Knowledge-Based Systems 184 (2019) 104901

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems☆

Xiwen Cai, Haobo Qiu, Liang Gao *, Chen Jiang, Xinyu Shao

The State Key Laboratory of Digital Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, 1037 Luoyu Road, 430074, Wuhan, PR China

ARTICLE INFO

Article history:
Received 9 January 2019
Received in revised form 27 July 2019
Accepted 28 July 2019
Available online 30 July 2019

Keywords:
Particle swarm optimization
Global surrogate
Local surrogate
Radial basis function
Surrogate-assisted particle swarm optimization
High-dimensional expensive problems
Design optimization

ABSTRACT

In this paper, an efficient surrogate-assisted particle swarm optimization algorithm is proposed to further improve the efficiency for optimization of high-dimensional expensive problems, which sometimes involve costly simulation analysis. Unlike several surrogate-assisted metaheuristic algorithms, the proposed algorithm can effectively balance the prediction ability of surrogates and the global search ability of particle swarm optimization in the optimization process. Specifically, the proposed algorithm efficiently uses the optima obtained from the global surrogate built in the entire design space and the local surrogate built in a neighbor region around the personal historical best particle to update the velocities of the particles. It should be stressed that the neighbor region partition strategy used to obtain the optimums of local surrogates is an essential aspect of the proposed algorithm. This strategy helps to obtain the predicted optima of local surrogates in the neighbor regions to guide the search of particle swarm optimization in the optimization process. In addition, the neighbor region partition strategy considers the diversity of personal historical best particles, which enables the proposed algorithm to efficiently search for different types of problems. Moreover, the optimization efficiency of the proposed algorithm can be enhanced by using the surrogate prescreening strategy. In order to validate the proposed algorithm, it is tested on several high-dimensional numerical benchmark problems and comprehensively compared with several optimization algorithms. The results show that the proposed algorithm is very promising for the optimization of high-dimensional expensive problems.

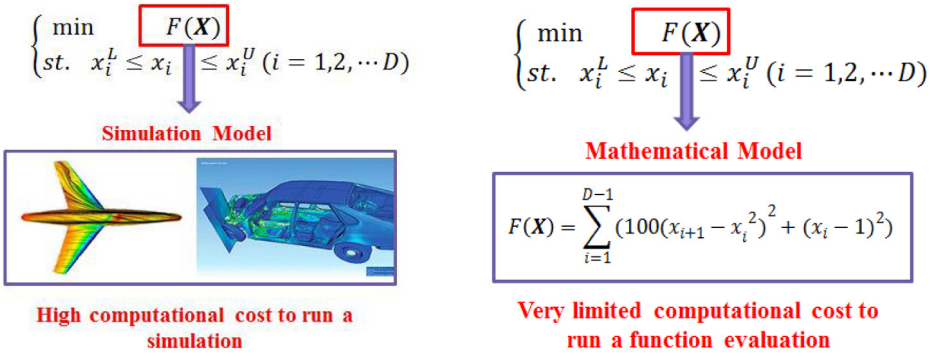
1. Introduction

Engineering optimization problems sometimes involve computationally expensive simulations, which cost minutes, hours, or even days [1,2]. Such problems are usually called as expensive optimization problems. Different from the common numerical optimization problems, the expensive problems have no explicit mathematical expression and require plenty of time to obtain a response value of the optimization objective. The difference between expensive optimization problems and common optimization problems are presented as shown in Fig. 1. The considerable computational cost involved in optimization of the expensive problems therefore limits the application of popular metaheuristic optimization algorithms [3,4] such as the genetic algorithm (GA) [5], differential evolution (DE) [6], particle swarm optimization (PSO) [7] and so on. This is because good optimizing results obtained by the metaheuristic optimization algorithms usually require hundreds of thousands of function calls, which implies that hundreds of thousands of simulations should be conducted in the expensive optimization problems. The computational time is therefore a limitation to the engineering design. To ease the computational burden, researchers attempt to combine computationally cheap approximation models (also referred to as surrogates or metamodels) with metaheuristic algorithms to optimize the expensive problems with dimensions greater than 20. The common surrogates include the kriging [8], radial basis function (RBF) [9], polynomial response surface (PRS) [10] and so on. The surrogate-assisted metaheuristic algorithms generally use the metaheuristic algorithms as the main optimization framework and the surrogates as an additional tool to accelerate the convergence of the basic metaheuristic algorithms. With respect to the acceleration of the convergence, the surrogate-assisted metaheuristic algorithms can be classified into three main categories: metaheuristic algorithms assisted by surrogate prescreening, metaheuristic algorithms assisted by the predicted

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.104901.

* Corresponding author.
E-mail address: gaoliang@mail.hust.edu.cn (L. Gao).

https://doi.org/10.1016/j.knosys.2019.104901
0950-7051/



a. Expensive optimization problems **b.** Common optimization problems

Fig. 1. Difference between the expensive and common optimization problems.

optimum of surrogates, and metaheuristic algorithms assisted by surrogate-based local search.

Metaheuristic algorithms assisted by surrogate prescreening are widely studied for optimization of expensive problems. Some algorithms are based on the kriging surrogate (also referred to as the Gaussian model). They usually utilize the kriging-based sampling infilling criteria such as the expected improvement (EI) [11,12], probability of improvement (PoI) [13] and lower confidence bound (LCB) [14–16] to prescreen promising candidate offspring points produced by the basic metaheuristic algorithms. Moreover, Tian et al. [17] proposed a kriging-based multi-objective infill criterion for prescreening. Li et al. [18] proposed a prescreening criterion based on prediction difference of multiple surrogates. In addition, several algorithms use the predicted response of surrogates for prescreening. They use the surrogate's response function to sort the candidate offspring points produced by the basic metaheuristic algorithms and then select the points with small predicted responses as the offspring points. For example, Praveen and Duvigneau [19] proposed an RBF-assisted PSO algorithm for the aerodynamic shape design. The algorithm can use the surrogate to screen more promising particles among the candidate offspring particles in each optimization iteration. Fonseca et al. [20] used the surrogate as fitness inheritance to assist the GA algorithm in solving optimization problems with a limited computational budget. Regis [21] proposed an optimization algorithm referred to optimization by particle swarm using surrogates (OPUS), which utilizes an RBF surrogate to identify the most promising trial position for each particle in the swarm. Mallipeddi et al. [22] also used surrogates to generate competitive offspring points among the trial offspring points. Sun et al. [23] used RBF to sort the candidate offspring particles and then screen some potential offspring particles with small response values in the RBF-assisted social learning PSO. Yu et al. [24] used the RBF to screen the best particle after many generations of social learning PSO. Wang et al. [25] and Cai et al. [26] also used the similar strategy to screen the best offspring point produced by the DE algorithm. Gong et al. [27] and Chen et al. [28] attempted to screen the promising candidate points based on probabilistic models. Moreover, several approaches [29,30] are based on comparing the candidate point's predicted response of surrogates with its parent point's exact response and then deciding whether the candidate point should be evaluated by the exact response function.

The metaheuristic algorithms assisted by the predicted optima of surrogates usually use the predicted optimum provided by the global surrogate to replace the current best point if the predicted optimum is better than the current best point. For example, Parno et al. [31] proposed a surrogate-assisted PSO algorithm (SPSO), which uses a kriging surrogate to improve the efficiency of the PSO. Tang et al. [32] used a hybrid global surrogate that consist of a quadratic polynomial and an RBF to develop a surrogate-assisted PSO algorithm. Hajikolaei et al. [33] used the global optimum provided by a high-dimensional model representation model to accelerate the convergence of the PSO algorithm. Yu et al. [30] used the optimum provided by a local RBF surrogate built around the current best particle to speed up the PSO's search process. The metaheuristic algorithms assisted by surrogate-based local search usually first use the surrogate-based local search and then use the search mechanism of basic metaheuristic algorithms for global optimization. For example, Ong et al. [34] employed a trust-region method for the interleaved use of exact models for the objective and constrained functions with computationally cheap RBF surrogates during the local search. The GA operator was then run for global optimization. The local search based on the trust region method and surrogates is common in surrogate-assisted metaheuristic algorithms. Additional information on these algorithms can be referred to [35–37]. Moreover, some researchers [38–40] attempts to integrate machine learning with surrogates and metaheuristic algorithms to solve expensive optimization problems. It should be noted that only a brief summary of the surrogate-assisted metaheuristic algorithms is presented in this study. A more comprehensive overview of these algorithms was provided by Jin et al. [41,42].

In this paper, a new efficient surrogate-assisted particle swarm optimization algorithm is proposed to further improve the optimization efficiency for high-dimensional expensive problems. The proposed algorithm can efficiently use the optimum information provided by the global and local surrogates to guide PSO to search in a relatively accurate and efficient manner. Moreover, a surrogate prescreening strategy can also be integrated to enhance the optimization efficiency of the proposed algorithm. Tests are conducted using several benchmark problems with dimensions ranging from 20 to 100. In addition, the proposed algorithm is compared with several other optimization algorithms. Results reveal that the proposed algorithm is promising for optimization of high-dimensional expensive problems.

The remainder of this paper is organized as follows: Section 2 presents the background of the proposed algorithm. Section 3 presents the proposed efficient surrogate-assisted particle swarm optimization algorithm. Section 4 presents the discussions and experimental results and Section 5 concludes the paper.

2. Background

2.1. Particle swarm optimization algorithm (PSO)

The PSO was originally proposed by Eberhart and Kennedy [7] to solve unconstrained optimization problems. It searches by

simulating the behavior of bird flocking or fish schooling. As an efficient optimizer, the PSO has a wide range of applications in many fields including the supply chains [43], finance [44], electric power systems [45], structures [46] and so on. The PSO starts with a population of N swarm particles randomly positioned in the search space. Each of the swarm particles has its own position and velocity. At each iteration, the position and velocity of the ith particle are updated as:

$$v_{id}(t+1) = v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 \left(p_{gd}(t) - x_{id}(t)\right) \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{2}$$

Where d is the dth dimension of a problem, and $\boldsymbol{v}_i(t) = [v_{i1}(t), v_{i2}(t), \ldots, v_{iD}(t)]$ and $\boldsymbol{x}_i(t) = [x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t)]$ are the velocity and position of the ith particle at the tth iteration, respectively. D represents the dimension of a problem. $\boldsymbol{p}_i(t) = [p_{i1}(t), p_{i2}(t), \ldots, p_{iD}(t)]$ is the personal historical best position found by the ith particle; $\boldsymbol{p}_g(t) = [p_{g1}(t), p_{g2}(t), \ldots, p_{gD}(t)]$ is the global best position found by all particles. r_1 and r_2 are two uniformly generated random numbers in the range of [0, 1]. c_1 and c_2 are positive constants referred to as acceleration coefficients.

To improve the efficiency of the PSO, a number of modified PSO algorithms were proposed to update the velocity parameters. Among them, the algorithm proposed by Clerc and Kennedy [47] (the constriction factor model) is commonly used. In this paper, the algorithm is referred to as the CPSO. It uses the following equation to update the velocity of the swarm particles:

$$v_{id}(t+1) = \chi(v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gd}(t) - x_{id}(t))) \tag{3}$$

$$\chi = \left| \frac{2k}{2 - \phi - \sqrt{\phi^2 - 4\phi}} \right| \tag{4}$$

Where $\phi = c_1 + c_2$. Generally, $\phi > 4$, and k is a real number in the range [0, 1].

2.2. Two efficient surrogate-assisted particle swarm optimization algorithms, SPSO and OPUS

The SPSO algorithm was proposed by Parno et al. [31], and a similar algorithm was proposed by Tang et al. [32]. The difference between the two is that the different surrogates are used in the PSO optimization process. The SPSO algorithm is a typical metaheuristic algorithm assisted by the optimum of the global surrogate. This algorithm uses the predicted optimum provided by the global surrogate to replace the current best particle if it is better than the current best particle. Under the CPSO structure, the velocities of the particles in the SPSO are updated as:

$$v_{id}(t+1) = \chi(v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p^*_{gd}(t) - x_{id}(t))) \tag{5}$$

Where

$$\boldsymbol{p}^*_g(t) = \begin{cases} X_{Gbest};\ if\ f(X_{Gbest}) < f\left(\boldsymbol{p}_g(t)\right), \\ \qquad X_{Gbest} = arg\ min\left(\hat{f}_G(x)\right); \\ \boldsymbol{p}_g(t);\ else; \end{cases} \tag{6}$$

Where $\hat{f}_G(x)$ is the predicted response function of the global surrogate built by all the particle points in the entire design space. x is the input variable of $\hat{f}_G(x)$, and X_{Gbest} is the predicted minimum point of $\hat{f}_G(x)$ in the design space.

The SPSO algorithm is more efficient than the PSO, as the surrogate can predict a better global optimum than the current global best particle in the optimization process, which correspondingly improves the social ability of the PSO. The other metaheuristic algorithms assisted by the optimum of a surrogate include the OPUS [21] and other surrogate-assisted PSO algorithms [30,33].

Compared with the SPSO, the OPUS uses not only the global optimum predicted by a local RBF surrogate (built in the region around the current best particle) but also a surrogate prescreening strategy in the optimization process. The OPUS prescreening strategy is presented in Fig. 2. First, the basic PSO updating mechanism is used to produce multiple (s) candidate offspring particles for each parent swarm particle. Thereafter, a global surrogate is used to prescreen the most promising candidate particle, which has the smallest predicted response, as the offspring particle. Under the CPSO structure, the velocity of each swarm particle $v_i(t)$ in the OPUS is updated as follows:

For $l = 1, \ldots, s$

$$v^{(l)}_{id}(t+1) = \chi(v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p^*_{gd}(t) - x_{id}(t)))$$

$$x^{(l)}_{id}(t+1) = x_{id}(t) + v^{(l)}_{id}(t+1)$$

End for.

Let $\boldsymbol{x}_i(t+1)$ be the point in $\left\{\boldsymbol{x}^{(1)}_i(t+1), \boldsymbol{x}^{(2)}_i(t+1), \ldots, \boldsymbol{x}^{(s)}_i(t+1)\right\}$ with the minimum predicted response of $\hat{f}_G$. $\boldsymbol{v}_i(t+1)$ will be the corresponding velocity in obtaining $\boldsymbol{x}_i(t+1)$.

The prescreening strategy based on surrogates is widely used in surrogate-assisted metaheuristic algorithms. The relative researches related to surrogate prescreening have been reviewed in introduction.

3. Proposed efficient surrogate-assisted particle swarm optimization algorithm

In this section, an efficient surrogate-assisted PSO algorithm is proposed to further improve the optimization efficiency for high-dimensional expensive problems. The optimization framework of the algorithm is given in Fig. 3. Compared with the basic PSO, the proposed algorithm presents two changes: (1) the surrogate-assisted PSO updating mechanism is proposed to produce the candidate offspring particles and (2) the surrogate prescreening strategy can also be integrated in the optimization process. In this paper, the proposed efficient surrogate-assisted particle swarm optimization algorithm without and with a prescreening strategy are referred to as the ESPSO and PESPSO, respectively.

3.1. Surrogate-assisted PSO updating mechanism

In the surrogate-assisted PSO updating mechanism, the equation for updating the velocity of the ith particle in the ESPSO (or PESPSO) is given as follows:

$$v_{id}(t+1) = \chi(v_{id}(t) + c_1 r_1 (p^*_{Nid}(t) - x_{id}(t)) + c_2 r_2 (p^*_{gd}(t) - x_{id}(t))) \tag{7}$$

Where

$$\boldsymbol{p}^*_g(t) = \begin{cases} X_{Gbest};\ if\ f(X_{Gbest}) < f\left(\boldsymbol{p}_g(t)\right), \\ \qquad X_{Gbest} = arg\ min\left(\hat{f}_G(x)\right); \\ \boldsymbol{p}_g(t);\ else; \end{cases} \tag{8}$$

$$\boldsymbol{p}^*_{Ni}(t) = \begin{cases} X_{Nbest};\ if\ \hat{f}_{Li}(X_{Nbest}) < f(\boldsymbol{p}_i(t)), \\ \qquad X_{Nbest} = arg\ min\left(\hat{f}_{Li}(x)\right); \\ \boldsymbol{p}_i(t);\ else; \end{cases} \tag{9}$$

Where $\hat{f}_G(x)$ is the predicted response function of the global surrogate built by all the evaluated particle points $[\boldsymbol{X}_{all}, f(\boldsymbol{X}_{all})]$

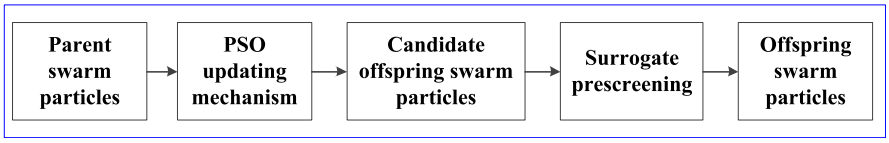


Fig. 2. Prescreening strategy of the OPUS.

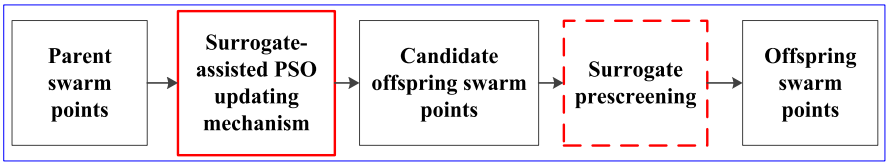


Fig. 3. Optimization framework of the proposed algorithm.

in the entire design space. $\hat{f}_{Li}(x)$ is the predicted response function of the local surrogate built by the evaluated particle points $[\boldsymbol{X}_{local}, f(\boldsymbol{X}_{local})]$ in the neighbor region of $\boldsymbol{p}_i(t)$ and X_{Nbest} is the predicted minimum point of $\hat{f}_{Li}(x)$ in the neighbor region.

The search mode of the ESPSO is presented in the schematic in Fig. 4. It can be seen that the ESPSO changes the search direction of the basic PSO from lines 0 and 1 to lines 0* and 1*. In the ESPSO, the optimum X_{Nbest} predicted by the local surrogate could be better than the personal historical best point $\boldsymbol{p}_i(t)$ of $\boldsymbol{x}_i(t)$. The aim is to improve the cognitive ability of the swarm particles. In addition, by smoothing out the local optima, the global surrogate is built in the entire design space. The global optimum X_{Gbest} predicted by the global surrogate could be better than the current best particle. The aim is to improve the social ability of the swarm particles. In summary, the objective of the local and global surrogates used in the ESPSO is to greatly improve the optimization efficiency of the basic PSO algorithm.

3.2. Neighbor region partition strategy for the personal historical best particles

The neighbor region partition for the personal historical best particles $\boldsymbol{p}_i(t), i = 1, 2 \ldots N$ is a special and essential characteristic of the ESPSO. In the optimization process of the ESPSO, the strategy uses the predicted optima obtained in the neighbor regions to guide the search of the PSO. Meanwhile, the strategy also considers the diversity of personal historical best points $\boldsymbol{p}_i(t), i = 1, 2 \ldots N$, which could have a significant effect on the global search ability of the proposed algorithm. The neighbor region of $\boldsymbol{p}_i(t)$ should be appropriately determined. If the region is excessively large, the neighbor regions may be duplicated, as shown in Fig. 5. Hence, the cognitive role and social role of the swarm particles in the ESPSO are duplicated, which could result in the ESPSO biased towards local search. Moreover, as shown in Fig. 6, if the region is excessively small, the optimum information provided by surrogates is not efficiently utilized, and the efficiency of the ESPSO algorithm is not significantly improved.

Therefore, an algorithm proposed by Kitayama et al. [48] is introduced to obtain a suitable size of the local neighbor regions. The radius r_i of the local region around particle $p_i(t)$ is determined using

$$r_i = \alpha \times \frac{D_{i,max}}{\sqrt{D}\sqrt[D]{N-1}}, \quad i = 1, 2, \ldots, N. \tag{10}$$

Where $D_{i,max}$ denotes the maximum distance between the ith personal historical best particle and the other personal historical best particles. D is the dimension of the design variable. N is the number of the whole personal historical best particles. α is an important parameter used to control the size of the local region. Its setting will be discussed in Section 4.1. The local search region of $\boldsymbol{p}_i(t)$ can then be defined as $[\boldsymbol{p}_i(t) - r_i, \boldsymbol{p}_i(t) + r_i] \cap [\mathbf{lb}, \mathbf{ub}]$, where **lb** and **ub** are the lower and the upper bounds of the design space, respectively. The minimum predicted response point X_{Nbest} can be obtained by using the global optimization algorithm to minimize $\hat{f}_{Li}(x)$ in the local region.

3.3. Surrogate prescreening strategy in the PESPSO

For generalization, the surrogate prescreening strategy, which is widely used in surrogate-assisted metaheuristic algorithms, can also be integrated in the proposed optimization framework. In this paper, the enhanced algorithm is referred to as the PESPSO. The prescreening strategy used in this study is based on the algorithm proposed by Sun et al. [29], which determines whether new particles $\boldsymbol{x}_{i,ESPSO}(t+1)$ produced by the ESPSO particle updating mechanism should be evaluated by the exact fitness function $f(\cdot)$. The evaluation criterion is defined as:

$$\begin{cases} If\ \hat{f}_G\left(\boldsymbol{x}_{i,ESPSO}(t+1)\right) < f(\boldsymbol{p}_i(t)) \\ \quad Evaluate\ \boldsymbol{x}_{i,ESPSO}(t+1)\ \ with\ f(\cdot), \mathrm{NFE} = \mathrm{NFE} + 1; \\ Else \\ \quad Don't\ evaluate\ \boldsymbol{x}_{i,ESPSO}(t+1)\ \ with\ f(\cdot), \mathrm{NFE} = \mathrm{NFE}; \\ End \end{cases} \tag{11}$$

Where $\hat{f}_G(\cdot)$ is the predicted response of the global RBF surrogate which is built by the whole evaluated particles $[\boldsymbol{X}_{all}, f(\boldsymbol{X}_{all})]$. NFE is the number of exact function evaluations, and $\boldsymbol{x}_{i,ESPSO}(t+1)$ is the candidate offspring particle produced by the ESPSO.

The particle-prescreening criterion in Eq. (11) uses the global surrogate to determine whether the new particles produced by the ESPSO should be evaluated by the exact function evaluations. If the predicted response of the new particle is smaller than the response of the corresponding personal historical best particle, the particle will be evaluated by the exact fitness function. Otherwise, no exact function evaluation is carried out for the particle. Therefore, through the appropriate prescreening and evaluation of the candidate particles produced by the ESPSO, PESPSO could be more efficient than the ESPSO. Furthermore, when compared with the PESPSO, the ESPSO can add the same number of new

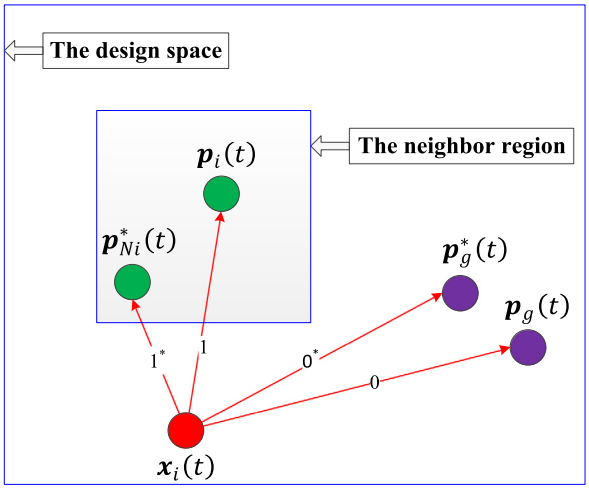


Fig. 4. Schematic diagram of the surrogate-assisted PSO updating mechanism.

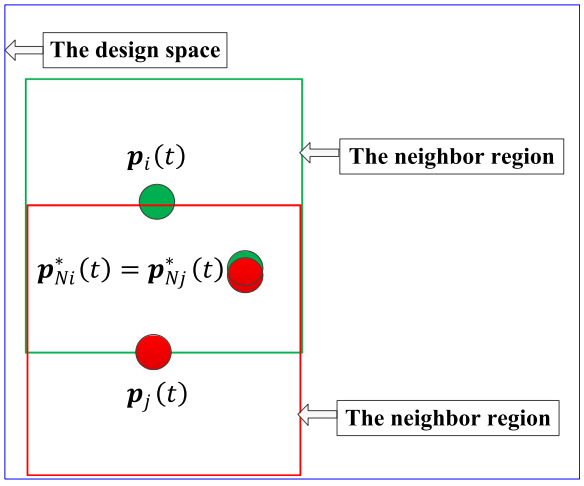


Fig. 5. Large size of neighbor regions belonging to the personal historical best particles $\boldsymbol{p}_i(t)$ and $\boldsymbol{p}_j(t)$.

particles in each iteration, which makes it applicable to parallel computation according to the discussion provided by Ong et al. [34].

3.4. Modeling tool in the optimization process

In this study, the RBF surrogate is used to build the global and local surrogates in the optimization process of the proposed algorithm given that it can generally obtain a more accurate approximation for high-dimensional problems [49–51]. Moreover, the RBF surrogate has a higher metamodeling speed than the kriging or Gaussian model. The RBF surrogate used in this paper is defined as follows.

Given n distinct points $x_1, x_2, \ldots, x_n \in R^D$ and the function values $f(x_1), f(x_2), \ldots, f(x_n)$, the interpolating form of the RBF can be expressed as follows:

$$\hat{f}(x) = \sum_{i=1}^{n} \lambda_i \Phi_i(\|x - x_i\|) + p(x) \tag{12}$$

Where $\Phi_i(\cdot)$ is the ith basis function, $\|\cdot\|$ is the Euclidean norm, and λ_i denotes the weight of the ith basis function. In this study, the cubic function is used as the basis function: $\Phi(r) = r^3$. Other possible choices of the radial basis functions include the thin plate spline, multi-quadric and Gaussian forms. Moreover, $p(x)$ is a linear polynomial function $b^T x + a$.

The unknown parameters ($\lambda_1, \lambda_2, \ldots, \lambda_n \in R^D$, $b \in R^D$, $a \in R$.) of the RBF can be obtained as the solution of the following linear equations:

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{13}$$

Where Φ is the $n \times n$ matrix with $\Phi_{ij} = \Phi(\|x_i - x_j\|)$ and

$$P = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_n^T & 1 \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}, c = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ a \end{pmatrix}, \text{ and } F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

If $rank(P) = D+1$, the matrix $\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix}$ is nonsingular and Eq. (13) has a unique solution [52]. Thus, a unique RBF model is obtained.

3.5. ESPSO and PESPSO procedures

Based on the above description, the pseudo code of the ESPSO is outlined as follows:

Pseudo code of the ESPSO:

//Initialization
1. NFE=0, t=0.
2. Use Latin hypercube sampling to obtain the initial swarm particles, which are expressed by the positions $\boldsymbol{x}_i(0)$, velocities $\boldsymbol{v}_i(0)$ and historical best positions $\boldsymbol{p}_i(0) = \boldsymbol{x}_i(0), i = 1,2,\cdots N$.
3. Calculate the responses $f(\boldsymbol{x}_i(0)), i = 1,2,\cdots N$. NFE=NFE+$N$. The global best position $\boldsymbol{p}_g(t+1)$ is the point in $\{\boldsymbol{p}_1(t+1), \boldsymbol{p}_2(t+1), \cdots, \boldsymbol{p}_N(t+1)\}$ with the minimum response value of $f(\cdot)$.

// Main loop
4. **While** *termination criteria (*NFE<Nmax *or* t<Tmax, Nmax is the maximum number of exact function evaluations, Tmax is the maximum number of iterations*) are not met* **do**

// Update global best position $\boldsymbol{p}_g(t)$
5. Build a global surrogate $\hat{f}_G(x)$ and obtain its predicted global best position X_{Gbest}. Then calculate its response $f(X_{Gbest})$. NFE=NFE+1.
6. Update the global best position $\boldsymbol{p}_g(t)$ using Eq. (8).

// Determine the new swarm particle positions
7. **For** $i = 1$ to N do
8. Build the local surrogate $\hat{f}_{Li}(x)$ in the local region around $\boldsymbol{p}_i(t)$, obtain the best neighbor point X_{Nbest} and update $\boldsymbol{p}_i(t)$ using Eq. (9).
9. Obtain the velocity of the new particle $v_i(t+1)$ using Eq. (7).
10. Obtain the position of the new particle $x_i(t+1) = x_i(t) + v_i(t+1)$.
11. **End for**.
12. Calculate the responses of new swarm particles $f(x_i(t+1))$, NFE=NFE+N.

// Update personal historical best position $\boldsymbol{p}_i(t)$ and global best position $\boldsymbol{p}_g(t+1)$
13. **For** $i = 1$ to N do
14. **If** $f(x_i(t+1)) < f(\boldsymbol{p}_i(t))$
15. $\boldsymbol{p}_i(t+1) = x_i(t+1)$.
16. **Else**
17. $\boldsymbol{p}_i(t+1) = \boldsymbol{p}_i(t)$.
18. **End If**
19. **End for.**
20. Let $\boldsymbol{p}_g(t+1)$ be the point in $\{\boldsymbol{p}_1(t+1), \boldsymbol{p}_2(t+1), \cdots, \boldsymbol{p}_N(t+1)\}$ with the minimum response value of $f(\cdot)$.
21. t=t+1.
22. **End While**

For PESPSO, the optimization procedure described in lines 7–20 above is different. The pseudo code of PESPSO is outlined as follows:

Pseudo code of the PESPSO:

//Initialization

1. NFE=0, t=0.
2. Use Latin hypercube sampling to obtain the initial swarm particles, which are expressed by the positions $\boldsymbol{x}_i(0)$, velocities $\boldsymbol{v}_i(0)$ and historical best positions $\boldsymbol{p}_i(0) = \boldsymbol{x}_i(0), i = 1,2,\cdots N$.
3. Calculate the responses $f(\boldsymbol{x}_i(0)), i = 1,2,\cdots N$. NFE=NFE+$N$. The global best position $\boldsymbol{p}_g(t+1)$ is the point in $\{\boldsymbol{p}_1(t+1), \boldsymbol{p}_2(t+1), \cdots, \boldsymbol{p}_N(t+1)\}$ with the minimum response value of $f(\cdot)$.

// Main loop

4. **While** *termination criteria (*NFE<Nmax *or* NIT<Tmax, Nmax is the maximum number of exact function evaluations, Tmax is the maximum number of iterations*) are not met* **do**
5. Build a global surrogate $\hat{f}_G(x)$ and obtain its predicted global best position X_{Gbest}. Then calculate its response $f(X_{Gbest})$. NFE=NFE+1.
6. Update the global best position $\boldsymbol{p}_g(t)$ using Eq. (8).
7. **For** $i = 1$ to N do
8. Build the local surrogate $\hat{f}_{Li}(x)$ and obtain the best neighbor point X_{Nbest} and update $\boldsymbol{p}_i(t)$ using Eq. (9).
9. Obtain the velocity of the new particle $v_i(t+1)$ using Eq. (7).
10. Obtain the position of the new particle $x_i(t+1) = x_i(t) + v_i(t+1)$.

//Determine whether $\boldsymbol{x}_i(t+1)$ is evaluated by the exact function $f(\cdot)$.

11. **If** $\hat{f}(x_i(t+1)) < f(\boldsymbol{p}_i(t))$
12. $f(x_i(t+1))$, NFE=NFE+1.
13. **End If**

//Update the personal historical best position $\boldsymbol{p}_i(t)$:

14. **If** $f(x_i(t+1)) < f(\boldsymbol{p}_i(t))$
15. $\boldsymbol{p}_i(t+1) = x_i(t+1)$;
16. **Else**
17. $\boldsymbol{p}_i(t+1) = \boldsymbol{p}_i(t)$;
18. **End If**

//Update the global best position $\boldsymbol{p}_g(t+1)$

19. **If** $f(\boldsymbol{p}_i(t+1)) < f(\boldsymbol{p}_g(t)$
20. $\boldsymbol{p}_g(t+1) = \boldsymbol{p}_i(t+1)$.
21. **Else**
22. $\boldsymbol{p}_g(t+1) = \boldsymbol{p}_g(t)$.
23. **End If**
24. **End for**
25. t=t+1.
26. **End While**

Given that the proposed algorithm is under the CPSO optimization structure, the parameters c_1, c_2, r_1, r_2 and k in the ESPSO and PESPSO are set the same as those in the CPSO. In this study, c_1 and c_2 are set as 2.05. r_1 and r_2 are two random numbers in the range of [0, 1]. k is set as 0.729. The swarm size is $N = 30$. To enhance the diversity of the swarm particles, the initial positions and velocities of the swarm particles are designed using the Latin hypercube sampling method (LHS), because the method is considered to sample more uniformly in the design space than the commonly used random sampling method. But with reference to our experimental studies, the CPSO with random initialization of velocities performs as similarly as that with the LHS initialization of velocities. In the process of obtaining the velocities and positions of new particles, if they are out of range, they will be set as the boundary values of the ranges. In this study, the cubic RBF surrogate is used in the optimization process for the proposed algorithm. The number of particle points used to build the local surrogate in ESPSO is set larger than 5D. If this requirement is not satisfied, the surrounding particle points of the corresponding personal historical best particle should be added to the local region for metamodeling until the requirement is satisfied. Moreover, if the number of the whole particles in the design space is smaller than 5D, the local surrogate will be built based on the whole particles. In the metamodeling process, the point in close proximity to other points (the threshold of the distance between any two points is set as 0.01) will not be utilized as it could result in an inaccurate RBF model with bad singularity. The CPSO algorithm is used to minimize $\hat{f}_G(x)$ and $\hat{f}_{Li}(x)$ to the predicted optima.

In summary, the particle updating mechanism of Eq. (7) and the neighbor partition strategy for $\boldsymbol{p}_i(t)$ in Eq. (10) are the main contributions for the proposed algorithm. They can increase the search efficiency of the ESPSO by establishing a balance between the prediction ability of surrogates and the global search ability of the PSO in the optimization process. In particular, the global and local surrogates used in the particle updating mechanism of Eq. (7) are able to accelerate the search speed of the ESPSO. The neighbor partition strategy for $\boldsymbol{p}_i(t)$ in Eq. (10) can enhance the diversity of personal historical best points and thus improve the global search ability of the ESPSO. This method of combining surrogates with particle swarm optimization distinguishes the ESPSO from several other surrogate-assisted metaheuristic algorithms. Moreover, the surrogate prescreening strategy which is widely used in surrogate-assisted metaheuristic algorithms can also be integrated into the proposed PESPSO algorithm.

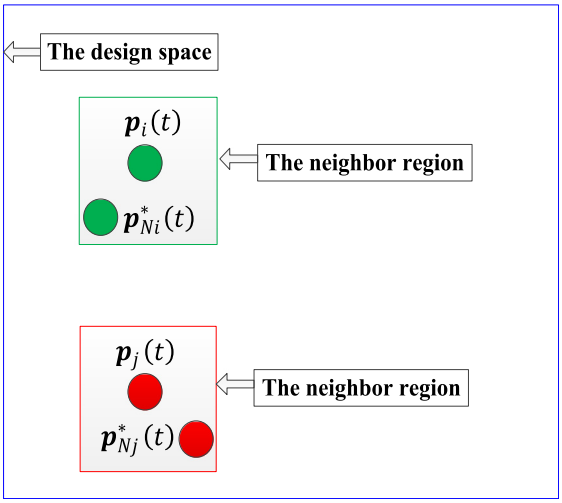


Fig. 6. Small size of neighbor regions belonging to the personal historical best particles $\boldsymbol{p}_i(t)$ and $\boldsymbol{p}_j(t)$.

4. Discussion and comparison

4.1. Discussion of parameter α

As is described in Section 3.2, the parameter α is very important to determine a suitable size of the local region around the personal historical best particle. In this study, the default value of parameter α is set as 0.5, which is suitable to prevent the duplication of local regions of the personal historical best particles and to ensure that their sizes are not excessively small. To validate this viewpoint, the personal historical best particles are assumed to be uniformly distributed. The radius of each personal historical best particle can be expressed as:

$$r = 0.5\frac{D_{max}}{\sqrt{D}\sqrt[D]{N}} \tag{14}$$

Where D_{max} represents the maximum distance between any two different personal historical best particles.

With respect to the K-level full factorial design, the distribution of the design points in a two dimensional (2-D) design space is depicted in Fig. 7, in which Δd is the regular interval between any two points. Moreover, D_{max} can be calculated using

$$D_{max} = \sqrt{D}(K-1)\Delta d, \quad where\ N = K^D. \tag{15}$$

The following then can be obtained:

$$\frac{r}{\Delta d} = 0.5\frac{\sqrt{D}(K-1)\Delta d}{\sqrt{D}\sqrt[D]{N}\Delta d} = 0.5\frac{K-1}{K} = 0.5 - \frac{0.5}{K} \tag{16}$$

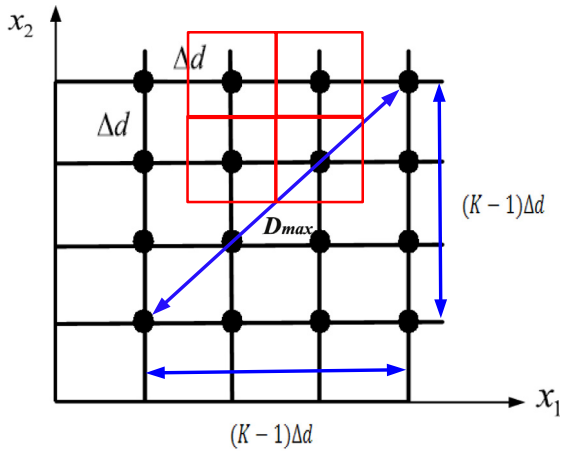


Fig. 7. Distribution of the K-level full factorial design points in the 2-D design space.

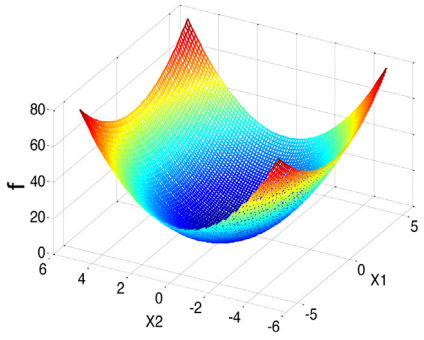


Fig. 8. Landscape of the 2-D Ellipsoid function.

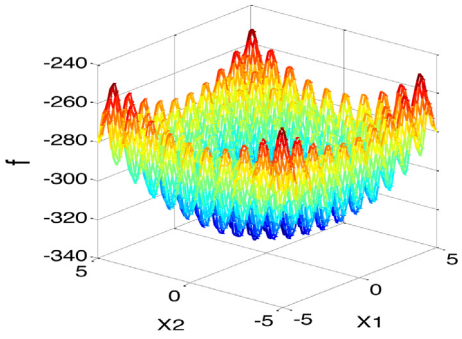


Fig. 9. Landscape of the 2-D modified Rastrigin function.

We can see from Eq. (16) that, with an increase in the sampling points ($K \to \infty$), the ratio $\frac{r}{\Delta d}$ approaches 0.5. Therefore, the local regions belonging to the personal historical best particles are not duplicated, as indicated by the red squares in Fig. 7.

The effect of parameter α is validated by testing the ESPSO on two types of functions including the 20 dimensional (20-D) unimodal Ellipsoid and the 20-D modified multimodal Rastrigin functions. The modified Rastrigin function is the function which downshifts the basic Rastrigin function to -330. The landscapes of these two functions with two variables are plotted in Figs. 8 and 9. To demonstrate the influence of parameter α on the optimization efficiency of the ESPSO, α is set as 0.1, 0.5, 2, 5 and 10, successively, in the ESPSO optimization. Moreover, 25 independent runs are carried out for each value of α. The corresponding convergence curves are depicted in Figs. 10 and 11. According to these two figures, we can find $\alpha = 0.5$ is really a good choice in optimization of functions with different characteristics as it can make the ESPSO perform the best. For the unimodal 20-D Ellipsoid function, the ESPSO with $\alpha = 0.5, 2, 5, 10$ always performs better than the ESPSO with $\alpha = 0.1$. This can be attributed to that the large neighbor region could make surrogates' information utilized sufficiently and make the improved personal historical best particles $\boldsymbol{p}^*_{Ni}, i = 1, 2, \ldots, N$ duplicated. The ESPSO will be biased to local search. However, the ESPSO with large values of $\alpha = 2, 5, 10$ usually does not perform better than the ESPSO with $\alpha = 0.5$. This phenomenon is strange. Because, in our consideration, the number of the duplicated improved personal historical best particles $\boldsymbol{p}^*_{Ni}, i = 1, 2, \ldots, N$ increases and the ESPSO will be greedier for local search with the increase of α ($\alpha > 0.5$). The ESPSO with a larger α will definitely perform better the ESPSO with a small α for the unimodal function. The reason for this strange phenomenon could be explained by analyzing the change of swarm diversity of the ESPSO in the optimization process. The change of swarm diversity of the ESPSO with different α on the 20-D Ellipsoid and modified Rastrigin functions are exemplified by Figs. 12–13, where the Mean-d represents the mean distance between the personal historical best particles and their cluster center. The Mean-d is used to show the swarm diversity. It can be seen from Fig. 12 that the Mean-d of the ESPSO with $\alpha > 0.5$ is larger than the ESPSO with $\alpha = 0.5$. This shows that very large neighbor region doesn't incur the duplication of the improved personal historical best particles and lower down the swarm diversity. The reason can be explained in two aspects. On the one hand, the CPSO is used to optimize the surrogate in the neighbor region. This algorithm has certain randomness and it tends to search locally. The global optimum of the surrogate in the neighbor region is difficult to be obtained. The larger neighbor region will make the obtainment of the global optimum of the surrogate become more difficult. On the other hand, accurate approximation for high-dimensional problems is very difficult. The local surrogates built in the neighbor regions could be not good. These surrogates provide inaccurate optima to replace the personal historical best particles. The two aspects all could incur the increase of the swarm diversity even though a large α is used. These two aspects could also be used to explain the bad performance of the ESPSO with large α for the 20-D modified Rastrigin function. It can be seen from Fig. 13 that the Mean-d of the ESPSO with $\alpha = 2, 5, 10$ is usually larger than that of the ESPSO with $\alpha = 0.1, 0.5$. However, the Mean-d of the ESPSO with $\alpha = 0.5$ is usually larger than that of the ESPSO with $\alpha = 0.1$. The reason could be attributed to that the modified Rastrigin function is a very complex multimodal function. This function has many local optima. Accurate approximation for this function is very difficult. The wrong optima provided by the surrogates in larger neighbor regions could increase the swarm diversity more easily. It can be seen from Figs. 11 and 13 that the ESPSO with $\alpha = 0.5$ can converge similarly to the ESPSO with $\alpha = 0.1$. This shows that $\alpha = 0.5$ has a reasonable control of swarm diversity and the use of surrogates' optima in the optimization of the ESPSO. To sum up, the performance of the ESPSO could be influenced by the accuracy of surrogates, the optimization algorithm used to optimize the surrogates and the value of α. In addition, $\alpha = 0.5$ is an appropriate choice for the ESPSO to optimize different types of problems. It can reasonably control the swarm diversity and the use of surrogates' optima in the ESPSO.

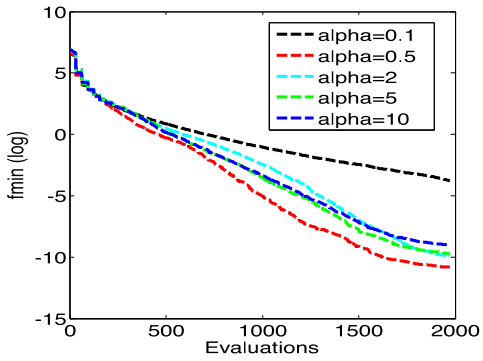


Fig. 10. Convergence curves of the ESPSO with different α for the 20-D Ellipsoid function.

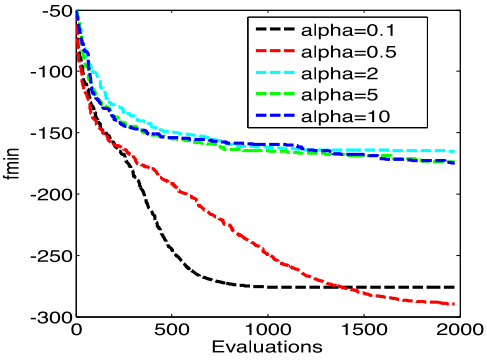


Fig. 11. Convergence curves of the ESPSO with different α for the 20-D modified Rastrigin function.

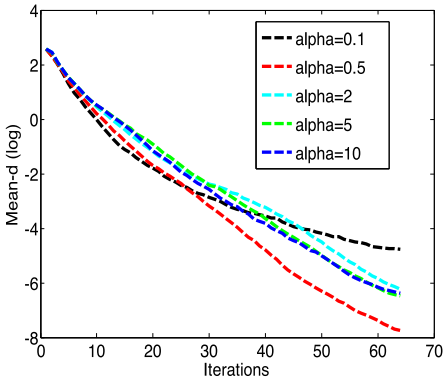


Fig. 12. Change of the swarm diversity of the ESPSO with different α for the 20-D Ellipsoid function.

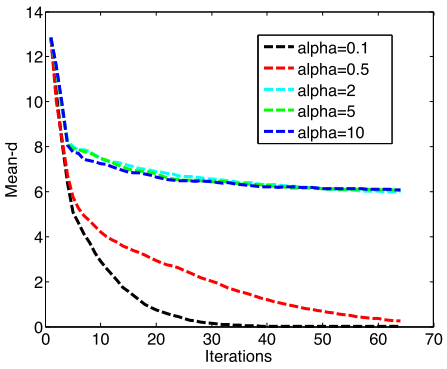


Fig. 13. Change of the swarm diversity of the ESPSO with different α for the 20-D modified Rastrigin function.

4.2. Testing and comparison

In order to validate the performance of the proposed algorithms, four common benchmark problems including the Ellipsoid, Rosenbrock, Ackley and Griewank functions and two other benchmark problems from CEC 2005 [53] are adopted for testing. These problems have different characteristics. They are unimodal, multimodal, highly complicated multimodal problems. Moreover, these problems were used by Lim et al. [35], Liu et al. [14], Sun et al. [23] and Yu et al. [24] to validate their proposed algorithms in studies conducted on the optimization for high-dimensional expensive problems. Detailed information on these problems is provided in Table 1.

For an overall comparison, the CPSO, covariance matrix adaptation evolution strategy (CMA-ES) [54], accelerated particle swarm optimization (APSO) [55], SPSO, OPUS and the proposed ESPSO and PESPSO algorithms are tested. The MATLAB codes of the CMA-ES and APSO are publicly available online. The other surrogate-assisted particle optimization algorithms including the SPSO, OPUS, ESPSO and PESPSO are all coded by the authors using the structure of the CPSO and the Cubic RBF. For the CPSO and the CPSO-based algorithms, the swarm size is $N = 30$. The parameters including c_1, c_2 and k are set as 2.05, 2.05 and 0.729 respectively.r_1 and r_2 are two random numbers in the range of [0, 1]. The initial swarm particles are obtained using the Latin hypercube sampling method in MATLAB 2011a. The maximum number of exact fitness evaluations is set as 2000. For the OPUS algorithm, the number of trial particle for each swarm particle is set as s=20. Furthermore, according to Regis [21], the local space of the OPUS should be defined to obtain the predicted optimum from the local surrogate. In this study, it is set as $\left[\boldsymbol{p}_g(t) - \xi, \boldsymbol{p}_i(t) + \xi\right] \cap [\mathbf{lb}, \mathbf{ub}]$, where $\xi = 0.1 \cdot (\mathbf{ub} - \mathbf{lb})$. To obtain robust and reliable results, the CPSO algorithm is used to obtain the optimum information from the surrogates for all the surrogate-assisted particle swarm optimization algorithms. For all compared optimization algorithms, 25 independent runs are carried out for each test problem in MATLAB 2011a. The test results are listed in Table 2. The results of the Wilcoxon rank sum test between the proposed ESPSO and PESPSO algorithms and the other algorithms are also calculated at a significance level of 0.05. They are listed after the mean optimum of the different algorithms in Table 2, where "$\approx\approx$" indicates that there is no statistically significant difference between the results obtained by the proposed ESPSO and PESPSO algorithms and the compared algorithms. In addition, "++" and "−" indicates that the ESPSO and PESPSO perform statistically better and worse than the compared algorithms. In addition, "+", "−" and "$\approx$" after the mean optimization results of the ESPSO represent that the PESPSO performs better than, worse than and comparable to the ESPSO. "Win/Lose/Tie" in the last row of Table 2 represents the number of problems, where the ESPSO and PESPSO perform better than, worse than and comparable to the other compared algorithms. "15/4/3" represents the number of problems, where the PESPSO performs better than, worse than and comparable to the ESPSO. Moreover, the convergence curves of different optimization algorithms for the 20-D, 30-D, 50-D and 100-D problems are plotted in Figs. 14–17, where evaluations in the horizontal axis means the

Table 1
Characteristics of the benchmark problems.

Fun.	Name	D	Design space	Global optimum	Property
F1–F4	Ellipsoid	20, 30, 50, 100	$X \in [-5.12, 5.12]^D$	0	Unimodal
F5–F8	Rosenbrock	20, 30, 50, 100	$X \in [-2.048, 2.048]^D$	0	Multimodal with narrow valley
F9–F12	Ackley	20, 30, 50, 100	$X \in [-32.768, 32.768]^D$	0	Multimodal
F13–F16	Griewank	20, 30, 50, 100	$X \in [-600, 600]^D$	0	Multimodal
F17–F19	Shifted rotated Rastrigin	30, 50, 100	$X \in [-5, 5]^D$	−330	Very complicated multimodal
F20–F22	Rotated hybrid composition function with a narrow basin for the global optimum	30, 50, 100	$X \in [-5, 5]^D$	10	Very complicated multimodal

evaluation times for the exact function and F_{min} in the vertical axis represents the obtained minimum function response.

As can be seen in Table 2 and Figs. 14–17, the proposed ESPSO and PESPSO algorithms achieve significantly better results than all the other algorithms for most of the benchmark problems. In order to further analyze the efficiency of the proposed algorithms, a detailed comparison is made between the proposed algorithms and the compared algorithms.

(1) The ESPSO algorithm versus non-surrogate-assisted optimization algorithms (CPSO, CMA-ES, APSO). The "Win/Lose/Tie" scores of the ESPSO over the CPSO, CMA-ES and APSO are respectively 22/0/0, 22/0/2 and 22/0/1. When compared with non-surrogate-assisted optimization algorithms, the ESPSO uses not only the global surrogate built in the entire design space but also the local surrogate in a local region around the personal historical best particle. The high efficiency of the ESPSO can be therefore attributed to the global and local surrogates used in its optimization process. The surrogates could accurately approximate the landscape of the tested function. Their predicted optima could provide accurate optimum information, and therefore significantly speed up the search process of the PSO. In an extreme situation where a surrogate is built accurately enough and equal to the tested function, the predicted optimum of the surrogate will be the global optimum, and the search process can be stopped.

(2) The effect of the global surrogate on the efficiency of the ESPSO. The effect of the global surrogate can be explained by comparing the SPSO with the CPSO, given that the ESPSO and SPSO all use the global surrogate, and SPSO only uses a global surrogate under the basic structure of the CPSO. The convergence curves in Figs. 14–17 show that the SPSO can achieve a faster convergence speed and higher accuracy than the CPSO. This is because the constructed global RBF surrogate can predict a better global optimum than the current best particle in the optimization process, thus improving the social ability and global search ability of the PSO. Therefore, the global surrogate can have a positive effect on improving the efficiency of the surrogate-assisted PSO. In contrast, the bad RBF surrogate could have a negative effect. For example, for the 100D Ackley function (F12), the mean optimal solution obtained by the CPSO reaches 15.48. However, the mean optimal solution obtained by the SPSO reaches 15.78. The SPSO performs slightly worse than the CPSO. This is because the built RBF surrogate could not properly approximate the problem and the predicted optimum is worse than the current global best particle. The additional exact function evaluation of wrong optima of surrogates could increase the number of function evaluations of the CPSO and thus lowers down the optimization efficiency of the CPSO.

(3) The effect of the local surrogates on the efficiency of the ESPSO. The effect of local surrogates can be explained by comparing the ESPSO with the SPSO, given that they all use the global surrogate in the optimization process. While the ESPSO adds to use the optimum information from the local surrogates built around each personal historical best point. The "Win/Lose/Tie" score of the ESPSO over the SPSO is 22/0/0. This shows that the ESPSO performs much better than the SPSO, which indicates that the local surrogates improve the optimization efficiency of the ESPSO. The local surrogates correspondingly improve the cognitive ability and global search ability of the PSO.

(4) Comparison of two search modes: ESPSO versus OPUS. In this paper, the OPUS represents the typical search mode of the surrogate-assisted metaheuristic optimization. OPUS regards the surrogate as a prescreening tool for the selection of potential offspring points among the candidate points produce by the basic metaheuristic algorithms. However, the ESPSO represents a new search mode. The ESPSO tightly integrates the surrogate with the PSO algorithm itself. The surrogate prescreening strategy is not necessary to obtain the offspring particles. The "Win/Lose/Tie" score of the ESPSO over the OPUS is 22/0/0. The significantly higher optimization efficiency of the ESPSO in comparison with that of the OPUS can be attributed to the high efficiency of the new search mode.

(5) PESPSO vs ESPSO. The test results reveal that the surrogate-based particle-prescreening strategy can be adequately integrated into the proposed optimization framework, as the PESPSO obtains more accurate global optimum than ESPSO for 15 out of 22 the problems and achieves faster convergence than the ESPSO according to the convergence figures. However, the prescreening strategy could also have adverse effects in some rare cases. For the 20D Griewank function (F13), the final optimization accuracy of the PESPSO is lower than that of the ESPSO, although it initially converges faster than the ESPSO. The reason for this is explained as follows. Through using the prescreening strategy, the quality of the offspring particles can be improved, which makes the PESPSO initially converge faster than the ESPSO. However, the diversity of particle points decreases. For specific cases, the removed particle points that are not selected by the prescreening strategy could have good optimum information. They could be closer to the global optimum than the selected particle points. Hence, the final optimization accuracy of the PESPSO is lower than that of the ESPSO for the F13 function.

4.3. Comparison with the metaheuristic algorithm assisted by surrogate-based local search

The current generalized metaheuristic algorithm assisted by the surrogate-based local search was firstly proposed by Ong et al. [34] and then developed by other researchers [13,35,36]. The algorithm involves the construction of local surrogates in the optimization process. The optimization framework is presented in Fig. 18. This algorithm uses the surrogate-based trust region method (STR) for the local search, and finds the local optimum around each population point. Thereafter, the global search ability of the GA is utilized to find the global optimum. The authors referred to the search mode that involves the balance between the prediction ability of surrogates and global search ability of metaheuristic algorithms as Lamarckian learning. In this paper, the metaheuristic algorithm assisted by the STR local search is referred to as the STR-GA. The basic optimization procedures of the STR-GA are outlined as follows:

Table 2
Performance of different optimization algorithms.

Fun	Metric	CPSO (2002)	CMA-ES (2007)	APSO (2011)	SPSO (2012)	OPUS (2014)	ESPSO	PESPSO
	Best	1.2103E+01	4.0039E−04	1.4635E+00	6.9180E−01	1.0895E−01	6.8963E−06	**5.1662E−06**
	Mean	3.3828E+01 (++)	1.1260E−02 (++)	3.4234E+00 (++)	1.7079E+00 (++)	2.0936E+01 (++)	1.2542E−02 (+)	**3.7908E−05**
F1	Median	2.9288E+01	7.7496E−03	2.7506E+00	1.4879E+00	1.3299E+01	6.4283E−04	**1.5898E−05**
	Worst	8.1716E+01	4.1999E−02	8.2334E+00	3.1583E+00	8.4332E+01	1.0367E−01	**2.2948E−04**
	Std.	1.6846E+01	1.2027E−02	1.9348E+00	7.1548E−01	2.4190E+01	2.5051E−02	**4.8480E−05**
	Best	2.5678E+01	4.5035E−01	6.4216E+00	7.4320E+00	1.1811E+00	3.5518E−05	**2.6998E−06**
	Mean	6.2455E+01 (++)	1.4954E+00 (++)	1.8263E+01 (++)	1.3929E+01 (++)	3.1225E+01 (++)	8.8943E−03 (+)	**5.7732E−04**
F2	Median	5.3736E+01	1.3971E+00	1.7056E+01	1.2432E+01	1.0213E+01	1.4037E−03	**3.3003E−04**
	Worst	1.5257E+02	2.7916E+00	4.4038E+01	2.6943E+01	1.5880E+02	5.8403E−02	**2.3769E−03**
	Std.	3.3101E+01	6.8102E−01	9.6680E+00	5.0048E+00	3.8287E+01	1.5445E−02	**6.5383E−04**
	Best	4.0998E+02	1.9732E+01	1.3521E+02	8.0863E+01	1.1580E+02	4.6343E−01	**1.4438E−03**
	Mean	6.6825E+02 (++)	3.7864E+01 (++)	2.1434E+02 (++)	1.4946E+02 (++)	4.4264E+02 (++)	6.4854E+00 (+)	**3.2574E−01**
F3	Median	6.3997E+02	3.4907E+01	2.0163E+02	1.5115E+02	4.3121E+02	4.8776E+00	**1.0697E−01**
	Worst	1.0308E+03	6.3490E+01	3.8760E+02	2.1688E+02	8.2382E+02	2.1027E+01	**1.8888E+00**
	Std.	1.7857E+02	1.1833E+01	6.8580E+01	3.4181E+01	1.9465E+02	5.6923E+00	**5.3750E−01**
	Best	1.9979E+03	8.5290E+02	2.3319E+03	8.5637E+02	1.0419E+03	7.2849E+00	**2.3620E+00**
	Mean	3.4650E+03 (++)	1.4164E+03 (++)	3.6916E+03 (++)	1.3481E+03 (++)	2.1597E+03 (++)	7.7217E+01 (+)	**1.0243E+01**
F4	Median	3.3634E+03	1.4539E+03	3.7431E+03	1.2452E+03	2.1748E+03	7.3558E+01	**8.3994E+00**
	Worst	4.7989E+03	2.3515E+03	4.4826E+03	2.0991E+03	3.7725E+03	1.5975E+02	**2.5267E+01**
	Std.	6.9278E+02	4.0429E+02	4.8562E+02	3.3886E+02	6.6405E+02	3.9242E+01	**6.5189E+00**
	Best	3.5974E+01	1.5595E+01	1.7688E+01	2.1857E+01	1.8604E+01	1.5260E+01	**8.4264E+00**
	Mean	7.9852E+01 (++)	1.7883E+01 (≈+)	3.7121E+01 (++)	2.8375E+01 (++)	2.7162E+01 (++)	1.7319E+01 (+)	**1.2288E+01**
F5	Median	8.1688E+01	1.7608E+01	2.1529E+01	2.8650E+01	2.1002E+01	1.7530E+01	**1.2387E+01**
	Worst	1.1736E+02	1.9505E+01	1.2222E+02	4.1211E+01	1.0289E+02	1.9079E+01	**1.4470E+01**
	Std.	2.3893E+01	1.1744E+00	2.8840E+01	4.7444E+00	1.7197E+01	**1.0671E+00**	1.1528E+00
	Best	5.2754E+01	2.6695E+01	3.3604E+01	4.3078E+01	3.2476E+01	**2.2540E+01**	2.3251E+01
	Mean	9.4846E+01 (++)	2.8442E+01 (++)	7.6980E+01 (++)	6.1481E+01 (++)	4.1554E+01 (++)	**2.4796E+01** (≈)	2.4927E+01
F6	Median	8.8395E+01	2.8529E+01	6.5428E+01	5.9575E+01	3.5131E+01	**2.4613E+01**	2.4696E+01
	Worst	1.7104E+02	2.9577E+01	1.8673E+02	8.4111E+01	7.6120E+01	**2.9366E+01**	2.9404E+01
	Std.	3.0952E+01	8.9349E−01	4.5553E+01	1.1352E+01	1.3146E+01	1.1803E+00	**1.1444E+00**
	Best	2.7910E+02	5.1630E+01	1.8834E+02	1.2630E+02	9.1657E+01	4.6872E+01	**4.5691E+01**
	Mean	4.8129E+02 (++)	6.4041E+01 (++)	3.4270E+02 (++)	1.8576E+02 (++)	1.8390E+02 (++)	4.8332E+01 (≈)	**4.7901E+01**
F7	Median	4.5003E+02	5.5913E+01	3.5413E+02	1.8694E+02	1.7249E+02	4.8422E+01	**4.7685E+01**
	Worst	8.5633E+02	1.1713E+02	5.2478E+02	2.5442E+02	4.6601E+02	4.9894E+01	**4.9202E+01**
	Std.	1.4522E+02	1.8065E+01	1.0131E+02	3.4924E+01	7.4042E+01	**6.8128E−01**	9.6587E−01
	Best	9.2897E+02	3.6089E+02	1.2588E+03	3.9095E+02	3.4033E+02	1.0249E+02	**9.5938E+01**
	Mean	1.5578E+03 (++)	5.3688E+02 (++)	2.2463E+03 (++)	6.8249E+02 (++)	5.8579E+02 (++)	1.3589E+02 (+)	**9.7815E+01**
F8	Median	1.4792E+03	5.3053E+02	2.1255E+03	6.9145E+02	5.4058E+02	1.3029E+02	**9.7967E+01**
	Worst	2.1758E+03	1.0228E+03	3.7070E+03	9.6034E+02	1.1325E+03	1.7711E+02	**9.8705E+01**
	Std.	3.4079E+02	1.4247E+02	6.2050E+02	1.6966E+02	1.8457E+02	2.3963E+01	**7.1547E−01**
	Best	8.0462E+00	1.9108E+01	2.8826E+00	3.8113E+00	1.6311E+00	6.0960E−04	**1.2183E−04**
	Mean	1.1007E+01 (++)	1.9509E+01 (++)	3.7688E+00 (++)	5.6449E+00 (++)	2.9078E+00 (++)	**3.2581E−02**(−)	4.6706E−02
F9	Median	1.0986E+01	1.9515E+01	3.7886E+00	5.5529E+00	2.9847E+00	8.9104E−04	**5.2021E−04**
	Worst	1.3700E+01	1.9740E+01	4.6412E+00	8.7407E+00	3.9726E+00	**7.9224E−01**	1.1551E+00
	Std.	1.6763E+00	1.5693E−01	4.3275E−01	1.1710E+00	6.0362E−01	**1.5826E−01**	2.3093E−01
	Best	9.0053E+00	1.8989E+01	4.1256E+00	6.3332E+00	3.5649E+00	6.6624E−04	**2.8130E−04**
	Mean	1.2306E+01 (++)	1.9461E+01 (++)	4.8638E+00 (++)	9.0941E+00 (++)	5.2639E+00 (++)	1.2518E−01 (+)	**4.9020E−04**
F10	Median	1.2515E+01	1.9500E+01	4.7632E+00	9.2411E+00	4.7147E+00	1.1387E−03	**4.3696E−04**
	Worst	1.4990E+01	1.9692E+01	5.8296E+00	1.2513E+01	9.8027E+00	9.3162E−01	**1.0533E−03**
	Std.	1.4324E+00	1.8989E−01	4.6481E−01	1.2325E+00	1.5035E+00	2.6162E−01	**1.6574E−04**
	Best	1.1894E+01	1.9272E+01	8.1379E+00	1.1261E+01	6.6024E+00	3.4054E−03	**4.1042E−04**
	Mean	1.4392E+01 (++)	1.9546E+01 (++)	1.1252E+01 (++)	1.4030E+01 (++)	1.1742E+01 (++)	1.9988E+00 (+)	**6.1862E−01**
F11	Median	1.4712E+01	1.9558E+01	1.0966E+01	1.4061E+01	1.0307E+01	1.8030E+00	**2.0214E−03**
	Worst	1.6279E+01	1.9730E+01	1.9031E+01	1.6872E+01	1.8054E+01	5.1685E+00	**3.1071E+00**
	Std.	1.2398E+00	1.1039E−01	2.3568E+00	1.4473E+00	3.9788E+00	1.3228E+00	**8.9213E−01**
	Best	1.4310E+01	1.9376E+01	1.5312E+01	1.4328E+01	9.5666E+00	4.8846E+00	**4.5719E+00**
	Mean	1.5482E+01 (++)	1.9538E+01 (++)	1.8475E+01 (++)	1.5781E+01 (++)	1.4084E+01 (++)	8.1299E+00 (+)	**6.7980E+00**
F12	Median	1.5396E+01	1.9531E+01	1.8359E+01	1.5937E+01	1.4248E+01	8.2249E+00	**6.3709E+00**
	Worst	1.6835E+01	1.9659E+01	2.0519E+01	1.6808E+01	1.8448E+01	1.2944E+01	**1.0201E+01**
	Std.	6.6780E−01	**7.7859E−02**	1.3474E+00	6.8108E−01	2.3624E+00	1.7950E+00	1.5616E+00
	Best	9.5007E+00	2.8442E−01	1.2300E+00	1.1461E+00	4.9473E−01	5.2103E−07	**1.8246E−07**
	Mean	1.5014E+01 (++)	6.5833E−01 (++)	1.3982E+00 (++)	1.3384E+00 (++)	8.1523E−01 (++)	**3.8246E−03 (−)**	7.5716E−03
F13	Median	1.4004E+01	6.2221E−01	1.4051E+00	1.2656E+00	8.0215E−01	1.8600E−05	**5.0579E−07**
	Worst	3.7170E+01	1.0135E+00	1.5693E+00	1.8653E+00	1.0319E+00	4.6822E−02	**4.4154E−02**
	Std.	5.4205E+00	1.7665E−01	8.3971E−02	1.9346E−01	1.4989E−01	**1.0087E−02**	1.4673E−02

(*continued on next page*)

Table 2 (*continued*).

Fun	Metric	CPSO (2002)	CMA-ES (2007)	APSO (2011)	SPSO (2012)	OPUS (2014)	ESPSO	PESPSO
F14	Best	1.6704E+01	1.2797E+00	1.8810E+00	2.5683E+00	1.1069E+00	8.9084E−06	**1.8851E−07**
	Mean	3.5975E+01 (++)	1.6012E+00 (++)	2.3602E+00 (++)	4.2949E+00 (++)	1.2277E+00 (++)	3.7022E−03 (+)	**2.1727E−03**
	Median	3.7172E+01	1.5357E+00	2.2548E+00	4.1919E+00	1.2176E+00	4.5463E−04	**2.8233E−06**
	Worst	5.0587E+01	2.4715E+00	3.0343E+00	6.6072E+00	1.3897E+00	2.4782E−02	**1.9724E−02**
	Std.	7.9143E+00	2.7020E−01	3.4147E−01	1.0665E+00	6.0483E−02	6.5375E−03	**4.9467E−03**
F15	Best	5.5768E+01	2.2312E+01	7.5443E+00	1.0998E+01	2.5755E+00	5.6447E−03	**3.0049E−05**
	Mean	1.0180E+02 (++)	3.7773E+01 (++)	1.8315E+01 (++)	1.8336E+01 (++)	3.1436E+01 (++)	5.9282E−02 (+)	**7.0855E−04**
	Median	1.0003E+02	3.5122E+01	1.6436E+01	1.6872E+01	3.4389E+00	4.5783E−02	**2.4895E−04**
	Worst	1.5582E+02	6.1502E+01	3.6246E+01	3.1689E+01	9.2752E+01	2.1146E−01	**8.8495E−03**
	Std.	2.3751E+01	1.0828E+01	6.5295E+00	5.0568E+00	4.1443E+01	4.9680E−02	**1.7261E−03**
F16	Best	2.1679E+02	4.1374E+02	1.8711E+02	5.0448E+01	3.5015E+01	2.2505E−01	**1.3366E−01**
	Mean	2.8369E+02 (++)	5.3100E+02 (++)	3.1878E+02 (++)	8.4695E+01 (++)	1.6705E+02 (++)	1.8675E+00 (+)	**2.9118E−01**
	Median	2.8230E+02	5.3096E+02	3.2277E+02	8.2083E+01	1.8722E+02	1.9382E+00	**3.1420E−01**
	Worst	3.8206E+02	6.1685E+02	4.3449E+02	1.2172E+02	2.4942E+02	2.9595E+00	**4.0390E−01**
	Std.	4.9181E+01	4.4378E+01	6.0098E+01	2.0769E+01	6.2889E+01	5.8172E−01	**7.4617E−02**
F17	Best	−1.0116E+00	−9.7309E+01	−1.3326E+02	−1.0692E+02	−8.3598E+01	**−2.9279E+02**	−2.7647E+02
	Mean	9.3605E+01 (++)	−2.7218E+01 (++)	−7.1408E+01 (++)	−3.6290E+01 (++)	7.3498E+01 (++)	−1.9705E+02 (≈)	**−2.1186E+02**
	Median	7.6727E+01	−1.0606E+01	−9.7571E+01	−3.8740E+01	7.6294E+01	−2.0834E+02	**−2.0934E+02**
	Worst	2.6151E+02	3.4551E+01	6.2496E+01	2.1410E+01	2.5410E+02	−4.6540E+01	**−1.4283E+02**
	Std.	7.2888E+01	4.8484E+01	5.8312E+01	3.1038E+01	1.0153E+02	5.5787E+01	**3.4905E+01**
F18	Best	5.0303E+02	1.5424E+02	1.5021E+02	2.6610E+02	3.2605E+02	−1.1942E+02	**−1.5584E+02**
	Mean	6.5759E+02 (++)	2.5457E+02 (++)	2.7797E+02 (++)	5.6706E+02 (++)	5.8717E+02 (++)	7.8538E+00 (≈)	**−2.8593E+01**
	Median	6.2804E+02	2.7163E+02	2.7694E+02	6.1205E+02	5.8230E+02	1.2714E+01	**−2.3548E+01**
	Worst	8.6401E+02	3.2371E+02	4.2569E+02	8.8143E+02	7.5975E+02	2.1568E+02	**6.9995E+01**
	Std.	1.0989E+02	**4.5416E+01**	8.4426E+01	1.6194E+02	1.0342E+02	8.5761E+01	6.2026E+01
F19	Best	1.6647E+03	9.4649E+02	1.1748E+03	1.7573E+03	1.8041E+03	1.0397E+03	**6.9459E+02**
	Mean	1.9372E+03 (++)	1.7039E+03 (++)	2.0635E+03 (++)	1.9803E+03 (++)	1.9595E+03 (++)	1.4771E+03 (+)	**8.0017E+02**
	Median	1.9329E+03	1.7254E+03	2.0028E+03	1.9610E+03	1.9368E+03	1.4299E+03	**7.9900E+02**
	Worst	2.1499E+03	3.0278E+03	2.6691E+03	2.3539E+03	2.1707E+03	1.9130E+03	**9.7679E+02**
	Std.	1.2222E+02	5.8411E+02	3.4154E+02	1.4727E+02	1.1158E+02	2.2410E+02	**6.5687E+01**
F20	Best	1.0483E+03	1.0321E+03	9.4489E+02	9.3311E+02	9.7025E+02	9.2428E+02	**9.1741E+02**
	Mean	1.1818E+03 (++)	1.0550E+03 (++)	9.7298E+02 (++)	1.0428E+03 (++)	1.0556E+03 (++)	9.6860E+02 (+)	**9.3079E+02**
	Median	1.1952E+03	1.0565E+03	9.6549E+02	1.0391E+03	1.0460E+03	9.6333E+02	**9.2339E+02**
	Worst	1.2996E+03	1.0773E+03	1.0351E+03	1.1452E+03	1.1912E+03	1.0581E+03	**9.7584E+02**
	Std.	6.7246E+01	**1.1700E+01**	2.5010E+01	5.7231E+01	6.0165E+01	3.2974E+01	1.5084E+01
F21	Best	1.2023E+03	1.0770E+03	1.0048E+03	1.1605E+03	1.0490E+03	**9.8360E+02**	1.0066E+03
	Mean	1.2921E+03 (++)	1.0966E+03 (++)	1.0588E+03 (≈≈)	1.2425E+03 (++)	1.1884E+03 (++)	**1.0512E+03**(≈)	1.0649E+03
	Median	1.3006E+03	1.0907E+03	1.0627E+03	1.2408E+03	1.1773E+03	**1.0394E+03**	1.0603E+03
	Worst	1.3897E+03	1.1287E+03	1.1067E+03	1.3226E+03	1.3222E+03	**1.1759E+03**	1.2015E+03
	Std.	4.5686E+01	1.6657E+01	**3.1855E+01**	4.2384E+01	8.0434E+01	4.7506E+01	4.2129E+01
F22	Best	1.3479E+03	1.3271E+03	1.4284E+03	1.3580E+03	1.3549E+03	1.2279E+03	**1.0043E+03**
	Mean	1.4140E+03 (++)	1.3759E+03 (≈+)	1.4851E+03 (++)	1.4196E+03 (++)	1.4074E+03 (++)	1.3397E+03 (+)	**1.2721E+03**
	Median	1.4224E+03	1.3518E+03	1.4766E+03	1.4178E+03	1.4064E+03	1.3526E+03	**1.3158E+03**
	Worst	1.4623E+03	1.4885E+03	1.5452E+03	1.4870E+03	1.4692E+03	1.4125E+03	**1.3882E+03**
	Std.	3.3441E+01	5.3560E+01	3.4502E+01	2.7014E+01	**2.6302E+01**	5.0123E+01	1.1255E+02
Win/Lose/Tie		22/0/0, 22/0/0	20/0/2, 22/0/0	21/0/1, 21/0/1	22/0/0, 22/0/0	22/0/0, 22/0/0	15/4/3	N/A

Basic optimization procedures of the STR-GA:

//Initialization

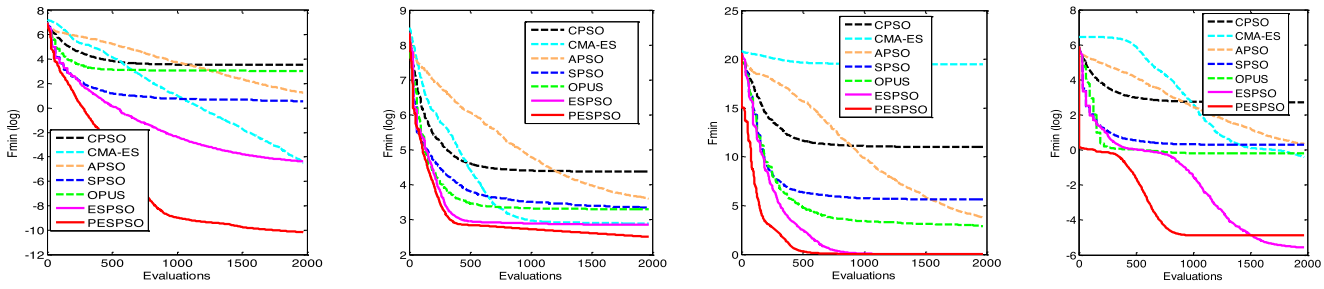
1. Generate a database that includes the parent population points by the design of an experiment such as the Latin hypercube sampling method.

// Main loop

While *termination criterion* (computational budget is not exhausted) *is not met* **do**

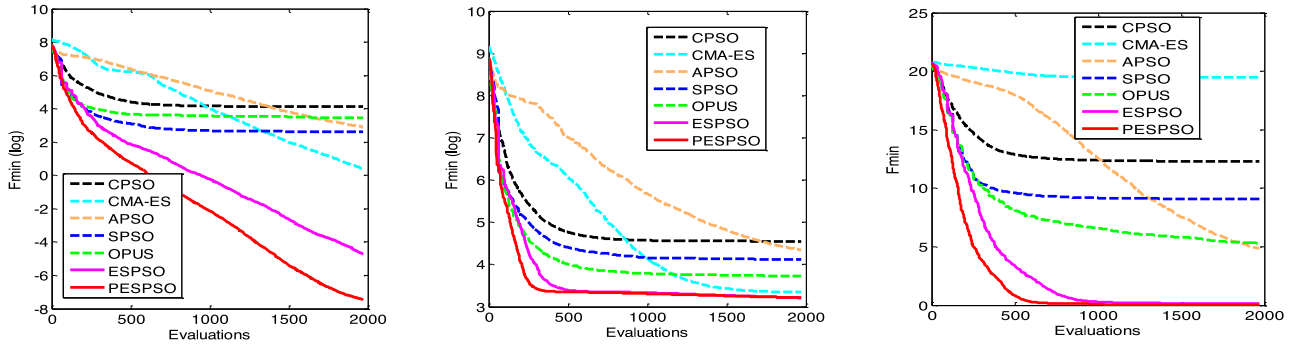
2. Evaluate all individuals in the population using the exact function evaluation.
3. **For** each nonduplicated individual in the population.
4. Apply trust-region-based local search to each individual in the population by interleaving the exact and local surrogate models for the objective functions.
5. Update the database with any new design points generated during the trust-region iterations and evaluate their exact function values.
6. Replace the individuals in the population with the locally improved solution in the spirit of Lamarckian learning.
7. **End For**
8. Apply standard GA operators to create a new child population.
9. **End While**

Based on the above descriptions, the STR-GA also balances the prediction ability of local surrogates and the global search ability of the GA algorithm in the optimization process. The main difference between the STR-GA and ESPSO is that the STR-GA does not use the neighbor partition strategy, which could weaken the diversity of its population points and create a bias towards local search. In addition, the STR-GA is expected to obtain a highly accurate local optimum using the trust region method, whereas the ESPSO only requires the local surrogate to provide a predicted optimum. This could cause the STR-GA to conduct more exact function evaluations than the ESPSO given that the trust region method in the STR-GA requires multiple exact function evaluations. The performance of the ESPSO is compared with that of the STR-GA by Lim et al. [35]. The same problems used to test the ESPSO, including F6, F10, F14, F17 and F20 in Table 1, are also tested by Lim et al. [35]. The results of the STR-GA, which are extracted from the converging figures obtained by Lim et al. [35], are used for comparison. The comparison results are listed in Table 3. It can be seen that the ESPSO demonstrated superior performance to that of STR-GA for the specific problems. For F6, F10 and F14, the ESPSO obtains a better optimal result than the STR-GA within 2000 exact function evaluations. Moreover, the STR-GA would require over 8000 exact function evaluations to obtain the same optimal result as the ESPSO. The same was observed for the cases of F14 and F20. The reason why the ESPSO performs better than the STR-GA could be attributed to that the different surrogates are used in their optimization processes.



(a) F1 (b) F5 (c) F9 (d) F13

Fig. 14. Convergence curves of different optimization algorithms on 20-D benchmark functions.



(a) F2 (b) F6 (c) F10

(d) F14 (e) F17 (f) F20

Fig. 15. Convergence curves of different optimization algorithms on 30-D benchmark functions.

Table 3
Comparison between the STR-GA's optimization results in Lim et al. [35] and the ESPSO's optimization results.

Fun.	STR-GA (2010)		ESPSO
	Mean, NFE = 2000	NFE, $\text{Mean}_{(\text{STR-GA})} \approx \text{Mean}_{(\text{ESPSO})}$	Mean, NFE = 2000
F6	665	> 8000	**2.4796E+01**
F10	13	> 8000	**1.2518E−01**
F14	54	4500	**3.7022E−03**
F17	> 50	> 8000	**−1.9705E+02**
F20	> 1119	5000	**9.6860E+02**

In addition, the STR-GA algorithm, coded by us, is also tested to make a fair comparison with the ESPSO. The STR-GA uses the same type of RBF as that used in the ESPSO. Moreover, for consistency with the ESPSO, the number of training points used to build an accurate local surrogate is set to 5D in the STR local search of the STR-GA. The settings of the other parameters in the STR-GA such as the GA parameters and iterations of the STR local search are the same as those used by Lim et al. [35]. The test results for the 30D and 50D functions are listed in Table 4 and the convergence curves for F17 and F11 are shown in Figs. 19 and 20. The optimization results of the two algorithms are validated by the Wilcoxon rank sum test. "+", "−" and "≈" represent that ESPSO performs better than, worse than and comparable to STR-GA. "Win/Lose/Tie" in the last column of Table 4 represents the number of tested problems where ESPSO performs significantly better than, worse than and comparable to STR-GA. From Table 4, it can be seen that ESPSO demonstrates superior performance to that of the STR-GA for 10 out of 12 the tested problems.

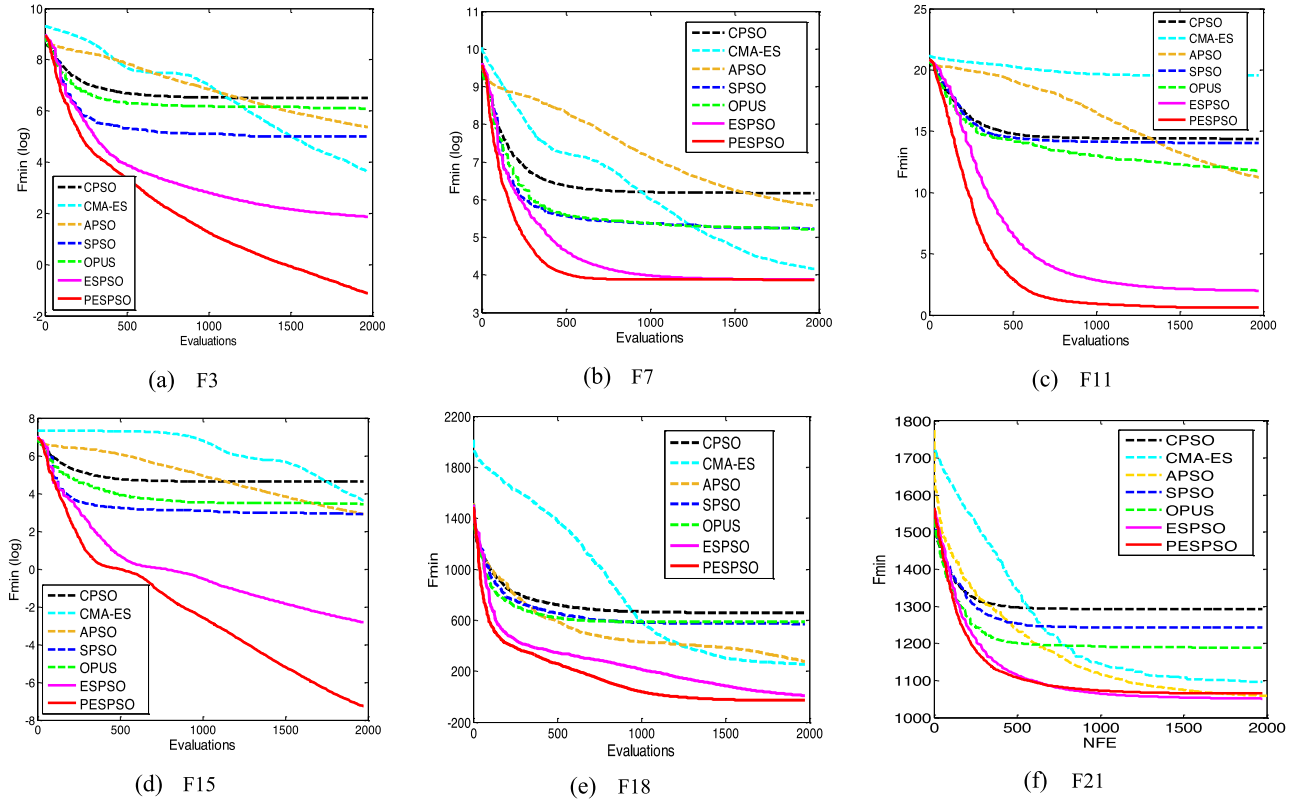


Fig. 16. Convergence curves of different optimization algorithms on 50-D benchmark functions.

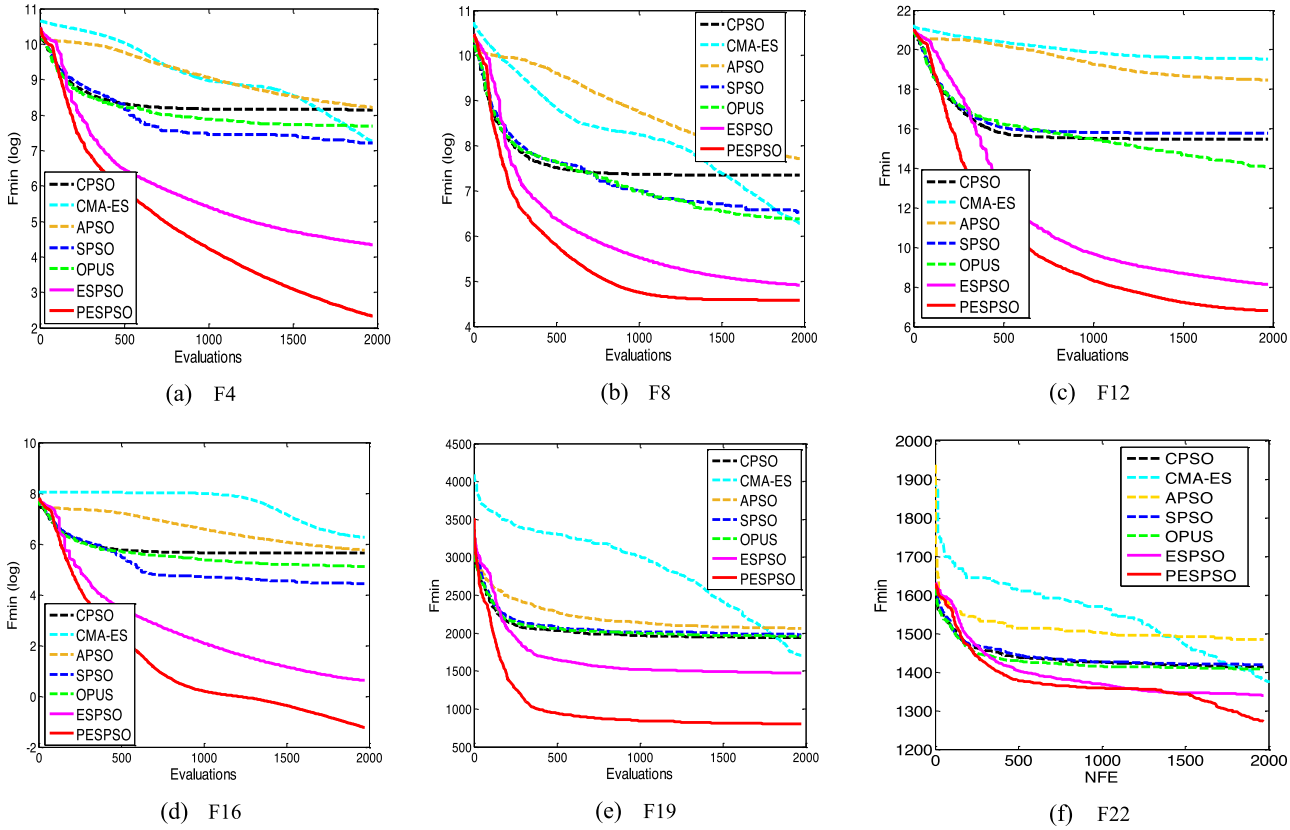


Fig. 17. Convergence curves of different optimization algorithms on 100-D benchmark functions.

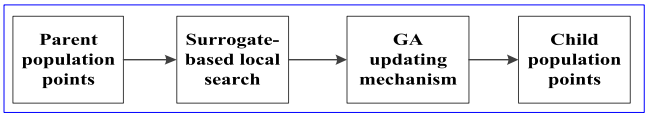


Fig. 18. Flowchart of the STR-GA.

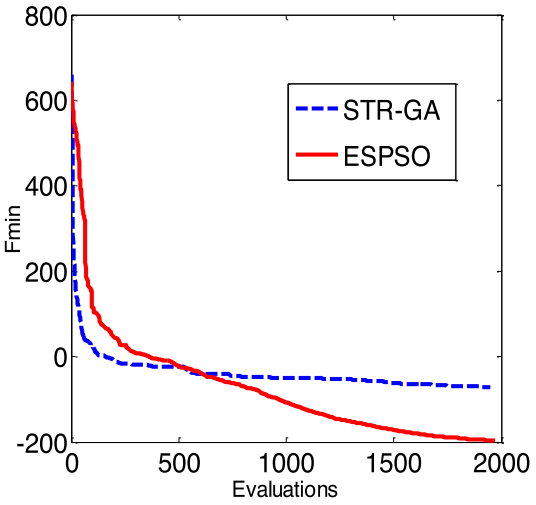


Fig. 19. Convergence curves for the 30-D Shifted Rotated Rastrigin function (F17).

Furthermore, for certain problems such as F6, F10 and so on, the STR-GA is robust by obtaining smaller std. values although its performance is worse than ESPSO. This implies that the STR-GA could be trapped into a local optimum. From the convergence curves of F17 and F11 in Figs. 19 and 20, it can be seen that STR-GA initially has a faster converging speed but finally performs worse than ESPSO. The surrogate-based trust region (STR) method is a local search method, which can speed up the convergence of the STR-GA to a local optimum. However, running the STR for the population points could decrease their diversity, thus weakening the global search ability of the STR-GA. Owing to the proposed neighbor region partition strategy, the ESPSO achieves a more appropriate balance between the prediction ability of surrogates and global search ability of metaheuristic algorithms. Therefore, the ESPSO has better global search ability than the STR-GA. However, the STR-GA could perform better than the ESPSO under the situation of using very limited exact function evaluations, because the STR method makes the STR-GA biased towards local search.

4.4. Comparison with other surrogate-assisted metaheuristic algorithms

To further demonstrate the performance of the proposed algorithm, two recent surrogate-assisted algorithms GPEME [14] and GORS-SSLPSO [24] are used for comparison. The PESPSO is selected for a fair comparison, as the three algorithms (GPEME, GORS-SSLPSO and PESPSO) all use the surrogate prescreening strategy in the optimization process. The commonly tested problems are listed in Table 1.

The GPEME is proposed to handle medium-scale problems with dimensions less than 50, because building the kriging surrogate is very costly for high-dimensional problems. The GPEME can be considered as a typical kriging–prescreening-based evolutionary algorithm. The algorithm uses the LCB infill criterion of kriging to prescreen potential offspring points among the candidate offspring points produced by the basic differential evolution (DE)

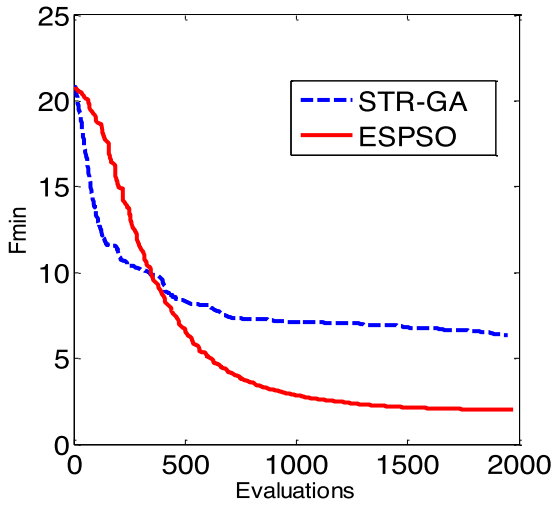


Fig. 20. Convergence curves for the 50-D Ackley function (F11).

algorithm. The comparative results of the PESPSO and GPEME for 20–50D problems are listed in Table 5. The optimization results of GPEME are cited from the research paper of Liu et al. [14]. The t-test results at 0.05 significance level that compares the PESPSO to the GPEME are also listed in Table 5, where '+', '−', and '≈' indicates that the PESPSO is significantly better than, significantly worse than, or comparable to the compared algorithm respectively. The results show that the PESPSO performs significantly better than the GPEME, as its performance is superior to that of the GPEME for 12 out of the 14 tested problems. The PESPSO has a great improvement on the optimization accuracy for most of the problems. Take the 30-D Shifted Rotated Rastrigin function (F17) for an example. The mean optimal solution obtained by the PESPSO reaches −180.0357. The mean optimal solution obtained by the GPEME only reaches −21.8610. The optimization accuracy of the PESPSO increases about 700% over that of the GPEME. However, for the 20-D and 30-D Ellipsoid functions F1 and F2, the GPEME can obtain a higher accurate optimum than the PESPSO. This is because the optimization of the GPEME is carried out under the DE-best/1 optimization framework, which is biased to the local search. Given that the Ellipsoid is a unimodal function, it is more probable for the GPEME to find a more accurate estimate of the global optimum.

The GORS-SSLPSO is a surrogate-assisted PSO algorithm for handling high-dimensional expensive problems whose dimensions can reach 100. Similar with the PESPSO, the algorithm uses the same cubic RBF surrogate in the optimization process. The comparative results of the PESPSO and GORS-SSLPSO for the 50D and 100D benchmark problems are listed in Table 6. The optimization results of the GORS-SSLPSO are cited from the research paper of Yu et al. [24]. The p-values are obtained by pairwise t-test at 0.05 significance level. "+", "−" and "≈" represent that the PESPSO performs better than, worse than and comparable to the GORS-SSLPSO. The results show that the PESPSO algorithm is slightly better than the GORS-SSLPSO, as the performance of the PESPSO is better than, worse than and comparable to that of the GORS-SSLPSO for 5, 4 and 3 problems respectively. However, the comparable performance of the PESPSO and GORS-SSLPSO for the F3, F4 and F7 functions can be attributed to the use of t-test. For example, the mean and the std. values of the optimal solutions obtained by the PESPSO reach 3.9041 and 4.6179 for the 50-D Ellipsoid function (F3). The mean and the std. values of the optimal solutions obtained by the GORS-SSLPSO only reaches

Table 4
Comparative results tested on the STR-GA and ESPSO.

Fun.	STR-GA(2010, Self-coded), NFE = 2000		ESPSO, NFE = 2000	
	Mean	Std.	Mean	Std.
F2	3.8523E−01	4.5424E−01	**8.8943E−03** (+)	**1.5445E−02**
F3	5.0635E+01	1.3742E+01	**6.4854E+00** (+)	**5.6923E+00**
F6	2.8403E+01	**1.0257E+00**	**2.4796E+01** (+)	1.1803E+00
F7	8.5963E+01	9.3496E+00	**4.8332E+01** (+)	**6.8128E−01**
F10	3.2141E−01	**1.4153E−01**	**1.2518E−01** (+)	2.6162E−01
F11	6.3480E+00	**5.1272E−01**	**1.9988E+00** (+)	1.3228E+00
F14	4.4210E−01	1.9533E−01	**3.7022E−03** (+)	**6.5375E−03**
F15	7.9076E+00	2.8174E+00	**5.9282E−02** (+)	**4.9680E−02**
F17	−7.1750E+01	**1.8005E+01**	**−1.9705E+02** (+)	**5.5787E+01**
F18	1.5838E+02	**2.2087E+01**	**7.8538E+00** (+)	8.5761E+01
F20	**9.2594E+02**	**1.8976E+00**	9.6860E+02 (−)	3.2974E+01
F21	**9.3082E+02**	**6.3349E+00**	1.0649E+03 (−)	4.2129E+01
Win/Lose/Tie		10/2/0		

49.098 and 148.4, which are much larger than those of the PE-SPSO. According to mean and std. values of optimization results for the three problems, the PESPSO performs better and more robust than the GORS-SSLPSO. Overall, the PESPSO exhibits good optimization ability for high-dimensional problems under limited computational budget.

4.5. Analysis of the PESPSO's computational complexity

In this subsection, we present an analytical study on the computational complexity of the PESPSO when it is applied for optimization of expensive problems. The total computational cost, referred to as T_{comp}, of the PESPSO is formulated as follows:

$$T_{comp} = \left(T_{exp} + \sum_{i=1}^{N} F_i\right) + (NIT - 1) \cdot \left(\sum_{i=1}^{L} F_i + F_{gbest} + \sum_{i=1}^{N} FLS_i + FGS + T_{overhead}\right) \tag{17}$$

Where:

T_{exp}	Time of experiment design to generate the initial swarm particles, which is often negligible
F	Exact function evaluation cost
NIT	The number of iteration times
L	The number of exact function evaluations in each iteration
FLS	Time required to build and optimize the local RBF surrogates
FGS	Time required to build and optimize the global RBF surrogate
$T_{overhead}$	Additional time costs such as time required for response predictions and particle updating procedures, which are often negligible

Although there are several elements in Eq. (17), it should be noted that when working with computationally expensive problems, the most significant contributor to the total computational cost is F. When F is significantly large, the other elements in Eq. (17) can be negligible. Specifically, the T_{exp} and $T_{overhead}$ take less than 1 s. The FLS and FGS take more time because the approximation and optimization for the surrogates is a little expensive. However, because the RBF surrogate has fast modeling speed, the time for the FLS and FGS are not much. They only take several seconds in one iteration. To further analyze the computational complexity of the proposed algorithm, we take the optimization of the Rosenbrock function for an example and assume the Rosenbrock optimization problem as an expensive optimization problem. Moreover, it is assumed that one exact evaluation of the Rosenbrock function, which means a simulation in certain engineering optimization problems, needs 5 min (300 s). The time costs of the PESPSO for the Rosenbrock function with different dimensions are reported in Table 7. It can be seen that the rate of time cost for the exact function evaluations $F/(F + T_{comp})$ is higher than 99% for all the tested Rosenbrock functions. This confirms that F is the main contributor to the total computational cost when it is significantly large. The time cost of implementation of codes of the PESPSO can be negligible for the expensive optimization problems.

5. Conclusions

In this paper, an efficient surrogate-assisted particle swarm optimization algorithm was proposed. The algorithm efficiently used the optimum information provided by the global surrogate built in the entire design space and the local surrogate built in a neighbor region around the personal historical best particle. The surrogates guided the PSO to search in a relatively accurate and efficient manner. Moreover, the optimization efficiency of the proposed algorithm was enhanced by a widely used surrogate prescreening strategy. To validate the proposed algorithm, it was tested on several widely used high-dimensional numerical benchmark problems and compared with several other optimization algorithms. The results showed that the proposed algorithm performed much better than the other compared algorithms for most of tested problems.

In summary, the proposed surrogate-assisted particle swarm optimization algorithm is promising for optimization of the high-dimensional expensive problems. It is expected to be implemented in practical engineering applications in the future. Furthermore, more efficient metamodeling techniques for high-dimensional problems are suggested for the enhancement of the proposed algorithm. Moreover, the proposed algorithm is tested under the structure of the CPSO in this study. The proposed algorithm could be enhanced by optimization based on the structure of other PSO algorithms or other metaheuristic algorithms.

Acknowledgments

This study is financially supported by the National Natural Science Foundation for Distinguished Young Scholars of China under Grant No. 51825502 and the National Natural Science Foundation of China under Grant No. 51805180, No. 51721092 and No. 51675198 is gratefully acknowledged. The research is supported by Program for HUST Academic Frontier Youth Team.

Table 5
Comparative results tested on the GPEME and PESPSO.

Fun.	GPEME (2014), NFE = 1000				PESPSO, NFE = 1000			
	Best	Mean	Worst	Std.	Best	Mean	Worst	Std.
F1	**1.27E−06**	**1.3E−05**	**7.2E−05**	**2.18E−05**	3.6982E−05	1.3531E−04 (−)	5.9376E−04	1.2587E−04
F2	0.0155	**0.0762**	**0.1647**	**0.0401**	**2.5520E−05**	0.1380 (−)	0.6030	0.1842
F3	134.0681	221.0774	372.5567	81.6123	**0.0156**	**3.9041** (+)	**21.1919**	**4.6179**
F5	15.1491	22.4287	75.8806	18.7946	**10.4669**	**15.4503** (+)	**18.5884**	**1.4636**
F6	26.2624	46.1773	88.2325	25.5199	**25.7980**	**27.4812** (+)	**29.4040**	**0.8743**
F7	172.3547	258.2787	401.4187	80.1877	**46.5659**	**48.4644** (+)	**49.4365**	**0.6796**
F9	0.0037	0.1990	1.8403	0.5771	**9.7652E−04**	**0.0481** (+)	**1.1552**	**0.2306**
F10	1.9491	3.0105	4.9640	0.9250	**0.0015**	**0.0986** (+)	**0.6233**	**0.2040**
F11	9.2524	13.2327	14.9343	1.5846	**0.0068**	**0.9515** (+)	**3.9372**	**0.9842**
F13	0.0002	0.0307	0.2234	0.0682	**1.1101e−5**	**0.0077** (+)	**0.0444**	**0.0147**
F14	0.7368	0.9969	1.0761	0.1080	**5.8190e−5**	**0.0130** (+)	**0.0572**	**0.0170**
F15	22.5456	36.6459	64.9767	13.1755	**0.0053**	**0.0883** (+)	**0.2735**	**0.0746**
F17	−57.0678	−21.8610	18.0327	36.4492	**−268.4769**	**−180.0357** (+)	**−102.6981**	**44.6448**
F20	933.1601	958.5939	992.8618	25.6946	**918.1656**	**931.5115** (+)	**976.0358**	**15.1766**
Win/Lose/Tie					12/2/0			

Table 6
Comparative results tested on the GORS-SSLPSO and PESPSO.

Fun	GORS-SSLPSO (2019), NFE = 1000		PESPSO, NFE = 1000		p-value
	Mean	Std.	Mean	Std.	
F3	49.098	148.4	**3.9041**	**4.6179**	6.7264E−02 (≈)
F4	170.64	318.79	**75.4627**	**41.2080**	7.2364E−02 (≈)
F7	79.187	122.50	**48.4644**	**0.6796**	1.0803E−01 (≈)
F8	**100.53**	**10.801**	119.3011	18.8568	1.2335E−05 (−)
F11	3.7053	**0.9442**	**0.9515**	0.9842	5.9415E−15 (+)
F12	9.0422	1.7945	**8.4462**	**1.4973**	9.6145E−02 (+)
F15	**0.0021**	**0.0042**	0.0883	0.0746	2.7025E−08 (−)
F16	9.2403	27.599	**1.2804**	**0.1835**	7.7906E−02 (+)
F18	**−65.813**	**53.086**	47.0483	81.1611	4.4119E−08 (−)
F19	1042.9	129.61	**850.2688**	**74.4305**	1.0835E−08 (+)
F21	**1019.7**	66.189	1074.0	**41.331**	3.9504E−04 (−)
F22	1421.73	**23.635**	**1358.66**	35.354	8.4297E−11 (+)
Win/Lose/Tie			5/4/3		

Table 7
Time cost of the PESPSO for Rosenbrock functions.

Fun.	Dimension	T_{comp} (s)	F (s)	Rate ($F/(F+T_{comp})$)
Rosenbrock (NFE = 2000)	20D	1565.0	2000*300 = 6E+05	99.74%
	30D	806.6		99.87%
	50D	1399.5		99.77%
	100D	2836.5		99.53%

References

[1] Y. Le Guennec, J.-P. Brunet, F.-Z. Daim, M. Chau, Y. Tourbier, A parametric and non-intrusive reduced order model of car crash simulation, Comput. Methods Appl. Mech. Engrg. 338 (2018) 186–207.

[2] X. Cai, H. Qiu, L. Gao, P. Yang, X. Shao, An enhanced RBF-HDMR integrated with an adaptive sampling method for approximating high dimensional problems in engineering design, Struct. Multidiscip. Optim. 53 (2016) 1209–1229.

[3] Z. Beheshti, S.M.H. Shamsuddin, A review of population-based meta-heuristic algorithms, Int. J. Adv. Soft Comput. Appl. 5 (2013) 1–35.

[4] I. Fister Jr., X.-S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, arXiv preprint arXiv:1307.4186, 2013.

[5] M. Mitchell, An Introduction to Genetic Algorithms, MIT press, 1998.

[6] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.

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

[8] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. 13 (1998) 455–492.

[9] H.-M. Gutmann, A radial basis function method for global optimization, J. Global Optim. 19 (2001) 201–227.

[10] J.R. Edwards, Polynomial regression and response surface methodology, in: Perspectives on Organizational Fit, 2007, pp. 361–372.

[11] M.T. Emmerich, K.C. Giannakoglou, B. Naujoks, Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels, IEEE Trans. Evol. Comput. 10 (2006) 421–439.

[12] X. Cai, L. Gao, X. Li, Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems, IEEE Trans. Evol. Comput. (2019).

[13] Z. Zhou, Y.-S. Ong, P.B. Nair, A.J. Keane, K.Y. Lum, Combining global and local surrogate models to accelerate evolutionary optimization, IEEE Trans. Syst. Man Cybern. C 37 (2007) 66–76.

[14] B. Liu, Q. Zhang, G.G. Gielen, A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, IEEE Trans. Evol. Comput. 18 (2014) 180–192.

[15] B. Liu, Q. Chen, Q. Zhang, G. Gielen, V. Grout, Behavioral study of the surrogate model-aware evolutionary search framework, in: Evolutionary Computation, CEC, 2014 IEEE Congress on, IEEE, 2014, pp. 715–722.

[16] B. Liu, Q. Zhang, G. Gielen, A surrogate-model-assisted evolutionary algorithm for computationally expensive design optimization problems with inequality constraints, in: Simulation-Driven Modeling and Optimization, Springer, 2016, pp. 347–370.

[17] D. Guo, Y. Jin, J. Ding, T. Chai, Heterogeneous ensemble-based infill criterion for evolutionary multiobjective optimization of expensive problems, IEEE Trans. Cybern. (2018).

[18] F. Li, X. Cai, L. Gao, Ensemble of surrogates assisted particle swarm optimization of medium scale expensive problems, Appl. Soft Comput. (2018).

[19] C. Praveen, R. Duvigneau, Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design, Comput. Methods Appl. Mech. Engrg. 198 (2009) 1087–1096.

[20] L.G. Fonseca, A.C. Lemonge, H.J. Barbosa, A study on fitness inheritance for enhanced efficiency in real-coded genetic algorithms, in: Evolutionary Computation, CEC, 2012 IEEE Congress on, IEEE, 2012, pp. 1–8.
[21] R.G. Regis, Particle swarm with radial basis function surrogates for expensive black-box optimization, J. Comput. Sci. 5 (2014) 12–23.
[22] R. Mallipeddi, M. Lee, An evolving surrogate model-based differential evolution algorithm, Appl. Soft Comput. 34 (2015) 770–787.
[23] C. Sun, Y. Jin, R. Cheng, J. Ding, J. Zeng, Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems, IEEE Trans. Evol. Comput. 21 (2017) 644–660.
[24] H. Yu, Y. Tan, C. Sun, J. Zeng, A generation-based optimal restart strategy for surrogate-assisted social learning particle swarm optimization, Knowl.-Based Syst. 163 (2019) 14–25.
[25] X. Wang, G.G. Wang, B. Song, P. Wang, Y. Wang, A novel evolutionary sampling assisted optimization method for high dimensional expensive problems, IEEE Trans. Evol. Comput. (2019).
[26] X. Cai, L. Gao, X. Li, H. Qiu, Surrogate-guided differential evolution algorithm for high dimensional expensive problems, Swarm Evol. Comput. 48 (2019) 288–311.
[27] W. Gong, A. Zhou, Z. Cai, A multioperator search strategy based on cheap surrogate models for evolutionary optimization, IEEE Trans. Evol. Comput. 19 (2015) 746–758.
[28] Y. Chen, X. Sun, D. Gong, Y. Zhang, J. Choi, S. Klasky, Personalized search inspired fast interactive estimation of distribution algorithm and its application, IEEE Trans. Evol. Comput. 21 (2017) 588–600.
[29] C. Sun, Y. Jin, J. Zeng, Y. Yu, A two-layer surrogate-assisted particle swarm optimization algorithm, Soft Comput. 19 (2015) 1461–1475.
[30] H. Yu, Y. Tan, J. Zeng, C. Sun, Y. Jin, Surrogate-assisted hierarchical particle swarm optimization, Inform. Sci. 454 (2018) 59–72.
[31] M. Parno, T. Hemker, K. Fowler, Applicability of surrogates to improve efficiency of particle swarm optimization for simulation-based problems, Eng. Optim. 44 (2012) 521–535.
[32] Y. Tang, J. Chen, J. Wei, A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions, Eng. Optim. 45 (2013) 557–576.
[33] K. Haji Hajikolaei, A. Safari, G. Wang, H.G. Lemu, Surrogate-assisted self-accelerated particle swarm optimization, in: 10th AIAA Multidisciplinary Design Optimization Conference, 2014, p. 1486.
[34] Y.S. Ong, P.B. Nair, A.J. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA J. 41 (2003) 687–696.
[35] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comput. 14 (2010) 329–355.
[36] M.N. Le, Y.S. Ong, S. Menzel, Y. Jin, B. Sendhoff, Evolution by adapting surrogates, Evol. Comput. 21 (2013) 313–340.
[37] X. Lu, T. Sun, K. Tang, Evolutionary optimization with hierarchical surrogates, Swarm Evol. Comput. (2019).
[38] X. Sun, D. Gong, W. Zhang, Interactive genetic algorithms with large population and semi-supervised learning, Appl. Soft Comput. 12 (2012) 3004–3013.
[39] X. Sun, D. Gong, Y. Jin, S. Chen, A new surrogate-assisted interactive genetic algorithm with weighted semisupervised learning, IEEE Trans. Cybern. 43 (2013) 685–698.
[40] H. Wang, Y. Jin, J. Doherty, Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems, IEEE Trans. Cybern. 47 (2017) 2664–2677.
[41] Y. Jin, Surrogate-assisted evolutionary computation: Recent advances and future challenges, Swarm Evol. Comput. 1 (2011) 61–70.
[42] Y. Jin, H. Wang, T. Chugh, D. Guo, K. Miettinen, Data-driven evolutionary optimization: An overview and case studies, IEEE Trans. Evol. Comput. (2018).
[43] Y. Gao, G. Zhang, J. Lu, H.-M. Wee, Particle swarm optimization for bi-level pricing problems in supply chains, J. Global Optim. 51 (2011) 245–254.
[44] S.C. Chiam, K.C. Tan, A.A. Mamun, A memetic model of evolutionary PSO for computational finance applications, Expert Syst. Appl. 36 (2009) 3695–3711.
[45] M.R. AlRashidi, M.E. El-Hawary, A survey of particle swarm optimization applications in electric power systems, IEEE Trans. Evol. Comput. 13 (2008) 913–918.
[46] A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi, Metaheuristic Applications in Structures and Infrastructures, Newnes, 2013.
[47] M. Clerc, J. Kennedy, The particle swarm-explosion stability and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002) 58–73.
[48] S. Kitayama, M. Arakawa, K. Yamazaki, Sequential approximate optimization using radial basis function network for engineering optimization, Opt. Eng. 12 (2011) 535–557.
[49] R. Jin, W. Chen, T.W. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, Struct. Multidiscip. Optim. 23 (2001) 1–13.
[50] A.A. Mullur, A. Messac, Metamodeling using extended radial basis functions: a comparative approach, Eng. Comput. 21 (2006) 203.
[51] A. Díaz-Manríquez, G. Toscano, C.A.C. Coello, Comparison of metamodeling techniques in evolutionary algorithms, Soft Comput. 21 (2017) 5647–5663.
[52] M.J. Powell, Radial basis functions in 1990, Adv. Numer. Anal. 2 (1992) 105–210.
[53] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, KanGAL report, 2005005, 2005, p. 2005.
[54] N. Hansen, The CMA Evolution Strategy: A Tutorial (2009), 2010, URL www.bionik.tu-berlin.de/user/niko/cmatutorial.pdf.
[55] X.-S. Yang, S. Deb, S. Fong, Accelerated particle swarm optimization and support vector machine for business optimization and applications, in: International Conference on Networked Digital Technologies, Springer, 2011, pp. 53–66.