Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 1676879, 15 pages
https://doi.org/10.1155/2021/1676879

WILEY | Hindawi

Research Article

A Compact Adaptive Particle Swarm Optimization Algorithm in the Application of the Mobile Sensor Localization

Wei-Min Zheng, Ning Liu, Qing-Wei Chai, and Shu-Chuan Chu

College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

Correspondence should be addressed to Shu-Chuan Chu; scchu0803@gmail.com

Received 12 September 2021; Accepted 20 October 2021; Published 9 November 2021

Academic Editor: Chi-Hua Chen

The mobile sensor network can sense and collect the data information of the monitored object in real time in the monitoring area. However, the collected information is meaningful only if the location of the node is known. This paper mainly optimizes the Monte Carlo Localization (MCL) in mobile sensor positioning technology. In recent years, the rapid development of heuristic algorithms has provided solutions to many complex problems. This paper combines the compact strategy into the adaptive particle swarm algorithm and proposes a compact adaptive particle swarm algorithm (cAPSO). The compact strategy replaces the specific position of each particle by the distribution probability of the particle swarm, which greatly reduces the memory usage. The performance of cAPSO is tested on 28 test functions of CEC2013, and compared with some existing heuristic algorithms, it proves that cAPSO has a better performance. At the same time, cAPSO is applied to MCL technology to improve the accuracy of node localization, and compared with other heuristic algorithms in the accuracy of MCL, the results show that cAPSO has a better performance.

1. Introduction

Metaheuristic algorithms are algorithms inspired by the life habits of various creatures in nature. Metaheuristic algorithms can effectively solve many problems in life [1] and are widely used in finance, transportation, physics, chemistry, military, and other fields [2–8]. The No Free Lunch Theorem [9, 10] proves that any optimization algorithm cannot suit all situations. Therefore, various metaheuristic algorithms and their improved algorithms are constantly being proposed to solve more complicated problems [11–13].

The particle swarm optimization (PSO) is one of the most important metaheuristic algorithms proposed by Kennedy and Eberhart [14, 15]. They observed and analyzed the foraging behavior of birds and then proposed this algorithm. There is a global optimal position and an individual optimal position in the particle swarm optimization algorithm. These two positions are updated according to the fitness value in each iteration, so that the algorithm is closer to the optimal solution of the problem. The characteristics of PSO, such as few parameters, simple structure, and fast search speed, make it applied to many fields. There are also many improved PSO algorithms, such as constricted particle swarm optimization (CPSO) [16], fully informed particle swarm optimization (FIPSO) [17], comprehensive learning particle swarm optimization (CLPSO) [18], intelligence single particle optimization (ISPO) [19], and adaptive particle swarm optimization (APSO) [20]. The PSO can solve many problems, such as optimizing neural networks [21], solving vehicle routing problems [22, 23], scheduling workflow scheduling [24], and locating wireless sensor nodes [25]. Based on the original PSO algorithm, the APSO algorithm introduces evolutionary state evaluation strategies, elite learning strategies, and system adaptive parameter strategies to improve the original PSO algorithm. The APSO solves the problem of slow convergence speed and easy to fall into the local optimum of the original PSO algorithm. This paper mainly tries to combine the compact strategy with APSO to improve the accuracy of mobile sensor localization.

The application of a mobile sensor network is to provide services to people when the location information of the node is known [26]. The data measured by the node without

location information is meaningless in many situations [27], such as forest fire detection [28]. Therefore, to make full use of the monitored data, it is necessary to know the location information of the node [29, 30]. Installing GPS for each node is the best way to solve this problem, but this way is expensive and energy-consuming. Therefore, a small number of nodes should be randomly selected to install GPS, and then, the positions of other nodes should be obtained by using positioning technology through the location of the GPS nodes [31]. The most significant difference between a mobile sensor and a fixed wireless sensor is its mobility [32, 33]. The mobility enables the sensor to collect effective information in the specified area better and solves the problem that the information in a certain area cannot be collected due to the damage of a specific location node. The deployment of mobile sensors is more convenient and does not require detailed design like fixed wireless sensor deployment nodes [34].

The compact idea is to use the behavior probability of the particle swarm to replace the position and velocity of each individual to express the particle swarm. The compact algorithm can effectively save memory space [20] and has applications in small robots [35], remote office [36], and space shuttle control [37]. Algorithms using compact ideas have also been continuously proposed, such as compact artificial bee colony (cABC) [38], compact sine cosine algorithm (cSCA) [39], compact bat algorithm (cBA) [40], and compact particle swarm optimization (cPSO) [41]. However, the combination of compact algorithm and APSO has not been mentioned. This paper hopes to combine the idea of compact with APSO and propose a cAPSO algorithm that uses small memory and fast convergence and applies it to mobile sensor localization.

The rest of the paper is organized as follows. The second section introduces the basis of related work and briefly introduces APSO algorithm and mobile sensor localization technology. The third section presents the improvement methods and steps of the algorithm. The fourth section tests the performance of the algorithm and compares it with similar algorithms. The fifth section applies the improved algorithm to mobile sensor localization. The sixth section gives the conclusion of this paper.

2. Related Work

This section will briefly introduce APSO and mobile sensor localization technology MCL.

2.1. Particle Swarm Optimization. The PSO imitates the foraging behavior of birds. The food location is unknown, and each bird is affected by the surrounding birds and keeps approaching the bird with the best position during the foraging process [14, 15]. Depending on the solution of the problem, the position of the optimal individual is constantly updated iteratively. Suppose the problem dimension is D, the current position of the i-th individual is $X_i = (X_{i1}, X_{i2}, \cdots, X_{iD})$, and the current speed of the i-th individual is $V_i = (V_{i1}, V_{i2}, \cdots, V_{iD})$. In each iteration, the particle swarm retains the global optimal position and the current optimal position, and each particle is affected by these two optimal particles and approaches these two particles. The iterative formulas for updating the position and velocity of the next generation of particles are shown in Equations (1) and (2).

$$V_i^{g+1} = wV_i^g + c_1 \times \text{rand}() \times \left(p\text{Best}_i - X_i^g\right) + c_2 \times \text{rand}() \times \left(g\text{Best} - X_i^g\right), \tag{1}$$

$$X_i^{g+1} = X_i^g + V_i^{g+1}, \tag{2}$$

where c_1 and c_2 are two learning factors, w is the inertia weight, and rand () is a random number between (0,1). The iterative update of $p\text{Best}_i$ and $g\text{Best}$ are performed through the fitness value comparison. The pseudo-code of the PSO is shown in Algorithm 1.

2.2. Adaptive Particle Swarm Optimization. The APSO is based on the original PSO by introducing state estimation strategy, elite learning strategy, and parameter adaptation strategy to improve it [20]. The improved algorithm can find the solution faster and more stable. The three strategies will be briefly introduced below.

2.2.1. State Estimation Strategy. The APSO divides the entire search process into four states, namely, exploration, exploitation, convergence, and jump-out, which are represented by S1, S2, S3, and S4. The division basis is the value of the evolution factor f. To calculate the evolution factor f, we must first calculate the Euclidean distance d_i of each particle. The calculation formula of d_i is shown in Equation (3).

$$d_i = \frac{1}{N-1} \times \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(X_i^k - X_j^k\right)^2}, \tag{3}$$

where d_i represents the Euclidean distance of the i-th particle, N represents the total number of particle swarms, and D represents the dimension of the problem.

After obtaining the Euclidean distance of each particle, we find the minimum value $d_{\min}$, maximum value $d_{\max}$, and optimal value $d_{g\text{best}}$. Then, calculate the evolution factor f by Equation (4).

$$f = \frac{d_{g\text{best}} - d_{\min}}{d_{\max} - d_{\min}}. \tag{4}$$

According to the value of the evolution factor f, the state is divided according to Figure 1. In addition, the state division is also related to the state of the previous iteration. For example, the value of the evolution factor f of this iteration is 0.55. It is assumed that the influence of the state of the previous iteration is not considered. In that case, the current state will be set as S1, but considering that the state of the previous iteration will have an impact on this state, so if the state of the previous iteration is S1 or S4, the current state will be set to S1. If the state of the previous iteration is S2 or S3, the current state will be set to S2.

```
while i < particles do
   Initialize the position X_i and velocity V_i of each particle
   Calculate the fitness value of each particle fitness(i)
   i = i +1
end
Initialize the pBest_i = X_i
Initialize the gBest = min(pBest_i)
for g =1 to iterMax do
      for i =1 to particles do.
         Update the V_i and X_i of the particles by Equations (1) and (2)
         Calculate the fitness value of the new particle fitness(i)
         if fitness(i) < fitness(pBest_i) then
            pBest_i = X_i
         end
         if fitness(i) < fitness(gBest) then
            gBest= X_i
         end
      end
end
```

Algorithm 1: The pseudo-code of the PSO.

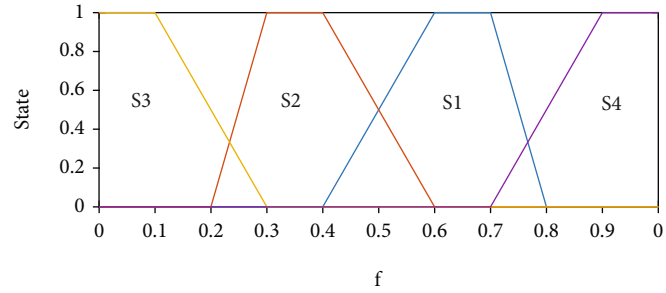


Figure 1: The division diagram of four states.

2.2.2. Parameter Adaptation Strategy. Three dynamic parameters are involved in APSO: inertial weight w, the individual learning factor c_1, and the global learning factor c_2. The weight of inertia changes with the evolutionary state. Its relationship with evolution factor f is shown in Equation (5).

$$w = \frac{1}{1 + 1.5e^{-2.6f}}. \tag{5}$$

APSO initializes w to 0.9. In the exploration state and the jump-out state, f is more extensive, which leads to a more oversized w, which is conducive to the global optimal search. In the convergence state and the exploitation state, f is more minor, which leads to a smaller w, which is conducive to the local convergence.

The c_1 and the c_2 are initialized to 2.0, and the two learning factors are adjusted according to the different evolutionary state. In the exploration state, increase c_1 and decrease c_2. This ensures that the individual learning factors play a leading role in helping the particles explore their own best individuals and avoid falling into the local optimum. In the exploitation state, slightly increase c_1 and slightly decrease c_2. Particles in the exploration phase gradually approach the local optimum. Increasing c_1 can more effectively enable the particles to explore around the individual optimum. Since the local optimum found may not be the global optimum, c_2 should be slightly reduced to prevent premature convergence because this will easily lead to the problem of the population falling into the local optimum. In the convergence stage, slightly increase c_1 and c_2. Increasing c_2 means that the particle swarm has found the global optimum at this stage, and the particle swarm can converge to this global optimum. A slight increase of c_1 is to prevent the learning factor from reaching the upper limit prematurely. If the upper bound is reached prematurely, the particles will treat the local optimum as the global optimum and quickly converge. In the jump-out state, reduce c_1 and increase c_2. It is helpful for the global optimum particle to jump out of the convergence zone and find a better position; other particles will follow the global optimal particle and converge to a better position. Table 1 shows the dynamic changes of the two learning factors in different states.

2.2.3. Elite Learning Strategy. The elite learning strategy makes the optimal global particles jump out of the convergence zone and finds a more superior position in the state

Table 1: Parameter adaptive strategy factor factors.

State	c_1	c_2
Exploration	Increase	Decrease
Exploitation	Increase slightly	Decrease slightly
Convergence	Increase slightly	Increase slightly
Jump-out	Decrease	Increase

of convergence. The elite learning strategy is determined by the elite learning rate σ. The calculation formula of σ is shown in Equation (6).

$$\sigma = \sigma_{\max} - (\sigma_{\max} - \sigma_{\min}) \times \frac{g}{\text{iterMax}}, \tag{6}$$

where g is the current number of iterations, $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum values of the elite learning rate σ, taking 1.0 and 0.1, respectively, and iterMax is the maximum number of iterations. After obtaining the elite learning rate, we randomly select a dimension for Gaussian perturbations to get the global optimal particle out of the convergence region. The formula is shown in Equation (7).

$$T^d = T^d + \left(X^d_{\max} - X^d_{\min}\right) \times \text{Gaussian}\left(\mu, \sigma^2\right). \tag{7}$$

2.2.4. The Pseudo-Code of APSO. The APSO introduces the above three strategies on the basis of PSO to optimize, and the optimized APSO can find the optimal solution to the problem better and faster when solving the problem.

The pseudo-code of APSO that combines the three strategies is shown in Algorithm 2.

2.3. Mobile Sensor Localization. This section mainly introduces the MCL mobile node location method [42, 43]. The mobile nodes in the mobile sensor network move with random speed and random direction. The biggest advantage of mobile nodes over fixed nodes lies in their mobility. The mobile node solves the problem that the information in a certain area cannot be collected due to the damage of the node at a specific location. The MCL method is mainly divided into three stages: the initialization stage, the prediction stage, and the filtering stage [44]. In the initialization stage, the moving area and the maximum moving speed are specified for each node. In the prediction stage, a preliminary estimate of the location of the mobile node is made. The speed and direction of node movement are uncertain. The position after the movement is within a circle. The center of this circle is the position of the last node, and the radius is the product of the speed and the positioning interval time, as shown in Figure 2.

The filtration stage is the most critical stage of MCL. Firstly, according to the distance between the anchor node and the unknown node, MCL determines which anchor nodes are the one-hop beacon nodes of the unknown node and which anchor nodes are the two-hop beacon nodes of the unknown node. Then, MCL obtains the one-hop beacon node set B_1 and the two-hop beacon node set B_2. Secondly, randomly select points in possible areas, and filter the nonconforming points according to whether the selected points are within the range of one-hop and two-hop beacon nodes. The filter condition is shown in Equation (8).

$$\begin{aligned}\text{filer}(\text{node}) = {} & (\forall b_1 \in B_1, \text{distance}(\text{node}, b_1) \\ & \le R) \cup (\forall b_2 \in B_2, \text{distance}(\text{node}, b_2) \\ & \le 2R).\end{aligned} \tag{8}$$

The gray areas in Figure 3 are the sets of points that meet the filter conditions. The MCL locates the position of the unknown node in a small space according to the number of hops from the unknown node to the anchor node. Equation (8) filters out the points that fall in this small space. In order to prevent contingency, the coordinates of all points in this small space are averaged as the position of the unknown node initially predicted by MCL. Finally, the qualified points after filtering are estimated by Equation (9) to estimate the position of the unknown node.

$$\text{Position}(b) = \frac{\left(\sum_{i=1}^{N} \text{node}_i\right)}{N}, \tag{9}$$

where N represents the total number of points that meet the filter conditions, and node represents the position of node that meet the filter conditions.

3. Compact Adaptive Particle Swarm Optimization

This section mainly introduces the idea of compact strategy and how to apply compact strategy to adaptive particle swarm optimization algorithm.

3.1. Compact Strategy. The primary purpose of the compact strategy is to reduce memory usage without changing the performance of the original algorithm or even improving the performance of the original algorithm. The running speed will naturally be improved if the memory usage is reduced. The compact strategy uses PV perturbation vectors to represent the overall motion state of the population instead of simply using the position and velocity of each individual to represent the population state. The disturbance vector PV is defined as $\text{PV}^g = (\mu^g, \sigma^g)$, where μ represents the average value of the disturbance vector, σ represents the standard deviation of the disturbance vector, and g represents the current iteration update times.

The compact strategy ultimately returns a value between (0,1). The PV vector is composed of μ and σ, and the probability distribution function (PDF) can be calculated through μ and σ. Then, the cumulative distribution function (CDF) can be calculated by PDF. The calculation formulas of PDF and CDF are in Equations (10) and (11).

$$\text{PDF} = \frac{e^{-(x-\mu)^2/2\sigma^2 \times \sqrt{2/\pi}}}{\sigma \times \left(\text{erf}\left(\mu + 1/\sqrt{2\sigma}\right) - \text{erf}\left(\mu - 1/\sqrt{2\sigma}\right)\right)}, \tag{10}$$

```
while i < particles do
   Initialize the position X_i and velocity V_i of each particle
   Calculate the fitness value of each particle fitness(i)
   i = i +1
end
Initialize the pBest_i = X_i
Initialize the gBest = min(pBest_i)
for g =1 to iterMax do
      for i =1 to particles do
            Update the V_i and X_i of the particles by Equations (1) and (2)
            Calculate the fitness value of the new particle fitness(i)
            if fitness(i) < fitness(pBest_i) then
               pBest_i= X_i
            end.
            if fitness(i) < fitness(gBest) then
               gBest= X_i
            end
end
Calculate the d_i by Equation (3)
Calculate the f by Equation (4)
Determine the evolutionary state S by Figure 1
w is adjusted according to Equation (5)
c_1 and c_2 are adjusted according to Table 1
if evolutionary state S == convergence state S3 then
         σ = σ_max − (σ_max − σ_min) × g/iterMax
         T^d = T^d + (X^d_max − X^d_min) × Gaussian(μ, σ^2)
         if fitness(T) < fitness(gBest) then
               gBest= T
         end
      end
end
```

Algorithm 2: The pseudo-code of APSO.

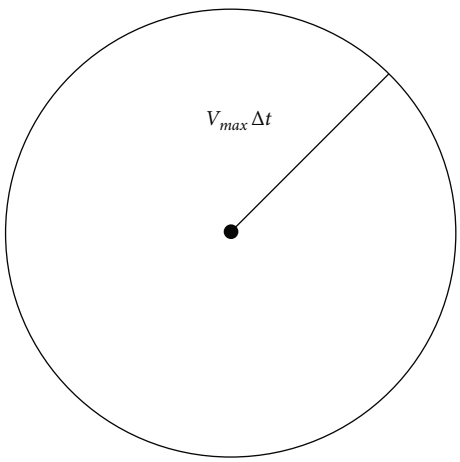


Figure 2: Rang of node movement.

$$\mathrm{CDF} = \int_{-\infty}^{x} \frac{e^{-(x-\mu)^2/2\sigma^2 \times \sqrt{2/\pi}}}{\sigma \times \left(\mathrm{erf}\left(\mu + 1/\sqrt{2\sigma}\right) - \mathrm{erf}\left(\mu - 1/\sqrt{2\sigma}\right)\right)} dx. \tag{11}$$

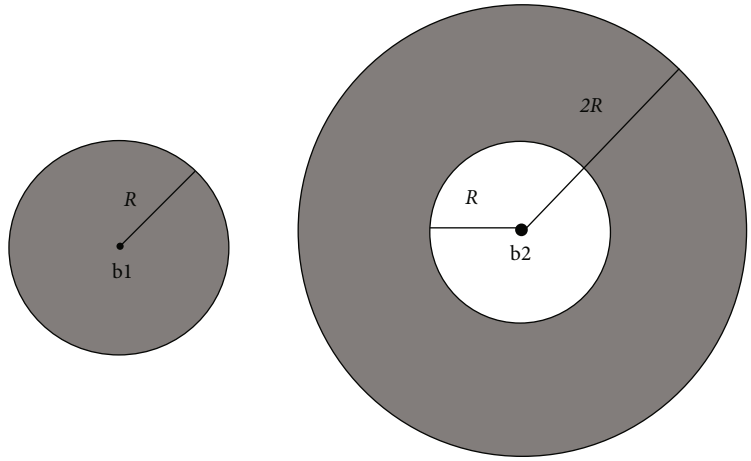
The CDF value range of the cumulative distribution function is (0,1), which is also the value range returned by the compact strategy. Taking the standard normal distribution as an example, PDF and CDF of the standard normal distribution are shown in Figure 4.

Another important content of the compact strategy is the iterative update of the PV disturbance vector. The compact strategy is based on comparison, and the winner and loser are obtained through a competitive game mechanism. Then, update it through the update iteration strategy. The update formulas are shown in Equations (12) and (13).

$$\mu^{g+1} = \mu^{g} + \frac{1}{N_p}(\mathrm{winner} - \mathrm{loser}), \tag{12}$$

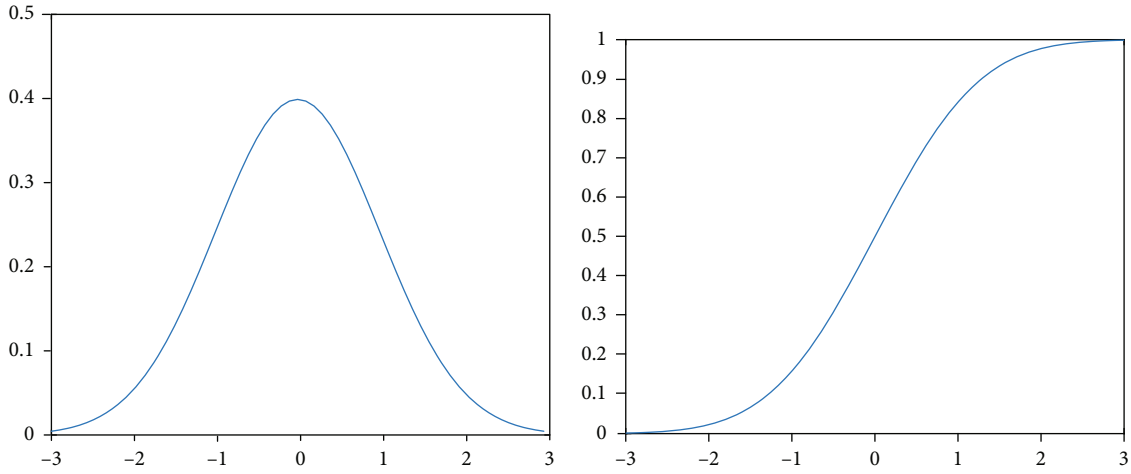
$$\sigma^{g+1} = \sqrt{(\sigma^{g})^2 + (\mu^{g})^2 + \frac{2}{N_p}\left(\mathrm{winner}^2 - \mathrm{loser}^2\right)}, \tag{13}$$

where g represents the current iteration times and N_p represents the number of virtual populations. The mean value μ in the PV disturbance vector is generally set to 0, and the standard deviation σ in the PV disturbance vector is generally set to 10 to avoid the contingency of the local optimum during initialization. After a large number of experiments, it has been proven that the effect achieved



(a) One-hop beacon node locations (b) Two-hop beacon node locations

Figure 3: One-hop and two-hop beacon node locations.



(a) Standard normal distribution PDF (b) Standard normal distribution CDF

Figure 4: PDF and CDF of standard normal distribution.

when the number of virtual populations N_p is set to 300 is the best. So N_p is generally set to 300.

3.2. Implementation of the Compact Strategy in APSO. The compact strategy is based on comparison. In other words, there must be a game mechanism of size two so that the winner and loser can be generated, and then, μ and σ in the PV disturbance vector can be updated. APSO can meet this condition through the comparison of fitness values, so APSO meets the primary conditions of the compact strategy, and the compact strategy can be combined with APSO to achieve the goal of reducing memory and improving operating speed. For example, if there are 30 particles in the particle swarm and the dimension to solve the problem is also 30 dimensions, then APSO needs 900 storage units to store the position information of each particle in each dimension. However, cAPSO only needs 60 storage units to store the particle swarm in each dimension using PV perturbation vectors, which significantly saves storage space. The saving of storage space reduces the number of reads and writes to the memory, and the speed of the algorithm will also be improved accordingly.

The cAPSO expresses the position of the particle swarm through the overall probability distribution, and the value range of the compact strategy is between (0,1), assuming that y is the return value of the compact strategy. Then, the return value of the compact strategy should be corresponded to the actual position range by Equation (14).

$$X = \frac{1}{2} \times (X_{\max} - X_{\min}) \times y + \frac{1}{2} \times (X_{\max} - X_{\min}). \quad (14)$$

The evolution factor f in the original APSO is calculated by Equations (3) and (4). However, the distribution probability is used to represent the position of the

Initialize the PV(μ, σ) disturbance vector parameter
Initialize a particle swarm position $X = X_{\min} + (X_{\max} - X_{\min}) \times \text{rand}()$
Initialize a particle swarm velocity $V = V_{\min} + (V_{\max} - V_{\min}) \times \text{rand}()$
Initialize pBest = X
Initialize gBest = X
for g =1 to iterMax do
    pBest = $1/2 \times (X_{\max} - X_{\min}) \times \text{compact}(\mu, \sigma) + 1/2 \times (X_{\max} - X_{\min})$
    Update the V and X of the particles by Equations (1) and (2)
    Calculate the fitness value of the new particle fitness(X)
    (winner, loser) = compare(fitness(pBest), fitness(X))
    Update the PV disturbance vector by Equations (12) and (13)
    Calculate the f by Equation (15)
    Determine the evolutionary state S by Figure 1
    w is adjusted according to Equation (5)
    c_1 and c_2 are adjusted according to Table 1
    if evolutionary state S == convergence state S3 then
        $\sigma = \sigma_{\max} - (\sigma_{\max} - \sigma_{\min}) \times g/\text{iterMax}$
        $T^d = T^d + (X^d_{\max} - X^d_{\min}) \times \text{Gaussian}(\mu, \sigma^2)$
        (winner, loser) = compare(fitness(gBest), fitness(X))
        gBest= winner
    end
end

Algorithm 3: The pseudo-code of cAPSO.

particle swarm in cAPSO, so the calculation of the evolution factor f is replaced by Equation (15).

$$f = \frac{\mu_{\text{mean}} - \mu_{\min}}{\mu_{\max} - \mu_{\min}}. \tag{15}$$

The cAPSO compares the fitness value of the position of this iteration generated by the PV disturbance vector with the fitness value of the optimal position of the present individual and generates winner and loser. The generated winner and loser use the Equations (12) and (13) to update the disturbance vector PV and then generate the position probability distribution of the particle swarm in the next iteration.

The pseudo-code of cAPSO is shown in Algorithm 3.

4. The Performance Test of cAPSO

In this section, the cAPSO algorithm is mainly tested in 28 test functions of CEC2013 [45]. This paper compares the cAPSO with common heuristic algorithms and common compact algorithms. The 28 test functions in CEC2013 include 8 mixed functions, 15 multimodal functions, and 5 unimodal functions. These 28 functions are very representative. The 28 functions are represented by f1 to f28. Every experiment keeps the common parameters consistent during the comparison process to ensure the fairness of the comparison.

4.1. Performance Comparison of cAPSO and Common Heuristic Algorithms. In this section, the cAPSO is compared with genetic algorithm (GA) [46], differential evolution algorithm (DE) [47], whale optimization algorithm (WOA) [48], bat algorithm (BA) [49], and sine cosine algorithm (SCA) [50] on 28 test functions of CEC2013. At the same time, the overall performance of each algorithm compared with the cAPSO algorithm was measured at a significant level $\alpha = 0.05$ under the Wilcoxon signed rank test. Twenty tests are carried out on each test function, and then, the average value is taken to avoid the occurrence of chance. To ensure the fairness of the algorithm during testing, the dimension of the problem is set to 50, the number of populations is set to 60, and the number of iterations is set to 3000, and the search range requirement in CEC2013 is [-100,100]. Table 2 shows the performance comparison of cAPSO and common heuristic algorithms. In addition, Table 2 also shows that each algorithm is measured under the Wilcoxon signed rank test, with a significance level of $\alpha = 0.05$. The symbol ">" represents that the performance of the cAPSO is better than the other heuristic algorithm, the symbol "=" represents that the performance of the cAPSO is the same as the other heuristic algorithm, and the symbol "<" represents that the performance of the cAPSO is worse than the other heuristic algorithm. The last row of Table 2 summarizes the comparison results of all test functions.

Table 2 shows that compared with DE, the test performance of cAPSO is better than DE in 20 functions, the same as DE in 2 functions, and worse than DE in 6 functions. Compared with BA, the test performance of cAPSO is better than BA in 21 functions, the same as BA in 1 function, and worse than BA in 6 functions. Compared with GA, WOA, and SCA, cAPSO has the same performance as these three algorithms in function f8, but it is better than others in other functions. It can be seen that the performance of the cAPSO algorithm combined with the compact strategy is greatly improved compared with the common heuristic algorithm.

In order to further describe the effect of the algorithm, this paper uses the convergence curve for evaluation. Since the convergence of many algorithms is very similar, the performance is not obvious in the convergence curve, so this

Table 2: The performance comparison between cAPSO and common heuristic algorithms and the Wilcoxon signed rank test of each algorithm at the significance level $\alpha = 0.05$.

Function	GA		DE		BA		WOA		SCA		cAPSO
f1	1.61*E*-05	(>)	-1.40*E*-03	(=)	-1.39*E*-03	(>)	-1.36*E*-03	(>)	2.70*E*-04	(>)	-1.40*E*-03
f2	5.34*E*-09	(>)	2.39*E*-08	(>)	5.19*E*-06	(<)	8.10*E*-07	(>)	5.07*E*-08	(>)	7.33*E*-06
f3	7.38*E*-19	(>)	1.87*E*-10	(>)	5.10*E*-08	(<)	4.11*E*-10	(>)	1.09*E*-11	(>)	3.69*E*-09
f4	6.55*E*-05	(>)	1.10*E*-05	(>)	1.98*E*-04	(>)	6.18*E*-04	(>)	6.37*E*-04	(>)	7.64*E*-03
f5	8.99*E*-04	(>)	-9.86*E*-02	(>)	-9.96*E*-02	(>)	-8.07*E*-02	(>)	2.44*E*-03	(>)	-1.00*E*-03
f6	2.88*E*-04	(>)	-8.54*E*-02	(<)	-8.40*E*-02	(>)	-6.66*E*-02	(>)	1.21*E*-03	(>)	-8.20*E*-02
f7	6.09*E*-06	(>)	-6.67*E*-02	(>)	9.86*E*-02	(>)	-3.18*E*-02	(>)	-6.06*E*-02	(>)	-6.71*E*-02
f8	-6.79*E*-02	(=)	-6.79*E*-02	(=)	-6.79*E*-02	(=)	-6.79*E*-02	(=)	-6.79*E*-02	(=)	-6.79*E*-02
f9	-5.19*E*-02	(>)	-5.31*E*-02	(>)	-5.33*E*-02	(>)	-5.29*E*-02	(>)	-5.26*E*-02	(>)	-5.42*E*-02
f10	2.18*E*-04	(>)	-4.31*E*-02	(>)	-4.96*E*-02	(>)	-2.17*E*-02	(>)	3.42*E*-03	(>)	-4.98*E*-02
f11	2.11*E*-03	(>)	-2.79*E*-02	(>)	7.61*E*-02	(>)	4.02*E*-02	(>)	3.09*E*-02	(>)	-3.95*E*-02
f12	1.96*E*-03	(>)	1.22*E*-02	(>)	8.98*E*-02	(>)	6.50*E*-02	(>)	4.56*E*-02	(>)	7.37*E*-01
f13	2.03*E*-03	(>)	2.29*E*-02	(<)	1.16*E*-03	(>)	8.43*E*-02	(>)	5.53*E*-02	(>)	2.79*E*-02
f14	1.64*E*-04	(>)	2.13*E*-03	(>)	8.93*E*-03	(>)	8.92*E*-03	(>)	1.36*E*-04	(>)	1.98*E*-03
f15	1.61*E*-04	(>)	1.45*E*-04	(>)	8.81*E*-03	(<)	1.15*E*-04	(>)	1.44*E*-04	(>)	1.01*E*-04
f16	2.05*E*-02	(>)	2.04*E*-02	(>)	2.02*E*-02	(<)	2.03*E*-02	(>)	2.03*E*-02	(>)	2.02*E*-02
f17	5.26*E*-03	(>)	5.14*E*-02	(>)	2.65*E*-03	(>)	1.47*E*-03	(>)	1.31*E*-03	(>)	3.51*E*-02
f18	5.29*E*-03	(>)	8.74*E*-02	(>)	2.94*E*-03	(>)	1.51*E*-03	(>)	1.40*E*-03	(>)	8.54*E*-02
f19	2.59*E*-07	(>)	5.21*E*-02	(>)	5.68*E*-02	(>)	6.74*E*-02	(>)	4.29*E*-04	(>)	5.02*E*-02
f20	6.25*E*-02	(>)	6.23*E*-02	(<)	6.25*E*-02	(>)	6.25*E*-02	(>)	6.24*E*-02	(>)	6.25*E*-02
f21	1.26*E*-04	(>)	1.23*E*-03	(<)	1.53*E*-03	(>)	1.86*E*-03	(>)	4.67*E*-03	(>)	1.60*E*-03
f22	1.86*E*-04	(>)	3.63*E*-03	(<)	1.27*E*-04	(>)	1.28*E*-04	(>)	1.54*E*-04	(>)	4.59*E*-03
f23	1.81*E*-04	(>)	1.55*E*-04	(>)	1.22*E*+04	(<)	1.39*E*-04	(>)	1.61*E*-04	(>)	1.26*E*-04
f24	2.08*E*-03	(>)	1.37*E*-03	(>)	1.45*E*-03	(>)	1.41*E*-03	(>)	1.43*E*-03	(>)	1.36*E*-03
f25	1.76*E*-03	(>)	1.48*E*-03	(>)	1.47*E*-03	(<)	1.53*E*-03	(>)	1.55*E*-03	(>)	1.51*E*-03
f26	1.79*E*-03	(>)	1.52*E*-03	(<)	1.69*E*-03	(>)	1.65*E*-03	(>)	1.59*E*-03	(>)	1.64*E*-03
f27	4.68*E*-03	(>)	3.29*E*-03	(>)	3.50*E*-03	(>)	3.54*E*-03	(>)	3.66*E*-03	(>)	3.18*E*-03
f28	1.72*E*-04	(>)	2.77*E*-03	(>)	1.08*E*-04	(>)	8.79*E*-03	(>)	6.84*E*-03	(>)	2.73*E*-03
>/=/<	27/1/0		20/2/6		21/1/6		27/1/0		27/1/0		

paper selects several scattered representative curves for display. Figure 5 shows the convergence process of the algorithm on some test functions. The horizontal axis represents iteration times, and the vertical axis represents the fitness value of different algorithms. The smaller the fitness value, the better the performance on this test function. It can be seen from Figure 5 that the performance of the proposed cAPSO algorithm is better than other heuristic algorithms on f4, f9, f11, f17, and f27. But the performance of the test function on f13 and f22 is worse than DE, and the performance on f17 is not as good as BA.

4.2. Performance Comparison of cAPSO and Common Compact Algorithms. In this section, the cAPSO is compared with cPSO [41], cABC [38], cSCA [39], and cBA [40] on 28 test functions of CEC2013. At the same time, the overall performance of each algorithm compared with the cAPSO algorithm was measured at a significant level $\alpha = 0.05$ under the Wilcoxon signed rank test. Twenty tests are carried out on each test function, and then, the average value is taken to avoid the occurrence of chance. To ensure the fairness of the algorithm during testing, the dimension of the problem is set to 50, the number of populations is set to 60, the number of iterations is set to 3000, the virtual number of populations is set to 300, and the search range requirement in CEC2013 is [-100,100]. Table 3 shows the performance comparison of cAPSO and common compact algorithms. In addition, Table 3 also shows that each algorithm is measured under the Wilcoxon signed rank test, with a significance level of $\alpha = 0.05$. The symbols ">," "=," and "<" have the same meaning as in Section 4.1. The last row of Table 3 summarizes the comparison results of all test functions.

Table 3 shows that compared with cPSO, the test performance of cAPSO is better than cPSO in 24 functions, the same as cPSO in 2 functions, and worse than cPSO in 2 functions. Compared with cBA, the test performance of cAPSO is better than cBA in 20 functions, the same as cBA in 3 functions, and worse than cBA in 5 functions.

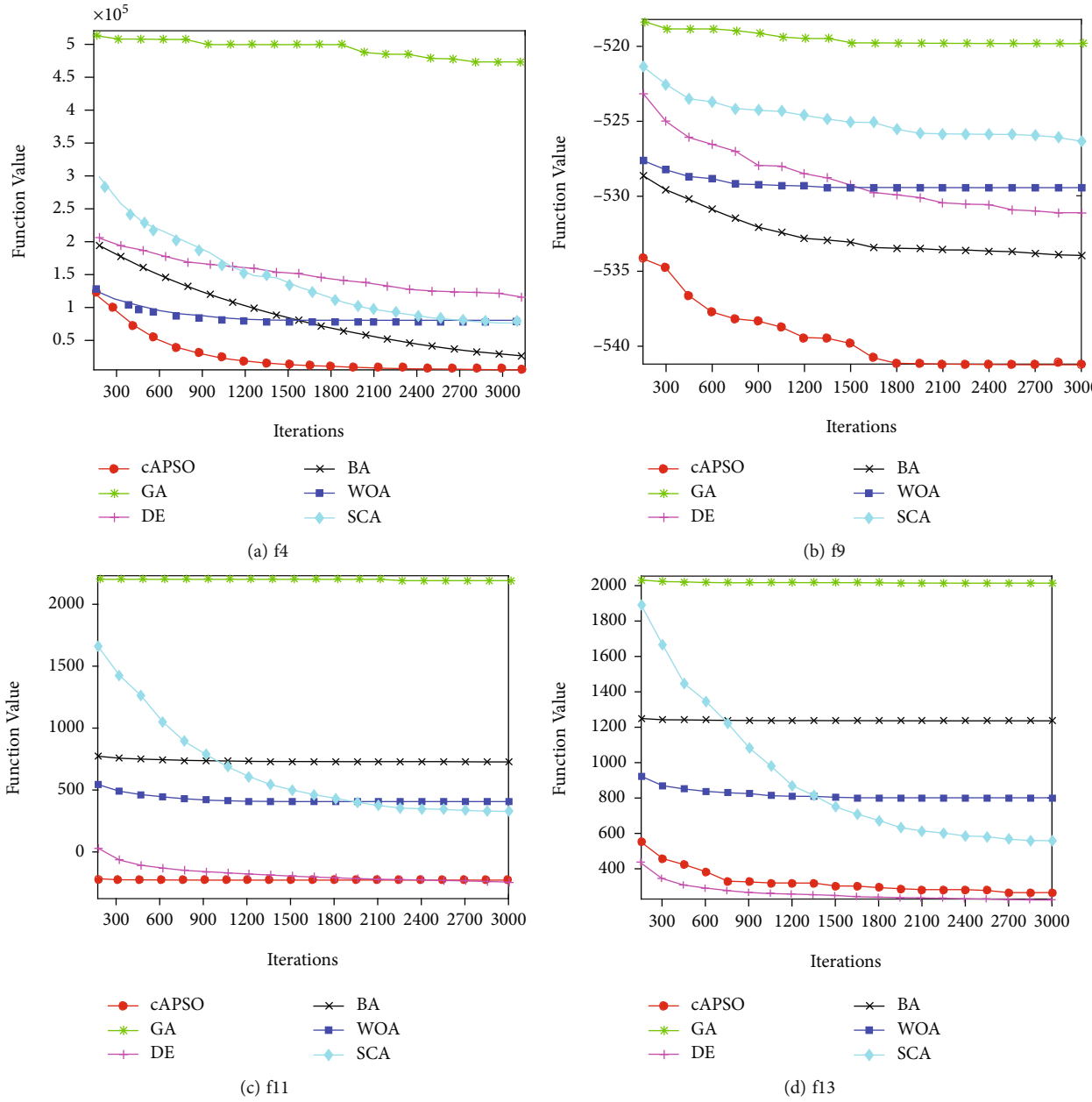


Figure 5: Continued.

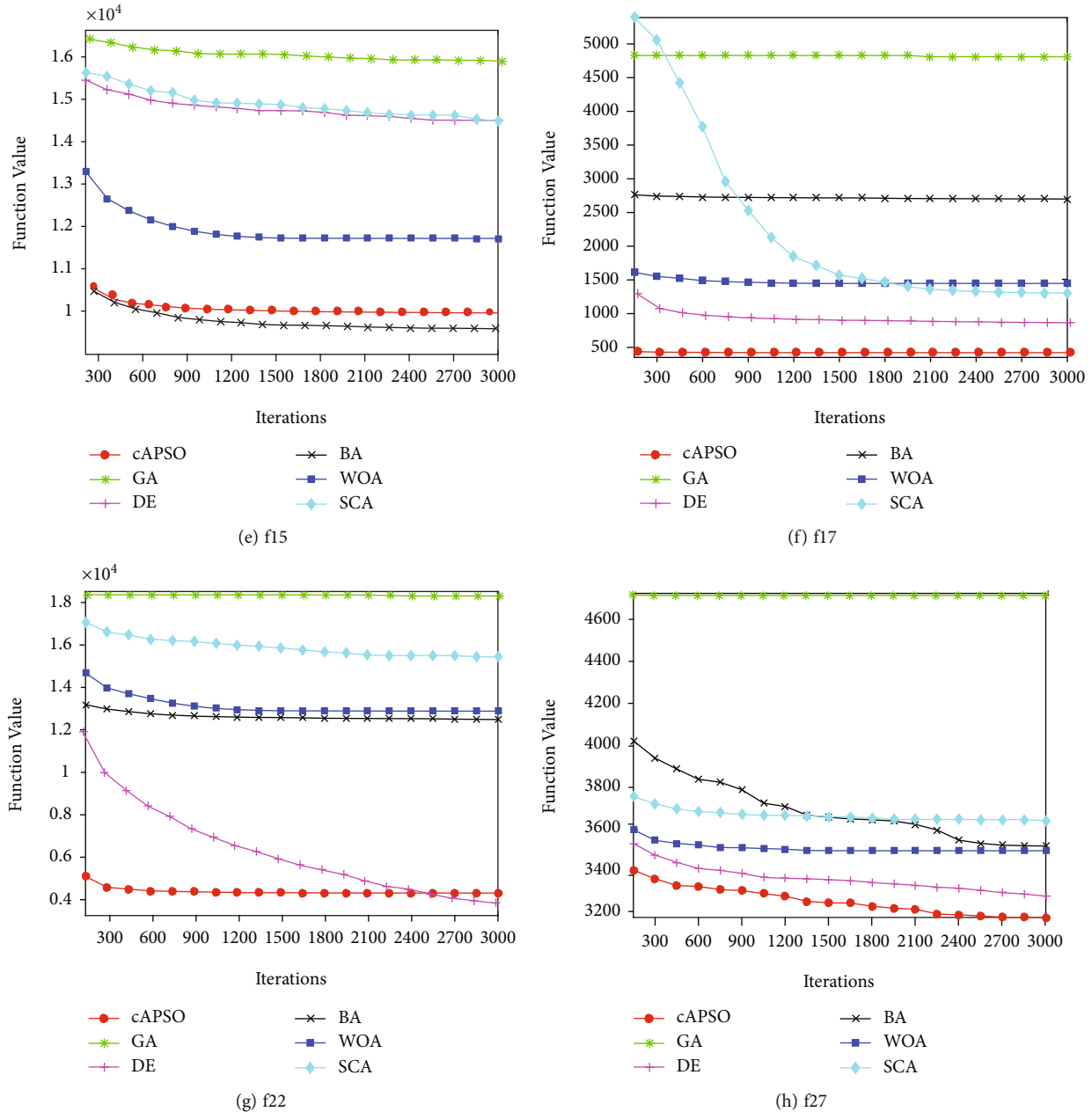


Figure 5: Performance comparison between cAPSO and common heuristic algorithms.

Compared with cABC and cSCA, cAPSO has the same performance as these two algorithms in function f8, but it is better than these two algorithms in other functions. It can be seen that cAPSO has strong competitiveness in compact algorithms and has advantages over other compact algorithms in performance.

As in Section 4.1, Figure 6 shows representative convergence curves of the proposed cAPSO and other compact algorithms. It can be seen from Figure 6 that the performance of the proposed cAPSO algorithm is better than other compact algorithms on f12, f15, f20, and f25. But the performance of the test function on f16 is not as good as cBA, and the performance on f23 is worse than cPSO.

5. Application of cAPSO in Mobile Sensor Localization

This section mainly applies cAPSO to the mobile sensor localization technology MCL and compares it with the original MCL, WOA-based MCL, and BA-based MCL under different anchor node numbers and different communication radius. It takes a lot of time and computing power to directly find a position with a small error through the MCL technology. A position with a large error is initially obtained through MCL technology, and then, cAPSO is applied around the obtained position for further optimization. The cAPSO broadcasts nodes around it, and the broadcast nodes move according to the idea of the cAPSO. The position after

Table 3: The performance comparison between cAPSO and common compact algorithms and the Wilcoxon signed rank test of each algorithm at the significance level $\alpha = 0.05$.

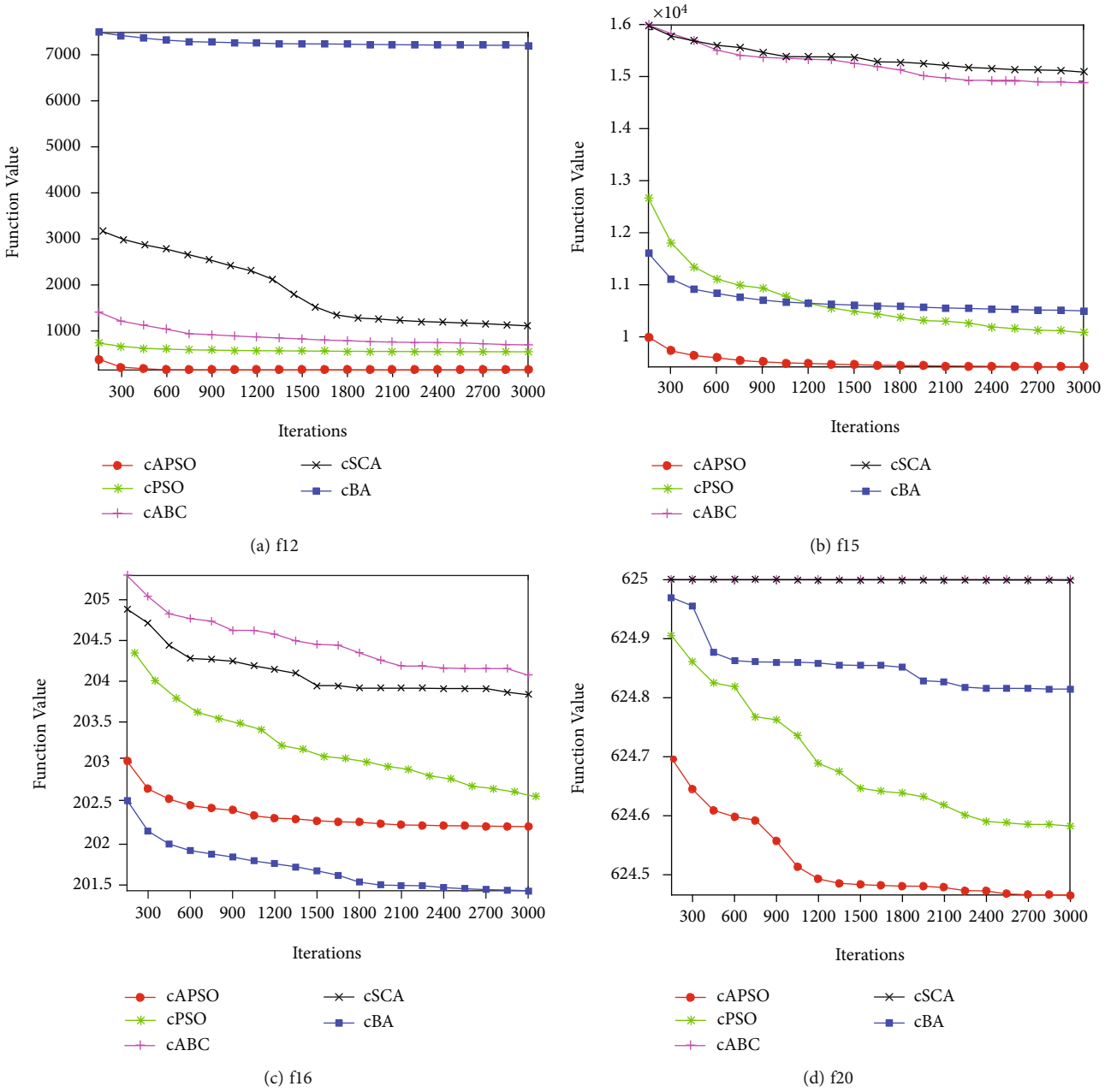
Function	cPSO		cABC		cSCA		cBA		cAPSO
f1	-1.40*E*+03	(=)	2.50*E*+04	(>)	5.20*E*+04	(>)	-1.40*E*+03	(=)	-1.40*E*+03
f2	2.54*E*+07	(>)	7.05*E*+08	(>)	1.02*E*+09	(>)	3.31*E*+06	(<)	8.07*E*+06
f3	7.45*E*+09	(>)	2.07*E*+11	(>)	2.52*E*+15	(>)	5.34*E*+09	(>)	3.78*E*+09
f4	3.76*E*+03	(<)	1.44*E*+05	(>)	1.84*E*+05	(>)	2.87*E*+05	(>)	8.68*E*+03
f5	-9.88*E*+02	(>)	5.94*E*+03	(>)	1.44*E*+04	(>)	-1.00*E*+03	(=)	-1.00*E*+03
f6	-8.05*E*+02	(>)	1.28*E*+03	(>)	3.13*E*+03	(>)	-8.49*E*+02	(<)	-8.30*E*+02
f7	-5.68*E*+02	(>)	-4.55*E*+02	(>)	8.72*E*+03	(>)	3.26*E*+12	(>)	-6.75*E*+02
f8	-6.79*E*+02	(=)	-6.79*E*+02	(=)	-6.79*E*+02	(=)	-6.79*E*+02	(=)	-6.79*E*+02
f9	-5.37*E*+02	(>)	-5.24*E*+02	(>)	-5.26*E*+02	(>)	-5.11*E*+02	(>)	-5.41*E*+02
f10	-4.76*E*+02	(>)	4.31*E*+03	(>)	7.25*E*+03	(>)	-4.99*E*+02	(<)	-4.98*E*+02
f11	3.33*E*+02	(>)	4.32*E*+02	(>)	4.35*E*+02	(>)	2.84*E*+03	(>)	-3.96*E*+02
f12	5.47*E*+02	(>)	6.96*E*+02	(>)	9.21*E*+02	(>)	7.20*E*+03	(>)	1.52*E*+02
f13	4.48*E*+02	(>)	8.20*E*+02	(>)	1.14*E*+03	(>)	8.65*E*+03	(>)	2.65*E*+02
f14	9.05*E*+03	(>)	1.45*E*+04	(>)	1.60*E*+04	(>)	8.59*E*+03	(>)	2.11*E*+03
f15	1.01*E*+04	(>)	1.50*E*+04	(>)	1.52*E*+04	(>)	1.06*E*+04	(>)	9.48*E*+03
f16	2.03*E*+02	(>)	2.04*E*+02	(>)	2.04*E*+02	(>)	2.01*E*+02	(<)	2.02*E*+02
f17	1.06*E*+03	(>)	1.91*E*+03	(>)	1.45*E*+03	(>)	1.50*E*+04	(>)	3.51*E*+02
f18	1.07*E*+03	(>)	2.28*E*+03	(>)	1.59*E*+03	(>)	1.65*E*+04	(>)	9.16*E*+02
f19	5.47*E*+02	(>)	6.82*E*+04	(>)	1.86*E*+04	(>)	8.46*E*+02	(>)	5.02*E*+02
f20	6.25*E*+02	(>)	6.25*E*+02	(>)	6.25*E*+02	(>)	6.25*E*+02	(>)	6.24*E*+02
f21	1.61*E*+03	(>)	5.58*E*+03	(>)	5.10*E*+03	(>)	1.62*E*+03	(>)	1.58*E*+03
f22	1.26*E*+04	(>)	1.66*E*+04	(>)	1.81*E*+04	(>)	1.10*E*+04	(>)	4.31*E*+03
f23	1.29*E*+04	(<)	1.72*E*+04	(>)	1.68*E*+04	(>)	1.40*E*+04	(>)	1.32*E*+04
f24	1.42*E*+03	(>)	1.42*E*+03	(>)	1.52*E*+03	(>)	1.69*E*+03	(>)	1.37*E*+03
f25	1.57*E*+03	(>)	1.60*E*+03	(>)	1.57*E*+03	(>)	1.65*E*+03	(>)	1.50*E*+03
f26	1.61*E*+03	(>)	1.64*E*+03	(>)	1.71*E*+03	(>)	1.40*E*+03	(<)	1.59*E*+03
f27	3.54*E*+03	(>)	3.65*E*+03	(>)	3.81*E*+03	(>)	5.28*E*+03	(>)	3.18*E*+03
f28	4.16*E*+03	(>)	7.40*E*+03	(>)	8.71*E*+03	(>)	9.08*E*+04	(>)	2.18*E*+03
>/=/<	24/2/2		27/1/0		27/1/0		20/3/5		

move is compared with the historical optimal position to update the optimal position. After a certain number of iterations, an optimal position is obtained as the final position of the MCL positioning technology. Mobile sensor positioning technology is a technology that uses the information of anchor nodes to estimate the location of unknown nodes, so the position error becomes the key to the technology. The smaller the position error, the more conducive to the accuracy of the information. Applying heuristic algorithms to MCL technology can better locate unknown nodes and reduce localization errors. The more accurate position coordinates are iteratively updated by the error function. The error function is defined as Equation (16).

$$\text{error} = \frac{\sum_{j=1}^{M}\left(\sum_{i=1,i\neq j}^{N}\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} - D_{ij}\right)}{M}, \tag{16}$$

where (x_i, y_i) represents the estimated location of the unknown node i, (x_j, y_j) represents the location of the anchor node j, M represents the total number of unknown nodes, and N represents the total number of anchor nodes. D_{ij} represents the distance between each unknown node i and each anchor node j. This paper assumes that the anchor node j can obtain the distance between the anchor node j and the unknown node i by the strength of the signal received from the unknown node i. The smaller the error value, the higher the accuracy of localization.

5.1. Influence Analysis of Different Anchor Nodes. In the simulation experiment, 200 nodes are randomly scattered within a range of 200 m × 200 m. The number of anchor nodes is 5, 10, 15, 20, 25, 30, 35, and 40, so the number of unknown nodes is 195, 190, 185, 180, 175, 170, 165, and 160. The communication radius is set to 50 m, and the simulation experiments with the same parameters were tested 20 times, and the average value was calculated as the error



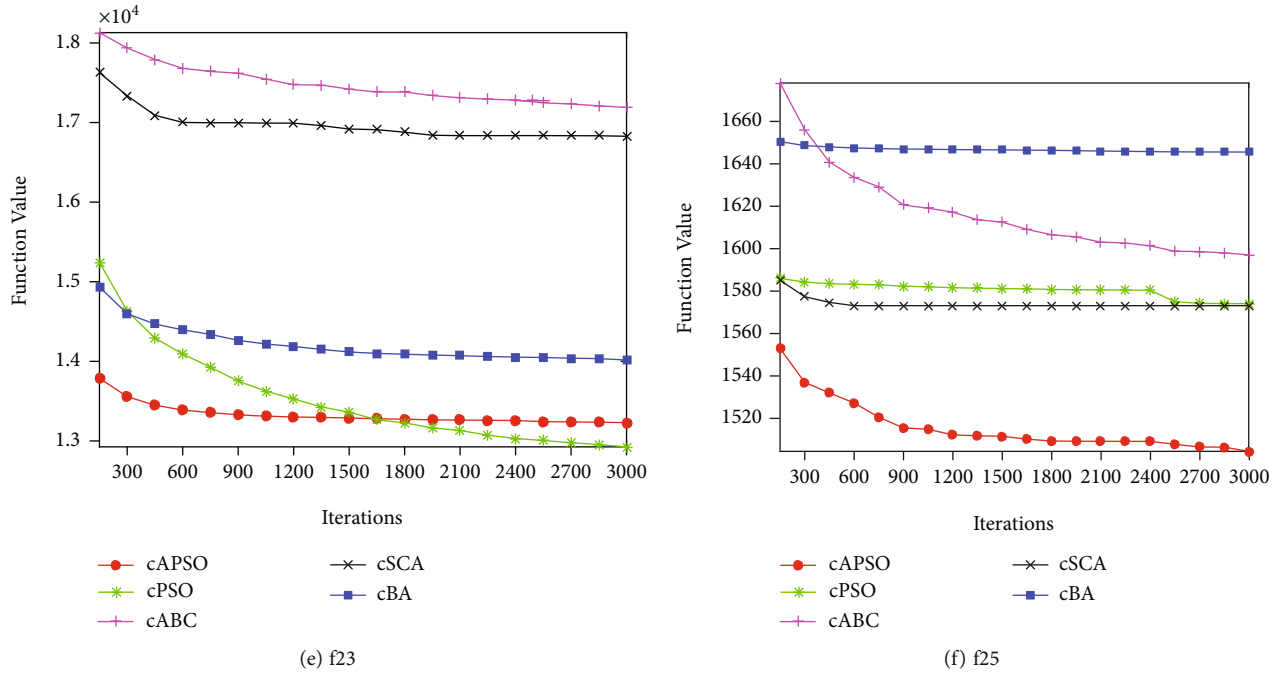
(a) f12

(b) f15

(c) f16

(d) f20

Figure 6: Continued.



(e) f23 (f) f25

Figure 6: Performance comparison between cAPSO and common compact algorithms.

Table 4: Experimental results of the localization error of different anchor nodes.

Anchor	MCL		cAPSO MCL		WOA MCL		BA MCL	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	22.0428	2.4357	12.5510	2.4495	12.6486	2.4621	12.5604	2.4466
10	15.5169	1.2365	6.8567	1.0768	6.9593	1.0758	6.8692	1.0787
15	11.4610	0.8525	3.4532	0.7155	3.5635	0.7134	3.4695	0.7231
20	3.4568	0.7152	2.0050	0.4179	2.1148	0.4147	2.0199	0.4179
25	7.6420	0.5372	1.1901	0.3881	1.3114	0.3877	1.2074	0.3878
30	1.2074	0.3878	0.6402	0.1856	0.7603	0.1832	0.6591	0.1852

Table 5: Experimental results of the localization error of different communication radius.

Radius	MCL		cAPSO MCL		WOA MCL		BA MCL	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10	17.7767	3.7450	8.4470	3.2895	8.5827	3.2828	8.4567	3.2881
20	13.2522	0.7495	4.9535	0.5931	5.0626	0.5932	4.9637	0.5932
30	9.5178	0.5585	2.7765	0.4599	2.8918	0.4639	2.7952	0.4622
40	7.8710	0.4849	1.8598	0.3892	1.9788	0.3893	1.8773	0.3889
50	1.8773	0.3889	1.1117	0.3746	1.2293	0.3744	1.1303	0.3743

result. Apply cAPSO, WOA, and BA algorithms to the simulation experiment, respectively. The experimental results are shown in Table 4.

Table 4 shows that when the total number of nodes is 200 and the communication radius is 50 m, the larger the number of anchor nodes, the more accurate the location of unknown nodes and the smaller the error value. Table 4 clearly shows that the positioning error after optimization by the heuristic algorithm is much smaller than the original MCL. In the combined different heuristic algorithms, it can be clearly seen that although the cAPSO algorithm is not better than other algorithms in standard deviation every time, and it has better results than other heuristic algorithms in error mean.

5.2. Influence Analysis of Different Communication Radius. The simulation experiment randomly scatter 200 nodes in the range of 200 m × 200 m, the number of anchor nodes is set to 30, and the communication radius is set to 10 m, 20 m, 30 m, 40 m, and 50 m. The simulation experiments with the same parameters were tested 20 times, and the average value was calculated as the error result. Apply cAPSO,

WOA, and BA algorithms to the simulation experiment, respectively. The experimental results are shown in Table 5.

Table 5 shows that when the total number of nodes is 200 and the number of anchor nodes is 30, the greater the communication radius, the more accurate the location of unknown nodes, and the smaller the error value. Table 5 clearly shows that the positioning error after optimization by the heuristic algorithm is much smaller than that of the original MCL. In the combined different heuristic algorithms, it can also be clearly seen that the cAPSO algorithm is better than other heuristic algorithms in the comparison of the error mean.

6. Conclusion

In this paper, an improved APSO algorithm combined with compact strategy is proposed and applied to mobile sensor localization. The compact strategy no longer stores the position of each particle in each dimension but describes the distribution characteristics of the particles in each dimension through the operation of probability model. The compact strategy can reduce the use of memory very well. This paper tests the performance of cAPSO on 28 test functions of CEC2013 and compares it with the common heuristic algorithms GA, DE, BA, WOA, and SCA and the common compact strategy heuristic algorithms cPSO, cABC, cSCA, and cBA. The comparison results show that cAPSO has better test performance. Finally, cAPSO is applied to mobile sensor localization technology MCL, and it is also compared with WOA-based MCL and BA-based MCL. The results show that MCL based on cAPSO is more effective in solving this problem.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgments

This project was funded by the National Key Research and Development Program of China under Grant No. 11974373.

Supplementary Materials

Reference URL of compact strategy thought: https://blog.csdn.net/Liu_Ning_666/article/details/118308477. Reference URL of many improved PSO algorithms: https://blog.csdn.net/Liu_Ning_666/article/details/120174723 *(Supplementary Materials)*

References

[1] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[2] L. Tonutti, B. Dalla Costa, H. Decolatti, G. Mendow, and C. Querini, "Determination of kinetic constants for glycerol acetylation by particle swarm optimization algorithm," *Chemical Engineering Journal*, vol. 424, article 130408, 2021.

[3] A. Kokhanovskiy, E. Kuprikov, A. Bednyakova, I. Popkov, S. Smirnov, and S. Turitsyn, "Inverse design of mode-locked fiber laser by particle swarm optimization algorithm," *Scientific Reports*, vol. 11, pp. 1–9, 2021.

[4] F. Zhou, S. Bian, and D. Wang, "Optimal dispatching of microgrid based on improved particle swarm optimization," *Journal of Physics: Conference Series. IOP Publishing*, vol. 1871, article 012141, 2021.

[5] E. Bas, E. Egrioglu, and E. Kolemen, "Training simple recurrent deep artificial neural network for forecasting using particle swarm optimization," *Granular Computing*, pp. 1–10, 2021.

[6] J. Jafari-Asl, B. S. Kashkooli, and M. Bahrami, "Using particle swarm optimization algorithm to optimally locating and controlling of pressure reducing valves for leakage minimization in water distribution systems," *Sustainable Water Resources Management*, vol. 6, no. 4, pp. 1–11, 2020.

[7] H. C. Huang, S. C. Chu, J. S. Pan, C. Y. Huang, and B. Y. Liao, "Tabu search based multi-watermarks embedding algorithm with multiple description coding," *Information Sciences*, vol. 181, no. 16, pp. 3379–3396, 2011.

[8] Z. Meng and J. S. Pan, "QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): an enhanced structure for differential evolution," *Knowledge-Based Systems*, vol. 155, pp. 35–53, 2018.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] T. Joyce and J. M. Herrmann, "A review of no free lunch theorems, and their implications for metaheuristic optimisation," *Nature-inspired algorithms and applied optimization*, vol. 44, pp. 27–51, 2018.

[11] P. C. Song, S. C. Chu, J. S. Pan, and H. Yang, "Simplified Phasmatodea population evolution algorithm for optimization," *Complex & Intelligent Systems*, pp. 1–19, 2021.

[12] J. S. Pan, N. Liu, S. C. Chu, and T. Lai, "An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems," *Information Sciences*, vol. 561, pp. 304–325, 2021.

[13] J. Wang, B. Pan, Q. R. Wang, and Q. Ding, "A chaotic key expansion algorithm based on genetic algorithm," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 10, pp. 289–299, 2019.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, Perth, WA, Australia, 1995.

[15] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, 1995.

[16] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[17] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[18] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[19] J. Zhou, Z. Ji, and L. Shen, "Simplified intelligence single particle optimization based neural network for digit recognition," in *2008 Chinese Conference on Pattern Recognition*, pp. 1–5, Beijing, China, 2008.

[20] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE transactions on systems, man, and cybernetics. Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[21] P. Zhang, Z. Cui, Y. Wang, and S. Ding, "Application of BPNN optimized by chaotic adaptive gravity search and particle swarm optimization algorithms for fault diagnosis of electrical machine drive system," *Electrical Engineering*, pp. 1–13, 2021.

[22] Y. Marinakis, M. Marinaki, and A. Migdalas, "A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows," *Information Sciences*, vol. 481, pp. 311–329, 2019.

[23] J. S. Pan, X. Wang, S. C. Chu, and T. Nguyen, "A multi-group grasshopper optimisation algorithm for application in capacitated vehicle routing problem," *Data Science and Pattern Recognition*, vol. 4, pp. 41–56, 2020.

[24] Z. J. Wang, Z. H. Zhan, W. J. Yu et al., "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE transactions on cybernetics*, vol. 50, pp. 2715–2729, 2019.

[25] H. Wu, J. Liu, Z. Dong, and Y. Liu, "A hybrid mobile node localization algorithm based on adaptive MCB-PSO approach in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2020, 17 pages, 2020.

[26] R. Abdul Razak, S. Sukumar, and H. Chung, "Scalar field estimation with mobile sensor networks," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 9, pp. 4287–4305, 2021.

[27] Y. Lv, L. Chen, D. Zhang, and J. Liu, "Research on indoor mobile node localization based on wireless sensor networks," *Engineering Letters*, vol. 29, 2021.

[28] C. I. Wu, H. Y. Kung, C. H. Chen, and L. C. Kuo, "An intelligent slope disaster prediction and monitoring system based on WSN and ANP," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4554–4562, 2014.

[29] J. S. Pan, L. Kong, T. W. Sung, P. W. Tsai, and V. Snášel, "α-Fraction first strategy for hierarchical model in wireless sensor networks," *Journal of Internet Technology*, vol. 19, pp. 1717–1726, 2018.

[30] J. S. Pan, L. Kong, T. W. Sung, P. W. Tsai, and V. Snášel, "A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set," *Journal of Internet Technology*, vol. 19, pp. 1111–1118, 2018.

[31] Q. W. Chai, S. C. Chu, J. S. Pan, and W. M. Zheng, "Applying adaptive and self assessment fish migration optimization on localization of wireless sensor network on 3-D Te rrain," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 11, pp. 90–102, 2020.

[32] Z. G. Du, J. S. Pan, S. C. Chu, H. J. Luo, and P. Hu, "Quasi-affine transformation evolutionary algorithm with communication schemes for application of RSSI in wireless sensor networks," *IEEE Access*, vol. 8, pp. 8583–8594, 2020.

[33] J. Wu, M. Xu, F. F. Liu, M. Huang, L. Ma, and Z. M. Lu, "Solar wireless sensor network routing algorithm based on multi-objective particle swarm optimization," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 12, pp. 1–11, 2021.

[34] X. Xue and J. S. Pan, "A compact co-evolutionary algorithm for sensor ontology meta-matching," *Knowledge and Information Systems*, vol. 56, no. 2, pp. 335–353, 2018.

[35] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE transactions on systems, man, and cybernetics. Part B (Cybernetics)*, vol. 34, no. 1, pp. 718–724, 2004.

[36] G. Cheung, W. T. Tan, and T. Yoshimura, "Real-time video transport optimization using streaming agent over 3G wireless networks," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 777–785, 2005.

[37] P. G. Norman, "The new AP101S general-purpose computer (GPC) for the space shuttle," *Proceedings of the IEEE*, vol. 75, no. 3, pp. 308–319, 1987.

[38] T. K. Dao, T. S. Pan, T. Nguyen, and S. C. Chu, *A Compact Artificial Bee Colony Optimization for Topology Control Scheme in Wireless Sensor Networks*, 2015.

[39] A. M. Wazwaz, "The sine-cosine method for obtaining solutions with compact and noncompact structures," *Applied Mathematics and Computation*, vol. 159, no. 2, pp. 559–576, 2004.

[40] T. Nguyen, J. S. Pan, and T. K. Dao, "A compact bat algorithm for unequal clustering in wireless sensor networks," *Applied Sciences*, vol. 9, no. 10, p. 1973, 2019.

[41] F. Neri, E. Mininno, and G. Iacca, "Compact particle swarm optimization," *Information Sciences*, vol. 239, pp. 96–121, 2013.

[42] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pp. 29–39, New York, USA, 2008.

[43] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 35, pp. 78–92, 2005.

[44] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 45–57, New York, USA, 2004.

[45] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, vol. 201212, pp. 281–295, 2013.

[46] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*, pp. 43–55, Springer, 2019.

[47] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 4–31, 2011.

[48] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[49] X. S. Yang and X. He, "Bat algorithm: literature review and applications," *International Journal of Bio-inspired computation*, vol. 5, no. 3, pp. 141–149, 2013.

[50] S. M. Mirjalili, S. Z. Mirjalili, S. Saremi, and S. Mirjalili, "Sine cosine algorithm: theory, literature review, and application in designing bend photonic crystal waveguides," *Nature-inspired optimizers*, vol. 811, pp. 201–217, 2020.