithm with global optimization capability but may affect fine-grained local search during later iterations. The term $c_1 r_1[p_{id}-x_{ij}(n)]$ constitutes the particle' s cognitive component, representing self-cognition after each iteration—i.e., learning from its own experience. The term $c_2 r_2[p_{gd}-x_{ij}(n)]$ represents the particle' s social component, demonstrating information sharing among particles—i.e., learning from the entire population. These three components collectively determine the search capability of particles in feasible space: the first balances global and local search abilities, the second enables strong local

search capability, and the third represents inter-particle information exchange. In Equation (1), particles update their velocity based on the k-th iteration' s velocity and cognition, while Equation (2) controls particles' movement to new positions.

The PSO algorithm proceeds as follows:

a) Initialize algorithm parameters, randomly generate each particle' s position x_{id} within bounds, and randomly generate initial particle velocities v_{id}.

b) Calculate each particle' s fitness value F_{id} after iteration.

c) For particles in the region, compare fitness value F_{id} with individual best position p_{id}. If $F_{id} > p_{id}$, assign F_{id} to p_{id}.

d) For the population in the region, compare fitness value F_{id} with global best position p_{gd}. If $F_{id} > p_{gd}$, assign F_{id} to p_{gd}.

e) Update particle velocity and position according to the formulas.

f) If termination conditions are satisfied, output results; otherwise, repeat steps b)-e).

1.2 Inertial Particle Swarm Algorithm

Two concepts frequently used in PSO are: first, *exploration*, which means particles depart from original search trajectories to some extent, seeking new directions—reflecting the ability to develop into unknown regions, similar to global search; second, *exploitation*, which means particles continue more detailed searches along original trajectories to some degree, primarily referring to further searching of regions explored during exploration, similar to local search.

The population seeks optimal solutions through exploration and exploitation. How to control these two search processes directly affects PSO efficiency. To better control these searches, Shi et al. [11] introduced inertia weight into the original PSO algorithm, proposing the inertia weight PSO algorithm. Its velocity update formula is:

$$v_{ij}(n+1) = wv_{ij}(n) + c_1 r_1[p_{id} - x_{ij}(n)] + c_2 r_2[p_{gd} - x_{ij}(n)] \tag{3}$$

When $w = 0$, particle velocity depends solely on p_{id} and p_{gd}. During iteration, particles already at the global optimal position remain stationary, while other particles fly toward the weighted direction of p_{id} and p_{gd}. Once the population converges to p_{gd}' s position, the algorithm enters a state of local convergence.

When $w \neq 0$, the current particle' s movement direction and velocity are influenced by its previous state, acquiring the ability to expand search space and explore new regions. By adjusting w, the algorithm can balance global and local search capabilities. The inertia weight w is typically a constant value, often in the range $[0.9, 1.2]$. Moreover, inertia weight can be dynamically adjusted during the search process: a larger positive value can be given at the algorithm' s beginning, gradually decreasing linearly as the search progresses. This ensures that during the initial phase, each particle can quickly detect better regions globally, while in later stages, smaller weight values ensure fine-grained searches around extreme points, giving the algorithm greater convergence probability toward the global optimum. The currently more widely used dynamic linear weight expression is:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T_{\max}} \times t \tag{4}$$

where $T_{\max}$ represents maximum iteration count, $w_{\max}$ represents maximum inertia weight, $w_{\min}$ represents minimum weight, and t represents current iteration count. In most cases, $w_{\max} = 0.9$ and $w_{\min} = 0.4$.

2.1 Overview of Quasi-Physical Algorithm

The quasi-physical algorithm was first proposed by Huang Wenqi [12] to efficiently solve certain NP-complete problems, representing an optimization algorithm inspired by nature. This section combines quasi-physical algorithms with PSO, proposing a quasi-physical algorithm-based PSO that improves particle velocity updates through quasi-physical forces. This enables particles to change direction, accelerate convergence, and reasonably adjust inter-particle distances, thereby reducing redundant coverage. Compared with basic PSO, the quasi-physical force-guided PSO exhibits stronger global search capability, faster convergence to global optimum, reduced algorithmic time consumption, and decreased redundant coverage.

1) Quasi-Gravitational Model [13]

Assume a plane in a horizontal position and absolutely smooth, with a finite number of points imagined as unit-mass particles fixed in this plane. Consider each disk as absolutely smooth, unit-mass, and uniformly distributed around its circumference. If these disks are randomly placed on this smooth plane, due to gravitational force between particles and disks, each particle attracts disks to cover itself, thus solving the coverage problem.

However, to solve coverage problems with limited disks, we must conserve disk usage. The strategy involves informing each particle that if it has already been covered (i.e., coverage requirements are satisfied), it should no longer attract any disks, thereby providing opportunities and resources for other particles.

This strategy' s spirit aligns with the finite damage priority method in modern mathematical logic recursion theory and possesses a physical background—the shielding phenomenon.

According to the law of universal gravitation, after ignoring some purely mechanical details and integrating them, we obtain the gravitational potential of a simple system composed of particles and disks:

$$U = \sum_{i=1}^{M} \sum_{j=1}^{m} u_{ij} \tag{5}$$

where u_{ij} represents the potential energy of the whole formed by the i-th particle and j-th disk, and the summation over particles only includes uncovered particles, no longer calculating already covered particles.

From the above equation, potential energy U is always non-negative under any circumstances. When $U > 0$, it indicates that disk positions do not satisfy coverage conditions for all particles; when $U = 0$, it indicates that disk positions have satisfied coverage requirements for all particles. In this case, states satisfying $U = 0$ are feasible points in the state space. Defining U as the state space' s guiding function and correcting it after algorithmic computation yields feasible points in the state space, where the array contained in this feasible point represents disk placement positions.

Two main contradictions affect the application of quasi-physical algorithms to coverage problems: first, why no gravitational force exists between disks; second, why disks in the same planar space can overlap and embed without any resistance or obstruction, contradicting the impenetrability of physical objects. These contradictions add considerable difficulty to quasi-physical algorithms for coverage problems.

In nature, many examples exist where substances have minimal interaction with their own kind but produce significant forces with other substances—iron and magnet interaction being one such case. We can thus assume that particles and disks are composed of these two substances, resolving the first contradiction.

Regarding the second contradiction, a seemingly far-fetched idea can explain it: stretch the original plane into countless parallel planes with very small intervals, with all particles embedded in one plane and other disks distributed across different planes. Under this assumption, disk movement produces no mutual influence, and what we observe is the effect of all planes projecting onto the same plane—precisely the image produced by disks moving under particle influence while overlapping and embedding.

Having resolved these issues, we can confirm that quasi-physical algorithms can be properly applied to coverage problems. Following this approach, we consider the exploration area as having infinitely uniform $m \times n$ pixels, with each pixel treated as a particle, forming $m \times n$ particles. Each sensor node

is regarded as a regular disk with radius r. Assuming every uncovered particle exerts gravitational force on disks within a certain range, while covered particles exert no force, we obtain the "quasi-gravitational force" expression:

$$f_{jk}^{g} = \begin{cases} \frac{1}{d_{jk}^{2}}, & 0 < r < R \text{ and } d_{jk} > r \\ 0, & \text{else} \end{cases} \tag{6}$$

where d_{jk} represents the distance between disk and particle, r represents disk radius, and R limits gravitational force to only uncovered particles and nearby disks.

2) Quasi-Coulomb Force Model [14]

Similarly, assume a plane in a horizontal position and absolutely smooth, with a finite number of points imagined as unit-mass particles fixed in this plane, and a finite number of absolutely smooth disks scattered uniformly across the plane.

If these disks are randomly placed on this smooth plane, due to Coulomb repulsion between disks, each disk repels other disks that are too close, thus solving redundant coverage problems.

Assuming all particles are covered by disks, we need to control overlapping coverage among disks to minimize redundant coverage rates. The strategy involves informing each disk that when all particles on the plane are covered, it should begin repelling other disks that are too close, thereby reducing overlapping areas.

According to Coulomb' s law, after ignoring some purely mechanical details and integrating them, we obtain the repulsive potential function of a simple system composed of disks:

$$U = \sum_{i=1}^{m} u_i \tag{7}$$

where u_i represents the potential energy of the whole formed by disks, and this expression only takes effect when coverage conditions are satisfied, being invalid when coverage is incomplete.

From the above equation, potential energy U is always non-negative. When $U > 0$, it indicates that relative distances between disks are too close, though smaller U means less redundant coverage in principle; when $U = 0$, it indicates no redundant coverage exists among disks, though this generally does not occur in practice. Therefore, we only need to make U approach 0 as closely as possible.

Following this approach, we define quasi-Coulomb force as a repulsive force with far lower priority than quasi-gravitational force. It only takes effect when all particles are covered (i.e., when no quasi-gravitational force exists), becoming the dominant force controlling disk movement. Quasi-Coulomb force achieves

more uniform disk coverage and reduces unnecessary redundant coverage by repelling two disks that are too close. The "quasi-Coulomb force" expression is:

$$f_{ij}^{d} = \begin{cases} \frac{1}{d_{ij}^{2}}, & d_{ij} < R_d \\ 0, & \text{else} \end{cases} \tag{8}$$

where r_i and r_j represent the masses of two disks respectively, d_{ij} represents the distance between disk centers, and R_d limits the control range of quasi-Coulomb force.

Applying quasi-physical force theory to node coverage problems in sensor networks, using universal gravitation and Coulomb force to achieve node attraction and repulsion, can optimize node position distribution within the population under the basic PSO framework, enabling sensor nodes to more effectively achieve efficient target area coverage. In WSNs, nodes need to receive, store, and forward data—isolated nodes are meaningless, while overly dense nodes waste network resources. Thus, quasi-physical force theory can appropriately optimize inter-node distances.

2.2 Quasi-Physical Optimization PSO Coverage Scheme Design

This paper combines quasi-physical force algorithms with PSO, proposing a quasi-physical force-guided PSO algorithm. In PSO, particle convergence speed is influenced to some extent by initialized positions and velocities, particularly noticeable when particle numbers are relatively small. To accelerate particle swarm convergence and guide rapid convergence, we incorporate quasi-physical force influence into particle velocity iterations based on quasi-physical force guidance strategies. The velocity expression updates as follows:

$$v_{ij}(n+1) = wv_{ij}(n)+c_1r_1[p_{id}-x_{ij}(n)]+c_2r_2[p_{gd}-x_{ij}(n)]+c_3r_3g_{ij}^{n}+c_4r_4g_{ij}^{d} \tag{9}$$

where w, c_1, c_2, r_1, and r_2 are identical to those in Equation (3); c_3 is the quasi-gravitational acceleration coefficient, c_4 is the quasi-Coulomb force acceleration coefficient, and r_3, r_4 are random numbers between 0 and 1.

g_{ij}^{n} represents the resultant quasi-gravitational force, calculated by first computing the gravitational function value of each sensor to a certain particle, then summing the gravitational function values of all particles:

$$g_{ij}^{n} = \sum_{k=1}^{n} \frac{r^2}{d_{jk}^2} \tag{10}$$

g_{ij}^d represents the resultant quasi-Coulomb force, calculated by first computing the Coulomb function value of each node to other sensor nodes, then calculating the total Coulomb force function value between this node and other nodes. Its priority is lower than quasi-gravitational force:

$$g_{ij}^d = \sum_{j=1}^{n} \frac{r^2}{d_{ij}^2} \tag{11}$$

After integration, the repulsive potential function of the simple system between disks can be obtained as:

$$U = \sum_{i=1}^{m} \sum_{j=1}^{m} u_{ij} \tag{12}$$

Introducing quasi-gravitational and quasi-Coulomb forces into basic PSO can reduce redundant coverage and accelerate convergence speed. The basic flowchart is shown in [Figure 1: see original paper].

The algorithm proceeds as follows:

a) Initialize algorithm parameters, randomly generate each particle' s position x_{id} and initial velocity v_{id} within bounds, and calculate original coverage rates.

b) Update particle velocity and position according to basic PSO steps, and recalculate coverage rates.

c) If updated particle position coverage exceeds historical individual optimal position, set historical individual optimal position to current coverage.

d) Find the global optimal position based on each particle' s individual optimal position.

e) Determine whether the area is fully covered—if covered, apply quasi-Coulomb force once; if not fully covered, apply quasi-gravitational force once.

f) If termination conditions are satisfied, output results; otherwise, repeat steps b)-e).

2.3 Inertial Weight Quasi-Physical PSO Algorithm Design

Building upon basic PSO development, we combine inertia weight with the quasi-physical PSO algorithm, further refining the velocity update formula:

$$v_{ij}(n+1) = wv_{ij}(n)+c_1r_1[p_{id}-x_{ij}(n)]+c_2r_2[p_{gd}-x_{ij}(n)]+c_3r_3g_{ij}^n+c_4r_4g_{ij}^d \quad (13)$$

Through this formula, inertia weight is introduced into PSO with quasi-physical algorithms, adding particle inertia' s influence on velocity. This allows timely adjustment of particles' global and local search capabilities while maintaining accelerated optimization and reduced coverage. As shown in subsequent simulation results, larger w values yield stronger global optimization but weaker local optimization, and vice versa. The inertia weight value is a constant. Additionally, inertia weight can be dynamically adjusted during the search process: a larger positive value can be given at the algorithm' s start, gradually decreasing linearly as search progresses. This ensures that during the initial phase, each particle can quickly detect better regions globally, while in later stages, smaller weight values ensure fine-grained searches around extreme points, giving the algorithm greater convergence probability toward the global optimum. The expression is:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{T_{\max}} \times t \quad (14)$$

where $T_{\max}$ represents maximum iteration count, $w_{\max}$ represents maximum inertia weight, $w_{\min}$ represents minimum weight, and t represents current iteration count. Because quasi-physical forces accelerate convergence, to reduce inertia weight' s interference with convergence, data compression is necessary. In most cases, $w_{\max} = 0.7$ and $w_{\min} = 0.3$.

3.1 Simulation Environment Setup

This paper uses MATLAB as the simulation tool to test the improved algorithm' s performance and compare it with basic PSO. To accurately test and improve the hybrid algorithm' s performance and ensure fair comparison conditions, consistent parameters are set when simulating various algorithms. In algorithm coverage range simulation and comparison, each algorithm runs 10 times to obtain average values reflecting coverage performance. Simulation parameters are shown in .

Table 1 Simulation Parameters

Parameter	Value
Population size (pop)	30
Iterations (maxgen)	1000
Area	100×100 m²
Sensor node sensing radius (r)	12 m

Parameter	Value
Particle maximum velocity (vmax)	20 m/s
Individual learning factor (c)	2.0
Social learning factor (c)	2.0

3.2 Simulation Results and Analysis

1) Network Coverage Rate

To verify the improved algorithm' s coverage effect, this paper assumes wireless sensor network nodes with a 12 m radius in a homogeneous network. Simulation results are shown in the following figures: [Figure 2: see original paper] shows the initial sensor node coverage diagram, [Figure 3: see original paper] shows basic PSO simulation coverage, [Figure 4: see original paper] shows standard PSO with inertia weight simulation coverage, and [Figure 5: see original paper] shows quasi-physical PSO with inertia weight simulation coverage. The figures use rectangles to represent the monitored area of the wireless sensor network, "+" symbols to indicate individual sensor node positions, and circles to represent coverage ranges of individual sensor nodes within the monitoring area.

As shown in Figures 2–5, wireless sensor network nodes are initially randomly deployed in the monitoring area. After optimization by basic PSO and standard PSO with inertia weight, sensor node distribution and overall network monitoring coverage improve compared to initial conditions. However, uneven particle distribution still causes overlapping coverage areas and numerous coverage blind spots. After optimization by quasi-physical PSO with inertia weight, node distribution becomes more uniform, significantly reducing both redundant coverage areas and coverage blind spots. The improved algorithm demonstrates noticeably better regional coverage effectiveness compared to basic algorithms.

2) Convergence Speed

The previous section presented coverage simulation diagrams of the three algorithms under 1,000 iterations, with coverage variation curves shown in [Figure 6: see original paper]. Simulation results indicate that basic PSO tends to converge after 102 iterations, achieving 76.8% coverage with 43.4% redundant coverage. In standard PSO with inertia weight, the dynamic inertia coefficient ensures fine-grained searches around extreme points during later iterations. In simulation, this manifests as continuing refined searches after reaching 76.8% coverage at 48 iterations, ultimately achieving 84.4% coverage peak at 265 iterations with 37.5% redundant coverage. In quasi-physical PSO with inertia weight, the "quasi-gravitational" pull and "quasi-Coulomb" repulsion enable the particle swarm to reach convergence faster and greatly reduce redundant coverage, surpassing 95% coverage in just 30 iterations and achieving 96.6% coverage

after 469 iterations of fine-grained search, with 28.8% redundant coverage. Simulation data demonstrate that under identical particle numbers and iteration counts, quasi-physical PSO with inertia weight offers larger regional coverage and better coverage effectiveness compared to both basic PSO and standard PSO with inertia weight.

4 Conclusion

To enhance wireless sensor network deployment effectiveness, expand WSN coverage areas under equivalent node counts, improve WSN coverage rates, and significantly enhance coverage effectiveness, this paper integrates inertia weight and quasi-physical algorithms into PSO through research and analysis, proposing a quasi-physical PSO algorithm based on inertia weight. This algorithm incorporates the late-stage fine-grained exploration capability from inertia weight and the rapid convergence and redundant coverage reduction capabilities from quasi-physical algorithms, simultaneously possessing the superiority of both individual algorithms and achieving, to some extent, an effect where the whole exceeds the sum of its parts.

To validate the effectiveness of quasi-physical PSO with inertia weight, we conducted simulation analyses from three aspects: coverage performance, convergence speed, and redundant coverage rate, comparing results with both basic PSO and standard PSO with inertia weight. Simulation results fully demonstrate that compared to the two existing algorithms, quasi-physical PSO with inertia weight achieves faster global convergence, higher coverage rates, and lower redundant coverage ratios, avoiding to some extent the problem of falling into local optima due to algorithm "premature" convergence during iterations, and greatly enhancing WSN deployment effectiveness.

References

[1] Mostafaei H, Montieri A, Persico V, et al. A sleep scheduling approach based on learning automata for WSN partial coverage [J]. Journal of Network & Computer Applications, 2017, 80 (C): 67-78.
[2] Qiu Chenxi, Shen Haiying, Chen Kang. An energy-efficient and distributed cooperation mechanism for k-coverage hole detection and healing in WSNs [J]. IEEE Trans on Mobile Computing, 2018, 17 (6):
[3] Song Mingzhi, Yang Le. Application of improved VFPSO algorithm in WSN node random deployment [J]. Computer Engineering and Application, 2016, 52 (2): 141-145.
[4] Rauf A, Aleisa E A. PSO based automated test coverage analysis of event driven systems [J]. Intelligent Automation & Soft Computing, 2015, 21 (4): 491-502.

[5] Kennedy J, Eberhart R. Particle swarm optimization [C]// Proc of International Conference on Neural Networks. Piscataway, NJ: IEEE Press, 2002: 1942-1948.
[6] Srinivas M, Patnaik L M. Genetic algorithms: a survey [J]. Computer, 2002, 27 (6): 17-26.
[7] Khatami A, Mirghasemi S, Khosravi A, et al. A new PSO-based approach to fire flame detection using K-Medoids clustering [J]. Expert Systems with Applications, 2017, 68 (C): 69-80.
[8] Mistry K, Zhang L, Neoh S C, et al. A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition [J]. IEEE Trans on Cybernetics, 2017, 47 (6): 1496-1509.
[9] Kumar S U, Inbarani H H. PSO-based feature selection and neighborhood rough set-based classification for BCI multiclass motor imagery task [J]. Neural Computing & Applications, 2017, 28 (11):
[10] Garg H. A hybrid PSO-GA algorithm for constrained optimization problems [J]. Applied Mathematics & Computation, 2016, 274 (11):
[11] Shi, Y, Eberhart R C. Empirical study of particle swarm optimization [C]// Proc of Congress on Evolutionary Computation. 1999: 1945–1950.
[12] Huang Wenqi, Zhao Xiaowu. Simulation algorithm for orthogonal array problem [J]. Computer Research and Development, 2002, 39 (2): 205-212.
[13] Cheng Aihua, Ge Baozhong, Ji Zhongheng. Simulation and humanization optimization of wireless sensor network area coverage [J]. Journal of Sensor Technology, 2007, 20 (12): 2668-2673.
[14] Tao Xiaoshou, Li Xunbo. Improved round routing algorithm based on area covering of wireless sensor network [C]//Proc of International Conference on Electrical and Control Engineering. Piscataway, NJ: IEEE Press, 2010: 5274-5277.

Note: Figure translations are in progress. See original paper for figures.