Complex & Intelligent Systems
https://doi.org/10.1007/s40747-024-01460-w

ORIGINAL ARTICLE

Surrogate-assisted sine Phasmatodea population evolution algorithm applied to 3D coverage of mobile nodes

Shu-Chuan Chu[1] · LuLu Liang[1] · Jeng-Shyang Pan[1,2] · LingPing Kong[3] · Jia Zhao[4]

Received: 29 November 2023 / Accepted: 17 April 2024

Abstract
Deploying static wireless sensor nodes is prone to network coverage gaps, resulting in poor network coverage. In this paper, an attempt is made to improve the network coverage by moving the locations of the nodes. A surrogate-assisted sine Phasmatodea population evolution algorithm (SASPPE) is used to evaluate the network coverage. A 50×50 hill simulation environment was tested for the number of nodes of 30 and 40 and radii of 3, 5 and 7, respectively. The results show that the SASPPE algorithm has the highest coverage, which can be up to 23.624% higher than the PPE algorithm, and up to 5.196% higher than the PPE algorithm, ceteris paribus. The SASPPE algorithm mixes the GSAM with LSAMs, which balances the computational cost of the algorithm and the algorithm's ability to find optimal results. The use of hierarchical clustering enhances the stable type of the LSAMs. In addition, LSAMs are easy to fall into local optimality when they are modeled with local data, and the use of sine Phasmatodea population evolution algorithm (Sine-PPE) for searching in LSAMs alleviates the time for the algorithm to fall into local optimality. On 30D, 50D, and 100D, the proposed algorithm was tested by 7 test functions. The results show that the algorithm has significant advantages on most functions.

Keywords Phasmatodea population evolution · Surrogate-assisted · Radial basis function networks · Removable nodes

Introduction

Wireless sensor networks (WSNs) are task-oriented networks whose ultimate goal is to detect the monitored area to obtain the required data [1, 2]. To ensure the accuracy and validity of the data, researchers have investigated the deployment strategy [3, 4], energy efficiency [5, 6], and coverage of the sensors [7, 8]. The deployment strategy of sensor nodes will directly affect the performance of the network; energy efficiency will directly determine the lifetime of the network; coverage is the basis for accomplishing the task of target monitoring and information acquisition, which directly affects the energy consumption of sensors and network lifetime, and is also a hot spot and focus of current research [9].

Network coverage can be classified in several ways depending on the application of wireless sensors. Based on how the nodes are deployed and based on whether the nodes are in motion are two common classifications [8, 10]. According to the deployment method of nodes, coverage can be divided into two categories: deterministic coverage and random coverage. Suppose the environment to be covered is known or does not change much. In that case, the location

✉ Jeng-Shyang Pan
jengshyangpan@gmail.com

Shu-Chuan Chu
scchu0803@gmail.com

LuLu Liang
lianglulu0403@163.com

LingPing Kong
lingping_kong@yahoo.com

Jia Zhao
zhaojia925@163.com

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

2 Department of Information Management, Chaoyang University of Technology, Taichung, Taiwan

3 Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava, Czech Republic

4 School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China

Published online: 11 May 2024

of the nodes can be determined in advance, and this case is suitable for deterministic coverage. If the environment to be covered is harsh, and the environmental conditions cannot be predicted in advance, only random deployment can be used for coverage. Random deployment of nodes produces random coverage effects, which may achieve optimal coverage, cause coverage redundancy, or make too many coverage blind areas [11, 12]. Whether or not the nodes are in motion, they can be divided into static and dynamic network coverage. Fixed networks are defined as networks in which the nodes are not in motion [13]. Dynamic network coverage means the nodes have some mobile capability to provide better monitoring services to the area [14]. Regardless of the coverage method, the goal is to maximize network coverage. Efficient target coverage algorithms can improve the performance of wireless sensors [15]. This paper uses the coverage method of mobile nodes to increase the network. Move the available sensor nodes to fill the void when a coverage gap occurs [16]. There are various ways to move the nodes, such as random, fixed, or adaptive. The advantage of random movement is that the movement method is simple and does not require planning the movement path. The disadvantage is that the stability of coverage cannot be guaranteed. Fixed movement means that the movement path of the nodes is scheduled in advance. This method can guarantee coverage to a certain extent. Adaptive movement automatically adjusts the nodes' direction based on the real-time status. This approach ensures the maximum coverage of the network but is more energy intensive. The choice of movement method needs to be selected on a case-by-case basis, or a combination of the above three methods can be adapted to specific scenarios. Meta-heuristic algorithms are often considered effective measures to solve optimization problems and practical algorithms to improve the coverage of wireless sensor networks [17–19]. In this paper, we focus on the coverage of movable nodes based on the improved meta-heuristic algorithm, and the following is an introduction to the meta-heuristic algorithm.

Meta-heuristic algorithms emerged to solve practical problems and are characterized by the ability to find approximate solutions to optimal solutions globally, with exploration and exploitation at their core. Various meta-heuristic algorithms have been proposed, for example, the gray wolf optimizer (GWO) [20], the whale optimization algorithm (WOA) [21], and the Phasmatodea population evolution (PPE) [22–24]. In some aspects, they have advantages over classical genetic algorithms (GA) [25], particle swarm optimization (PSO) [26–28], differential evolution (DE) [29] and other algorithms. Regardless of the type of meta-heuristic algorithm, random variables are introduced in the iterative process to prevent the algorithm from deviating from the optimal value [30]. This is the most significant difference between meta-heuristic algorithms and traditional methods. However, no single algorithm can be applied to all practical problems. Researchers have therefore proposed various improvements. For example, add restart policy [31], orthogonal learning strategy [32], transfer learning [33], or parallel strategy [34], etc. Zhu [35] proposed a multigroup-based Phasmatodea population evolution algorithm, which further improved the overall performance of PPE. Similarly, Liang [34] proposed a parallel Phasmatodea population evolution algorithm (PPPE) with a communication strategy. Chaotic models are often used to improve various algorithms, which can generate chaotic numbers between 0 and 1, and have been applied to applications. For example, Dynamic analysis of chaotic digital maps via state-mapping networks [36] and Exponential chaotic model for generating robust chaos [37], proposed in 2019. Bird swarm algorithms with chaotic mapping proposed in 2020 [38]. All the articles mentioned above can prove that chaotic mapping has good results.

In addition, adding a surrogate model not only improves the accuracy of the algorithm, but also solves the time-consuming problem of the algorithm in computing the expensive optimization problem. Compared to 2D coverage, 3D coverage is computationally expensive and is well suited for the use of surrogate models. To date, many existing regression models have been used as proxy models, such as Kriging models [39], radial basis function networks [40], polynomial regression [41], and artificial neural networks [42, 43]. However, all types of regression models have limitations, and no single regression model is suitable for all problems. For example, the Kriging model is a typical interpolation estimation method sensitive to numerical noise in the computation process. The modeling process could be more computationally heavy and may result in construction failure for problems with large sample sizes [39]. What makes RBFN unique from other neural network models is the introduction of a new variable, Euclidean distance, which transforms the fitted prediction problem in a multidimensional design space into a one-dimensional problem with itself as the independent variable. It is shown that the RBFN model works very well for problems with many design variables and a high degree of nonlinearity [44, 45]. At the moment, there are many articles using RBFN as a surrogate model. For example, Sun [46] proposed a surrogate-assisted cooperative particle swarm optimization algorithm (SACOSO) that uses two surrogate models to search for the global optimum collaboratively. Experiments show that the algorithm can find the best solution for problems of relatively high dimensionality in a limited number of calculations,fully levering the local approximation ability of RBFN. Wu [47] proposed a new surrogate assisted evolutionary algorithm MASTO based on adaptive technology. This algorithm fully utilizes the learning and local approximation capabilities of RBFN to dynamically establish the most promising RBF for each target, thereby improving the reliability of alternative model

Table 1 Introduction to various surrogate assisted algorithms

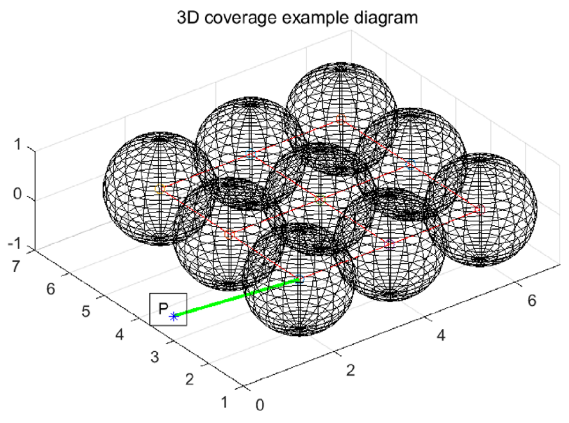
Algorithm	Definition	Model
SAPPE	Surrogate assisted phasmatodea population evolution algorithm	RBF
SAALO	Surrogate assisted ant lion optimizer	RBF
SA-COSO	Surrogate assisted cooperative swarm optimization	RBF, FES
RBFPSO	RBF-assisted particle swarm optimization	RBF
SHPSO	Surrogate-assisted Hierarchical particle swarm optimization	RBF
RFMOPSO	Random forest multi-objective particle swarm optimization	RF
SAMSO	Surrogate assisted multiswarm optimization algorithm	RBF
EASO	Evolutionary sampling assisted optimization	RBF
DSCPSO-EMM	Dual-surrogate-assisted cooperative particle swarm optimization algorithm	RBF, BP
FHSAPPSO	Fuzzy Hierarchical surrogate assists probabilistic particle swarm optimization	RBF
SAEA	Surrogate-assisted reference vector guided evolutionary algorithm	GP
HSAEA	A study on hierarchical surrogate-assisted evolutionary algorithm	PR, GP

predictions. Regis [48] proposed an algorithm for particle swarm optimization using a surrogate model (RBFPSO), which uses RBF to construct a surrogate model to identify the most promising particle positions. In addition, the RBF-PSO algorithm has shown outstanding performance in 36-D groundwater bioremediation problems, 14-D watershed calibration problems, and ten mostly 30-D testing problems. Liang [49] proposed a two-level substitution-assisted PPE algorithm (SAPPE), which improves the PPE algorithm using two surrogate models constructed by RBFN. The algorithm was applied to the 3D coverage problem of wireless sensors and achieved good results. RBFN has strong nonlinear mapping ability and can handle very complex nonlinear problems. The following is a summary of the above algorithms and other similar algorithms, please refer to Table 1 for details.

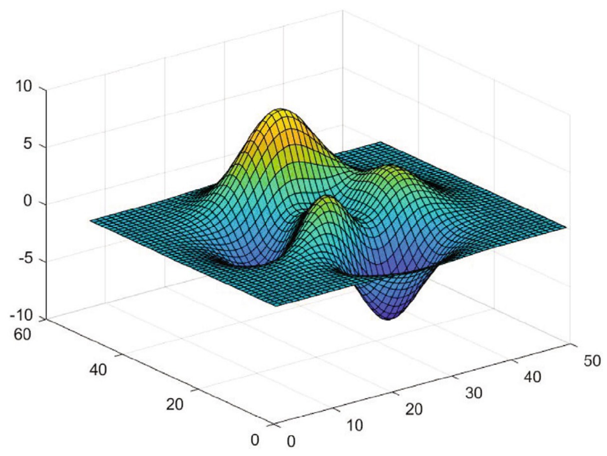
This does not mean that only RBFN is effective. When selecting a proxy model, it is necessary to consider factors such as algorithm, characteristics of the problem to be solved, data size, and computing resources comprehensively. Gu [50] proposed a random forest surrogate-assisted adaptive multi-objective particle swarm optimization (RFMOPSO). In this algorithm, random forests are used to speed up the algorithm's search; new adaptive rules are used to update the particle states. The experiments show that the algorithm has some applications for combinatorial optimization problems. On the one hand, Random Forest needs to build multiple decision trees with longer training time, which is not as simple as RBFN to build the model. On the other hand, random forest has high prediction accuracy and can handle data well under adaptive conditions. Ji [51] proposed a dual-surrogate-assisted cooperative particle swarm optimization algorithm (DSCPSO-EMM) and searched for several optimal solutions with significantly reduced individual evolutionary costs. To verify the effect of the surrogate model on the performance of the DSCPSO-EMM, two variants, DSCPSO-EMM-LS and DSCPSO-EMM-BP, are proposed in this literature. The former surrogate models all take the form of RBFN-LS, while the latter uses a single-layer surrogate-assisted model combining RBFN-BP. The algorithm has significant advantages over other algorithms of the same type in solving expensive optimisation problems, but is structurally complex using multiple surrogate models.

Based on the above considerations, it is good to use surrogate assisted techniques to solve the computationally expensive problem. Therefore, in this paper, we propose A surrogate-assisted sine Phasmatodea population evolution algorithm (SASPPE) to solve the expensive optimization problem. We evaluate the proposed SASPPE algorithm on multiple dimensions of 7 benchmark functions as well as a real-world optimization problem. Comparative results show that the proposed algorithm SASPPE is effective. The main work of this paper is as follows.

1. This article proposes a new algorithm for solving the three-dimensional coverage problem of wireless sensors. At the same time, to increase network coverage, this article adds a mobile mechanism to sensor nodes.
2. SASPPE adopts an alternation mechanism. LSAMs and GSAMs are used alternately, which effectively balances the computational cost and the algorithm's optimization seeking ability.
3. The use of hierarchical clustering method to divide the solid LSAMs ensures the balance of building LSAM data. In LSAMs, the use of the Sine-PPE algorithm to search the solution space avoids the algorithm from falling into local optimality to a certain extent.



(a) 3D coverage example



(b) The simulated 3D terrain

Fig. 1 3D coverage schematic

The rest of this paper is structured as follows, section "Related work" introduces the relevant components of this paper, section "The proposed algorithm" mainly describes the proposed algorithm SASPPE, section "Experiment" presents the experimental results of the algorithm, section "3D coverage of removable nodes" is the application of the algorithm to the 3D coverage of movable nodes, and section "Conclusion and future work" is the summary and outlook.

Related work

Wireless sensor networks

Sensor nodes are the cornerstone of WSNs, and their sensing models significantly impact the detection and coverage capabilities of the network [7, 8]. Currently, the main sensing models of sensor nodes are binary and probabilistic sensing models. The binary sensing model is an ideal sensing model that ignores the attenuation of the signal during transmission, i.e., the ability to transmit the signal is constant [49]. The probabilistic sensing model takes into account the fading of the nodes during the message of the signal, i.e., the ability to transmit the signal decreases with increasing distance. This paper uses a binary sensing model, 0–1 sensing model, or Boolean sensing model. It is covered as shown in Fig. 1a, and all areas within this sphere can be detected. Assuming that the point $P(x_p,y_p,z_p)$ does not fall exactly within the sensing range of a sensor S_i, the coverage value C of the point P is recorded as 0, and vice versa as 1, as shown in Eq. (1).

Where $d(P, S_i)$ denotes the Euclidean distance between point P and sensor node S_i, and r denotes the sensing radius of sensor S_i. The red line in Fig. 1a is r, and the green line is d. To ensure maximum coverage, the nodes are movable for coverage in this paper. The energy of each sensor node is limited and cannot be moved indefinitely. It is assumed that each node is full of energy, gradually consumed as the nodes move. At the time when some sensor nodes run out of electrical power, mobile nodes can be used to fill them and ensure the connectivity of the network. In this paper, the nodes move a constant distance each time, i.e., the energy consumed is also constant.

$$C(P, S_i) = \begin{cases} 1, & \text{if } d(P, S_i) \le r \\ 0, & \text{otherwise} \end{cases}, \tag{1}$$

Radial basis function networks

RBFN, one of the methods for constructing surrogate-assisted models, is often used to compute more costly optimization problems [40, 46]. RBF neural network is a three-layer neural network. The transformations from the input layer to the hidden layer are nonlinear and are connected by radial basis functions. The transformation from the hidden layer to the output layer is linear and is connected by weights. The basic principle of RBFN is to divide the input space into N non-overlapping subspaces, each described by an independent RBF. The value of the RBF function depends only on the distance from the origin, which is generally adopted as the Euclidean distance [45]. The RBF method combines a series of exact interpolation methods, i.e., the surface must be sampled through each measured value. The interpolation function based on the radial basis function is shown in Eq. (2).

$$\check{y} = \sum_{i=1}^{N} \lambda_i \phi(\mathbf{x} - \mathbf{x_i}) + p(x), \tag{2}$$

where N is the number of RBF established and the number of nodes in the hidden layer. $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_N]^T$ is the correlation weight. $\phi(\cdot)$ is the kernel function, and $\|\cdot\|$ is the Euclidean distance. There are various activation functions for radial basis functions, such as Gaussian, polynomial, linear, etc. The formula of the Gaussian function is shown in Eq. (3), and the formula of the cubic surface function is shown in Eq. (4).

This paper uses the least squares method to solve for the unknown parameters. p is either a polynomial or a constant. Suppose the number of polynomials of p is m, then $\mathbf{b} = [b_1, b_2, b_3, \ldots, b_m]^T$, $\Phi_{ij} = \phi(x_i - x_j), (i = 1, 2, 3, \ldots, N), (j = 1, 2, 3, \ldots, N)$, $\mathbf{P}_{ij} = p_j(x_i), (i = 1, 2, \ldots, N), (j = 1, 2, \ldots, m)$, $\mathbf{b}$ and $\boldsymbol{\lambda}$ are solved in the way shown in Eq. (5).

$$\phi(r) = exp\left(-\frac{r^2}{2\sigma^2}\right), \tag{3}$$

$$\phi(r) = r^3, \tag{4}$$

$$\begin{bmatrix} \Phi & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \tag{5}$$

PPE algorithm

PPE differs from other algorithms in that each solution represents a population and has two properties: number and growth rate [22–24]. The solution is generated randomly at the beginning of the evolutionary stage of the population. The PPE algorithm selects the k nearest optimal solutions to avoid losing solutions during evolution to guide the subsequent population movement. The k is determined by the number of solutions N. The population position update formula is Eq. (6).

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{ev}, \tag{6}$$

where $\mathbf{ev}$ represents the evolutionary trend of the population, three types of updates are available for ev affected by path dependence, nearest optimum, and competition, and the update formula is shown in Eq. (7). Where $\mathbf{m}$ denotes the population mutation; $f(\mathbf{x})$ denotes the fitness value of the current population i; $\mathbf{A}$ represents the nearest optimal solution; $\mathbf{B}$ means an n-dimensional random variable generated from a normal distribution, and st is determined by the upper and lower bounds of the search space.

$$\mathbf{ev}^{t+1} = \begin{cases} (1 - \mathbf{p}^{t+1})\mathbf{A} + \mathbf{p}^{t+1}(\mathbf{ev}^t + \mathbf{m}), & \text{if path dependence} \\ rand \cdot \mathbf{A} + st \cdot \mathbf{B}, & \text{if nearest optimum} \\ \mathbf{ev}^{t+1} + \frac{f(\mathbf{x}_j) - f(\mathbf{x}_i)}{f(\mathbf{x}_j)}(\mathbf{x}_j - \mathbf{x}_i), & \text{if competition} \end{cases}, \tag{7}$$

The update of population size is shown in Eq. (8), and a denotes the population growth rate, which is set as a constant 1.1 in this paper. Population competition occurs when two populations are close enough to each other. The competition formula is shown in Eq. (9).

$$\mathbf{p}^{t+1} = a^{t+1}\mathbf{p}^t(1 - \mathbf{p}^t), \tag{8}$$

$$\mathbf{p}_i = \mathbf{p}_i + a_i \mathbf{p}_i (1 - \mathbf{p}_i - \frac{f(\mathbf{x}_j)}{f(\mathbf{x}_i)}\mathbf{p}_j),\ if\ d(\mathbf{x}_i, \mathbf{x}_j) < G, \tag{9}$$

The pseudo-code of the PPE is Algorithm 1.

Algorithm 1 PPE algorithm

```
Require: The dimension $D$, the upper bound $UB$ and the lower bound $LB$ of the search space, the number of particle groups $Np$.
1: The value of $k$ is calculated from $Np$.
2: Initialize particle position $\mathbf{x}$, evolutionary trend $\mathbf{ev}$.
3: while The termination conditions are met do
4:     $\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{ev^t}$
5:     for i = 1: N do
6:         if $f(\mathbf{x}^{t+1}) \le f(\mathbf{x}^t)$ then
7:             $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t$
8:             $\mathbf{p}_i^{t+1} = a_i^{t+1}\mathbf{p}_i^t(1 - \mathbf{p}_i^t)$
9:             $\mathbf{ev}_i^{t+1} = (1 - \mathbf{p}_i^{t+1})\mathbf{A} + \mathbf{p}_i^{t+1}(\mathbf{ev}_i^t + \mathbf{m})$
10:        else
11:            if $rand \le \mathbf{p}_i^{t+1}$ then
12:                $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t$
13:                $\mathbf{p}_i^{t+1} = a_i^{t+1}\mathbf{p}_i^t(1 - \mathbf{p}_i^t)$
14:            end if
15:            $\mathbf{ev}_i^{t+1} = rand \cdot \mathbf{A} + st \cdot \mathbf{B}$
16:        end if
17:        if $dist(\mathbf{x}_i, \mathbf{x}_j) \le G$ then
18:            $\mathbf{p}_i = \mathbf{p}_i + a_i\mathbf{p}_i(1 - \mathbf{p}_i - \frac{f(\mathbf{x}_j)}{f(\mathbf{x}_i)}\mathbf{p}_j)$
19:            $\mathbf{ev}^{t+1} = \mathbf{ev}^{t+1} + \frac{f(\mathbf{x}_j) - f(\mathbf{x}_i)}{f(\mathbf{x}_j)}(\mathbf{x}_j - \mathbf{x}_i)$
20:        end if
21:    end for
22: end while
Ensure: Output: The optimal value $\mathbf{G}$.
```

Chaotic model

In the optimization field, chaotic mappings can be used instead of pseudo-random number generators that produce random numbers in the range of 0–1 [52, 53]. The random numbers generated by chaotic mappings have the characteristic of being highly dependent on the initial values, i.e., two similar initial values will yield completely different sequences of random numbers [36]. This is also the nature of chaotic mappings, and such property will affect the generation of optimal solutions of the algorithm. When the algorithm generates multiple solutions, searching for the global optimal solution is more accessible. Chaotic mapping can be used not only in the initialization phase of the population but also in the selection, crossover, and mutation operations on the population. This operation also affects the whole evolution process of the algorithm, getting better results than the original algorithm. Chaotic mappings commonly used in meta-heuristic algorithms are logistic mapping, piecewise mapping, singer mapping, sine mapping, etc. The sine mapping is single-peaked and has a value domain of [− 1, 1]. Adding the sine mapping ensures that the population is adjusted in a small range, so the sine mapping is chosen in this paper. The original formula of sine mapping is shown in Eq. (10), representing the chaotic parameters. In this paper, the value of β is set to 4 and Z_0 to 0.01.

$$Z_{t+1} = \frac{4}{\beta} sin(\pi Z_t), 0 < \beta \leq 4. \tag{10}$$

Hierarchical clustering

Clustering is commonly used in the initial stage of data exploration or data mining and is often used for data preprocessing. Hierarchical clustering can be divided into two opposite approaches: agglomerative hierarchical clustering and divisive hierarchical clustering [54, 55]. Agglomerative clustering treats each object as a cluster and merges these clusters to form a larger cluster based on conditions. Divisive clustering treats all objects as one cluster and then gradually divides them into smaller and smaller clusters. The advantage of hierarchical clustering is that it is easy to implement similarity measures or distance measurements to draw a tree diagram. Another advantage is that the number of clusters can be specified in the clustering process to achieve a more desirable clustering effect.

This paper chooses to use agglomerative clustering that requires merging clusters. Generally speaking, the two clusters with the smallest distance are effortlessly merged. The definition methods of the distance between clusters determine different clustering results. The most commonly used distance measurement methods include complete-linkage, single-linkage, and ward-linkage [56]. The single-linkage may be combined due to the proximity of extreme data points. The complete-linkage may not be able to be combined due to the distance between their extreme values. The single-linkage may be grouped due to the proximity of different data classes; complete-linkage may not be grouped due to the distance of similar data. Therefore, ward-linkage is chosen in this paper, and the calculation formula is shown in Eq. (11).

$$d(r, s) = \sqrt{\frac{2n_r n_s}{(n_r + n_s)}} \|\bar{x}_r - \bar{x}_s\|_2, \tag{11}$$

where $\| \cdot \|_2$ denotes the Euclidean distance, $\bar{x}_r$ denotes the center of mass of r clusters, and n_r denotes the number of elements of r clusters. The number of clusters set in this paper is 3.

The proposed algorithm

Like most meta-heuristics, the PPE algorithm has the disadvantages of slow convergence and low accuracy. The PPE algorithm works well in 30 dimensions, and its effectiveness in higher dimensions is unknown. However, the efficacy of other meta-heuristics decreases exponentially as the dimensionality increases. The PPE algorithm has the same deficiency. Improve the problem, three solutions are proposed in this paper to solve the problem collaboratively. At first, to solve the problem of high computational cost, this paper adds a surrogate-assisted model to reduce the computational cost of the algorithm. The surrogate models include the global surrogate-assisted model and multiple local surrogate-assisted models. In this paper, the global surrogate-assisted model is abbreviated as GSAM, and the local surrogate-assisted model is abbreviated as LSAM. Second, multiple LSAMs are divided using agglomerative hierarchical clustering to speed up the algorithm's operations. Although adding the surrogate models improves the accuracy of the algorithm to some extent, it tends to make the algorithm premature. Finally, this paper adds a chaotic mapping mechanism to solve the algorithm's premature problem. In summary, a surrogate-assisted sine PPE algorithm is proposed in this paper.

Local surrogate-assisted model

LSAMs are used before GSAM to quickly find the current optimal solution and avoid premature algorithm maturity. Before constructing the LSAM, the data samples are preprocessed to improve the quality of the model. This paper uses agglomerative hierarchical clustering to cluster the data samples and the "ward" method to calculate the distance between two clusters. This method is simple to define the distance without specifying the center of the cluster in advance and generates non-spherical clusters. Multiple LSAMs are generated using agglomerative hierarchical clustering to ensure that the generated LSAMs are all in the best condition. The LSAM uses cubic surface functions as kernel functions and p takes the form of a polynomial. The local search space is explored using the Sine-PPE algorithm. Sine mapping is used throughout the LSAM and influences the generation of optimal solutions, improves the development capability of LSAM, and prevents the algorithm from falling into local optimum. After each LSAM generates the optimal particles, the excellent particles around the optimal particles are found using sine mapping and evaluated using the true evaluation function.

The specific process of LSAM is as follows:

- It is divided into three clusters by hierarchical clustering, and three LSAMs are constructed using the sample data in the clusters.
- The optimal solution is found using Sine-PPE, and the obtained optimal solution is added to the database (DB). Subsequently, the optimal solution and sine mapping are used to find the excellent particles around the optimal solution.
- The suitable particles are added to the DB, and the matrix DB is updated.

It is necessary to determine whether the end condition is reached each time the current optimal solution is added. If the state is reached, the whole algorithm is ended, and vice versa; the algorithm continues to run. The DB used is the same DB as the one used by the GSAM, and LSAM uses all the data in the DB.

The pseudo-code of the LSAM is Algorithm 3.

Sine-PPE algorithm

As shown in Song's literature [22], the overall performance of PPE on 30 functions of CEC 2014 is better than GWO, GSA, SA, SLPSO, BOA, and GA algorithms. To avoid premature algorithm, sine chaotic mapping is added to the PPE. Chaotic mappings can have an impact on the algorithm during the iterative process, and this impact will affect the subsequent iterative process. Adding chaotic mapping improves the possibility of the algorithm jumping out of the local optimum and avoids premature maturity. The evolutionary part of the PPE algorithm is modified using sine chaos mapping. The revised sine mapping is shown in Eq. (12).

$$Z_t = sin(\pi Z_{t-1}) + cos(\pi Z_{t-1}), \tag{12}$$

Considering the evolutionary process, the population will be influenced not only by the optimal solution and path dependence but also by the parents. In this paper, $x^{t'}$ is introduced as the influence of the previous generation on the current iteration, and the formula for calculating $x^{t'}$ is shown in Eq. (13).

$$x^{t'} = (a_t Z_t + 4cos(Z_t))(x^t - x^{t-1}), \tag{13}$$

Because the evolutionary trend takes into account the effects of path dependence, convergence to the nearest optimum, and competition, the variable S_t is introduced in this paper to balance the relationship between the evolutionary trend ev and $x^{t'}$. S_t is calculated as shown in Eq. (14). To ensure that the solution generated by the algorithm is in the search space, the value of S_t must be less than 1. In this paper, the solution overflow is prevented by using modulo operation, i.e., a remainder of S_t to generate S_t^*. The population evolution formula of the improved PPE algorithm is shown in Eq. (15).

$$S_t = sin(u \cdot \pi \cdot rand1 \cdot S_{t-1}) \\ + sin(u \cdot \pi \cdot rand2 \cdot S_{t-1}), \tag{14}$$

$$x^{t+1} = x^t + S_t \cdot ev + (1 - S_t^*) x^{t'}, \tag{15}$$

The pseudo-code of the Sine-PPE is Algorithm 2.

Algorithm 2 Sine-PPE algorithm

```
Require: The dimension $D$, the upper bound $UB$ and the lower bound
    $LB$ of the search space, the number of particle groups $N$.
1: Set the value of $k$.
2: Initialize particle position $\mathbf{x}$, evolutionary trend $\mathbf{ev}$, the variable $S_t$
   and $Z_0$.
3: while The termination conditions are met do
4:     $Z_t = sin(\pi Z_{t-1}) + cos(\pi Z_{t-1})$
5:     $\mathbf{x}^{t'} = (a_i^t Z_t + 4cos(Z_t))(\mathbf{x}^t - \mathbf{x}^{t-1})$
6:     $x^{t+1} = x^t + S_t \cdot ev + (1 - S_t^*) x^{t'}$
7:     Execute the "for" loop structure of Algorithm 1, which is lines
       5–21.
8: end while
Ensure: The current optimal positions $\mathbf{G}$ and its fitness value $f(\mathbf{G})$
```

Global surrogate-assisted model

The primary use of the GSAM is to ensure the stability of the model. The meta-heuristic algorithm is unstable and will have different results in different computation instances in the same problem, sometimes performing excellent and sometimes poorly. In case of poor performance, the GSAM is used to adjust the search capability of the SASPPE algorithm to ensure that the searched solutions do not deviate from the normal values. The RBF function is constructed using "newbe" in Matlab, with the kernel function as a Gaussian and p as a constant 0. But building GSAM needs to solve the following two problems. One is that GSAM requires a relatively large data sample. Another is that GSAM is to select a point on the whole search space, and the accuracy of the data cannot be guaranteed. To solve the above problems, the following

Algorithm 3 Local surrogate-assisted sine-PPE

```
Require: database $DB$, maximum number of real evaluations $MRE$, and $DB$ size $Nd$.
1: Use hierarchical clustering to divide the sample data to form $Nc$ sub-archives $DB_i$.
2: for i = 1: Nc do
3:     Constructing $FUN1$ instead of true fitness function $f(x)$ to evaluate populations.
4:     $@FUN1 = RBFN(DB_i, cubic)$;
5:     $\mathbf{G}_{L1}$ = Algorithm 2.
6:     The $f(\mathbf{G}_{L1})$ is obtained by evaluating $\mathbf{G}_{L1}$ using the true fitness function $f(x)$.
7:     Select two optimal particles $\mathbf{X_1}$ and $\mathbf{X_2}$ from the $DB_i$.
8:     Two variables $w_1$ and $w_2$ are randomly selected from $S_t$.
9:     $\mathbf{G}_{L2} = \mathbf{G}_{L1} - w_1(\mathbf{G}_{L1} - \mathbf{X}_1) - w_2(\mathbf{G}_{L1} - \mathbf{X}_2)$
10:    The $f(\mathbf{G}_{L2})$ is obtained by evaluating $\mathbf{G}_{L2}$ using the true fitness function $f(x)$.
11:    $f(\mathbf{G}_L) = min(f(\mathbf{G}_{L1}), f(\mathbf{G}_{L2}))$
12:    if $f(\mathbf{G}_L) \leq f_{max}(DB_i)$ then
13:        $max(DB_i) = \mathbf{G}_L$;
14:        $MRE = MRE + 2$;
15:    end if
16:    $DB = sort(DB \cup DB_i, 1)$;
17:    $DB = DB(1, Nd, :)$;
18: end for
Ensure: The current optimal positions $\mathbf{G}_{L1}, \mathbf{G}_{L2}$ and their fitness value $f(\mathbf{G}_{L1}), f(\mathbf{G}_{L2})$.
```

solutions are given in this paper. The PPE algorithm has the number and growth property, which is more advantageous in exploring the search space. In addition, the time complexity of the algorithm could be higher, which is suitable for the calculation of relatively large data samples. Therefore, the search space is explored in GSAM using the PPE algorithm. Besides, multiple LSAMs are constructed in this paper to develop the current search space and exploit the optimal values in the whole search space as much as possible.

To reduce the expenses and better find the optimal solution, the GSAM and the LSAM use the same database DB. The size of the database is positively correlated with the time to train the model; the more significant the database, the longer the time. A reasonable database size must be set for reasonable training model time. The complexity of the dimensionality also needs to be considered when designing the database size. Because as the dimensionality increases, more samples are required to ensure the accuracy of the model, then the time to train the model must increase. The particle size of the algorithm is assumed to be Np. After several experiments, the database size setting of Np and $2 \times Np$ is the best solution. When the search space dimension is 30 and 50, the size of DB is set to Np. At this time, GSAM uses all the data in the DB. When the dimension of the search space is 100, the size of DB is set to $2 \times Np$. To reduce the time of building the model, GSAM uses two-thirds of the data in the DB. The initial values of the DB are generated using the Latin hypercube sampling (LHS). Once the data is updated, the DB must be sorted once to delete the inferior particles and ensure that the number of particles in the DB is Np and $2 \times Np$.

The pseudo-code of the GSAM is Algorithm 4.

Algorithm 4 Global surrogate-assisted PPE

```
Require: database $DB$, maximum number of real evaluations $MRE$, data size $Ng$ required for GSAM, and $DB$ size $Nd$.
1: $DB_g = DB(1, Ng, :)$
2: Constructing $FUN2$ using all the data in the $DB_g$ instead of true fitness function f(x) to evaluate populations.
3: $@FUN2 = RBFN(DB_g, Gaussian)$;
4: $\mathbf{G}$ = Algorithm 1.
5: The $f(\mathbf{G})$ is obtained by evaluating $\mathbf{G}$ using the true fitness function f(x).
6: if $f(\mathbf{G}) \leq f_{max}(DB)$ then
7:     $max(DB) = \mathbf{G}$;
8:     MRE++;
9: end if
10: $DB = sort(DB \cup DB_g, 1)$;
11: $DB = DB(1, Nd, :)$;
Ensure: The current optimal position $\mathbf{G}$ and its fitness value $f(\mathbf{G})$
```

The entire algorithm (SASPPE)

The SASPPE algorithm integrates strategies such as surrogate, chaotic mapping, and hierarchical clustering to improve the overall performance of the PPE algorithm. Adding a surrogate mechanism saves time and cost for the algorithm. Adding chaotic mapping enhances the randomness and variability of the algorithm and avoids premature maturation of the algorithm. Adding hierarchical clustering facilitates classifying data and building a more reasonable surrogate model. Figure 2 shows the overall framework of the algorithm; the purple part is LSAM, and the green part is GSAM. The execution steps of the SASPPE algorithm are as follows. Where $f(x)$ represents the true fitness function, function $FUN1$ refers to the LSAM function constructed using RBF, and $FUN2$ denotes the GSAM function made using RBFN.

1. nitialize the data using LHS, then evaluate the data using $f(x)$, and select the appropriate amount of data stored in the DB.
2. LSAMs are first constructed in the following way. The selected data are divided into multiple sub-archived DB_i using hierarchical clustering, and LSAMs are constructed for each DB_i. In each LSAM, the function $FUN1$ is used instead of $f(x)$.
3. The Sine-PPE is executed to find the optimal particle in the current space, and the end condition of the Sine-PPE algorithm is executed 200 times, or the optimal value is constant for 20 consecutive times. The evaluation function of the Sine-PPE algorithm is the $FUN1$, not the $f(x)$.

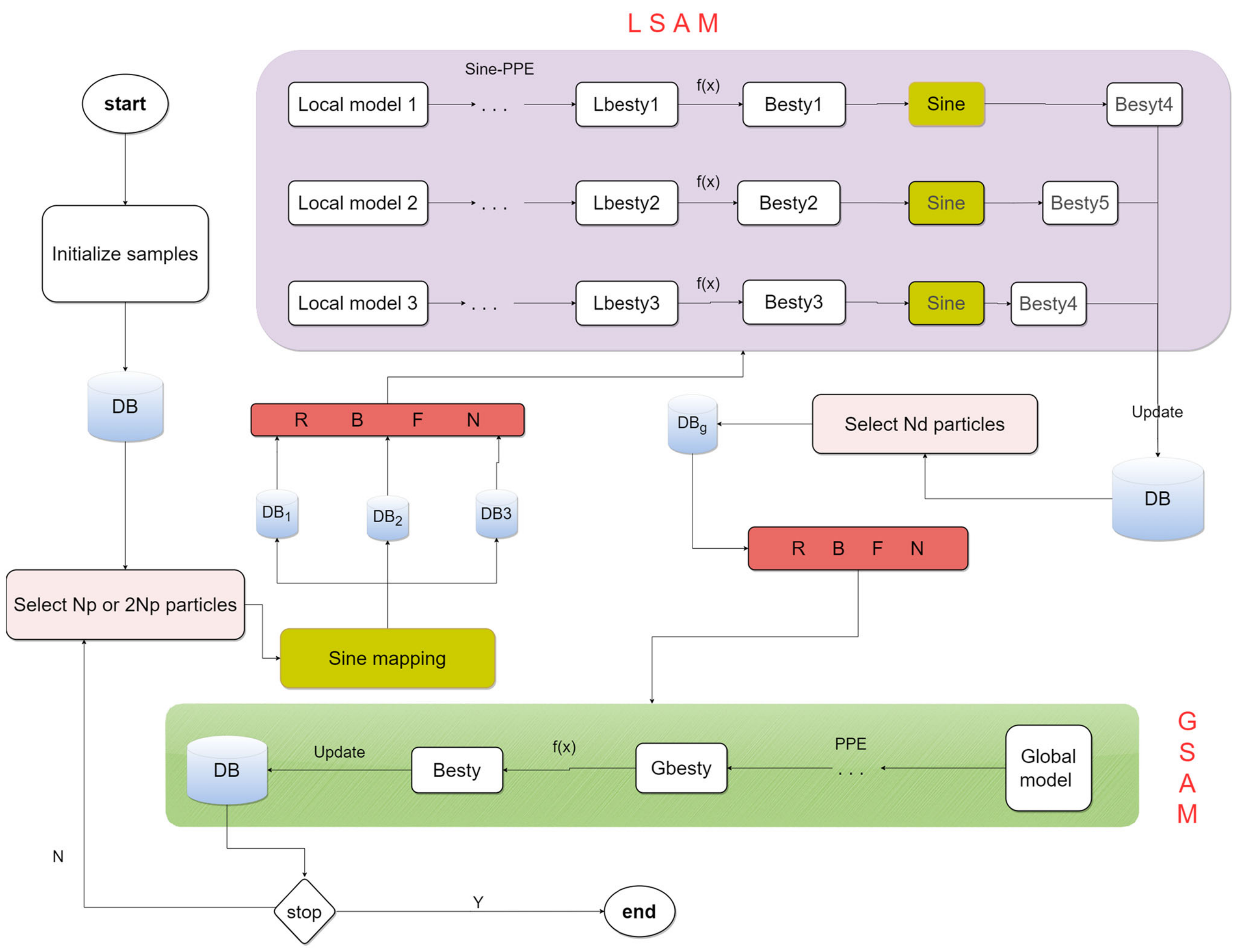


Fig. 2 Flowchart of SASPPE

4. Evaluate the particle using the $f(x)$. If the particle is superior to the particle in the DB, replace the worst particle in the DB. Subsequently, update the DB. Conversely, leave the status quo unchanged and perform the next step.
5. Sine mapping is used for the optimal particles selected by Sine-PPE to discover the best particles around the optimal particles, then 4 is executed.
6. Repeat 3–5 to ensure that the LSAM is executed once for each build.
7. At this point, the GSAM build phase is entered. Enough particles are selected from DB to form DB_g, and a GSAM is constructed using the particles in DB_g. The particles are evaluated using the $FUN2$ instead of the $f(x)$. Subsequently, the optimal value is searched over the entire search space using the PPE algorithm. The PPE uses $FUN2$ to evaluate the population, and the PPE algorithm ends with the same conditions as the Sine-PPE algorithm.
8. The evaluation process is the same as LSAM; refer to 4.
9. If the end condition is reached, the algorithm is finished; if not, the algorithm returns to 2 and executes the SASPPE.

The pseudo-code of the SASPPE is Algorithm 5.

Algorithm 5 Surrogate-assisted sine-PPE algorithm

```
Require: The function $f(x)$, the dimension $D$ and the upper bounds
    $UB$ and lower bounds $LB$ of the search space. Number of particles
    $Np$, number of hierarchical clusters $K$, and number of true evalua-
    tions $MRE_{max}$.
1: Set the $DB$ size $Nd$.
2: The datas are generated using the Latin hypercube sampling and
    evaluated using the function $f(x)$.
3: Select $Nd$ data to put into the $DB$.
4: $MRE = MRE + Nd$;
5: while Meet the termination conditions do
6:     Algorithm 3;
7:     Algorithm 4;
8: end while
Ensure: The current optimal position $\mathbf{G}$ and its fitness value $f(\mathbf{G})$
```

Table 2 Test functions and their parameters

Name	Type of function	Global optimal
F(1)/Ackley	Multimodal	0
F(2)/Griewank	Multimodal with narrow valley	0
F(3)/Rosenbrock		0
F(4)/Ellipsoid	Unimodal	0
F(5)/(Shifted rastrigin function) [57]	Very complicated multimodal	− 330
F(6)/(Shifted rotated rastrigin) [57]		− 330
F(7)/(Rotated hybrid composition function) [57]		− 120

Experiment

In order to verify the effectiveness of the proposed algorithm SASPPE algorithm, this paper chooses to test the algorithm with 7 test functions on 30D, 50D, and 100D.

Parameter description

Table 2 shows the test functions used for this experiment and their parameters. F(5), F(6) and F(7) in Table 2 are F(9), F(10) and F(16) in CEC2005 [57], respectively. The algorithms that participated in the comparison were SAPPE, PPE, RBFPSO, and SACOSO algorithms. All algorithms demonstrate their performance on 30D, 50D, and 100D, respectively. For a fair comparison, all algorithms were run 10 times independently on MATLAB 2020b, the number of particles for all algorithms was set to 100, and the number of real evaluations was 1000, which was also the condition for the end of the whole algorithm. SAPPE, SASPPE and PPE have the same parameters, please refer to PPE algorithm for details of the parameters. The sample volumes generated by SASPPE,SAPPE, RBFPSO and SACOSO using LHS are set to 100 (for the 30D and 50D states) and 200 (for the 100D state). The maximum iterations of the PPE and Sine-PPE algorithms used in SAPPE and SASPPE are 200. However, when the optimal value changes unchanged for 20 iterations, the current search is ended, and the next search is performed. The condition that the optimal value remains unchanged is that the difference between the current and last evaluated values is less than 1.00E−6.

Tables 3, 4, and 5 show the final results of all algorithms on 30D, 50D, and 100D, respectively, and Figs. 3, 4, and 5 show the experimental comparison plots of all algorithms on 30D, 50D, and 100D. Where "Best" represents the best value among all data, "Std." represents the standard deviation of the obtained data, and "Mean" represents the average of 10 runs. Bolded font indicates that the value outperforms equivalent types of data. Only the mean values are bolded in this paper to enhance the experimental persuasiveness. The "p" means the result of the Wilcoxon rank test at a 5% confidence level, "+" indicates a significant difference with the SASPPE algorithm, and "-" indicates no statistically significant advantage with SASPPE.

Results

30D and 50D experiment results

From Tables 3 and 4, it is clear that SASPPE can achieve the best on all function, whether it is the contingent "Best" or the persuasive "Mean." As shown in Table 3, the optimal value of the SASPPE algorithm is closest to that of SAPPE on F(1), F(3) and F(4) but closest to SACOSO and RBFPSO on the other functions. The optimal value of SASPPE is close to the ideal optimal value of 0 on F(2) and F(4), but the largest difference is with the ideal value of − 120 on F(7). On F(3), F(5), and F(7), the optimal values of SASPPE are all 100 or even 200 smaller than the optimal values obtained from SAPPE. Compared with "Std.", the SASPPE algorithm is minimum on F(1), F(2), F(3), F(4) and F(7); SACOSO is minimum on F(5); SAPPE is minimum on F(6). It can be seen that the SASPPE algorithm with the added surrogate model has the best stability. As shown in Table 3, the SASPPE algorithm is not significantly different from with RBFPSO only on F(5). As shown in Table 4, the SASPPE algorithm does not differ from with RBFPSO and SACOSO only on F(7). On 50D, the SASPPE mean value is similar to 30D, and it still cannot win on F(7). The value obtained by subtracting the mean value of 50D from the mean value of 30D is defined as the "mean difference". The minimum "mean difference" is SASPPE on all function. On F(1), the mean value of the PPE algorithm on 50D is superior to that on 30D, and the SASPPE algorithm has the largest "mean difference". On f (2), only the SAPPE algorithm's "mean difference" less than 1, all other algorithms are greater than 1 On F(3) and F(4), the "mean difference" is smaller for SASPPE and is larger for the other algorithms, even a hundred times that of SASPPE. The "mean difference" of all functions on F(5) and F(6) is greater than 100. From the comparison of SASPPE,SAPPE and SACOSO and PPE, RBFPSO, it can be seen that the addition of sur-

Table 3 Table of experimental results for SASPPE, SAPPE, PPE, RBFPSO and SACOSO on F(1)–F(7) (30D)

F(x)	Result	Algorithm				
		SASPPE	SAPPE	SACOSO	RBFPSO	PPE
F(1)	Mean	**3.495526078**	11.37248814	11.59342192	5.688499535	15.11612832
	Best	2.564183526	9.565912046	9.720089526	4.464412136	14.8170328
	Std	0.66145723	1.150136455	1.273984865	1.386538715	0.241443592
	p		+	+	+	+
F(2)	Mean	**0.190447349**	1.51649733	2.058770355	3.372291975	74.74007711
	Best	0.090759209	1.09465208	1.312402714	2.260646666	66.21825556
	Std	0.111155346	0.578627041	0.525284908	1.057300869	6.427753043
	p		+	+	+	+
F(3)	Mean	**28.77810542**	295.5495506	60.94190152	93.40133538	407.1258937
	Best	27.04809394	165.6078974	35.35286441	32.01262967	357.5274613
	Std	1.268688467	112.2856031	20.26145179	51.15019746	39.61284844
	p		+	+	+	+
F(4)	Mean	**0.232769301**	95.15189239	9.593160004	10.91116002	283.2476509
	Best	0.019503099	24.62844044	5.345372731	3.931230087	260.7739141
	Std	0.203343534	59.32376259	3.715983937	5.501803614	13.01531134
	p		+	+	+	+
F(5)	Mean	− **175.8540334**	− 37.9505387	− 23.79819711	− 145.3615498	38.55999165
	Best	− 275.2690174	− 118.7974381	− 50.39681654	− 212.0975493	24.53649519
	Std	76.09037241	36.95909468	28.047353	63.79585548	12.01125887
	p		+	+	−	+
F(6)	Mean	− **128.753451**	− 0.680229669	36.0226557	− 43.55979841	280.5706386
	Best	− 185.9044089	− 57.62162386	− 26.68250829	− 100.247818	210.3075556
	Std	44.20783831	31.313114	51.01355104	44.24715679	51.38921006
	p		+	+	+	+
F(7)	Mean	**517.6558656**	949.016479	732.7889183	537.5010761	1084.646434
	Best	412.4021637	776.6725279	650.4101946	409.2779187	818.3790769
	Std	79.49206449	129.6853859	55.61358267	88.28995299	119.2701856
	p		+	+	+	+

rogate model has a better ability to find the optimal; from the comparison of SASPPE and SACOSO and RBFPSO, it can be seen that the choice of surrogate model affects the stability of the algorithm, and the excellent surrogate model can reduce the influence of the dimensionality on the algorithm; from the comparison of SASPPE and SAPPE, it can be seen that the addition of the chaotic model and the reasonable clustering approach are conducive to the improvement of the performance of surrogate model.

As can be seen from Figs. 3 and 4, the fitness function gradually converges as the number of true evaluations increases. For a more intuitive demonstration of the convergence performance of the algorithm, the fitness function is taken logarithmically over F(1)–F(4). For the first 150 real evaluations on all test functions, the SAPPE and SASPPE algorithms are comparable in power, and their optimal values are similar. However, as the number of evaluations increases, the SASPPE algorithm gradually becomes more capable of exploitation, and the convergence speed is much faster than that of other algorithms. As can be seen from Fig. 3, the SASPPE algorithm is highly exploitable on F(1), F(2), F(3), F(4) and F(5). In particular, on F(1), the SASPPE algorithm has the strongest development ability and the fastest convergence speed, which is approximately linear descent. On F(3), SASPPE converges faster in the first 600 evaluations and tends to level off in the last 400 evaluations. However, the SASPPE algorithm has beaten the other algorithms to dominate at the 150 times evaluation. On F(6), the SASPPE and SAPPE algorithms converge at equivalent rates, and SASPPE, SAPPE and SACOSO start to level off after 300 times evaluation, and RBFPSO start to level off after 400 times evaluation. For the first 150 times On F(7), SASPPE and RBFPSO converge at the same rate and both algorithms have similar accuracy. Observing the minima of the seven

Table 4 Table of experimental results for SASPPE, SAPPE, PPE, RBFPSO and SACOSO on F(1)–F(7) (50D)

F(x)	Result	Algorithm				
		SASPPE	SAPPE	SACOSO	RBFPSO	PPE
F(1)	Mean	**8.136756987**	11.54759197	13.7211638	11.67724597	14.79949636
	Best	6.31072546	8.51439116	11.64830468	10.2865097	14.29194484
	Std	1.433421276	1.228845944	1.26849451	1.208584052	0.289575483
	p		+	+	+	+
F(2)	Mean	**0.908794101**	2.165303283	11.58771051	45.45729197	119.5079959
	Best	0.753096438	1.444559026	6.401010842	26.56627295	104.6702913
	Std	0.074526596	0.884423297	4.159164723	19.52757593	7.223740252
	p		+	+	+	+
F(3)	Mean	**51.12550687**	491.0191017	160.9698412	345.3451634	674.4381343
	Best	48.83888104	263.1311346	108.1424651	228.6849725	613.9892058
	Std	2.534372038	209.0487382	55.05975278	77.98535235	51.28071258
	p		+	+	+	+
F(4)	Mean	**0.937505614**	273.3087799	96.49818992	270.6689957	804.7577376
	Best	0.352205581	56.88392346	43.59496562	149.2887917	746.8562407
	Std	0.332916428	158.3955588	40.53713783	81.46979935	32.63861387
	p		+	+	−	+
F(5)	Mean	− **6.405052438**	159.9170002	297.9124034	78.46398169	390.8806543
	Best	− 168.0282182	118.1537865	248.2339313	4.614425016	350.8829146
	Std	107.8218163	33.11385837	54.80230389	58.48077352	33.24382459
	p		+	+	+	+
F(6)	Mean	**156.8954833**	312.1372693	518.081811	281.7386628	853.7998971
	Best	87.14846476	223.6152599	440.5308302	153.9154277	755.7346373
	Std	69.38952707	62.43818939	89.70486901	83.19332963	53.56049593
	p		+	+	+	+
F(7)	Mean	665.0302044	973.5573307	834.5379608	**566.4987852**	1222.955002
	Best	576.0996301	875.7383486	659.2756053	470.9609286	1062.753565
	Std	80.28213377	89.25147985	143.105016	48.14823989	93.01901489
	p		+	−	−	+

graphs in Fig. 3, it can be seen that the SASPPE algorithm wins with an absolute advantage on the F(1)–F(4) functions, and is similar to RBFPSO on the last three functions. It wins with a slight advantage on F(5) and F(6), and is similar to RBFPSO on F(7).It is tentatively concluded that the SASPPE algorithm has an advantage on single-peak, multiple-peak, and some of the complex functions.

As can be seen from Fig. 4, the convergence performance of the SASPPE algorithm on 50D remains strong. he SASPPE algorithm has much better convergence accuracy than the other algorithms on F(1), F(3), and F(4), but the convergence speed is not as fast as that of RBFPSO. As the number of evaluations increases, the SAPPE algorithm cannot escape the premature dilemma. In contrast, the SASPPE algorithm can break through the current optimal value and continue to converge until the end of the algorithm. On F(2) and F(5), the SASPPE and SAPPE algorithm converges at a similar rate and falls into a local optimum; however, after 400 evaluations, the SASPPE algorithm jumps out of the local optimum on F(2) and develops a new optimum. On F(6), the convergence speed of the SASPPE algorithm and SAPPE algorithm are similar, both tend to level off after 150 times, but the convergence speed of SASPPE is still stronger than that of SAPPE. On F(7), after 200 evaluations, RBFPSO is victorious, and if only the first 200 evaluations are considered, SASPPE can be victorious.

Overall, the SASPPE algorithm outperforms the other algorithms on 30D and 50D, and from the above analysis, it can be seen that the SASPPE algorithm will always win in one way or another on F(1)–F(6), and can be defeated by RBFPSO only on F(7). Combined with the graph and table analysis, the SASPPE algorithm has a robust search and exploitation ability to go beyond the local optimum to find better results. Comparing the algorithms of SASPPE, SAPPE, SACOSO and RBFPSO with added surrogate model and without added surrogate model, the algorithm with the

Table 5 Table of experimental results for SASPPE, PPE, SAPPE, RBFPSO and SACOSO on F(1)–F(7) (100D)

F(x)	Result	Algorithm				
		SASPPE	SAPPE	SACOSO	RBFPSO	PPE
F(1)	Mean	**9.442362995**	11.57347181	15.64910137	17.94458103	14.56762335
	Best	8.336888824	10.46181128	15.09038902	17.20787919	14.1472672
	Std	0.890122	0.79709703	0.575625483	0.46529122	0.242770341
	p		+	+	+	+
F(2)	Mean	**1.077061787**	8.513216977	90.36093453	548.829582	224.9467053
	Best	1.024010738	7.217047864	77.76201509	441.3434287	204.7090663
	Std	0.032511431	0.966156631	10.51487776	87.91391828	12.16736
	p		+	+	+	+
F(3)	Mean	**143.5768238**	800.3801819	739.2665662	3420.067419	1340.917828
	Best	110.901991	448.2791317	595.443892	2411.695801	1219.709836
	Std	36.88060975	287.956134	83.65571453	751.9179497	76.59563541
	p		+	+	+	+
F(4)	Mean	**30.29064781**	855.768873	1288.20173	6720.074642	3059.168558
	Best	16.86007898	152.9110936	899.5915251	4768.341461	2693.397759
	Std	9.339881888	457.5665209	370.0078168	1310.513775	185.667625
	p		+	+	+	+
F(5)	Mean	**749.1185524**	1076.389719	1091.282224	816.499191	1238.616687
	Best	552.2162366	914.6428098	993.9117639	632.0862516	1134.8732
	Std	84.8901753	87.0913887	58.19489004	94.95449825	50.96456403
	p		+	+	−	+
F(6)	Mean	**1110.363586**	1445.188782	1728.9304	1770.394451	1947.751646
	Best	998.7128332	1255.771899	1605.475128	1590.242232	1829.935291
	Std	92.03230146	106.3165667	83.17541528	127.9197986	76.3228178
	p		+	−	+	+
F(7)	Mean	**728.5847131**	780.9697883	849.3243529	877.973152	1329.061813
	Best	636.2621934	704.7497316	764.7152226	754.050824	1215.700809
	Std	48.29671322	47.16623423	73.39628473	106.0217367	70.20123402
	p		+	+	+	

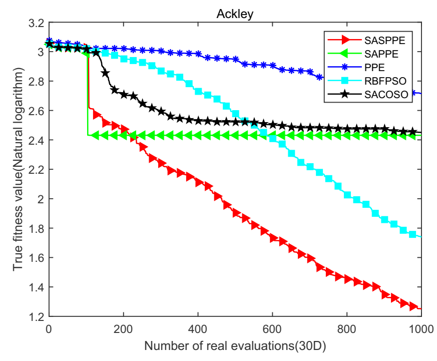
added surrogate model has more advantages. Compared with SASPPE and SAPPE, sine mapping and hierarchical clustering enhance the overall performance of the algorithm and solve the problem that adding surrogate models can easily fall into the local optimum.

100D experiment results

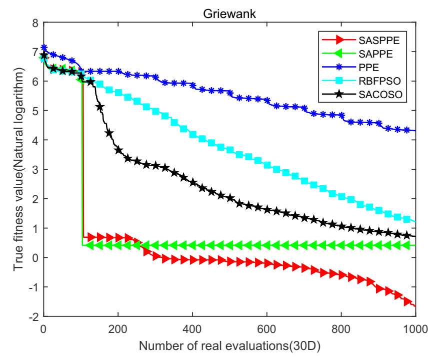
As can be seen from Table 5, SASPPE is not as dominant as 30D and 50D, especially on F(5) and F(6). Comparing the mean values in Table 5, SASPPE wins on the F(1)–F(4), F(6) and F(7) functions and is beaten by SAPPE on F(5)F(6). A horizontal comparison of “Best” shows that SASPPE wins on F(1)–F(5) and F(7), and SAPPE wins on F(6). Comparing the “p” results in the table, we can see that RBFPSO is not significantly different from SASPPE on F(5), SACOSO is not significantly different from SASPPE on F(6), and all other algorithms are significantly different from SASPPE. On F(1), the mean result of the PPE algorithm is inversely proportional to the dimensionality; the higher the dimensionality, the better the result, but it is not as good as SASPPE and SAPPE. Compared with “Std.”, SASPPE is the smallest in F(2), F(3), and F(4), and SAPPE gains an advantage in F(7); PPE is the smallest in F(1) and F(5); SACOSO is the smallest in F(6). It can be seen that the stability of the SASPPE algorithm does not decrease as the dimensionality increases.

As shown in Fig. 5, SASPPE continues to converge on F(1), F(5), and F(6), and continues to converge after several smooth evaluations on F(2), F(3), F(4), and F(7). On F(1), SAPPE achieves the optimal value in the early stage of the algorithm. As the evaluation proceeds, the SASPPE algorithm gradually catches up with the SAPPE algorithm and achieves the final victory. On F(2), F(3), and F(4), the SASPPE and SAPPE algorithms converge to the current optimal value in the early stages of the algorithm. As the algorithm proceeds, the SASPPE algorithm continues to converge

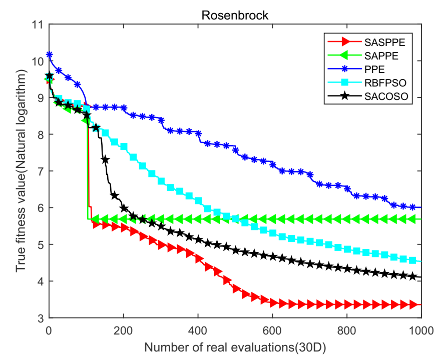
Fig. 3 Comparison of SASPPE, SAPPE, PPE, RBFPSO and SACOSO algorithms for F(1)–F(7) (30D)



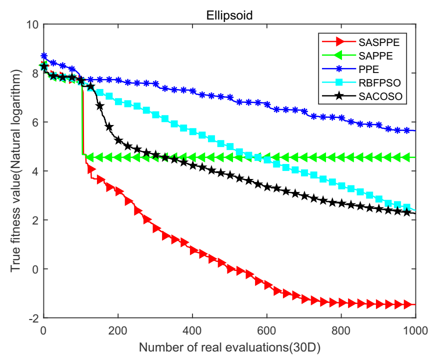
(a) F(1)



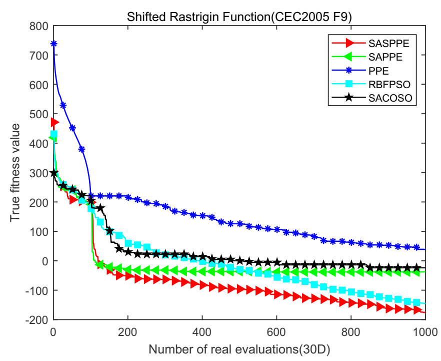
(b) F(2)



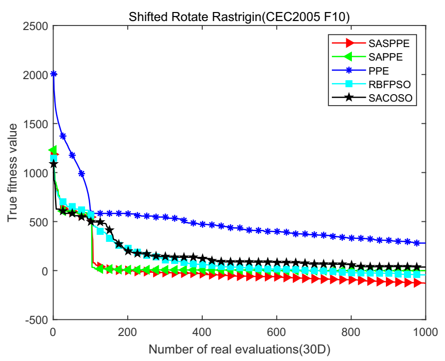
(c) F(3)



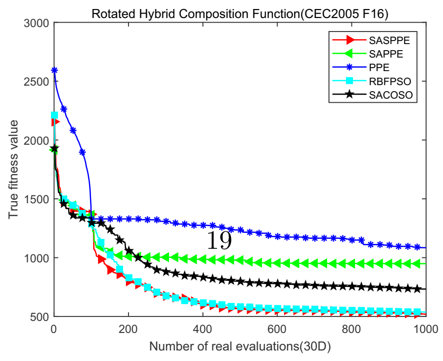
(d) F(4)



(e) F(5)

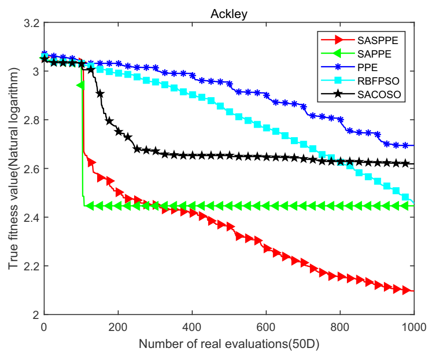


(f) F(6)

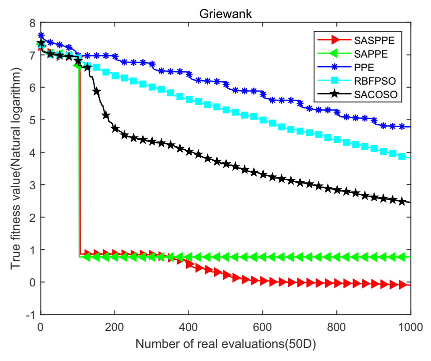


(g) F(7)

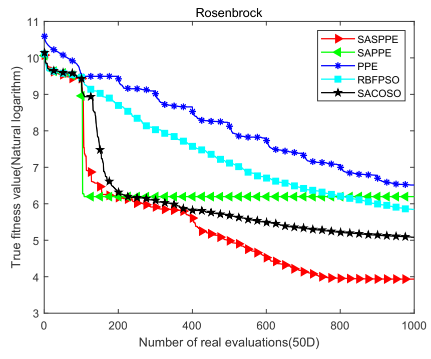
Fig. 4 Comparison of SASPPE, SAPPE, PPE, RBFPSO and SACOSO algorithms for F(1)–F(7) (50D)



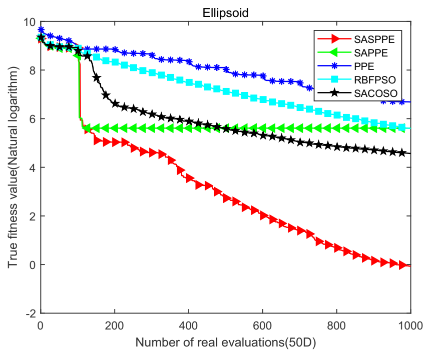
(a) F(1)



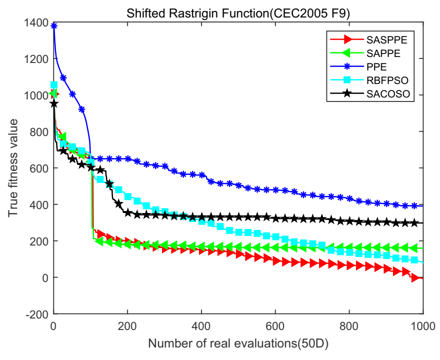
(b) F(2)



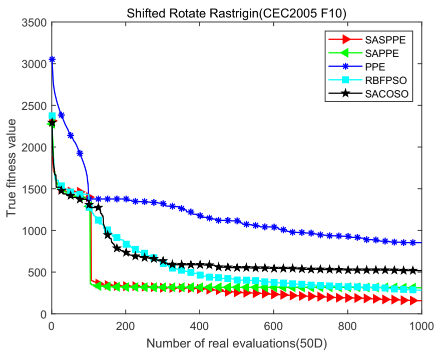
(c) F(3)



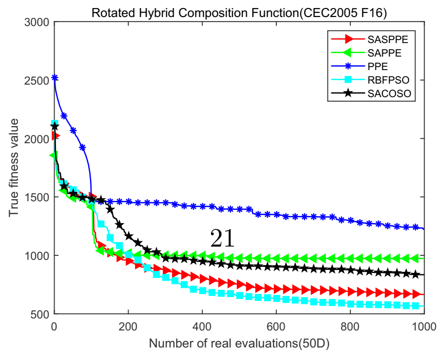
(d) F(4)



(e) F(5)

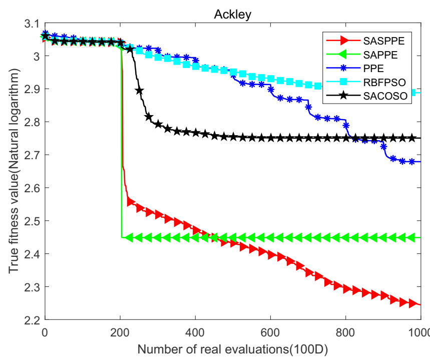


(f) F(6)

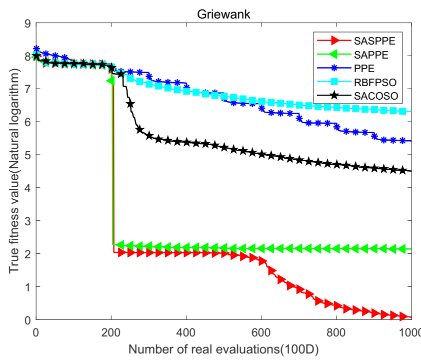


(g) F(7)

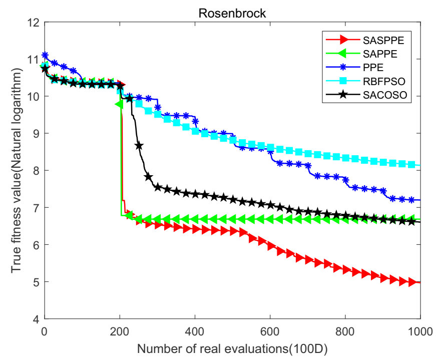
Fig. 5 Comparison of SASPPE, SAPPE, PPE, RBFPSO and SACOSO algorithms for F(1)–F(7) (100D)



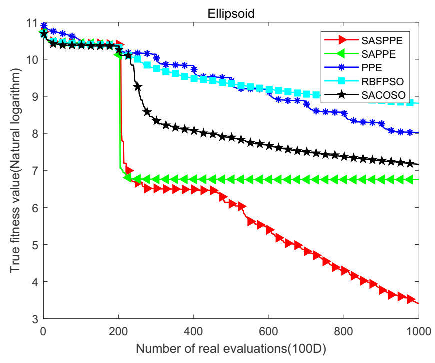
(a) F(1)



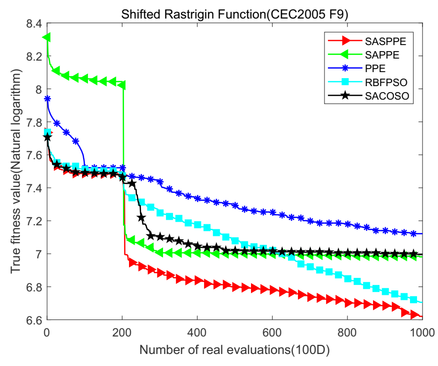
(b) F(2)



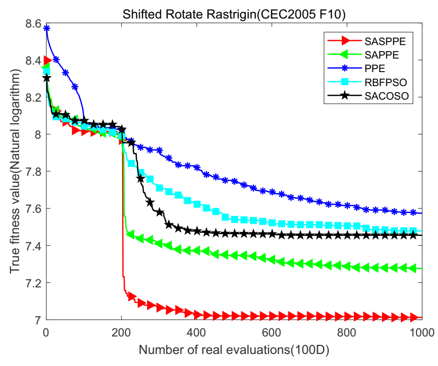
(c) F(3)



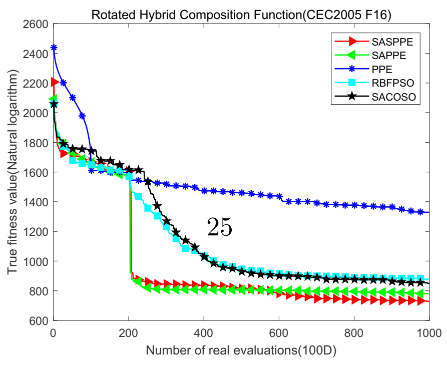
(d) F(4)



(e) F(5)



(f) F(6)



(g) F(7)

after approximately 600, 550, and 450 evaluations, respectively and successfully beats SAPPE. From Fig. 6e, it can be guessed that by increasing the number of evaluations, the SASPPE algorithm can catch up and catch up with SAPPE. On F(5), the SAPPE algorithm wins but matures too early and does not converge as well as the SASPPE algorithm. On F(6), the convergence rate of the SASPPE algorithm and the SAPPE algorithm gradually decreases and eventually plateaus after 150 evaluations. On F(5) and F(6), SASPPE converges to a better optimum than SAPPE, and SASPPE always has a better optimum than SAPPE for the same number of true evaluations. On F(7), the strength of the SASPPE algorithm and SAPPE algorithm are comparable, but the SASPPE algorithm catches up with the SAPPE algorithm at 600 evaluations by its characteristics.As the dimensions increase, the SASPPE algorithm gradually loses its advantage in the F(7) algorithm.

In conclusion, when the dimension is 100, the convergence performance advantage does not decrease from 30D and 50D. On F(1)–F(4), the SASPPE algorithm always wins with absolute advantage, and on F(7), the advantage of the SASPPE algorithm is gradually lost as the number of dimensions increases. On F(5) and F(6), the advantage of SASPPE algorithm is most obvious at 50D. However, SAPPE, SASPPE, SACOSO and RBFPSO outperform the other algorithms by enhancing exploration and exploitation through surrogate models. Adding sinusoidal mapping enhances population diversity, avoids premature maturation of the algorithms, and increases their probability of winning. Comparing the three dimensions, it can be seen that SASPPE has the highest optimal value and stability. It is not prone to fall into local optimality in any dimension. The next best algorithm is SAPPE. it can be seen that the application of the agent model is reasonable and the sinusoidal mapping and hierarchical clustering are effective. No single algorithm is suitable for all problems and the fact that the SASPPE algorithm consistently prevails on most functions is enough to show that the proposed algorithm SASPPE is feasible.

Stability analysis

Analysing Figs. 6, 7, and 8 shows that the SASPPE algorithm is the most stable. Regardless of that dimension, the SASPPE algorithm obtains absolute advantage on F(2)–F(4). On F(5), the SASPPE algorithm is poorly stable on 30D and 50D, but stable on 100D. On 30D, on F(1), the SASPPE algorithm is stable and the worst value is better than the optimal value of SAPPE and SACOSO, while the RBFPSO algorithm has deviating values and poor stability. With the increase of dimension, all algorithms can get better stability on F(7), and the advantage of SASPPE algorithm on F(6) gradually increases. However, the SASPPE algorithm is prone to deviating values on the complex function on 50D, which is prone to chance. On 100D, RBFPSO performs the worst and SACOSO is stable although it does not obtain the optimum value. Overall, the SASPPE algorithm has an advantage over other algorithms in terms of stability.

Empirical analysis of the computational complexity

The computational complexity of the various algorithms varies depending on the composition of the algorithms. The computational complexity of the SASPPE algorithm consists mainly of the actual evaluation time and the time required to train the model. Table 6 shows the computational time of this algorithm and the comparison algorithm on 30D, 50D and 100D respectively. It is easy to see from the table that the PPE algorithm is the most time efficient, this is because PPE does not need to construct a model, only the actual evaluation time. The SAPPE and SACOSO algorithms take more time than the SASPPE algorithm due to the fact that SAPPE and SACOSO do not use clustering and construct larger, more time-consuming models. Comparing RBFPSO, the SASPPE algorithm takes longer time, this is because the SASPPE algorithm has more training time for multiple local agent models than the RBFPSO algorithm. Comparing 100D and 30D shows that the algorithm increases the computation time due to the increase in dimensionality. The PPE algorithm has an insignificant increase in time on F(1)–F(4) and a significant increase in time on F(5)–F(7) complex functions by a factor of almost 20, while the SASPPE algorithm has an insignificant increase in time. Based on the above findings, it is not difficult to conclude that when the problem evaluation is relatively complex, most of the time consumed by the algorithm will be taken up by the fitness evaluation. When the problem size is relatively large, the actual evaluation is more time-consuming. Therefore, the time spent on RBFN model training prediction building is acceptable.

3D coverage of removable nodes

From the above experimental results, it is clear that the algorithm SASPPE proposed in this paper is effective, but it has yet to be tested in terms of practical applications. In this chapter, SASPPE is applied to 3D coverage of movable nodes to check the performance of the algorithm in practical applications.

Parameter settings

This paper uses a combination of random coverage and fixed mobile nodes to meet the dual objectives of low cost and high coverage efficiency. First, the sensor nodes are deployed randomly. Then, the coverage of the current position of the node is compared with the coverage of the place to be moved.

Fig. 6 Comparison of boxplots of the five algorithms for F(1)–F(7) (50D)

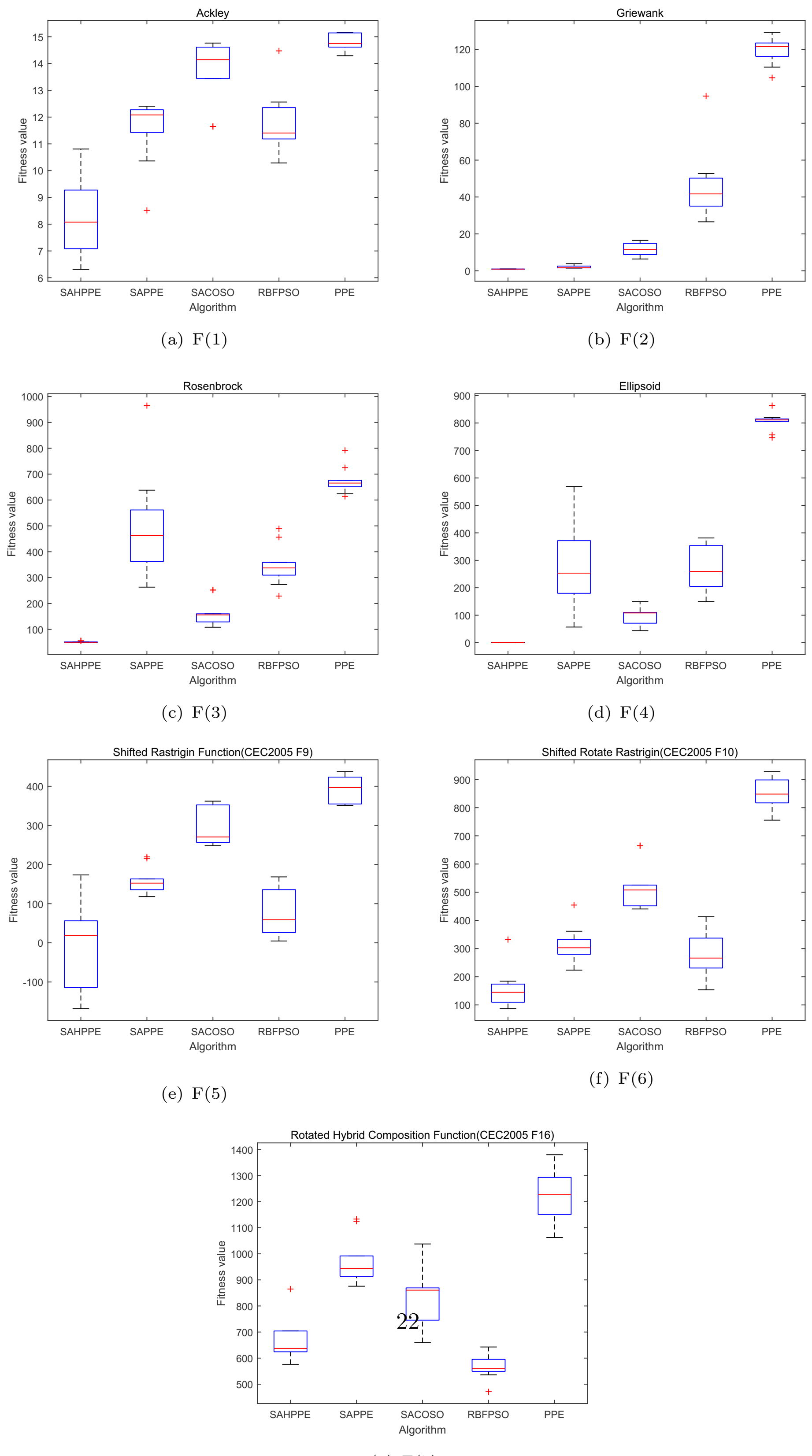
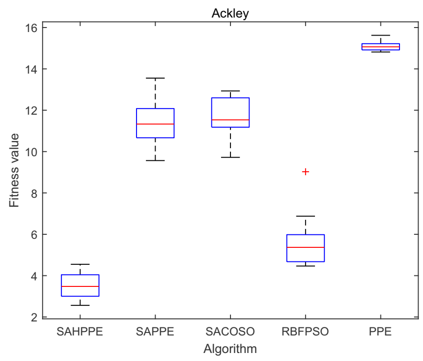
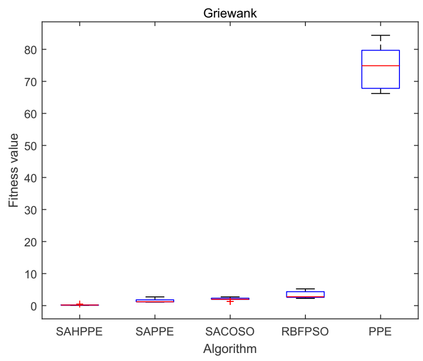


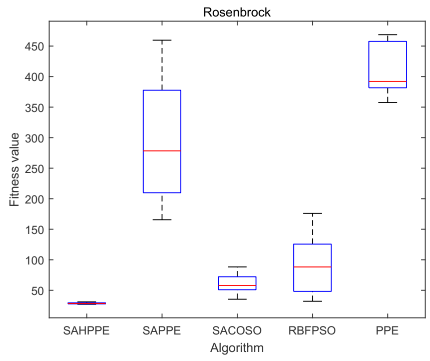
Fig. 7 Comparison of boxplots of the five algorithms for F(1)–F(7) (30D)



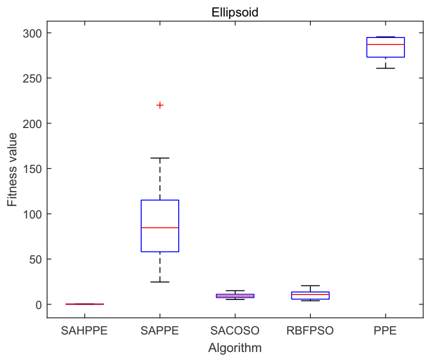
(a) F(1)



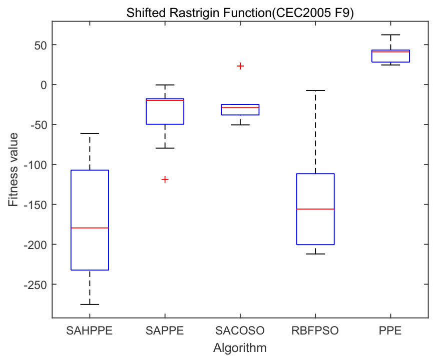
(b) F(2)



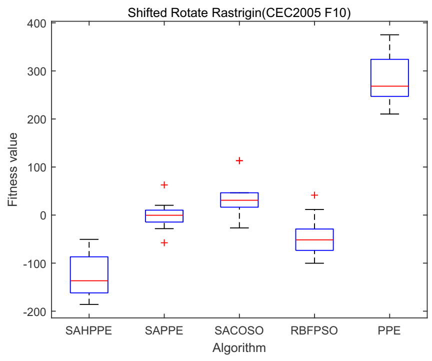
(c) F(3)



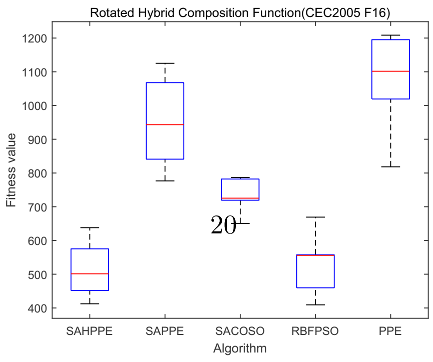
(d) F(4)



(e) F(5)



(f) F(6)



(g) F(7)

Fig. 8 Comparison of boxplots of the five algorithms for F(1)–F(7) (100D)

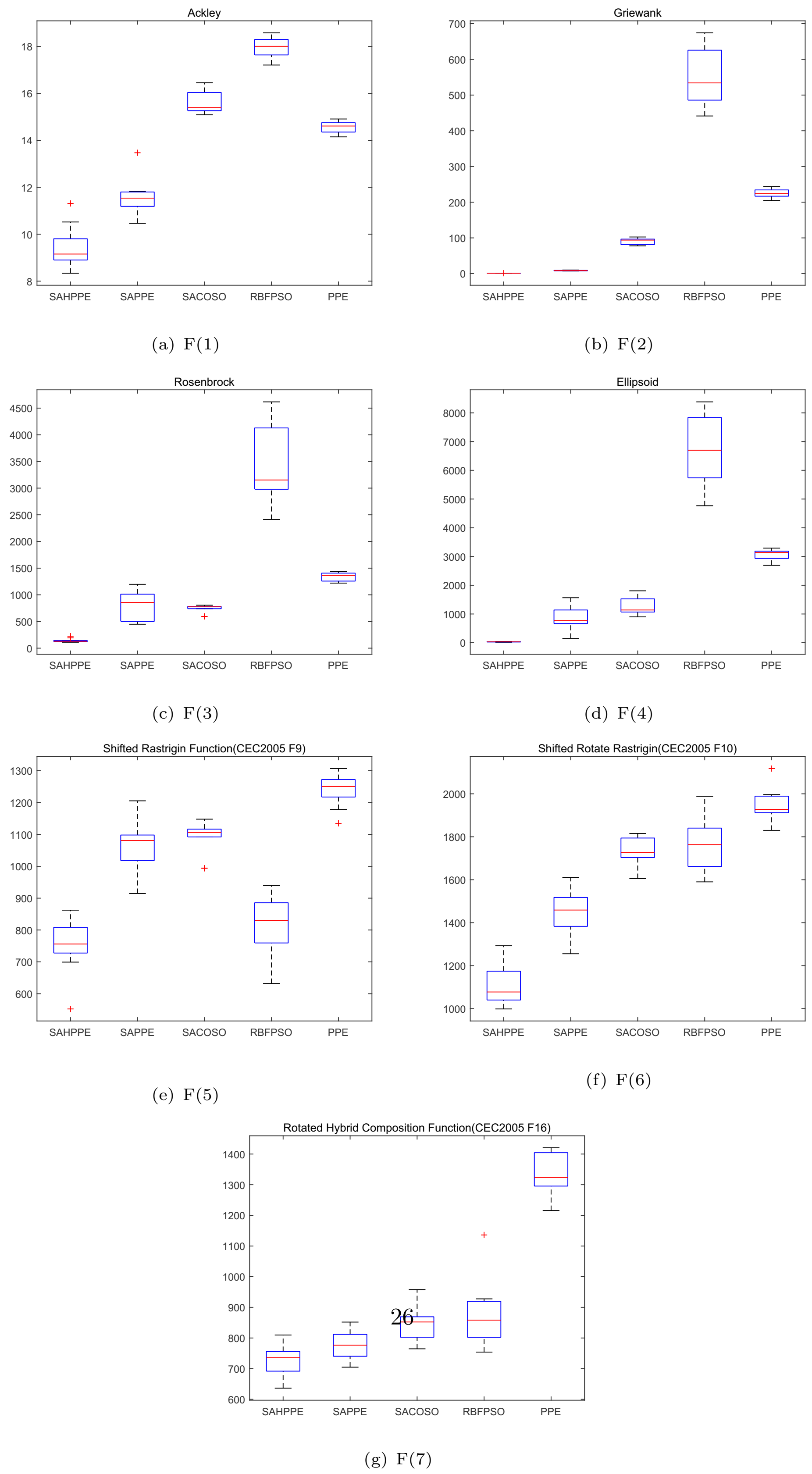


Table 6 Average calculation time required for 1000 fitness evaluations of the compared algorithms in 30D, 50D and 100D problems

Dimension	Algorithm	F(1)	F(2)	F(3)	F(4)	F(5)	F(6)	F(7)
30	SASPPE	1.47E+01	2.23E+01	1.91E+01	2.09E+01	2.27E+01	2.63E+01	1.57E+01
	SAPPE	5.58E+02	5.60E+02	5.64E+02	5.63E+02	5.70E+02	5.45E+02	5.76E+02
	SACOSO	2.63E+02	1.06E+02	1.46E+02	7.12E+01	1.90E+02	2.09E+02	1.54E+02
	RBFPSO	2.08E+00	1.66E+00	1.62E+00	1.61E+00	3.46E+00	3.11E+00	5.81E+00
	PPE	9.70E−02	8.58E−02	8.32E−02	8.81E−02	3.85E+00	1.01E+00	3.71E+00
50	SASPPE	3.46E+01	2.45E+01	3.51E+01	3.08E+01	3.17E+01	3.59E+01	3.19E+01
	SAPPE	5.85E+02	5.97E+02	5.74E+02	5.77E+02	5.75E+02	5.80E+02	5.97E+02
	SACOSO	1.80E+02	6.25E+01	8.54E+01	6.59E+01	2.51E+02	2.35E+02	2.34E+02
	RBFPSO	2.58E+00	2.07E+00	1.59E+00	1.55E+00	2.59E+00	2.83E+00	6.10E+00
	PPE	8.40E−02	8.35E−02	8.41E−02	8.29E−02	3.80E+00	9.90E−01	3.77E+00
100	SASPPE	3.77E+01	3.81E+01	2.81E+01	3.32E+01	4.00E+01	7.24E+01	3.86E+01
	SAPPE	5.61E+02	5.61E+02	5.63E+02	5.59E+02	5.62E+02	5.86E+02	5.62E+02
	SACOSO	2.64E+02	7.72E+01	1.03E+02	7.49E+01	1.97E+02	1.87E+02	2.00E+02
	RBFPSO	1.82E+00	1.53E+00	1.64E+00	1.65E+00	3.96E+00	1.18E+01	5.96E+01
	PPE	9.09E−02	9.14E−02	8.66E−02	8.67E−02	5.49E+01	5.82E+00	5.47E+01

Table 7 3D coverage parameter design for movable nodes

Algorithm	Parameter design	N	R	Rs
PPE	Np = 100; Dim = N * 2;Iteration = 1000	30, 40	3, 5, 7	Rs = R/2; Rs = Rs + 1
PSO				
PPSO				
SASPPE				

If the coverage of the position to be moved is smaller at this time, the node is moved to that location. Conversely, it is not moved. At last, the covered area is set to 1 to increase the network coverage. The energy of each movable sensor node is a constant value E. Each time it moves a distance d, the energy consumed is $E(d)$. The method to determine whether the node should be moved is as follows.

1. Convert 3D actual scenes into 2D.
2. Define the search length Rs $(R/2)$ to search in the four directions of node P.
3. If it is found that the coverage increases by moving to a location, move the node to that location, and conversely, increase the search length Rs to $Rs + 1$ and search again.
4. If the node is searched G times and still cannot be searched, the move is abandoned.

The application scene used in this paper is a 50×50 hill, and its simulation is shown in Fig. 1b. The values of the required parameters in this chapter are shown below. All experiments were run 10 times independently and averaged to avoid experimental chance.

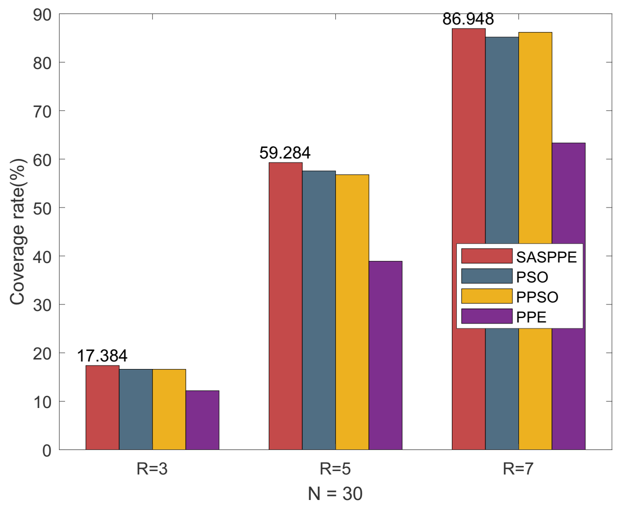
Table 8 Coverage comparison table of SASPPE, PPE, PSO, PPSO

R	N	Algorithm			
		PPE	PSO	PPSO	SASPPE
3	30	12.188	16.616	16.6	17.384
	40	16.148	20.872	20.956	22.532
5	30	38.92	57.548	56.78	59.284
	40	48.236	66.136	65.172	68.228
7	30	63.324	85.184	86.168	86.948
	40	74.324	91.664	91.592	92.572

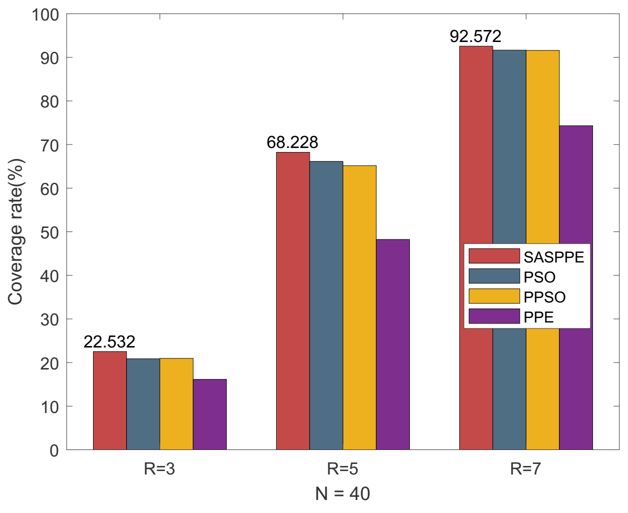
The result indicates the coverage of the algorithm under specific conditions. For example, 12.188 indicates that the average coverage of the PPE algorithm is 12.188% when the sensing radius is 3 and the number of nodes is 30

Results

Table 8 shows the results of SASPPE, PPE, PSO, and PPSO algorithms applied to the 3D coverage of removable nodes. As can be seen in Table 8, the SASPPE has the strongest coverage capability. When the sensor node sensing radius is 3, and the number of nodes is 30, its coverage is only 0.768 higher than PSO, and when the number of nodes increases to 40, its coverage is 1.576 higher than PPSO. When the



(a) N=30



(b) N=40

Fig. 9 Coverage comparison table of SASPPE, PPE, PSO, PPSO

sensing radius of the nodes is 5, the SASPPE algorithm has the greatest advantage and the greatest difference from other algorithms, especially when the number of nodes is 30. The difference with the suboptimal coverage is 2.504. When the sensing radius of the sensors is 5, it is recommended to use 30 sensor nodes when the cost of sensors is considered. When the sensor node sensing radius is 7, the difference between the SASPPE algorithm's and PPSO and PSO is insignificant due to too little uncovered area. To observe the coverage gap of all algorithms more visually, a histogram of coverage is provided in this paper. As shown in Fig. 9, SASPPE wins by a narrow margin among all algorithms, the PPE algorithm is the worst among all algorithms, and PPSO and PSO are comparable in capability. Comparing the two graphs in Fig. 9, it can be found that when the radius is the same, the more nodes, the greater the coverage. When the nodes are the same, the larger the radius, the more extensive the coverage.

Conclusion and future work

This paper proposes a method to enhance network coverage using mobile sensor nodes. All nodes in the network are mobile; however, to reduce energy consumption, nodes need to determine whether they should move before they do so. This not only reduces the energy consumption of the nodes but also maximizes the lifetime of the sensor nodes. The paper proposes a surrogate-assisted sine Phasmatodea population evolution algorithm to maximize network coverage. The GSAM smooth overall algorithm is used in this algorithm to ensure the search capability of the algorithm. This paper uses multiple LSAMs to develop the region in depth at a local scale, trying to find even better results. To better find the optimal value, GSAM, and LSAM use the same database, saving time and cost overhead. Using agglomerative hierarchical clustering to divide multiple LSAMs ensures the robustness of each LSAM. Sine mapping is added to LSAM to enhance the randomness of the algorithm. The algorithm was tested in 7 test functions in three dimensions and the results showed the feasibility of the algorithm is the algorithm. The proposed algorithm has the highest number of functions obtaining the optimum value on 30D, 50D and 100D and only on 50D the advantage is lost on F(7). The algorithm is applied to the 3D coverage of mobile nodes, and the experimental results show that the SASPPE has the highest coverage rate. The improved algorithm is at least 5.196% higher than the original PPE algorithm. However, this experiment leaves something to be desired. It consumes some energy to determine whether a node moves and the movement is fixed and can only be moved around it. The proposed algorithm SASPPE is only applied to the 3D coverage of movable nodes, and its effectiveness in other scenes is unknown. We hope the algorithm performs more robustly and the nodes move more rationally.

Funding The work is supported in part by the Natural Science Foundation of Shandong Province under Grant ZR2023MF109.

Data availability Data available on request from the authors.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Sun Y, Zhang L, Feng G et al (2019) Ant colony optimization. IEEE Internet Things J 6(3):5791–5802
2. Jaiswal SK, Dwivedi AK (2023) A security and application of wireless sensor network: a comprehensive study. In: 2023 International conference on IoT, communication and automation technology (ICICAT), pp 1–5
3. Boubrima A, Bechkit W, Rivano H (2017) Optimal WSN deployment models for air pollution monitoring. IEEE Internet Things J 16(5):2723–2735
4. Cao L, Yue Y, Cai Y, Zhang Y (2021) A novel coverage optimization strategy for heterogeneous wireless sensor networks based on connectivity and reliability. IEEE Access 9:18424–18442
5. Ayadi H, Zouinkhi A, Boussaid B et al (2016) Energy efficiency in WSN: IEEE 802.15.4. In: 2016 17th International conference on sciences and techniques of automatic control and computer engineering (STA), Sousse, pp 766–771
6. Narayan Vipul, Daniel AK, Chaturvedi Pooja (2023) E-FEERP: enhanced fuzzy based energy efficient routing protocol for wireless sensor network. Wirel Pers Commun 131:371–398
7. Tripathi A, Gupta HP, Dutta T et al (2018) Coverage and connectivity in WSNS: a survey, research issues and challenges. IEEE Access 6:26971–26992
8. Amutha J, Sharma S, Nagar J (2020) WSN strategies based on sensors, deployment, sensing models, coverage and energy efficiency: review, approaches and open issues. Wirel Pers Commun 111:1089–1115
9. Elhabyan R, Shi W, St-Hilaire M (2019) Coverage protocols for wireless sensor networks: review and future directions. J Commun Netw 21(1):45–60
10. Yarinezhad R, Hashemi SN (2023) A sensor deployment approach for target coverage problem in wireless sensor networks. J Ambient Intell Humaniz Comput 14(5):5941–5956
11. Wang L, Wu W, Qi J, Jia Z (2018) Wireless sensor network coverage optimization based on whale group algorithm. Comput Sci Inf Syst 15(3):569–583
12. Farsi M, Elhosseini MA, Badawy M et al (2019) Deployment techniques in wireless sensor networks, coverage and connectivity: a survey. IEEE Access 7:28940–28954
13. Sangaiah AK, Sadeghilalimi M, Hosseinabadi AAR, Zhang W (2019) Energy consumption in point-coverage wireless sensor networks via bat algorithm. IEEE Access 7:180258–180269
14. Elhoseny M, Tharwat A, Yuan X, Hassanien AE (2018) Optimizing k-coverage of mobile WSNs. Expert Syst Appl 92:142–153
15. Chen M, Xu A, Wang X (2019) Wireless sensor network energy efficient coverage method based on intelligent optimization algorithm. Discrete Contin Dyn Syst S 124&5:887–900
16. Mohamed SM, Hamza HS, Saroit IA (2017) Coverage in mobile wireless sensor networks (M-WSN): a survey. Comput Commun 110:133–150
17. Tang J, Liu G, Pan Q (2021) A review on representative swarm intelligence algorithms for solving optimization problems: applications and trends. IEEE/CAA J Autom Sin 8(10):1627–1643
18. Tsai CW, Hong TP, Shiu GN (2016) Metaheuristics for the lifetime of WSN: a review. IEEE Sens J 16(9):2812–2831
19. Kaur S, Mahajan R (2018) Hybrid meta-heuristic optimization based energy efficient protocol for wireless sensor networks. Egypt Inf J 16(3):145–150
20. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
21. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
22. Song PC, Chu SC, Pan JS, Yang H (2020) Phasmatodea population evolution algorithm and its application in length-changeable incremental extreme learning machine. In: 2020 2nd International conference on industrial artificial intelligence (IAI), Shenyang, pp 1–5
23. Wu TY, Li HN, Chu SC (2023) CPPE: an improved Phasmatodea population evolution algorithm with chaotic maps. Mathematics 11(9):1977
24. Song PC, Chu SC, Pan JS, Yang H (2021) The Phasmatodea population evolution algorithm and its application in 5G heterogeneous network downlink power allocation problem. J Internet Technol 22(6):1199–1213
25. Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–73
26. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. IEEE, pp 1942–1948
27. Kuo RJ, Li SS (2023) Applying particle swarm optimization algorithm-based collaborative filtering recommender system considering rating and review. Appl Soft Comput 135:110038
28. Dereli S, Köker R (2021) Strengthening the PSO algorithm with a new technique inspired by the golf game and solving the complex engineering problem. Complex Intell Syst 7(3):1515–1526
29. Price KV (2013) Differential evolution. Handbook of optimization: from classical to modern approach. Springer, Berlin, pp 187–214
30. Houssein EH, Hosney ME, Oliva D et al (2023) An efficient discrete rat swarm optimizer for global optimization and feature selection in chemoinformatics. Knowl Based Syst 275:110697
31. Liang JH, Oh C, Mathew M et al (2018) Machine learning-based restart policy for CDCL sat solvers. In: Theory and applications of satisfiability testing–SAT 2018: 21st international conference, SAT 2018, Held as part of the federated logic conference, FloC 2018, Oxford, UK, July 9–12, 2018, Proceedings 21, pp 94–110
32. Ma L, Cheng S, Shi Y (2020) Enhancing learning efficiency of brain storm optimization via orthogonal learning design. IEEE Trans Syst Man Cybern Syst 51(11):6723–6742
33. Lu J, Behbood V, Hao P et al (2015) Transfer learning using computational intelligence: a survey. Knowl Based Syst 80:14–23
34. Liang LL, Du ZG, Shieh CS et al (2022) A new PPE algorithm based on parallel communication strategy. In: Advances in intelligent information hiding and multimedia signal processing: proceeding of the IIH-MSP 2021 and FITAT 2021, Kaohsiung, pp 289–298
35. Zhu Y, Yan F, Pan JS et al (2022) Mutigroup-based Phasmatodea population evolution algorithm with multistrategy for IoT electric bus scheduling. In: Wireless communications and mobile computing 2022
36. Li C, Feng B, Li S et al (2019) Dynamic analysis of digital chaotic maps via state-mapping networks. IEEE Trans Circ Syst I Regul Pap 66(6):2322–2335
37. Hua Z, Zhou Y (2019) Exponential chaotic model for generating robust chaos. IEEE Trans Syst Man Cybern Syst 51(6):3713–3724

38. Varol Altay E, Alatas B (2019) Bird swarm algorithms with chaotic mapping. Artif Intell Rev 53:1373–1414
39. Han ZH (2012) Hierarchical kriging model for variable-fidelity surrogate modeling. AIAA J 50(9):1885–1896
40. Yu H, Tan Y, Zeng J et al (2018) Surrogate-assisted hierarchical particle swarm optimization. Inf Sci 454:59–72
41. Cheng K, Lu Z, Ling C, Zhou S (2014) Surrogate-assisted global sensitivity analysis: an overview. Struct Multidiscip Optim 61:1187–1213
42. Eason J, Cremaschi S (2014) Adaptive sequential sampling for surrogate model generation with artificial neural networks. Comput Chem Eng 68:220–232
43. Tahkola M, Keränen J, Sedov D et al (2023) Surrogate modeling of electrical machine torque using artificial neural networks. IEEE Access 8:220027–220045
44. Pan JS, Zhang LG, Chu SC et al (2023) Surrogate-assisted hybrid meta-heuristic algorithm with an add-point strategy for a wireless sensor network. Entropy 25(2):317–2023
45. Razavi S, Tolson BA, Burn DH (2012) Review of surrogate modeling in water resources. Water Resour Res 48(7):2012
46. Sun C, Jin Y, Cheng R et al (2017) Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Trans Evol Comput 21(4):644–660
47. Wu M, Wang L, Xu J et al (2022) Adaptive surrogate-assisted multi-objective evolutionary algorithm using an efficient infill technique. Swarm Evol Comput 75:101170
48. Regis RG (2014) Particle swarm with radial basis function surrogates for expensive black-box optimization. J Comput Sci 5(1):12–23
49. Liang LL, Chu SC, Du ZG, Pan JS (2023) Surrogate-assisted Phasmatodea population evolution algorithm applied to wireless sensor networks. Wirel Netw 29(2):673-675
50. Gu Q, Wang Q, Li X, Li X (2021) A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems. Knowl Based Syst 223:107049
51. Ji X, Zhang Y, Gong D, Sun X (2021) Dual-surrogate-assisted cooperative particle swarm optimization for expensive multimodal problems. IEEE Trans Evol Comput 25(4):794–808
52. Altan A, Karasu S, Bekiros S (2019) Digital currency forecasting with chaotic meta-heuristic bio-inspired signal processing techniques. Chaos Solitons Fractals 126:325–336
53. Syama S, Ramprabhakar J, Anand R et al (2023) A hybrid extreme learning machine model with Lévy flight chaotic whale optimization algorithm for wind speed forecasting. Results Eng 19:10124
54. Ran XC, Xi Y, Lu YG et al (2023) Comprehensive survey on hierarchical clustering algorithms and the recent developments. Artif Intell Rev 56(8):8219–8264
55. Brzozowski Ł, Siudem G, Gagolewski M (2023) Community detection in complex networks via node similarity, graph representation learning, and hierarchical clustering. arXiv preprint 56(8):8219–8264. arXiv:2303.12212
56. Dogan A, Birant D (2022) K-centroid link: a novel hierarchical clustering linkage method. Appl Intell 52:1–24
57. Suganthan PN, Hansen N, Liang JJ et al (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Rep 2005005(2005):2005–2005