Research Article

A Novel Compact Particle Swarm Optimization for Optimizing Coverage of 3D in Wireless Sensor Network

Ning Liu, Qing-Wei Chai, Shangkun Liu, and Wei-Min Zheng

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

Correspondence should be addressed to Wei-Min Zheng; zhengwm901@126.com

Received 20 August 2022; Revised 7 October 2022; Accepted 10 October 2022; Published 18 October 2022

Academic Editor: Shaohua Wan

Copyright © 2022 Ning Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The problem of 3D coverage in a wireless sensor network (WSN) has always been an urgent problem to be solved. A novel compact particle swarm optimization algorithm (ncPSO) to solve this problem is proposed in this paper. This algorithm uses a Pareto distribution to describe the position of particle swarms. The ncPSO reduces memory usage compared to traditional heuristics. The ncPSO using the Pareto distribution is less likely to fall into local optima than other compact algorithms using the Gaussian distribution. We also add Gaussian perturbation strategy to the algorithm to better avoid the algorithm falling into local optimum. Among the test functions of CEC2013, the ncPSO achieves remarkable optimization ability on most test functions. Finally, we apply ncPSO to the 3D coverage problem of sensors. Compared with other algorithms, the ncPSO achieves satisfactory results.

1. Introduction

The heuristic algorithms develop rapidly, and many heuristic algorithms and their improved algorithms are proposed, such as particle swarm optimization (PSO) [1], genetic algorithm (GA) [2], whale optimization algorithm (WOA) [3], Black Hole (BH) [4], Artificial Bee Colony (ABC) [5], Sine Cosine Algorithm (SCA) [6], and Bat Algorithm (BA) [7]. The theorem of No Free Lunch (NFL) shows that no algorithm is applicable to all problems [8, 9]. Intelligent computing is applied in many fields such as architecture, transportation, education, medicine, and economics [10, 11]. Many improved heuristic algorithms have been proposed to solve different problems [12, 13]. Ji et al. proposed a multisurrogate-assisted multitasking particle swarm optimization to solve the problem of expensive multimodal optimization. Ji et al. introduced multitasking evolution into PSO, which makes the algorithm have faster convergence speed and better optimization ability in high-dimensional problems [14]. Song et al. proposed variable-size cooperative coevolutionary particle swarm optimization to solve the problem of “dimensional disaster.” Using this algorithm, the feature selection problems with high-dimensional data are solved [15]. For the problem of mixed continuous and discrete data, Wang et al. proposed a new particle swarm optimization algorithm to solve this problem. Wang et al. tested the performance of the proposed algorithm with other algorithms on 28 test functions of CEC2013. The results show that the proposed algorithm by Wang et al. has better performance than other algorithms [16]. Fan et al. proposed a random reselection particle swarm optimization algorithm and applies it to the parameter optimization of solar photovoltaic modules [17]. The problem with existing heuristic algorithms is that they need large memory space. In order to solve this problem, compact Artificial Bee Colony (cABC) [18], compact Particle Swarm Optimization (cPSO) [19], compact Sine Cosine Algorithm (cSCA) [20], and compact Bat Algorithm (cBA) [21] are proposed. These algorithms with compact strategy all use normal distribution to represent the position of population. Although they can reduce the usage of memory space, they fall into local optimum simply. We propose a novel compact strategy to solve this problem.

The WSN has impacted every aspect of the life of people and has attracted the attention of many scholars [22, 23]. The goal of the WSN is to send collected information from
monitoring area to base station for data analysis [24]. There
are many types of sensors such as humidity, temperature,
and brightness. Due to the many kinds of sensors which
can collect different kinds of information, the WSN is
applied in military, healthcare, environmental studies,
and start home [25–27]. The coverage problem of the WSN is
one of the most important problems to be solved. There
are two main types of the coverage problem, namely, full
coverage and partial coverage [28]. In this paper, we mainly
research the partial coverage problems. However, the partial
coverage problem in a given area includes two categories: the
maximum coverage rate under a certain number of sensor
nodes and the minimum number of sensor nodes to achieve
a certain coverage rate. Increasing coverage rate by increas-
ing the number of sensors will reduce the lifetime of the
WSN. So it is a very vital topic to achieve maximum cover-
age rate with the minimum number of nodes.

Yoon and Kim first proposed that the coverage problem
of WSN is NP-hard [29]. The coverage problem of WSN be
studied in 3D is lesser than that in 2D. Researchers have pro-
posed many methods to solve this problem of WSN in 2D.
However, many solutions that perform well on 2D problems
do not work well in 3D. The 3D coverage problems are more
difficult than the 2D coverage problems, and the 3D cover-
age problems conform closer to the real world. In the 3D
coverage problem, the sensor will encounter obstacles and
affect the coverage of the area. The obstacle blocks the trans-
mission of the sensor signal, so that the sensor cannot mon-
itor the information of the area blocked by the obstacle. In
this paper, we discuss how to solve the two-class partial cov-

The main contributions of this paper are as follows:

1. Propose a novel compact strategy to solve the prob-
lem of large memory usage of heuristic algorithm.
The novel compact strategy uses the Pareto distri-
bution to describe the position of the whole particle
swarm instead of the position of each particle to
reduce memory usage

2. Use Pareto perturbation to better avoid algorithm
falling into local optimum. The Pareto perturbation
is a heavy-tailed distribution. This distribution
makes it easier for the algorithm to find the position
outside the local optimum. This distribution enables
particles to get the solution vector which deviates
from the current optimum

3. Combine the novel compact strategy and Pareto per-
turbation with PSO to solve the 3D coverage prob-
lem of sensor network. The performance of the
ncPSO and other algorithms is compared on the
CEC2013, and the optimization effect of ncPSO is
also compared with other algorithms in 3D coverage
problem

The rest of the paper is structured as follows. Section 2
covers a related work. We introduce the process of coming
up with the ncPSO in Section 3. We show the performance
test results in Section 4. Section 5 applies the ncPSO to the
coverage problem of WSN. We give conclusion and outlook
in Section 6.

2. Related Work

We introduce the PSO, cPSO, and current research progress
on WSN coverage in this section. The coverage of WSN is
one of the most important issues in current research.
Because of the small size of the sensor, algorithms that
require large memory are not suitable for it. Because intelli-
gent computing can effectively solve the 3D coverage prob-
lem of WSN, the ncPSO is proposed to conveniently use
for sensor in this paper. The traditional heuristic algorithms
have the disadvantage of large memory usage. The proposed
compact algorithms using Gaussian distribution have the
disadvantage of easily falling into local optimum. In view
of these two shortcomings, we propose the ncPSO that uses
Pareto distribution to represent the position of particle
swarm. Because the ncPSO solves the problems of large
memory usage and easy falling into local optimum, the
ncPSO is suitable for solving 3D coverage problems in WSN.

2.1. Particle Swarm Optimization. The PSO is proposed by
Kennedy and Eberhart in 1995 according to the foraging
behavior of birds [1]. The PSO is the most representative
heuristic algorithm. Each particle records its own position,
denoted by \( X_i \). Every particle is a solution to the problem.
The PSO will generate a current optimal solution pBest
and a historical optimal solution gBest in each iteration.
Each particle affected by current optimal position and his-
torical optimal position will have a flight speed, and accord-
ing to the current position and flight speed, they will move
to a new position in the next iteration. The positions of par-
ticles are updated according to

\[
V_{i}^{g+1} = w V_{i}^{g} + c_1 \times \text{rand} \times (\text{pBest}_i - X_{i}^{g}) \\
+ c_2 \times \text{rand} \times (\text{pBest} - X_{i}^{g}),
\]

\[
X_{i}^{g+1} = X_{i}^{g} + V_{i}^{g+1},
\]

where \( X_{i}^{g} \) represents the position in the \( g \)th iteration of the \( i \)
th particle, \( V_{i}^{g} \) represents the velocity in the \( g \)th iteration of
the \( i \)th particle, \( w \) is a inertia weight, \( c_1 \) and \( c_2 \) are two ran-
donumber in \([0,1]\), pBest, represents the current optimal
solution, and gBest represents the historical optimal solution. The movement process of the particle swarm is shown in Figure 1.

\[ V_1, V_2, \text{ and } V_3 \text{ represent three speeds in Equation (1). } V_1 \text{ represents the } \omega V^{\mu}_g, V_2 \text{ represents the (pBest} - X^g_i), \text{ and } V_3 \text{ represents the (pBest} - X^g_i). V^{\mu+1}_g \text{ is the combination of the three speeds. The new position of } X^{g+1}_i \text{ is based on } X^g_i \text{ and } V^{\mu+1}_i. \text{ The pseudocode of PSO is described in Algorithm 1.}

2.2. Compact Particle Swarm Optimization. Microhardware has strict requirements on memory, and excessive memory space affects the normal use of hardware. The cPSO is proposed by Ferrante to adapt limited hardware availability [19]. The cPSO uses the distribution probability of the population to describe the position of the population. For each dimension, cPSO only uses perturbation vectors \( PV^{\mu}(\mu, \sigma) \) to describe the swarm. The PV is a Gaussian distribution, \( \mu \) is the mean of the PV, and \( \sigma \) is the standard deviation of the PV. The compact strategy produces a winner and a loser based on the comparison. \( \mu \) and \( \sigma \) will update according to winner and loser in each iteration. The update formulas of \( \mu \) and \( \sigma \) are shown as Equations (3) and (4).

\[
\begin{align*}
\mu^{\mu+1} &= \mu^g + \frac{1}{Np}(\text{winner} - \text{loser}), \\
\sigma^{\mu+1} &= \sqrt{(\sigma^g)^2 + (\mu^g)^2 - (\mu^{\mu+1})^2 + \frac{1}{Np}(\text{winner}^2 - \text{loser}^2)},
\end{align*}
\]

where \( Np \) is the virtual number of particles. A large number of experiments show that when \( Np \) is set to 300, the algorithm has the best effect. The memory usage of PSO and cPSO is shown in Table 1.

![Algorithm: The pseudocode of PSO.](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>Particles ( \times ) dimension</td>
</tr>
<tr>
<td>cPSO</td>
<td>2 ( \times ) dimension</td>
</tr>
</tbody>
</table>

Table 1 shows that the usage of PSO memory depends on the number of particle and the dimension of problem, but the usage of cPSO memory only depends on the dimension of problem. The process of compact strategy is introduced. Firstly, generate corresponding probability distribution function (PDF) according to PV vector. Then, calculate cumulative distribution function (CDF) by PDF and normalize CDF within the range of \([0,1]\). Then a random number will be generated to calculate a inverse cumulative distribution function (iCDF). The value of iCDF is represented as the position of the particle swarm. The equations of PDF and CDF of the Gaussian distribution are shown in

\[
\begin{align*}
\text{PDF} &= \frac{e^{-((x-\mu)^2/2\sigma^2)}/\sqrt{2\pi}}{\text{erf} \left( (\mu + 1)/\sqrt{2}\sigma \right) - \text{erf} \left( (\mu - 1)/\sqrt{2}\sigma \right)}, \\
\text{CDF} &= \frac{\text{erf} \left( (\mu + 1)/\sqrt{2}\sigma \right) + \text{erf} \left( (\mu - 1)/\sqrt{2}\sigma \right)}{\text{erf} \left( (\mu + 1)/\sqrt{2}\sigma \right) - \text{erf} \left( (\mu - 1)/\sqrt{2}\sigma \right)},
\end{align*}
\]

where the erf is the error function. The pseudocode of cPSO that combining compact strategy into PSO is shown in Algorithm 2.

2.3. Coverage of WSN in 3D. The current research problems on sensor network coverage problem mainly include 2D and 3D. Scholars study problems on 2D more than 3D. The 3D
coverage problem is more in line with real life and should be the focus of research. Amulya et al. propose a modified particle swarm optimization to achieve maximum coverage in 2D. The result shows that using this algorithm could get bigger coverage than using conventional PSO [30]. Nguyen et al. proposed that different sensors have different transmission radius. They discussed how the sensors work together under different sensor transmission radius to achieve maximum coverage in 2D [31]. Huynh et al. applied the improved cuckoo algorithm to the 2D sensor coverage problem and achieved good results [32]. Osamah et al. used the bee algorithm to 2D coverage to reduce resource usage [33]. Junaid et al. proposed a method that dynamically adjust the state of the sensors to achieve maximum coverage while extending the lifetime of the sensor network [34]. Wang et al. added the idea of reverse learning to WOA to speed up the global search speed and make the sensor reach the maximum coverage faster in 2D [35].

Pan et al. improved the BH and applied it to the problem of 3D sensor network coverage with obstacles [36]. The feasibility of BH is proved by simulation experiments. However, there is still room for improvement in the optimization ability. Riham et al. proposed a new network configuration strategy for the 3D sensor network coverage problem [37]. Riham et al. proved that this strategy can better optimize the coverage problem than other similar strategies through simulation experiments. Wang et al. applied the improved virtual force algorithm to the $k$-coverage problem in underwater 3D and proved its feasibility through simulation experiments [38]. Cao et al. proposed a particle swarm optimizer with distributed parallelism and used it in the industrial field to achieve the 3D coverage problem with obstacle [39]. William et al. proposed a new strategy to achieve dynamic 3D coverage with limited energy using mobile sensors. Simulation experiments show that the strategy worked well [40].

3. A Novel Compact Particle Swarm Optimization

The original cPSO uses a Gaussian distribution to describe distribution statistics on the position of particle swarms. Gaussian distribution is a thin-tailed distribution. The biggest problem with this is that using a Gaussian distribution can easily lead to getting stuck in a local optimal solution. Considering this deficiency, we propose a novel compact particle swarm optimization (ncPSO). We use the Pareto distribution instead of the Gaussian distribution to express the position of particle swarms. Pareto distribution is a heavy-tailed distribution. It falls more slowly than Gaussian distribution for $x \to \infty$. So the Pareto distribution has a greater probability of taking values that deviate from the normal level. This feature makes it fall into local optimum difficult in the search. The formulas of PDF and CDF of the Pareto are shown in

$$
\text{PDF} = \frac{1}{\sigma} \left(1 + k \frac{(x - \theta)}{\sigma}\right)^{-1-k},
$$

Algorithm 2: The pseudocode of cPSO.
CDF = \begin{cases} 
1 - \left( 1 + k \frac{(x - \theta)}{\sigma} \right)^{-1/k}, & k \neq 0, \\
1 - \exp \left( - \frac{(x - \theta)}{\sigma} \right), & k = 0.
\end{cases}

The generalized Pareto distribution has three parameters: threshold parameter \( \theta \), scale parameter \( \sigma \), and shape parameter \( k \). The figures of Pareto PDF and Pareto CDF are shown in Figure 2. The descending speed of the right tail of the Pareto distribution is significantly slower than that of the Gaussian distribution.

The ncPSO uses uniform distribution function to randomly generate the initial positions and initial velocities of particles within a feasible solution range. Because the Pareto distribution is one-sided function, we also generate a symmetric position on the other side at each iteration, then compare the two positions and take the better one as the next pBest. Because the Pareto distribution is a heavy-tailed distribution, it may occur that the CDF value does not reach 1 within the feasible solution range, resulting in the value of the inverse cumulative distribution function out of bounds. We solve this problem by regenerating a new random numbers to generate iCDF. Among the three
parameters of the Pareto distribution, \( k > 0 \) is guaranteed, \( \theta \) determines the convergence position, and \( \sigma \) determines the convergence speed. The ncPSO no longer restricts that the position of particle must be normalized to \([-1,1]\), and the position of the particle is obtained directly in feasible solution space.

It is possible that only one dimension falling into the local optimum makes bad result of whole swarm. The ncPSO selects one dimension of the particle randomly to perform a Gaussian perturbation to better avoid the algorithm falling into local optimum in each iteration. The perturbation formula is shown in

\[
X_d = \text{normrnd} \left( X^d, C \right) \tag{9}
\]

where the \( C \) is a constant and determined by the actual problem to be solved. After Gaussian perturbation, the particle moves according to Equations (1) and (2) to a new position. Then, ncPSO compares the position of pBest and new position of particle to generate a winner and loser to update \( \theta \) and \( \sigma \) of the Pareto distribution. The update formulas of \( \theta \) and \( \sigma \) are shown as

\[
\theta_{g+1} = \sigma_{g+1} = \frac{1}{Np} \left( \text{winner} - \text{loser} \right), \tag{10}
\]

\[
\sigma_{g+1} = \sqrt{\left( \theta^{g+1} \right)^2 + \left( \sigma^{g+1} \right)^2 - \left( \theta^{g+1} \right)^2} + \frac{1}{Np} \left( \text{winner}^2 - \text{loser}^2 \right). \tag{11}
\]
After updating $\theta$ and $\sigma$, the pBest will be generated. As the number of iterations increases, $\sigma$ gets smaller and smaller, and the ncPSO converges at the convergence position $\theta$ faster and faster.

The difference between ncPSO and other algorithms is that ncPSO uses Pareto distribution of heavy-tailed distribution to describe the position of the whole particle swarm instead of Gaussian distribution. The advantage of using Pareto distribution is that it can reduce the probability of the algorithm falling into local optimum. Another difference is that ncPSO uses mathematical distribution to represent the position of particle swarm instead of storing the position of each particle, which greatly reduces the use of memory units. Taking 30 particles and 40 dimensions as an example, the traditional PSO needs $40 \times 30 = 1200$ memory units to store the position of each particle in each dimension, while ncPSO only needs $40 \times 2 = 80$ memory units to express the position of the whole particle swarm. The ncPSO has a good applicability to microdevices, and the excellent optimization performance of ncPSO is a new optimization method for 3D coverage of WSN. The pseudocode of ncPSO is shown in Algorithm 3.

4. The Performance Tests of Algorithms

We mainly compare the performance of ncPSO with conventional heuristic algorithms and other compact algorithms in this section. The CEC2013 test functions are used for performance test. The functions in CEC2013 are diverse and very convincing [41]. The 28 functions in CEC2013 are represented by $f_1 - f_{28}$, respectively. $f_1 - f_5$ are 5 unimodal functions, $f_6 - f_{20}$ are 15 multimodal functions, and $f_{21} - f_{28}$ are 8 composition functions.
Figure 3: Continued.
Figure 3: Continued.
Figure 3: Continued.
4.1. Comparison between ncPSO and Conventional Heuristic Algorithms. Firstly, we compare ncPSO with conventional heuristic algorithms. We choose PSO, ABC, WOA, and BH. All algorithms are run 20 times on each test function, and then, the average is calculated as the experiment result. Simultaneously Wilcoxon’s sign rank test is taken with a significant level $\alpha = 0.05$. The memory and function calls of the algorithms are shown in Table 2.
Table 5: The parameter settings of all compact algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>cPSO</td>
<td>$D = 50$, virtual $N_p = 300$, $w = 0.2$, $r_1 = -0.07$, $r_2 = 3.74$</td>
</tr>
<tr>
<td>cABC</td>
<td>$D = 50$, virtual $N_p = 300$, limit = 100, food number = 3</td>
</tr>
<tr>
<td>cSCA</td>
<td>$D = 50$, virtual $N_p = 300$, probability switch = 0.5</td>
</tr>
<tr>
<td>cBA</td>
<td>$D = 50$, virtual $N_p = 300$, loudness rate = 0.5, pulse rate = 0.5, $f_{\min} = 1$, $f_{\max} = 2$</td>
</tr>
<tr>
<td>ncPSO</td>
<td>$D = 50$, virtual $N_p = 300$, $r_1 = 2.0$, $r_2 = 2.0$</td>
</tr>
</tbody>
</table>

Table 6: Comparison of performance testing and Wilcoxon’s sign test of compact algorithms.

<table>
<thead>
<tr>
<th></th>
<th>cABC</th>
<th>cPSO</th>
<th>cBA</th>
<th>cSCA</th>
<th>ncPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$2.42E + 04$</td>
<td>$-1.39E + 03$</td>
<td>$-1.40E + 03$</td>
<td>$5.33E + 04$</td>
<td>$-1.40E + 03$</td>
</tr>
<tr>
<td>$f_2$</td>
<td>$6.92E + 08$</td>
<td>$2.71E + 07$</td>
<td>$4.24E + 06$</td>
<td>$7.31E + 08$</td>
<td>$1.74E + 06$</td>
</tr>
<tr>
<td>$f_3$</td>
<td>$2.33E + 11$</td>
<td>$5.53E + 09$</td>
<td>$4.82E + 09$</td>
<td>$1.88E + 15$</td>
<td>$1.04E + 09$</td>
</tr>
<tr>
<td>$f_4$</td>
<td>$1.42E + 05$</td>
<td>$4.80E + 03$</td>
<td>$2.81E + 05$</td>
<td>$1.81E + 05$</td>
<td>$8.12E + 04$</td>
</tr>
<tr>
<td>$f_5$</td>
<td>$5.79E + 03$</td>
<td>$-9.91E + 02$</td>
<td>$-1.00E + 03$</td>
<td>$1.43E + 04$</td>
<td>$-1.00E + 03$</td>
</tr>
<tr>
<td>$f_6$</td>
<td>$1.34E + 03$</td>
<td>$-8.23E + 02$</td>
<td>$-8.51E + 02$</td>
<td>$3.01E + 03$</td>
<td>$-8.23E + 02$</td>
</tr>
<tr>
<td>$f_7$</td>
<td>$-4.44E + 02$</td>
<td>$-5.90E + 02$</td>
<td>$3.63E + 12$</td>
<td>$1.48E + 04$</td>
<td>$-6.84E + 02$</td>
</tr>
<tr>
<td>$f_8$</td>
<td>$-6.79E + 02$</td>
<td>$-6.79E + 02$</td>
<td>$-6.79E + 02$</td>
<td>$-6.79E + 02$</td>
<td>$-6.79E + 02$</td>
</tr>
<tr>
<td>$f_9$</td>
<td>$-5.24E + 02$</td>
<td>$-5.37E + 02$</td>
<td>$-5.10E + 02$</td>
<td>$-5.26E + 02$</td>
<td>$-5.44E + 02$</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>$4.37E + 03$</td>
<td>$-4.75E + 02$</td>
<td>$-4.99E + 02$</td>
<td>$7.01E + 03$</td>
<td>$-4.99E + 02$</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>$4.47E + 02$</td>
<td>$3.16E + 02$</td>
<td>$2.47E + 03$</td>
<td>$4.60E + 02$</td>
<td>$-3.84E + 02$</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>$7.07E + 02$</td>
<td>$4.95E + 02$</td>
<td>$7.91E + 03$</td>
<td>$9.23E + 02$</td>
<td>$2.73E + 02$</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>$8.51E + 02$</td>
<td>$5.27E + 02$</td>
<td>$6.87E + 03$</td>
<td>$9.93E + 02$</td>
<td>$3.51E + 02$</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>$1.44E + 04$</td>
<td>$9.17E + 03$</td>
<td>$8.67E + 03$</td>
<td>$1.60E + 04$</td>
<td>$1.84E + 03$</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>$1.51E + 04$</td>
<td>$9.98E + 03$</td>
<td>$1.03E + 04$</td>
<td>$1.54E + 04$</td>
<td>$9.55E + 03$</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>$2.04E + 02$</td>
<td>$2.03E + 02$</td>
<td>$2.01E + 02$</td>
<td>$2.04E + 02$</td>
<td>$2.02E + 02$</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>$1.95E + 03$</td>
<td>$1.03E + 03$</td>
<td>$1.66E + 04$</td>
<td>$1.46E + 03$</td>
<td>$3.51E + 02$</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>$2.32E + 03$</td>
<td>$1.07E + 03$</td>
<td>$1.58E + 04$</td>
<td>$1.59E + 03$</td>
<td>$9.85E + 02$</td>
</tr>
<tr>
<td>$f_{19}$</td>
<td>$6.32E + 04$</td>
<td>$5.51E + 02$</td>
<td>$8.67E + 02$</td>
<td>$1.83E + 04$</td>
<td>$5.02E + 02$</td>
</tr>
<tr>
<td>$f_{20}$</td>
<td>$6.25E + 02$</td>
<td>$6.24E + 02$</td>
<td>$6.25E + 02$</td>
<td>$6.25E + 02$</td>
<td>$6.25E + 02$</td>
</tr>
<tr>
<td>$f_{21}$</td>
<td>$5.69E + 03$</td>
<td>$1.58E + 03$</td>
<td>$1.61E + 03$</td>
<td>$5.17E + 03$</td>
<td>$1.64E + 03$</td>
</tr>
<tr>
<td>$f_{22}$</td>
<td>$1.66E + 04$</td>
<td>$1.22E + 04$</td>
<td>$1.11E + 04$</td>
<td>$1.79E + 04$</td>
<td>$4.31E + 03$</td>
</tr>
<tr>
<td>$f_{23}$</td>
<td>$1.71E + 04$</td>
<td>$1.28E + 04$</td>
<td>$1.34E + 04$</td>
<td>$1.69E + 04$</td>
<td>$1.40E + 04$</td>
</tr>
<tr>
<td>$f_{24}$</td>
<td>$1.42E + 03$</td>
<td>$1.43E + 03$</td>
<td>$1.67E + 03$</td>
<td>$1.51E + 03$</td>
<td>$1.36E + 03$</td>
</tr>
<tr>
<td>$f_{25}$</td>
<td>$1.59E + 03$</td>
<td>$1.56E + 03$</td>
<td>$1.64E + 03$</td>
<td>$1.58E + 03$</td>
<td>$1.60E + 03$</td>
</tr>
<tr>
<td>$f_{26}$</td>
<td>$1.64E + 03$</td>
<td>$1.62E + 03$</td>
<td>$1.40E + 03$</td>
<td>$1.71E + 03$</td>
<td>$1.48E + 03$</td>
</tr>
<tr>
<td>$f_{27}$</td>
<td>$3.66E + 03$</td>
<td>$3.42E + 03$</td>
<td>$5.29E + 03$</td>
<td>$3.82E + 03$</td>
<td>$3.32E + 03$</td>
</tr>
<tr>
<td>$f_{28}$</td>
<td>$7.13E + 03$</td>
<td>$3.48E + 03$</td>
<td>$8.61E + 04$</td>
<td>$8.68E + 03$</td>
<td>$5.16E + 03$</td>
</tr>
</tbody>
</table>

In Table 2, we can see that ncPSO requires less memory and function calls than other algorithms. Table 3 shows the parameter settings of all algorithms.

The test results of these algorithms on CEC2013 and Wilcoxon’s sign test results are shown in Table 4. The symbol “>” indicates that the performance of ncPSO is better than other algorithms. The symbol “=” indicates that the performance of ncPSO is as good as other algorithms. The symbol “<” indicates that the performance of ncPSO is worse than other algorithms.
Figure 4: Continued.
Figure 4: Continued.
Figure 4: Continued.
According to the data from Table 4, we can see that the ncPSO has better performance than PSO on 19 functions. And compared with ABC, the ncPSO achieves better results on 19 functions too. The WOA is better than ncPSO in only two functions. The ncPSO performs better than BH on 21 functions. Because some images have very...

Figure 4: The convergence of some test functions of compact algorithms.
little discrimination, we select a part of representative images of the convergence process for display. The convergence of some test functions is shown in Figure 3.

We can see that ncPSO has faster convergence speed and better ability to find optimization on $f_{11}$, $f_{14}$, $f_{17}$, $f_{22}$, $f_{26}$. The ncPSO does not have a better optimization ability than ABC on $f_{13}$. Although the optimization ability of ncPSO is not as good as that of PSO, the convergence speed of ncPSO is faster on $f_{15}$ and $f_{23}$. The BH performs better than ncPSO on $f_{23}$.

4.2. Comparison between ncPSO and Other Compact Algorithms. We compare ncPSO with other compact algorithms in this subsection. We choose cPSO, cABC, cSCA, and cBA. We also performed 20 experiments for each test function and also take Wilcoxon’s sign rank test. The memory and function calls are the same for all these compact algorithms. Table 5 shows the parameter settings of all algorithms.

Table 6 shows the test results of the algorithm on CEC2013 and Wilcoxon’s sign test results. According to Table 6, we can find that ncPSO performs better than cPSO on 20 functions. The cABC and cSCA are only better than ncPSO on $f_{25}$. The ncPSO has better performance than cBA on 18 functions and has same performance with cBA on 5 functions.

We also select some representative images of the convergence process for display in Figure 4. According to the figures from Figure 4, we can find that ncPSO has faster convergence speed and better ability to find optimization on $f_{9}$, $f_{14}$, $f_{15}$, $f_{17}$, $f_{22}$. The convergence ability of ncPSO is not as good as cBA in $f_{16}$, but it is better than cABC, cSCA, and cPSO. On the contrary, the convergence ability of ncPSO is better than cBA on $f_{25}$, but not as good as cABC, cSCA, and cPSO. On $f_{23}$, the convergence ability of ncPSO is worse than that of cPSO and cBA, but better than that of cSCA and cABC.

5. Application in Optimizing Coverage of 3D in Wireless Sensor Network

The problem of sensor coverage in 3D environment is more complicated than that in 2D. Compared with 2D coverage, 3D coverage needs to consider the influence of terrain on coverage effect. The complicated terrain makes it difficult for sensors to collect information. We use the 3D model generated by “peak” function in MATLAB as the environment for simulation experiments. The control variables of the experiment in this section are the 3D simulation environment and the communication radius of sensors. The simulation environment is shown in Figure 5. The communication radius of sensors are set to 5 m.

In this simulation environment, as long as the first and second coordinates are known, the height in the 3D environment can be known, and then, the position of the sensor can be known. The ncPSO sets the dimension of particle swarm to twice the number of sensors. Two dimensions represent two coordinates of a sensor, and then, the value of the third dimension of the sensor in the terrain can be calculated by
the information of the topographic map, so that the position of the sensor can be obtained. The particle of ncPSO can be represented in the form of Table 7.

There is a monitoring obstacle problem in the 3D problem. The existence of obstacles makes it impossible for the sensor to pass through the obstacles to collect information about the location to be measured. If the terrain between the sensor and the monitoring node is higher than the line connecting the two points, the signal will be blocked and the monitoring point will not be monitored. The obstacle model in WSN is shown in Figure 6.

We suppose that nodes 1 and 2 are both in the radius of sensor in Figure 6; it can been seen that node 1 could be covered but node 2 could not be covered due to the obstacle. The judgment formula for whether the monitoring point is covered is shown in Equation (12). The formula for the coverage rate of the whole environment is shown in Equation (13).

\[
\text{Coverage}(s, n) = \begin{cases} 
1, & \text{distance}(s, n) \leq \text{radius} \text{ and there no obstacle}, \\
0, & \text{distance}(s, n) > \text{radius} \text{ or there are obstacles},
\end{cases}
\] (12)

\[
\text{Coverage rate} = \frac{1}{M} \cdot \left( \sum_{i=1}^{M} \left( \sum_{j=1}^{N} \text{Coverage}(s_j, n_i) \right) \right),
\] (13)

where \( s \) represents the sensor and \( n \) represents the measured node, \( N \) represents the total number of sensors, and \( M \) represents the total measured nodes. Then, we will carry out simulation experiments on the two problems of achieving the maximum coverage under a certain node and the minimum number of sensors to achieve a certain coverage.

5.1. The Maximum Coverage Rate under a Certain Number of Sensor Nodes. Different algorithms are used to optimize the problem of maximum coverage rate under a certain number of sensor nodes in this subsection. The radius of sensors is set to 5 m, and the number of sensors is set to 30, 40, 50, and 60, respectively. When the number of sensors is the same, different algorithms are used to optimize the coverage rate. The larger the coverage rate, the better the solution effect of the algorithm. Each group of experiments is carried out 10 times to ensure the accuracy of experiments, and the mean value is used as the experimental result. At the same time, we also calculate the standard deviation. Table 8 shows the results of simulation experiments.
Radius = 5 and the number of sensors = 30

Coverage rate vs. Iterations

(a) Sensor number = 30

Radius = 5 and the number of sensors = 40

Coverage rate vs. Iterations

(b) Sensor number = 40

Figure 7: Continued.
Radius = 5 and the number of sensors = 50

Figure 7: The convergence process of algorithms.
The maximum coverage rate and the minimum standard deviation of all algorithms are set in italics in Table 8. When the number of sensors is 30, 40, 50, and 60, respectively, the coverage of ncPSO to the simulation environment can reach 55.07%, 65.46%, 72.65%, and 79.02%, respectively. Table 8 shows that although the standard deviation of ncPSO is not the smallest, the optimization effect of ncPSO is much better than that of other algorithms. In other words, the ncPSO has the best optimization effect on 3D coverage because of its excellent optimization ability. It can be seen from Table 8 that as the number of sensors increases, the coverage rate of simulation environment also increases. In order to show the operation process of the algorithm, the optimization processes are shown in Figure 7.

Figure 7 shows that algorithms constantly update the positions of sensors, so that the coverage rates constantly increase. Compared with other algorithms, ncPSO has a greater improvement in coverage rate with different number of sensors. In the whole process, ncPSO has a strong ability to jump out of the local optimum, so it improves the coverage rate greatly compared with other algorithms.

5.2. The Minimum Number of Sensor Nodes to Achieve a Certain Coverage Rate. How to use the minimum number of sensors to achieve a certain coverage rate will be discussed in this subsection. When the number of sensors is limited, it is necessary to consider how to arrange the sensor positions to minimize the number of sensors to achieve the specified coverage. The radius of sensors is set to 5 m, and the coverage rates are set to 70% and 80%, respectively. We also conduct 10 simulation experiments per group to ensure the accuracy of the data. The number of sensors required by different algorithms is tested by simulation experiments. Table 9 shows the results of simulation experiments.

The minimum number of sensors to achieve the specified coverage and the minimum standard deviation are set in italics in Table 9. It can be seen from Table 9 that in order to achieve 70% coverage, ncPSO needs 58.2 sensors on average and cPSO needs 57.4 sensors on average. Compared with other algorithms, ncPSO requires fewer sensors to achieve 70% coverage. Because of good optimization ability of ncPSO, ncPSO requires fewer sensors than other algorithms to achieve 80% coverage.

6. Conclusion

We propose the ncPSO in this paper. This algorithm reduces memory usage and also improves the problem that traditional compact algorithms tend to fall into local optima. This novel algorithm uses a Pareto distribution of the long-tailed distribution to describe the position of particle swarms. The ncPSO uses Pareto distribution and Gaussian perturbation to avoid feasible solutions falling into local optima. Then, we conduct the performance test of the proposed ncPSO on CEC2013 and compare it with the conventional heuristic and compact algorithms. The results show that the ncPSO has good optimization ability in most cases compared with other algorithms. Finally, we apply the ncPSO to the 3D sensor coverage problem and compare it with other algorithms. The simulation results show that the ncPSO can achieve better results in solving these problems.

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 is funded by the National Natural Science Foundation of China (No. 619320).

References


Wireless Communications and Mobile Computing 23


