Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems

Chaoli Sun, *Member, IEEE*, Yaochu Jin, *Fellow, IEEE*, Ran Cheng, Jinliang Ding, *Senior Member, IEEE*, and Jianchao Zeng

***Abstract*—Surrogate models have shown to be effective in assisting metaheuristic algorithms for solving computationally expensive complex optimization problems. The effectiveness of existing surrogate-assisted metaheuristic algorithms, however, has only been verified on low-dimensional optimization problems. In this paper, a surrogate-assisted cooperative swarm optimization algorithm is proposed, in which a surrogate-assisted particle swarm optimization (PSO) algorithm and a surrogate-assisted social learning-based PSO (SL-PSO) algorithm cooperatively search for the global optimum. The cooperation between the PSO and the SL-PSO consists of two aspects. First, they share promising solutions evaluated by the real fitness function. Second, the SL-PSO focuses on exploration while the PSO concentrates on local search. Empirical studies on six 50-D and six 100-D benchmark problems demonstrate that the proposed algorithm is able to find high-quality solutions for high-dimensional problems on a limited computational budget.**

***Index Terms*—Computationally expensive problems, fitness estimation strategy (FES), particle swarm optimization (PSO), radial-basis-function networks, surrogate models.**

Manuscript received July 30, 2016; revised December 16, 2016 and February 18, 2017; accepted February 23, 2017. Date of publication March 1, 2017; date of current version July 27, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61403272, Grant 61472269, Grant 61525302, and Grant 61590922, in part by the Joint Research Fund for Overseas Chinese, Hong Kong and Macao Scholars of the National Natural Science Foundation of China under the Grant 61428302, in part by the EPSRC under Grant EP/M017869/1, and in part by the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, China. *(Corresponding author: Yaochu Jin.)*

C. Sun is with the Department of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China, and also with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110004, China, on leave from the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: chaoli.sun.cn@gmail.com).

Y. Jin is with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K., and also with the Department of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China (e-mail: yaochu.jin@surrey.ac.uk).

R. Cheng is with the School of Computer Science, University of Birmingham, Edgbaston B15 2TT, U.K. (e-mail: ranchengcn@gmail.com).

J. Ding is with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110004, China (e-mail: jlding@mail.neu.edu.cn).

J. Zeng is with the School of Computer Science and Control Engineering, North University of China, Taiyuan 030051, China (e-mail: zengjianchao@263.net).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2017.2675628

I. Introduction

METAHEURISTIC optimization algorithms, such as genetic algorithms, differential evolution, ant colony optimization, and particle swarm optimization (PSO), have been empirically shown to perform well on many real-world optimization problems, ranging from job shop scheduling [1], [2], power systems [3], [4], wireless networks [5], [6], robotics [7], to training of artificial neural networks (ANNs) [8], and classification [9]. Most metaheuristic algorithms entail a large number of fitness evaluations (FEs) before they can locate the global optimum or a near-optimal solution, which poses a serious barrier in applying metaheuristic algorithms to computationally expensive optimization problems widely seen in structural optimization [10] and computational fluid dynamic optimization [11], among many others. In solving these optimization problems, high-fidelity numerical analysis techniques, e.g., finite element analysis or computational fluid dynamics simulations are often involved to evaluate the performance of candidate solutions, which may consume minutes to hours, or even days of CPU time [12], [13].

Surrogate-assisted, also known as metamodel-assisted evolutionary algorithms, such as surrogate-assisted genetic algorithms [14], surrogate-assisted differential evolution [15], and surrogate-assisted PSO [16], have received increasing attentions in recent years. In surrogate-assisted evolutionary algorithms, surrogate models are employed to replace in part the time-consuming exact function evaluations for saving computational cost because the computational effort required to build and use surrogates is usually much lower than that for expensive FEs [17], [18]. The most commonly used surrogate models include polynomial regression [19], also known as response surface methodology [19], support vector machines [20]–[22], ANNs [12], [23], [24], radial basis function (RBF) networks [25]–[29], and Gaussian processes (GPs), also referred as to Kriging or design and analysis of computer experiment models [26], [30]–[34]. The surrogate-assisted metaheuristic algorithms reported in the literature can be largely classified into the following categories.

1) *Global-Surrogate Assisted Metaheuristic Algorithms:* Global-surrogate models, which aim to model the whole search space, were often used in the earlier stage of the research on surrogate-assisted evolutionary optimization. Ratle [35] proposed to use the Kriging interpolation as a function approximation model to replace the real function evaluation. Jin *et al.* [12] analyzed the convergence property of the ANN assisted evolutionary algorithms

and proposed an empirical criterion to switch between the expensive FEs and cheap fitness estimation during the search. A RBF network was proposed in [36] to assist an evolutionary algorithm for computationally expensive multiobjective problems by prescreening the most promising individuals to be exactly evaluated. Parno *et al.* [37] incorporated design and analysis of computer experiment surrogate model as a stand-in for the expensive objective function within a PSO framework. Regis [16] proposed to generate multiple trial velocities and positions for each particle in each iteration and then an RBF surrogate model was utilized to select the most promising trial position for each particle. Di Nuovo *et al.* [38] presented an empirical study on the use of fuzzy function approximation to evaluate candidate individuals to speed up evolutionary multiobjective optimization. A GP surrogate model was proposed by Liu *et al.* [39] to assist differential evolution to solve computationally expensive optimization problems, in which dimension reduction techniques were utilized to reduce the dimension of the GP surrogate model. Different to the surrogate model building, Gong *et al.* [40] proposed a cheap surrogate model based on density estimation for prescreening the candidate individuals in evolutionary optimization.

2) *Local-Surrogate Assisted Metaheuristic Algorithms:* Generally, it is difficult, in particular for high-dimensional problems to build a reliable global surrogate model due to the "curse of dimensionality" [18], [41]. To alleviate this difficulty, local surrogate models are thus intensively investigated in order to enhance the accuracy of the surrogates. Ong *et al.* [25] employed a trust-region method for an interleaved use of exact models for the objective and constrained functions with computationally cheap RBF surrogate models during the local search. Martinize and Coello [42] introduced a local search algorithm assisted by surrogate models to accelerate the convergence of a multiobjective evolutionary algorithm [28]. Fitness inheritance [43] proposed by Smith *et al.* [44] in genetic algorithm, can also be seen as an ad hoc local surrogate technique, where the fitness of an individual is inherited (estimated) from its parents. Hendtlass [45] adopted a fitness inheritance strategies in PSO and added a reliability measure to enhance the accuracy of fitness estimation. Taking into account of the positional relationships between particles, Sun *et al.* [46] proposed a fitness estimation strategy (FES) for PSO (FESPSO) to approximate the fitness of a particle by estimating the fitness not only from its parents, but also its progenitors and siblings. The FESPSO was extended later on by introducing a similarity measure to further reduce computationally expensive FEs [47].

3) *Ensemble-Surrogate Assisted Metaheuristic Algorithms:* Compared to global-surrogate models, local-surrogate models are more likely to produce accurate fitness estimations. However, local surrogates are not able to help evolutionary algorithms escape from local optima, thereby losing one important potential benefit of surrogates known as "blessing of uncertainty" [41]. It has been shown that in some cases, approximation errors introduced by a global surrogate model may help smooth out local optima or filter out noise in the fitness function, thereby very effectively accelerating the search. To take advantage of such potential benefit of global surrogate models, an ensemble surrogate consisting of a local surrogate and a global surrogate was proposed and demonstrated to outperform a single surrogate in most cases. Tenne and Armfield [48] suggested a memetic optimization framework using variable global and local surrogate-models for optimization of expensive functions. Also within a framework of memetic algorithms, Georgopoulou and Giannakoglou [49] proposed to perform a low-cost pre-evaluation of candidate solutions using RBF networks in global search and the gradient-based refinement of promising solutions during the local search. In [50], a global surrogate model was proposed for better preoffspring selection, and a local surrogate model was used to approximate the fitness in local search. Zhou *et al.* [26], [51] proposed a hierarchical surrogate-assisted evolutionary algorithm in which a GP was used as a global surrogate to prescreen promising individuals and an RBF network was utilized as the local surrogate to assist the trust-region enabled gradient-based search strategy to accelerate convergence. Lim *et al.* [41] proposed to unify diverse surrogate models in the local search phase of memetic algorithm. The ensemble model was used to attain reliable and accurate fitness values while the global smoothing model was utilized to speed up evolutionary search by traversing through the multimodal landscape of complex problems. A two-layer surrogate-assisted PSO algorithm was proposed by Sun *et al.* [29], in which a global and a number of local surrogate models were employed for fitness approximation.

Despite the success of various surrogate techniques reported in the literature, most of these techniques have been verified only on low-dimensional problems, mainly because a large number of training samples are needed to build a sufficiently accurate surrogate for high-dimensional problems, which is often not affordable. To the best of our knowledge, the highest dimension of computationally expensive problems ever solved by surrogate-assisted metaheuristic algorithms is 50 [39], where the principal component analysis technique is used to reduce the input dimension of the surrogate. This paper aims to push the boundary of surrogate-assisted optimization techniques by proposing a surrogate-assisted cooperative swarm optimization algorithm (SA-COSO), for solving high-dimensional time-consuming optimization problems up to a dimension of 100. The SA-COSO consists of two cooperative PSO variants, one being a PSO with a constriction factor [52] and the other a social learning-based PSO (SL-PSO) [53]. These two PSO variants cooperate in such a way that a particle in the PSO learns not only from its personal and global best particles, but also from the global best of the SL-PSO, whereas the particles in the SL-PSO may learn also from promising

solutions contributed by the PSO. On the other hand, the SL-PSO aims to perform exploratory search on the global surrogate model, while the PSO, assisted mainly by a local FES, focuses on fast local search. The proposed SA-COSO method is expected to be able to achieve a good performance for high-dimensional computationally expensive optimization problems mainly for the following two reasons.

1) The SL-PSO algorithm has been demonstrated to be effective in finding global optima of large-scale optimization problems. Assisted by a surrogate model providing the global contour of the objective functions, the SL-PSO is able to quickly identify the region in which the global optimum is located.
2) Whilst it is very unlikely to train an adequately accurate surrogate model for high-dimensional expensive problems because of the curse of dimensionality, the FES is more scalable to high-dimensional problems.

The above hypotheses are verified by the promising empirical results reported in this paper.

The rest of this paper is organized as follows. Section II briefly reviews the related background techniques including the PSO with a constriction factor, the SL-PSO, RBF networks, and the FESPSO. In Section III, the proposed SA-COSO is presented in detail. Section IV empirically assesses the proposed SA-COSO on six commonly used benchmark problems of a dimension 50 and 100, respectively. Section V concludes this paper with a summary and some ideas for future work.

II. Related Techniques

A. Particle Swarm Optimization Variants

PSO was originally proposed by Eberhart and Kennedy [54] to solve optimization problems by simulating collective behaviors of social animals such as bird flocking and fish schooling. PSO has been successfully applied to a number of applications owing to its simplicity and attractive search efficiency. Over the past decades, numerous PSO variants have been proposed in order to enhance the performance of the canonical PSO on both exploratory and exploitative search [55], [56]. One category of the PSO variants focuses on enhancing exploitation (convergence) capability, while the other concentrates more on improving the exploration (diversity) capability of PSO.

PSO with an inertia weight [57] and PSO with a constriction factor [52] are two early yet very popular PSO variants proposed to improve the convergence performance. Comparative studies [58] have shown that the PSO with a constriction factor usually perform better than the PSO with an inertia weight in particular in terms of convergence, indicating that the PSO with a constriction factor is better suited for efficient local search.

Another variant of PSO focuses on improving the diversity of the swarm in order to escape from local optima. In these variants, the comprehensive learning PSO (CLPSO) [59], the competitive swarm optimizer [60], and the SL-PSO [53] showed better performance on preserving the diversity of the swarm and discouraging the premature convergence. Experimental results in [53] showed that SL-PSO has a higher computational efficiency in comparison with some representative PSO variants including CLPSO.

In this paper, we integrate two PSO variants, the PSO with a constriction factor (PSO) proposed in [52] shown to be efficient for local search and the SL-PSO algorithm [53] suited for global search, to work cooperatively for solving high-dimensional expensive optimization problems. Without loss of generality, we consider a class of minimization problems as follows:

$$\begin{aligned} &\text{minimize: } f(\mathbf{x}) \\ &\text{subject to: } \mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \mathcal{R}^D$ is a vector of continuous decision variables and D is the dimension of the search space. $f(\mathbf{x})$ is a scalar-valued objective function, $\mathbf{x}_l$ and $\mathbf{x}_u$ are vectors of the lower and upper bounds of search space, respectively.

1) Particle Swarm Optimization With Constriction Factor: For simplicity, we denote the PSO algorithm with a constriction factor as PSO. The PSO algorithm starts with a population of particles randomly positioned in the search space, each of which has its own velocity and position. At each iteration, the position and velocity of a particle in PSO are updated as follows:

$$\begin{aligned} v_{id}(t+1) = \chi\big(&v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) \\ &+ c_2 r_2 \left(p_{gd}(t) - x_{id}(t)\right)\big) \end{aligned} \tag{2}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{3}$$

where $1 \le i \le m$, m is the swarm size of the PSO algorithm, and $1 \le d \le D$. $\mathbf{v}_i(t) = (v_{i1}(t), v_{i2}(t), \ldots, v_{iD}(t))$ and $\mathbf{x}_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t))$ are the velocity and position of particle i at iteration t, respectively. $\mathbf{p}_i(t) = (p_{i1}(t), p_{i2}(t), \ldots, p_{iD}(t))$ is the best historical position found by particle i (known as the personal best position), $\mathbf{p}_g(t) = (p_{g1}(t), p_{g2}(t), \ldots, p_{gD}(t))$ is the best historical position of the swarm (known as the global best position), r_1 and r_2 are two uniformly generated random numbers in the range $[0, 1]$, c_1 and c_2 are positive constants called cognitive and social coefficients, respectively. χ is the constriction factor, with

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

$\phi = c_1 + c_2$. In general, $\phi > 4$ and therefore, c_1 and c_2 are usually set to 2.05. k is a real number in the range $(0, 1]$.

2) Social Learning-Based Particle Swarm Optimization: In SL-PSO, the particles are first sorted in an increasing order of the fitness, i.e., from the worst to the best. Each particle j, except for the best particle, i.e., $1 \le j \le n-1$, where n is the swarm size of the SL-PSO algorithm, learns from a randomly chosen particle whose fitness is better than that of particle j, known as the demonstrator. Then, the position of the jth particle will be updated as follows:

$$x_{jd}(t+1) = \begin{cases} x_{jd}(t) + \Delta x_{jd}(t+1) & \text{if } \mathrm{pr}_j(t) \le \mathrm{pr}_j^L \\ x_{jd}(t) & \text{otherwise} \end{cases} \tag{5}$$

with

$$\Delta x_{jd}(t+1) = r_1 \cdot \Delta x_{jd}(t) + r_2 \cdot \big(x_{kd}(t) - x_{jd}(t)\big) + r_3 \cdot \epsilon \cdot \big(\bar{x}_d(t) - x_{jd}(t)\big) \quad (6)$$

where pr_j, $0 \le \mathrm{pr}_j \le 1$, is a randomly generated probability and pr_j^L is the probability threshold for particle j to update its position, r_1, r_2, and r_3 are three random numbers uniformly generated in the range $[0, 1]$, x_{kd} represents the dth ($1 \le d \le D$) element of particle k whose fitness is better than $f(\mathbf{x}_j)$, $\bar{x}_d(t) = (\sum_{j=1}^{n} x_{jd}(t)/n)$ is the mean position value on dth dimension of the swarm, ϵ is a parameter called the social influence factor that controls the influence of $\bar{x}_d(t)$.

Note that we use different notations for denoting the size of the PSO and SL-PSO in that typically, PSO uses a relatively small swarm size such as 30, while the size of SL-PSO is bigger, for example 200.

B. RBF Networks

The idea of using RBF networks as an approximation function was first proposed by Hardy [61] to fit irregular topographical data. It has been shown that the performance of RBF networks is relatively insensitive to the increase in the dimension of the function to be approximated [62], [63]. In this paper, the RBF network is employed as a global surrogate model to assist SL-PSO to quickly find the region where the global optimum might be located.

Let $\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, 2, \ldots, N\}$ denote the dataset for training the RBF network. $\mathbf{x}_i \in \mathcal{R}^D$ and $f(\mathbf{x}_i) \in \mathcal{R}$ are the inputs and output, respectively, N is the number of training data. An RBF network is a real-valued function $\Phi : \mathcal{R}^D \to \mathcal{R}$. There are several types of RBFs, including Gaussian, splines and multiquadrics. In this paper, the following Gaussian function is used as the basis function:

$$\varphi(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|}{\sigma^2}\right). \quad (7)$$

So the surrogate model can be written in the following form:

$$\Phi(\mathbf{x}) = \omega_0 + \sum_{k=1}^{\mathrm{CN}} \omega_k \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_k\|}{\sigma^2}\right) \quad (8)$$

where $\sigma > 0$ is the width of the Gaussian function. CN is the number of RBFs, each being associated with a different center $\mathbf{c}_k$. $\omega_k (k = 1, 2, \ldots, \mathrm{CN})$ is the coefficient, and ω_0 is a bias term, which can be set to the mean of the values of the known data points from the training set that are used to train the surrogate model, or set to 0.

C. Fitness Estimation Strategy for Particle Swarm Optimization

A computationally simple yet effective fitness approximation technique based on the positional relationship between the particles of the PSO was proposed in [46] for computationally expensive optimization problems. According to (2) and (3), the position of particle i and particle j, $1 \le i, j \le m$, is updated, respectively, as follows:

$$\begin{aligned}\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \chi\big(&\mathbf{x}_i(t) - \mathbf{x}_i(t-1) \\ &+ c_1\mathbf{r}_{i1}(t+1)(\mathbf{p}_i(t) - \mathbf{x}_i(t)) \\ &+ c_2\mathbf{r}_{i2}(t+1)(\mathbf{p}_g(t) - \mathbf{x}_i(t))\big) \quad (9)\end{aligned}$$

$$\begin{aligned}\mathbf{x}_j(t+1) = \mathbf{x}_j(t) + \chi\big(&\mathbf{x}_j(t) - \mathbf{x}_j(t-1) \\ &+ c_1\mathbf{r}_{j1}(t+1)\big(\mathbf{p}_j(t) - \mathbf{x}_j(t)\big) \\ &+ c_2\mathbf{r}_{j2}(t+1)\big(\mathbf{p}_g(t) - \mathbf{x}_j(t)\big)\big). \quad (10)\end{aligned}$$

We then introduce a virtual position at iteration $t+1$, denoted $\mathbf{x}_v(t+1)$, by combining and rearranging (9) and (10)

$$\begin{aligned}\mathbf{x}_v(t+1) &= \mathbf{x}_i(t+1) + \chi\mathbf{x}_i(t-1) \\ &\quad + \big(1 + \chi\big(1 - c_1\mathbf{r}_{j1} - c_2\mathbf{r}_{j2}\big)\big)\mathbf{x}_j(t) \\ &\quad + \chi c_1\mathbf{r}_{j1}\mathbf{p}_j(t) + \chi c_2\mathbf{r}_{j2}\mathbf{p}_g(t) \\ &= \mathbf{x}_j(t+1) + \chi\mathbf{x}_j(t-1) \\ &\quad + (1 + \chi(1 - c_1\mathbf{r}_{i1} - c_2\mathbf{r}_{i2}))\mathbf{x}_i(t) \\ &\quad + \chi c_1\mathbf{r}_{i1}\mathbf{p}_i(t) + \chi c_2\mathbf{r}_{i2}\mathbf{p}_g(t). \quad (11)\end{aligned}$$

Therefore, the fitness of the virtual position can be calculated either by taking a weighted average of the fitness values of $f(\mathbf{x}_i(t+1))$, $f(\mathbf{x}_i(t-1))$, $f(\mathbf{x}_j(t))$, $f(\mathbf{p}_j(t))$, and $f(\mathbf{p}_g(t))$, or by taking a weighted average of the fitness values of $f(\mathbf{x}_j(t+1))$, $f(\mathbf{x}_j(t-1))$, $f(\mathbf{x}_i(t))$, $f(\mathbf{p}_i(t))$, and $f(\mathbf{p}_g(t))$. Since these two fitness values should be the same, we can establish a relationship between $f(\mathbf{x}_i(t+1))$ and $f(\mathbf{x}_j(t+1))$ as follows:

$$f\big(\mathbf{x}_j(t+1)\big) = d_j(t+1) \cdot WF \quad (12)$$

where

$$\begin{aligned}WF = P_a\bigg(&\frac{1}{d_i(t+1)}f(\mathbf{x}_i(t+1)) + \frac{1}{d_i(t-1)}f(\mathbf{x}_i(t-1)) \\ &+ \frac{1}{d_j(t)}f\big(\mathbf{x}_j(t)\big) + \frac{1}{d_{pj}(t)}f\big(\mathbf{p}_j(t)\big) + \frac{1}{d_g(t)}f\big(\mathbf{p}_g(t)\big)\bigg) \\ &- \frac{1}{d_j(t-1)}f\big(\mathbf{x}_j(t-1)\big) - \frac{1}{d_i(t)}f(\mathbf{x}_i(t)) \\ &- \frac{1}{d_{pi}(t)}f(\mathbf{p}_i(t)) - \frac{1}{d_g(t)})f\big(\mathbf{p}_g(t)\big) \quad (13)\end{aligned}$$

and

$$P_a = \frac{\frac{1}{d_j(t+1)} + \frac{1}{d_j(t-1)} + \frac{1}{d_i(t)} + \frac{1}{d_{pi}(t)} + \frac{1}{d_g(t)}}{\frac{1}{d_i(t+1)} + \frac{1}{d_i(t-1)} + \frac{1}{d_j(t)} + \frac{1}{d_{pj}(t)} + \frac{1}{d_g(t)}} \quad (14)$$

where $d_i(t+1)$, $d_i(t-1)$, $d_j(t)$, $d_{pj}(t)$, $d_j(t+1)$, $d_j(t-1)$, $d_i(t)$, $d_{pi}(t)$, and $d_g(t)$ represent the distance between the virtual position $\mathbf{x}_v(t+1)$ and $\mathbf{x}_i(t+1)$, $\mathbf{x}_i(t-1)$, $\mathbf{x}_j(t)$, $\mathbf{p}_j(t)$, $\mathbf{x}_j(t+1)$, $\mathbf{x}_j(t-1)$, $\mathbf{x}_i(t)$, $\mathbf{p}_i(t)$, and $\mathbf{p}_g(t)$, respectively. We can find that if the fitness of particle i at the iteration $t+1$ ($f(\mathbf{x}_i(t+1))$) is known, then the fitness of particle j ($f(\mathbf{x}_j(t+1))$) can be approximated by (12), and vice versa. Note that the fitness of a virtual position does not always need

to be calculated. A more detailed description of the FES can be found in [46].

III. Surrogates-Assisted Cooperative Swarm Optimization

Multiswarm algorithms have shown to be effective in striking a good balance between exploration and exploitation [64]. Based on these findings, this paper proposes a surrogate-assisted cooperative swarm optimization algorithm, termed SA-COSO, by integrating a PSO [52] assisted by a FES and a surrogate assisted SL-PSO [53] for solving computationally expensive high-dimensional optimization problems. In SA-COSO, SL-PSO focuses on exploration while PSO concentrates on exploitation. To reduce the number of expensive FEs, an RBF network is adopted as a surrogate model capturing the global profile of the fitness landscape to assist SL-PSO to quickly find the region where the global optimum is located, whereas an FES is employed as a local estimation method to help the PSO perform local search.

In the following, we start with introducing general framework of the proposed SA-COSO algorithm, which integrates the FES assisted PSO (FES-assisted PSO) and RBF-assisted SL-PSO algorithms. Then, we describe in detail the fitness estimation algorithm in PSO and the surrogate management strategy in SL-PSO.

A. Coupling Between FES-Assisted PSO and RBF-Assisted SL-PSO in SA-COSO

Fig. 1 depicts the coupling between the FES-assisted PSO and RBF-assisted SL-PSO in the proposed SA-COSO. In Fig. 1, *DB* represents an archive for storing the positions (decision variables) and their corresponding fitness values evaluated using the computationally expensive real objective function. All data stored in *DB* will be utilized to train the RBF network for fitness approximation for all individuals in both FES-assisted PSO and RBF-assisted SL-PSO. Note that in the FESPSO presented in [46], the fitness of a particle is always evaluated using the real objective function if the condition for using the FES is not satisfied. In the FES-assisted PSO in this paper, however, the RBF network will usually be used for calculating the fitness of a particle if the condition for using the FES is not met to further reduce the expensive FEs. The real fitness function is used only if the estimated fitness of a particle is potentially promising, i.e., if it is better than the current personal best, or the estimated fitness has a large degree of uncertainty. Although the RBF surrogate is also involved, the fitness of most particles in the PSO is estimated using the FES. For simplicity, we term the PSO assisted mainly by the FES and sometimes also by the RBF network *FES-assisted PSO*. $g\text{best}_{\text{SL-PSO}}$ in Fig. 1 is the global best position found by the RBF-assisted SL-PSO. In the search process of PSO, the individuals not only learn from its own personal best position and the global best position of its own population, but also from the global best position obtained by the RBF-assisted SL-PSO in order to avoid premature convergence into a local optimum, since SL-PSO is meant for global search on a global

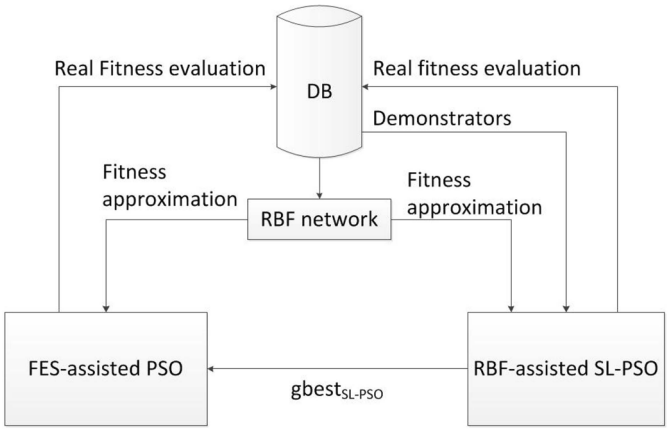

Fig. 1. Coupling between FES-assisted PSO and RBF-assisted SL-PSO in SA-COSO.

surrogate, i.e., the RBF network. Therefore, each individual in the FES-assisted PSO will update its velocity according to the following equation:

$$\begin{aligned} v_{id}(t+1) = \chi\big(&v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) \\ &+ c_2 r_2 \big(p_{gd}(t) - x_{id}(t)\big) \\ &+ c_3 r_3 \big(p_{rg,d}(t) - x_{id}(t)\big)\big) \end{aligned} \tag{15}$$

where $1 \le i \le m$, c_1, c_2, r_1, and r_2 are the same as defined in (2), c_3 is a positive constant also known as the social learning parameter, r_3 is a random number uniformly generated in the range $[0, 1]$, $\mathbf{p}_{rg} = (p_{rg,1}, p_{rg,2}, \ldots, p_{rg,D})$ is the global best position obtained by the RBF-assisted SL-PSO ($g\text{best}_{\text{SL-PSO}}$).

As we can see from Fig. 1, those solutions in both PSO and SL-PSO re-evaluated using the real fitness function are saved in the archive *DB*. To prevent the population from being misled by the errors introduced by the approximated fitness values, and to enhance the diversity of the swarm of the RBF-assisted SL-PSO, we also randomly choose n solutions (n is the swarm size of SL-PSO) from the archive so that the RBF-assisted SL-PSO can also use solutions in the *DB* as demonstrators. Consequently, the size of *DB* should be larger than n, the swarm size of SL-PSO. Therefore, the range of two parameters "j" and "k" should be modified in (6), where $1 \le j \le n$, $k \in K_j$, K_j is a subset of the union of n solutions in the current SL-PSO and n solutions randomly chosen from *DB* whose fitness values are better than that of the jth particle to be updated. Please note that if no other solution is better than particle j, then the original position of this particle will be kept and participate in the evolution in the next generation.

Algorithm 1 presents the pseudocode of the main components of SA-COSO. In Algorithm 1, $g$best represents the final output of global best position found by two swarm optimization algorithms and $g\text{best}_{\text{PSO}}$ is the global best position found by FES-assisted PSO.

In the following, we present the details of the FESPSO and the surrogate-management in SL-PSO, including training of the RBF network and update of the archive (*DB*).

Algorithm 1 Pseudocode of SA-COSO

1: Initialize a population pop_{PSO}: including velocity and position initialization, fitness evaluation using the real objective function, and assigning the position of each particle to its personal best position;
2: Initialize a population $pop_{SL\text{-}PSO}$: including position initialization, fitness evaluation using the real objective function;
3: Save positional information of all particles in an archive *DB*, and train an RBF network using these data;
4: $t = 0$;
5: **repeat**
6: Determine the global best position of the swarm pop_{PSO} ($gbest_{PSO}$) and of the swarm pop_{SL-PSO} ($gbest_{SL-PSO}$);
7: $gbest = \min\{gbest_{PSO}, gbest_{SL-PSO}\}$;
8: Run **FES-assisted PSO;**
9: Run **RBF-assisted SL-PSO;**
10: **Update of the archive** *DB***;**
11: Retrain a global RBF network using the data in the *DB*;
12: $t = t + 1$;
13: **until** the terminal condition is satisfied
14: Output *gbest*;

Algorithm 2 Pseudocode of the FES-Assisted PSO

1: Update velocity and position of each particle using Eq. (15) and Eq. (3), respectively;
2: Call the procedure to determine the fitness value of each particle; (see **Algorithm 3**)
3: Determine the personal best position of each particle (see **Algorithm 4**);
4: Call the procedure to determine the global best position; (see **Algorithm 5**)

Algorithm 3 Fitness Determination

1: **if** $t < 2$ **then**
2: Evaluate the fitness of each particle using the real objective function;
3: Save all particles into a temporary archive DB_t;
4: **else**
5: **for** $i = 1$ to m **do**
6: Set a label showing that the fitness of the *i*-th particle has not been estimated using the fitness estimation strategy;
7: Approximate the fitness of particle i using RBF network, denoted as $\widehat{f}_{RBF}(\mathbf{x}_i)$;
8: **end for**
9: **for** $i = 1$ to m **do**
10: **if** the fitness of the *i*-th particle has not been estimated using the fitness estimation strategy **then**
11: $f(\mathbf{x}_i) = \widehat{f}_{RBF}(\mathbf{x}_i)$;
12: **end if**
13: Find the nearest neighbor j of particle i, $\mathbf{x}_j \neq \mathbf{x}_i$;
14: **if** $j > i$ **then**
15: **if** the fitness of the *j*-th particle has not been estimated using the fitness estimation strategy **then**
16: $f(\mathbf{x}_j) = \widehat{f}_{FES}(\mathbf{x}_j)$;
17: Set a label showing that the fitness of the *j*-th particle has been estimated using the fitness estimation strategy;
18: **else**
19: $f(\mathbf{x}_j) = \min\{f(\mathbf{x}_j), \widehat{f}_{FES}(\mathbf{x}_j)\}$;
20: **end if**
21: **end if**
22: **end for**
23: **end if**

B. FES-Assisted PSO

Algorithm 2 lists the pseudocode of the main steps of the FES-assisted PSO, which is similar to a canonical PSO. Algorithm 3 describes the procedure for determining the fitness value of each particle. In Algorithms 3, $\widehat{f}_{\mathrm{FES}}(\mathbf{x})$ and $\widehat{f}_{\mathrm{RBF}}(\mathbf{x})$ denote a fitness value approximated by the FES and the RBF network, respectively.

In the first two iterations (lines 1–3 in Algorithm 3), all individuals in the FES-assisted PSO are evaluated using the computationally expensive real objective function. This is necessary as the FES requires the fitness values of all particles in the previous two iterations. In addition, the RBF network needs to be trained before it can be used for fitness approximation. An archive "DB_t" is used to temporarily save the particles (the position and its corresponding fitness value) calculated using the real function evaluations, which will be used to update the archive *DB* later on. The details for updating *DB* will be presented in Section III-D.

From the third iteration onward, the following procedure will be undertaken to evaluate the fitness of all particles (lines 5–22). The fitness of all particles in the current swarm will first be approximated by the RBF network and saved as $\widehat{f}_{\mathrm{RBF}}(\mathbf{x}_i), i = 1, 2, \ldots, m$, where m is the swarm size of the FES-assisted PSO (lines 5–8). Then, for the ith particle, if the fitness of the ith particle has not been estimated using the FES till now, the value approximated by the RBF network will be adopted as its fitness value (lines 10–12). Once the fitness of the *i*-particle is estimated using the RBF network, the particle that is closest to particle i, say, particle j, will be estimated using the FES). Note that j must be larger than i to avoid endless loops in fitness estimation. If the fitness of jth particle has not been approximated using the FES either, the fitness approximated by the FES is assigned to particle j and a label is set to show that the fitness of jth particle has been approximated using the FES. If, however, the fitness of the jth particle has already been estimated by another particle using the FES, then the current fitness value $f(\mathbf{x}_j)$ will be replaced by the fitness value estimated by particle i in case the fitness value estimated according to particle i is better than $f(\mathbf{x}_j)$. The above procedure repeats until the fitness of all particles are determined.

We can see from Algorithm 3 that the method for fitness determination in the FES-assisted PSO algorithm is almost the same as the one in [46] except that when the fitness of a particle has not yet been estimated using the FES, the fitness approximated by the RBF network will be adopted as the "real" fitness value to reduce the number of FEs. This might

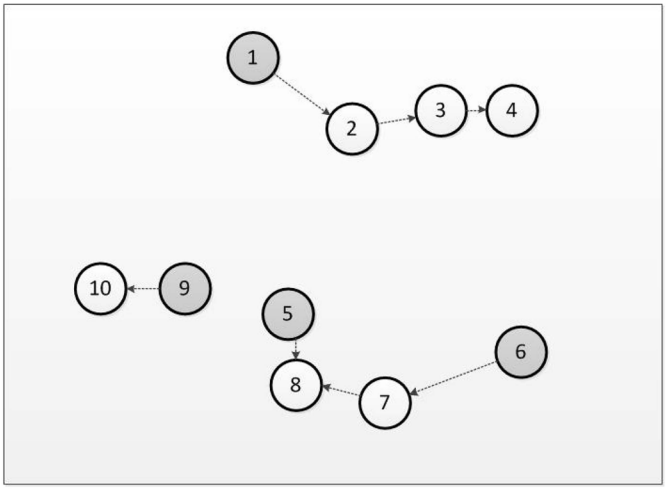

Fig. 2. Illustrative example of fitness determination using the RBF network and the FES for the FES-assisted PSO. A shaded circle denotes a particle whose fitness is calculated using the RBF network, while a blank circle stands for one whose fitness is estimated using the FES.

introduce a risk of degrading the accuracy of the estimated fitness and thereby misleading the FES-assisted PSO, however, this risk is reduced in that here the fitness value obtained by the RBF network is used as a "reference," and as soon as there is a large deviation between the fitness approximated by the RBF and the fitness estimated using the FES, the fitness of the particle will be re-evaluated using the real objective function.

Fig. 2 gives an example illustrating the fitness determination method described above. In the example, there are ten particles in the population, each being represented by a numbered circle. In the figure, an arrow in dotted line is utilized to point from a particle whose fitness is known to one whose fitness is approximated using the FES. In addition, a shaded circle indicates that the fitness of this particle is calculated using the RBF network, while a blank circle represents a particle whose fitness is estimated using the FES.

The fitness of each particle will be determined sequentially according to the number starting from particle 1. The fitness of particle 1 adopts the fitness value estimated by the RBF network by default. Assume particle 2 is the closest neighbor of particle 1, and since the condition $2 > 1$ is satisfied, the fitness of particle 2 will be estimated using the FES based on the fitness of particle 1. A label will be set to indicate that the fitness of particle 2 has been calculated using the FES. Then we proceed to determine the fitness of particle 2. As the fitness value of particle 2 is known, the fitness approximated by the FES will be used as its fitness. Now the fitness of particle 3, under the assumption that it is the closest neighbor of particle 2, will be estimated using the FES based on fitness of particle 2 and a label is set to indicate that the fitness of particle 3 has been calculated using the FES. This process continuous until the fitness of all particles are determined. Note, however, that the fitness of particle 8 is estimated using the FES according to the fitness of particle 5 as it is the closest neighbor of particle 5, and later on, its fitness can be estimated again according to the fitness of particle 7, as it is also the closest particle of particle 7. In this case, the fitness of particle 8 will be updated only if the fitness estimated by particle 7 is better than that estimated by particle 5. Note also that the fitness of particle 5 will not be re-estimated by particle 8 since 8 is larger than 5. In this way, endless loops can be avoided in fitness determination.

Algorithm 4 Personal Best Determination for FES-Assisted PSO

1: **for** $i = 1$ to m **do**
2: **if** $f(\mathbf{x}_i) = \widehat{f}_{RBF}(\mathbf{x}_i)$ **then**
3: **if** $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
4: $\mathbf{p}_i = \mathbf{x}_i$;
5: **end if**
6: **else**
7: **if** $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ and $\widehat{f}_{RBF}(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
8: Evaluate the fitness of particle i using the real objective function;
9: Save it into DB_t;
10: **if** $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
11: $\mathbf{p}_i = \mathbf{x}_i$;
12: **end if**
13: **end if**
14: **end if**
15: **end for**
16: **if** no particle's fitness is evaluated using the real objective function in current population **then**
17: Calculate the mean difference between two values approximated by RBF network and FES method using Eq. (16);
18: **for** $i = 1$ to m **do**
19: **if** $|f(\mathbf{x}_i) - \widehat{f}_{RBF}(\mathbf{x}_i)| > DF$ **then**
20: Evaluate the fitness of particle i using the real objective function;
21: Save it into DB_t;
22: **if** $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
23: $\mathbf{p}_i = \mathbf{x}_i$;
24: **end if**
25: **end if**
26: **end for**
27: **end if**

The next step is to update the personal best of all particles, which is described in Algorithm 4. The personal best of each particle will be replaced if its fitness value in the current iteration is calculated using the RBF network and is better than its personal best (lines 2–5). However, if the current fitness value is estimated using the FES and is better than the personal best, we will also compare the fitness estimated using the RBF network with the personal best. If both fitness values of the particle, one estimated using the FES and the other calculated using the RBF network, are better than the personal best, we will verify the fitness of this particle using the real objective function. So eventually, the personal best of this particle will be updated only if its real fitness value is better than the personal best (lines 7–13).

From the above description, we can see that a particle will be evaluated using the real fitness function only if both its fitness values estimated using the RBF network and using

Algorithm 5 Global Best Position Determination for FES-Assisted PSO

1: Find the best position ($best_{PSO}$) in personal best positions of all individuals in the pop_{PSO};
2: **if** $f(best_{PSO}) < f(gbest_{PSO})$ **then**
3: **if** $f(best_{PSO})$ is calculated using the real objective function **then**
4: Replace $gbest_{PSO}$ with $best_{PSO}$;
5: **else**
6: Evaluate the fitness of $best_{PSO}$ using the real objective function;
7: Save it into DB_t;
8: **if** $f(best_{PSO}) < f(gbest_{PSO})$ **then**
9: Replace $gbest_{PSO}$ with $best_{PSO}$;
10: **end if**
11: **end if**
12: **end if**

the FES are better than its personal best. If this situation does not occur, no particle in the current iteration will be estimated using the real fitness function, which is undesirable. To avoid false convergence, i.e., the PSO converges to a minimum of the surrogate that is not an optimum of the original fitness function, we will re-evaluate the particles if their fitness value estimated using the FES has a degree of uncertainty larger than the average. The average degree of uncertainty of the estimated fitness is defined as the average difference between the fitness calculated by the RBF network and the FES

$$DF = \sum_{i=1}^{m} \left| f(\mathbf{x}_i) - \widehat{f}_{\mathrm{RBF}}(\mathbf{x}_i) \right| / m. \tag{16}$$

Note that if the fitness of a particle is calculated using the RBF network, the difference will be 0. For particle i, if $|f(\mathbf{x}_i) - \widehat{f}_{\mathrm{RBF}}(\mathbf{x}_i)|$ is larger than the mean difference DF, it will be re-evaluated using the real fitness function and the personal best position of this particle will be updated only if the fitness value is better than the personal best (lines 16–27 in Algorithm 4).

Finally, the global best of the FES-assisted PSO needs to be updated, which is described in Algorithm 5. The main point here is that if the new global best is estimated using the RBF network or the FES, it will be re-evaluated using the real fitness function and replaces the current global best if the fitness value using the real fitness function is indeed better.

The FES described in Algorithm 3 (lines 14–21) is proposed to approximate the fitness of the closest neighbor of particle i once its fitness is known. Fitness estimation based on the positional relationships between the particles has been demonstrated to be an effective approach to reducing the number of FEs for expensive optimization problems [46]. As the PSO in SA-COSO is closely coupled with the SL-PSO, the mechanism for updating the velocity has been slightly modified as described in (15). Thus, the FES proposed in [46] must be adapted accordingly. Similar to (9) and (10), we can rewrite the equations for updating the position of particle i and particle j ($i, j \in \{1, 2, \ldots, m, i \neq j\}$), respectively, according to (3) and (15) as follows:

$$\begin{aligned}
\mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \chi\big((\mathbf{x}_i(t) - \mathbf{x}_i(t-1)) + c_1\mathbf{r}_{i1}(\mathbf{p}_i(t) - \mathbf{x}_i(t)) \\
&\quad + c_2\mathbf{r}_{i2}\big(\mathbf{p}_g(t) - \mathbf{x}_i(t)\big) \\
&\quad + c_3\mathbf{r}_{i3}(\mathbf{p}_{rg}(t) - \mathbf{x}_i(t))\big) \\
&= (1 + \chi(1 - c_1\mathbf{r}_{i1} - c_2\mathbf{r}_{i2} - c_3\mathbf{r}_{i3}))\mathbf{x}_i(t) \\
&\quad - \chi\mathbf{x}_i(t-1) + \chi c_1\mathbf{r}_{i1}\mathbf{p}_i(t) \\
&\quad + \chi c_2\mathbf{r}_{i2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{i3}\mathbf{p}_{rg}(t) \qquad (17) \\
\mathbf{x}_j(t+1) &= \mathbf{x}_j(t) + \chi\big(\big(\mathbf{x}_j(t) - \mathbf{x}_j(t-1)\big) \\
&\quad + c_1\mathbf{r}_{j1}\big(\mathbf{p}_j(t) - \mathbf{x}_j(t)\big) \\
&\quad + c_2\mathbf{r}_{j2}\big(\mathbf{p}_g(t) - \mathbf{x}_j(t)\big) \\
&\quad + c_3\mathbf{r}_{j3}\big(\mathbf{p}_{rg}(t) - \mathbf{x}_j(t)\big)\big) \\
&= \big(1 + \chi\big(1 - c_1\mathbf{r}_{j1} - c_2\mathbf{r}_{j2} - c_3\mathbf{r}_{j3}\big)\big)\mathbf{x}_j(t) \\
&\quad - \chi\mathbf{x}_j(t-1) + \chi c_1\mathbf{r}_{j1}\mathbf{p}_j(t) \\
&\quad + \chi c_2\mathbf{r}_{j2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{j3}\mathbf{p}_{rg}(t). \qquad (18)
\end{aligned}$$

By combining and rearranging (17) and (18), we can introduce a virtual position

$$\begin{aligned}
\mathbf{x}_v(t+1) &= \mathbf{x}_i(t+1) + \chi\mathbf{x}_i(t-1) \\
&\quad + \big(1 + \chi\big(1 - c_1\mathbf{r}_{j1} - c_2\mathbf{r}_{j2} - c_3\mathbf{r}_{j3}\big)\big)\mathbf{x}_j(t) \\
&\quad + \chi c_1\mathbf{r}_{j1}\mathbf{p}_j(t) + \chi c_2\mathbf{r}_{j2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{j3}\mathbf{p}_{rg}(t) \\
&= \mathbf{x}_j(t+1) + \chi\mathbf{x}_j(t-1) \\
&\quad + (1 + \chi(1 - c_1\mathbf{r}_{i1} - c_2\mathbf{r}_{i2} - c_3\mathbf{r}_{i3}))\mathbf{x}_i(t) \\
&\quad + \chi c_1\mathbf{r}_{i1}\mathbf{p}_i(t) + \chi c_2\mathbf{r}_{i2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{i3}\mathbf{p}_{rg}(t).
\end{aligned} \tag{19}$$

Consequently, the fitness of the virtual position can be approximated using the weighted average of $f(\mathbf{x}_i(t+1))$, $f(\mathbf{x}_i(t-1))$, $f(\mathbf{x}_j(t))$, $f(\mathbf{p}_j(t))$, $f(\mathbf{p}_g(t))$, and $f(\mathbf{p}_{rg}(t))$ or of $f(\mathbf{x}_j(t+1))$, $f(\mathbf{x}_j(t-1))$, $f(\mathbf{x}_i(t))$, $f(\mathbf{p}_i(t))$, $f(\mathbf{p}_g(t))$, and $f(\mathbf{p}_{rg}(t))$ in the following form:

$$f(\mathbf{x}_v(t+1)) = \frac{\mathrm{WS}_1}{\mathrm{WD}_1} = \frac{\mathrm{WS}_2}{\mathrm{WD}_2} \tag{20}$$

where

$$\begin{aligned}
\mathrm{WS}_1 &= \frac{f(\mathbf{x}_i(t+1))}{d_i(t+1)} + \frac{f(\mathbf{x}_i(t-1))}{d_i(t-1)} + \frac{f\big(\mathbf{x}_j(t)\big)}{d_j(t)} \\
&\quad + \frac{f\big(\mathbf{p}_j(t)\big)}{d_{pj}(t)} + \frac{f\big(\mathbf{p}_g(t)\big)}{d_g(t)} + \frac{f\big(\mathbf{p}_{rg}(t)\big)\big)}{d_{rg}(t)} \qquad (21) \\
\mathrm{WD}_1 &= \frac{1}{d_i(t+1)} + \frac{1}{d_i(t-1)} + \frac{1}{d_j(t)} + \frac{1}{d_{pj}(t)} \\
&\quad + \frac{1}{d_g(t)} + \frac{1}{d_{rg}(t)} \qquad (22) \\
\mathrm{WS}_2 &= \frac{f\big(\mathbf{x}_j(t+1)\big)}{d_j(t+1)} + \frac{f\big(\mathbf{x}_j(t-1)\big)}{d_j(t-1)} + \frac{f(\mathbf{x}_i(t))}{d_i(t)} \\
&\quad + \frac{f(\mathbf{p}_i(t))}{d_{pi}(t)} + \frac{f\big(\mathbf{p}_g(t)\big)}{d_g(t)} + \frac{f\big(\mathbf{p}_{rg}(t)\big)\big)}{d_{rg}(t)} \qquad (23) \\
\mathrm{WD}_2 &= \frac{1}{d_j(t+1)} + \frac{1}{d_j(t-1)} + \frac{1}{d_i(t)} + \frac{1}{d_{pi}(t)} \\
&\quad + \frac{1}{d_g(t)} + \frac{1}{d_{rg}(t)} \qquad (24)
\end{aligned}$$

where $d_i(t+1)$, $d_i(t-1)$, $d_j(t)$, $d_{pj}(t)$, $d_j(t+1)$, $d_j(t-1)$, $d_i(t)$, $d_{pi}(t)$, $d_g(t)$, and $d_{rg}(t)$ are the distances between the virtual position $\mathbf{x}_v(t+1)$ and $\mathbf{x}_i(t+1)$, $\mathbf{x}_i(t-1)$, $\mathbf{x}_j(t)$, $\mathbf{p}_j(t)$, $\mathbf{x}_j(t+1)$, $\mathbf{x}_j(t-1)$, $\mathbf{x}_i(t)$, $\mathbf{p}_i(t)$, $\mathbf{p}_g(t)$, and $\mathbf{p}_{rg}(t)$, respectively. Here, all distances are Euclidean distance.

Seen from (20) to (24), the relationship between the fitness values of $f(\mathbf{x}_i(t+1))$ and $f(\mathbf{x}_j(t+1))$ can be established as follows:

$$\hat{f}_{\mathrm{FES}}\big(\mathbf{x}_j(t+1)\big) = d_j(t+1)\cdot WF_{\mathrm{new}} \tag{25}$$

where

$$WF_{\mathrm{new}} = \frac{\mathrm{WD}_1 * \mathrm{WS}_1}{\mathrm{WD}_2} - \frac{f\big(\mathbf{x}_j(t-1)\big)}{d_j(t-1)} - \frac{f(\mathbf{x}_i(t))}{d_i(t)} - \frac{f(\mathbf{p}_i(t))}{d_{pi}(t)} - \frac{f\big(\mathbf{p}_g(t)\big)}{d_g(t)} - \frac{f\big(\mathbf{p}_{rg}(t)\big)}{d_{rg}(t)}. \tag{26}$$

Equation (25) can be used to estimate the fitness of any particle j in the swarm whose fitness has not yet been estimated using the FES, provided that the fitness of particle i in the same iteration is known.

C. RBF-Assisted SL-PSO

In the SL-PSO algorithm, a particle (termed imitator) learns from the behaviors of different particles in the current swarm that have better fitness values (termed demonstrators) than the imitator. In this paper, an RBF network is used to learn the global profile of the fitness landscape and therefore the fitness values of all particles in SL-PSO are estimated using the RBF network. Due to the fitness estimation errors introduced by the RBF network, the real fitness values of the demonstrators may be actually worse than the imitator. To avoid false convergence of the SL-PSO, n particles stored in the *DB* will be randomly chosen as potential demonstrators for updating the particles in the current swarm. Note that all particles stored in *DB* are evaluated using the real fitness function.

Once the position and velocity of all particles are updated, their fitness value will be estimated using the RBF network and update the personal best of each particle accordingly. If the best particle (according to the RBF network) is better than the current global best, the best particle is re-evaluated using the real fitness function. If the real fitness value of this particle is indeed better than the current global best, replace the global best with the best particle.

Algorithm 6 gives the pseudocode of the RBF-assisted SL-PSO. Note that in each iteration of the surrogate-assisted SL-PSO, at most one FE using the real objective function will be conducted.

Algorithm 6 Pseudocode of the RBF-Assisted SL-PSO

```
for i = 1 to n do
    Find its demonstrators from current swarm and demonstrators drawn from archive DB;
    Update its position using Eqs. (6) and (5);
    Estimate the fitness of particle i using the RBF network;
end for
Find the best position (best_{SL-PSO}) in all individuals in the current population;
if \hat{f}_{RBF}(best_{SL-PSO}) < f(gbest_{SL-PSO}) then
    Evaluate the fitness of best_{SL-PSO} using the real objective function;
    Save the position and corresponding fitness in the temporary archive DB_t;
    if f(best_{SL-PSO}) < f(gbest_{SL-PSO}) then
        Replace gbest_{SL-PSO} with best_{SL-PSO};
    end if
end if
```

D. Update the Archive DB

The archive *DB* is used to store the particles evaluated using the real fitness function, which plays an important role in model management in SA-COSO. The stored particles are used not only to serve as demonstrators in SL-PSO, but also to train the RBF network. Note that the RBF network is meant to serve as a global surrogate and the SL-PSO is supposed to perform global search. To this end, the samples for training the RBF network must be properly selected to ensure that the RBF network can model a slightly larger region than that covered by the SL-PSO, but still be most relevant to the current swarm. In addition, it is not desirable to use all particles evaluated using the real objective function to train the RBF network in order to reduce the computational time.

Recall that in each iteration, all particles re-evaluated using the real fitness function are saved in the temporary DB_t. In our method, whether a solution in DB_t is to be selected and put into *DB* mainly depends on the distance between this solution and those in $\mathrm{pop}_{\mathrm{SL\text{-}PSO}}$. Fig. 3 gives an example illustrating the strategy for updating the archive *DB*. In the figure, the horizontal axis represents the decision space and the vertical axis is the fitness value. The circles denote particles (data pairs) in *DB*, the two vertical dashed lines indicate the region in which the current swarm $\mathrm{pop}_{\mathrm{SL-PSO}}$ is located, and the triangles represent the particles that have been evaluated using the real objective function and stored in the temporary database DB_t. Assume there are three newly evaluated particles in DB_t, denoted by triangles 1–3, respectively. Among them, we can see that particle 3 is far away from the current location of the population $\mathrm{pop}_{\mathrm{SL-PSO}}$. If we add this data into *DB*, it will have little influence on the surrogate that covers the region of the current population. By contrast, if we include solution 1 or solution 2 into the database, the quality of the surrogate model will be improved more effectively.

Since the size of *DB* is fixed, the next question is whether the particles in DB_t should replace those in *DB* if *DB* is already full. An intuitive idea is to discard those particles that are far away from the location of the current swarm. For example in Fig. 3, if we must discard one particle in the archive *DB* for including a newly evaluated particle in DB_t, say the one denoted by triangle 1, we can see that the particle denoted by red circle "a" is most likely to be removed.

Note that the distance to the current population of SL-PSO is the only criterion for our method to determine whether a solution in the temporary DB_t will be saved in the archive or whether a solution will be discarded from *DB*. One question that may arise is whether a discarded solution is better than

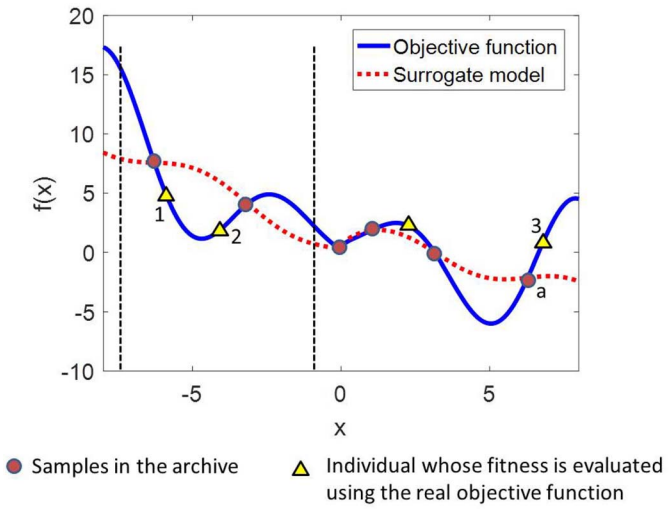

Fig. 3. Example to show the strategy for updating the archive *DB* using particles in DB_t.

the best solution found so far. This is very unlikely due to the following reasons. First, a discarded solution will not be better than the best solution of the RBF-assisted SL-PSO algorithm, because if it is, this solution must be an individual in the current population, and therefore it should not locate far away from the current population $\text{pop}_{\text{SL-PSO}}$. Second, this solution should not be the global optimum of the FES-assisted PSO, because as a local search algorithm, the population of the FES-assisted PSO is expected to be within or not far away from the region in which SL-PSO is in. Thus, discarding a solution that is far away from the current location of the SL-PSO should not lead to loss of important information for training the RBF network. Nevertheless, it might also be beneficial to store the data in a separate database as these data are computationally expensive and could be useful for future use in training the surrogate.

We must emphasize that the RBF network is a "global" surrogate relative to the decision space the current swarm is searching, rather than a global model that aims to account for the whole decision space. This can be clearly seen from the above description of the strategies for updating the training data in *DB*.

Algorithm 7 gives the pseudocode to update the data in the archive. Note that all distances in this paper are calculated using the Euclidean distance. The minimum distance to $\text{pop}_{\text{SL\_PSO}}$ denotes the minimum distance between the position of a solution stored in *DB* and the position of all individuals in the current population of SL-PSO.

IV. Experimental Results and Discussion

To investigate the effectiveness of the proposed SA-COSO, we conducted an empirical study on six widely used unimodal and multimodal benchmark problems. The characteristics of these test problems are listed in Table I. To the best of our knowledge, the largest dimension of computationally expensive optimization problems ever been handled using surrogate-assisted metaheuristic algorithms is 50, as reported in [39], in which a dimension reduction strategy has been used. In order to evaluate the effectiveness of SA-COSO for solving high-dimensional expensive optimization problems, we perform a set of experiments on 50-D and 100-D test problems listed in Table I and compare the performance of SA-COSO with that of PSO, FESPSO, SL-PSO, RBF-assisted SL-PSO, and COSO. Refer to Table II for the definition of the algorithms investigated here. Among them, FESPSO and RBF-assisted SL-PSO are two surrogate assisted particle swarm algorithms, while the rest are not. All experimental results are obtained over 20 independent runs in MATLAB R2014b.

Algorithm 7 Updating Archive *DB*

1: **for** each data i in the archive *DB* **do**
2: Find the minimum distance to the pop_{SL-PSO}, denoted by $DB\_pop_{\min}(i)$;
3: **end for**
4: Find the maximum distance in all minimum distance $\max(DB\_pop_{min})$;
5: **for** each particle k in the DB_t **do**
6: **if** the position that particle k occupies has not been saved in the archive **then**
7: **if** the archive is not full **then**
8: Save the position and corresponding fitness of particle k in *DB*;
9: **else**
10: Find the minimum distance to the the pop_{SL-PSO}, denoted by $ind\_pop_{\min}(k)$;
11: **if** the minimum distance is less than $\max(DB\_pop_{\min})$ **then**
12: Replace the particle that has the maximum value of $DB\_pop_{\min}$ with particle k in the DB_t;
13: **end if**
14: **end if**
15: **end if**
16: **end for**

TABLE I
Characteristics of Six Benchmark Problems

Benchmark Problem	Description	Characteristics	Global Optimum ($f(\vec{x}^*)$)
F1 [39]	Ellipsoid	Unimodal	0.0
F2 [41], [39]	Rosenbrock	Multimodal with narrow valley	0.0
F3 [41], [39]	Ackley	Multimodal	0.0
F4 [41], [39]	Griewank	Multimodal	0.0
F5 [41], [39]	Shifted Rotated Rastrigin	Very complicated multimodal	-330.0
F6 [41], [39]	Rotated hybrid Composition Function	Very complicated multimodal	120.0

A. Parameter Settings

The sizes of pop_{PSO} and $\text{pop}_{\text{SL-PSO}}$ in our algorithm are set to 30 and 200, respectively. The experimental results of SA-COSO with a different setup of the population sizes are given in the supplementary materials I to show that the setup of the population size in this paper is rational. In PSO [52],

TABLE II
DEFINITION OF ALGORITHMS THE SA-COSO COMPARED

Algorithms	Definision
PSO	Particle swarm optimization using Eq. (2) to update velocity
FESPSO	Fitness estimation strategy assisted PSO using approximation Eq. (12)
SL-PSO	A social learning based particle swarm optimization [53]
RBF-assisted SL-PSO	SL-PSO assisted by a radial basis function network
COSO	The cooperative swarm optimization without surrogate model assistance

the cognitive and social parameters are both set to 2.05. For the SA-COSO, these parameters are set the same as in PSO. However, as we can see from (2) and (15), the difference in updating the velocity of the canonical PSO [52] and the PSO used in SA-COSO is that in the latter there are two social coefficients, c_2 and c_3, which aim to cooperatively guide the particles toward the global best position. Two factors must be considered in setting the value of c_2 and c_3. First, both c_2 and c_3 are social coefficients, which are introduced to accelerate the convergence to the global best position. Therefore, the sum of c_2 and c_3 should remain to be 2.05 to be consistent with the setting in the canonical PSO. Second, the global best position of SA-COSO can be the global best of either the FES-assisted PSO or surrogate-assisted SL-PSO. Thus it is important to prevent the search dynamics of SA-COSO from being dominated by one of the two global best positions. Therefore, c_2 and c_3 are both set to 1.025.

The parameters pr_i^L and ϵ in RBF-assisted SL-PSO are set to 1 and 0, respectively. As the RBF network is utilized for learning the contour of the fitness landscape, the complexity of the RBF network should not be overly large. Therefore, in our experiments, the RBF network stops learning if the maximum number of its hidden nodes (max_node) reaches eight or the mean squared errors of the RBF is less than 0.1. Correspondingly, the minimum size of training data can be set to $\max\_\text{node} * D + \max\_\text{node}$ for the RBF network with eight hidden nodes. In our method, the size of the archive *DB* (N_{DB}) is set to $\max\_\text{node} * D + 10$, which is slightly larger than the minimum requirement on the size of training data. The width of the Gaussian function is set adaptively according to the number of data pairs (solutions) saved in archive *DB* as follows:

$$d\_\max(k,d) = \max\{x_{id} | i \in \{1,2,\ldots,N_{DB}\}\} \tag{27}$$

$$d\_\min(k,d) = \min\{x_{id} | i \in \{1,2,\ldots,N_{DB}\}\} \tag{28}$$

$$\sigma = \frac{\sum_{k=1}^{t} \sqrt{\sum_{d=1}^{D} \|d\_\max(k,d) - d\_\min(k,d)\|}}{t} \tag{29}$$

where $d\_\max(k,d)$ and $d\_\min(k,d)$ represent the maximum and minimum values of dimension d at iteration k in *DB*. t represents the current iteration, D is the dimension of the problem to be optimized and N_{DB} is the total number of particles (number of training samples) saved in the archive. The maximum number of FEs is set to 1000.

TABLE III
COMPARISONS OF THE STATISTICAL RESULTS ON 50-D BENCHMARK PROBLEMS

	Approach	Mean(Wilcoxon test)	Std.
F1	PSO	1.9679e+03(+)	3.1391e+02
	FESPSO	1.9912e+03(+)	5.4298e+02
	SL-PSO	2.8114e+03(+)	3.2779e+02
	RBF-assisted SL-PSO	6.2197e+02(≈)	7.2355e+02
	COSO	3.2459e+03(+)	4.0408e+02
	SA-COSO	**5.1475e+01**	1.6246e+01
F2	PSO	2.5187e+03(+)	8.6668e+02
	FESPSO	3.0534e+03(+)	6.2030e+02
	SL-PSO	4.2853e+03(+)	8.0166e+02
	RBF-assisted SL-PSO	1.3417e+03(+)	1.7562e+03
	COSO	4.2898e+03(+)	9.5552e+02
	SA-COSO	**2.5258e+02**	4.0744e+01
F3	PSO	1.8522e+01(+)	8.6328e-01
	FESPSO	1.8680e+01(+)	8.2140e-01
	SL-PSO	1.9098e+01(+)	3.7426e-01
	RBF-assisted SL-PSO	2.0742e+01(+)	9.1041e-02
	COSO	1.9381e+01(+)	3.1847e-01
	SA-COSO	**8.9318e+00**	1.0668e+00
F4	PSO	2.7561e+02(+)	6.4319e+01
	FESPSO	2.9842e+02(+)	6.1998e+01
	SL-PSO	4.4469e+02(+)	4.7173e+01
	RBF-assisted SL-PSO	5.9663e+01(≈)	9.0951e+01
	COSO	4.7394e+02(+)	4.4160e+01
	SA-COSO	**6.0062e+00**	1.1043e+00
F5	PSO	5.1157e+02(+)	1.0657e+02
	FESPSO	4.6461e+02(+)	9.0397e+01
	SL-PSO	5.9722e+02(+)	7.2165e+01
	RBF-assisted SL-PSO	3.1471e+02(+)	6.3568e+01
	COSO	6.8226e+02(+)	6.5694e+01
	SA-COSO	**1.9716e+02**	3.0599e+01
F6	PSO	1.1064e+03(≈)	6.1566e+01
	FESPSO	1.1415e+03(+)	6.0266e+01
	SL-PSO	1.3029e+03(+)	3.0287e+01
	RBF-assisted SL-PSO	1.3465e+03(+)	4.5098e+01
	COSO	1.2801e+03(+)	2.3818e+01
	SA-COSO	**1.0809e+03**	3.2859e+01

In order to make fair comparisons, the population size of both PSO and FESPSO is set to 30, the size of SL-PSO and SLPSO_RBF is set to 200, and the size of COSO is set to 230. The parameters of the RBF-assisted SL-PSO and in COSO are set the same as those in SA-COSO.

B. Experimental Results on 50-D Problems

Table III lists the statistical results of all algorithms under comparison averaged 20 independent runs, including the results of the Wilcoxon rank sum tests calculated at a significance level of $\alpha = 0.05$, where "≈" indicates that there is no statistically significant difference between the results obtained by SA-COSO and the compared algorithm, "+" indicates that the compared algorithm is significantly outperformed by SA-COSO according to a Wilcoxon rank sum test, while "−" means that SA-COSO is significantly outperformed by the compared algorithm.

It can be seen from Table III that SA-COSO has achieved significantly better or comparative results on all of the benchmark functions used in this paper. In order to examine the performance of SA-COSO, we plot the convergence profiles of the compared algorithms in Fig. 4. Note that as the population sizes of the algorithms under comparison are different,

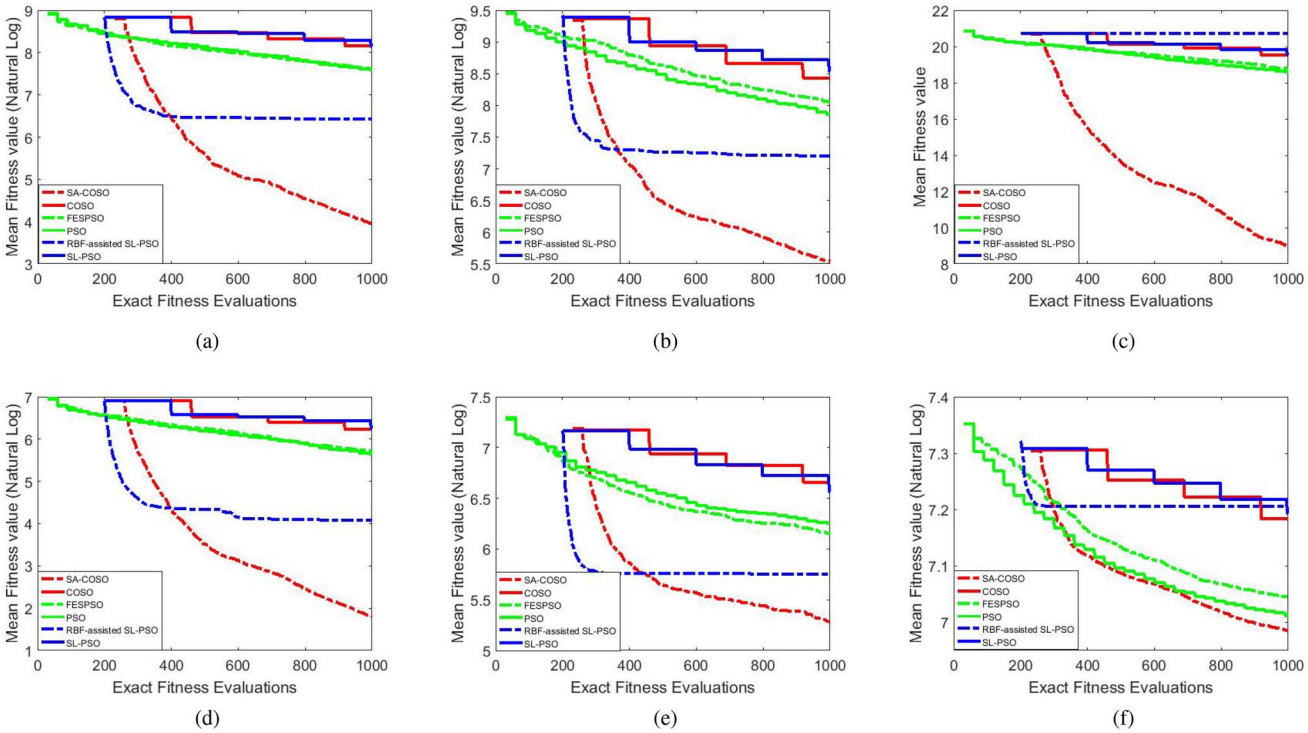

Fig. 4. Convergence trends for 50-D (a)–(f) $F1-F6$ from six algorithms.

the initial best fitness of the different algorithms are different. We can make the following observations regarding the performance of the compared algorithms. First, SA-COSO, which integrates the advantages of global surrogate model and FES, is able to continuously improve its performance over the generations. The performance improvement of SA-COSO is not as fast as the RBF-assisted SL-PSO in the early search stage, which might be attributed to the fact that SA-COSO considers not only the global best position of SL-PSO, but also that of PSO, resulting a larger degree of diversity in the population that contributes to more exploratory search. Compared to the RBF-assisted SL-PSO and FESPSO, SA-COSO performs the best on all test problems used in the comparisons. Second, we can see from the figures that the RBF-assisted SL-PSO (the blue line) converges rapidly in the beginning and then stagnates for all problems except for the Ackley function, which confirms that hypothesis that a global surrogate model is able to help the population quickly locate the region where an optimum is located. The reason why this does not happens to the Ackley function is due to the fact that its fitness landscape is nearly a plateau in most of the region close to the global optimum and the optimum is located in a very narrow region near the origin. Such fitness landscape is very hard for an RBF network to capture trained using only a limited number of samples. Third, the results of the FES-assisted PSO is only slightly better than PSO on 50-D test problems, the FES in SA-COSO is very helpful in that SA-COSO is able to continuously improve the results, while the RBF assisted SL-PSO stagnates in the later stage after its quick convergence in the early search stage.

To further demonstrate the performance of the SA-COSO, we also compare the results obtained by SA-COSO with those obtained by a surrogate-assisted differential evolution, termed GPEME (GP+DR) [39] on 50-D test problems $F1-F4$. GPEME (GP) uses a GP surrogate model to assist a differential evolution algorithm and GPEME (GP+DR) represents a GPEME variant using a dimension reduction technique. We are not able to compare the performance of the two algorithms on 50-D test problems F5 and F6 as no results were provided on these two test functions in [39]. The comparative results are given in Fig. 5. Seen from Fig. 5, we can see that SA-COSO achieves better results than both GPEME and GPEME(GP+DR) on $F1$, $F3$, and $F4$, and comparative results with GPEME(GP+DR) on $F2$, showing the promising performance of the proposed method.

C. Experimental Results on 100-D Problems

The main motivation of this paper is to push the boundary of surrogate-assisted metaheuristics for solving high-dimensional computationally problems. In the following, we examine the performance of the SA-COSO on 100-D test problems with limited computational budget in terms of the number of FEs using the real fitness function. Table IV summarizes the obtained optimal solutions together with the t-test results on the same set of test problems. Seen from these statistical results, we can conclude that the proposed SA-COSO shows superior performance to the compared algorithms on the 100-D optimization problems, similar to the results on the 50-D test problems. Fig. 6 plots the convergence profiles of the compared algorithms on 100-D test problems. As we can see, SA-COSO shows similar search dynamics as shown on the 50-D test problems. The RBF network is able to help the SA-COSO algorithm locate the region where the optimum is

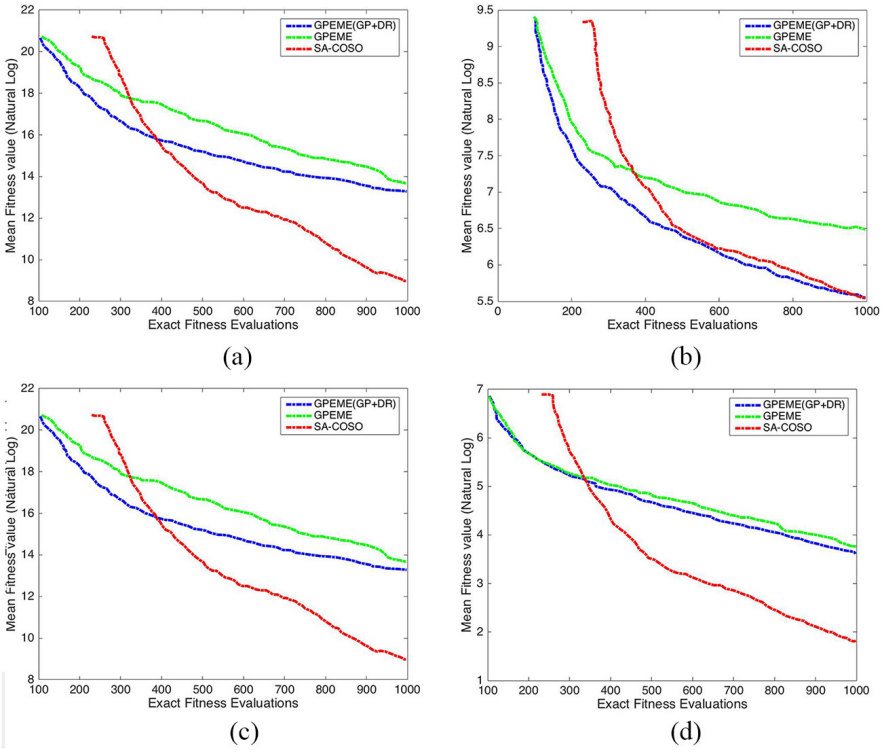

Fig. 5. Convergence trends comparison on (a)–(d) $F1-F4$.

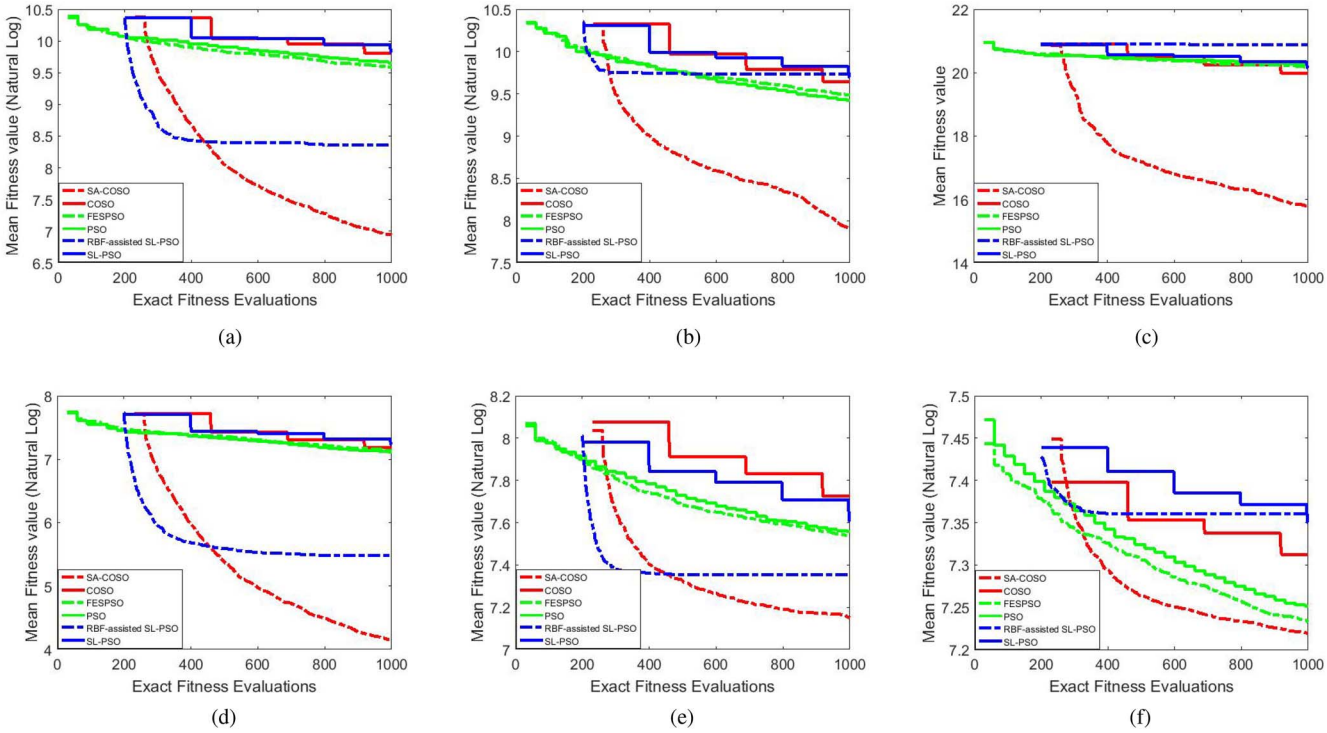

Fig. 6. Convergence profiles of the compared algorithms on 100-D (a)–(f) $F1-F6$.

and keep improving the solution, indicating the cooperative search of SL-PSO and PSO is very effective in finding an optimal solution. The experimental results on the 100-D problems confirm the competitive performance of SA-COSO on high-dimensional problems.

To further evaluate our method on higher dimensional problems, we have also conducted the comparisons on 200-D problems and the results are provided in the supplementary materials. These results confirm the good performance of the proposed method for solving high-dimensional problems.

TABLE IV
COMPARATIVE RESULTS ON 100-D BENCHMARK PROBLEMS

	Approach	Mean(Wilcoxon test)	Std.
F1	PSO	1.5309e+04(+)	1.7685e+03
	FESPSO	1.4574e+04(+)	1.8004e+03
	SL-PSO	1.6935e+04(+)	1.2746e+03
	RBF-assisted SL-SPO	4.2610e+03(+)	3.0987e+04
	COSO	1.7475e+04(+)	1.2135e+03
	SA-COSO	**1.0332e+03**	3.1718e+02
F2	PSO	1.2160e+04(+)	2.0188e+03
	FESPSO	1.2991e+04(+)	1.8186e+03
	SL-PSO	1.4755e+04(+)	1.5183e+03
	RBF-assisted SL-SPO	1.6895e+04(+)	6.1860e+03
	COSO	1.4405e+04(+)	1.4055e+03
	SA-COSO	**2.7142e+03**	1.1702e+02
F3	PSO	2.0239e+01(+)	1.8744e-01
	FESPSO	2.0178e+01(+)	3.5469e-01
	SL-PSO	1.9981e+01(+)	1.9843e-01
	RBF-assisted SL-SPO	2.0876e+01(+)	1.7703e-01
	COSO	1.9949e+01(+)	1.5436e-01
	SA-COSO	**1.5756e+01**	5.0245e-01
F4	PSO	1.2162e+03(+)	9.2716e+01
	FESPSO	1.2305e+03(+)	1.0561e+02
	SL-PSO	1.2232e+03(+)	9.7340e+01
	RBF-assisted SL-SPO	2.4023e+02(≈)	3.1646e+02
	COSO	1.2898e+03(+)	9.6918e+01
	SA-COSO	**6.3353e+01**	1.9021e+01
F5	PSO	1.8946e+03(+)	1.5227e+02
	FESPSO	1.8636e+03(+)	1.9079e+02
	SL-PSO	1.8604e+03(+)	1.3078e+02
	RBF-assisted SL-SPO	1.5629e+03(+)	1.3868e+02
	COSO	2.1028e+03(+)	5.6521e+01
	SA-COSO	**1.2731e+03**	1.1719e+02
F6	PSO	1.4083e+03(+)	5.2538e+01
	FESPSO	1.3810e+03(≈)	3.9465e+01
	SL-PSO	1.5407e+03(+)	2.4168e+01
	RBF-assisted SL-SPO	1.5721e+03(+)	7.5160e+01
	COSO	1.4852e+03(+)	2.6082e+01
	SA-COSO	**1.3657e+03**	3.0867e+01

TABLE V
AVERAGE COMPUTATION TIME (IN SECONDS) REQUIRED BY THE COMPARED ALGORITHMS FOR 50-D AND 100-D PROBLEMS WITH 1000 FES

	Approach	Mean time on 50-D problems(s)	Mean time on 100-D problems(s)
F1	PSO	6.08e-01	9.21e-01
	FESPSO	1.66e+00	2.16e+00
	SL-PSO	1.26e+01	1.71e+01
	RBF-assisted SL-SPO	1.48e+03	1.49e+03
	COSO	9.67e+01	1.64e+01
	SA-COSO	6.22e+02	8.16e+02
F2	PSO	8.71e-01	8.71e-01
	FESPSO	1.88e+00	1.89e+00
	SL-PSO	1.30e+01	1.30e+01
	RBF-assisted SL-SPO	1.49e+03	1.49e+03
	COSO	1.06e+01	1.06e+01
	SA-COSO	5.24e+02	5.24e+02
F3	PSO	6.77e-01	9.65e-01
	FESPSO	1.79e+00	2.24e+00
	SL-PSO	1.29e+01	1.71e+01
	RBF-assisted SL-SPO	1.29e+03	1.35e+03
	COSO	1.01e+01	1.61e+01
	SA-COSO	6.50e+02	9.00e+02
F4	PSO	8.03e-01	1.16e+00
	FESPSO	1.90e+00	2.40e+00
	SL-PSO	1.27e+01	1.88e+01
	RBF-assisted SL-SPO	1.58e+03	1.79e+03
	COSO	1.05e+01	1.59e+01
	SA-COSO	5.89e+02	8.21e+02
F5	PSO	5.38e-01	1.25e+00
	FESPSO	1.64e+00	4.37e+00
	SL-PSO	1.04e+01	1.79e+01
	RBF-assisted SL-SPO	1.32e+03	1.53e+03
	COSO	9.82e+00	1.54e+01
	SA-COSO	5.98e+02	9.81e+02
F6	PSO	1.20e+01	3.99e+01
	FESPSO	1.30e+01	4.07e+01
	SL-PSO	2.25e+01	5.65e+01
	RBF-assisted SL-SPO	1.34e+03	9.89e+02
	COSO	2.37e+01	6.65e+01
	SA-COSO	5.88e+02	9.61e+02

D. Empirical Analysis of the Computational Complexity

The computational complexity of the proposed SA-COSO is composed of three main parts, namely, the computation time for FEs, for training the RBF network, and for calculating the distances in updating the archive *DB*. In this section, we empirically compare computation time needed by compared algorithm for solving the 50-D and 100-D optimization problems. All algorithms are implemented on a computer with a 2.50-GHz processor and 8-GB in RAM. Table V presents the computation time of the algorithms under comparison averaged over 20 independent runs when a maximum of 1000 FEs using the real fitness function is allowed. From Table V, we find that PSO, whose computation time is mainly dedicated to FEs, requires the least computation time. For convenience, we use the time for PSO to search for an optimum on a fixed budget of 1000 FEs as the baseline for comparison. From Table V, we can see that the RBF-assisted SL-PSO requires the longest time, meaning that training the surrogate model takes up more time than calculating the distance in updating the archive *DB*. We can also find that the average time required by the proposed SA-COSO for solving 50-D and 100-D test problems are approximately 0.6 and 1.4 s per FE using the expensive fitness function, respectively. Compared to most time-consuming FEs in real world applications where each FE may take tens of minutes to hours, this increase in computation time for surrogate training and distance calculation can still be considered negligible.

V. CONCLUSION

An SA-COSO algorithm is proposed in this paper for solving high-dimensional computationally expensive problems. The PSO algorithm assisted by an FES and the SL-PSO assisted by a RBF network are integrated for computationally efficient and effective search of high-dimensional problems. The RBF-assisted SL-PSO aims to quickly find the region in which a global optimum is located, based on the hypothesis that SL-PSO is a powerful global search algorithm for large-scale optimization problems and that the RBF network is able to capture the global profile of the fitness landscape. FES-assisted PSO is meant help the algorithm to perform efficient local search assisted by the FES given limited computational budget. Experimental results on six 50-D and six 100-D benchmark problems demonstrated the efficiency and effectiveness of the proposed SA-COSO algorithm in comparison to a few PSO variants with and without using surrogates, and a surrogate-assisted differential evolution algorithm that is the only surrogate-assisted algorithm reported in the literature that have solved optimization problems up to a dimension of 50.

Although the proposed SA-COSO has shown promising performance on the six high-dimensional test problems up to a dimension of 100, several questions remain to be answered. First, the performance of the proposed algorithm is relatively poor on problems whose global optimum lies near the boundary of the decision space and problems whose global optimum is located very close to the local optima. Second, as the dimension of the problems increases, the computation time to train a surrogate model will dramatically increase too, and the capability of the surrogate model to capture the global profile of the fitness landscape will seriously degrade. Therefore, it is of great interest to develop efficient and effective training algorithms for large-scale problems. Finally, although the performance of SA-COSO has been convincingly demonstrated on six widely used test problems, it is of great interest to test its performance on real-world application problems such as aerodynamic design optimization.

REFERENCES

[1] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 1–14, Jan. 2015.

[2] T.-L. Lin *et al.*, "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2629–2636, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417409007696

[3] B. Ji *et al.*, "Multiobjective design optimization of IGBT power modules considering power cycling and thermal cycling," *IEEE Trans. Power Electron.*, vol. 30, no. 5, pp. 2493–2504, May 2015.

[4] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.

[5] Y. Yoon and Y.-H. Kim, "An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1473–1483, Oct. 2013.

[6] T.-Y. Wu and C.-H. Lin, "Low-SAR path discovery by particle swarm optimization algorithm in wireless body area networks," *IEEE Sensors J.*, vol. 15, no. 2, pp. 928–936, Feb. 2015.

[7] C.-H. Chen, T.-K. Liu, and J.-H. Chou, "A novel crowding genetic algorithm and its applications to manufacturing robots," *IEEE Trans. Ind. Informat.*, vol. 10, no. 3, pp. 1705–1716, Aug. 2014.

[8] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[9] S. Chernbumroong, S. Cang, and H. Yu, "Genetic algorithm-based classifiers fusion for multisensor activity recognition of elderly people," *IEEE J. Biomed. Health Inform.*, vol. 19, no. 1, pp. 282–289, Jan. 2015.

[10] Y. Jin and B. Sendhoff, "A systems approach to evolutionary multiobjective structural optimization and beyond," *IEEE Comput. Intell. Mag.*, vol. 4, no. 3, pp. 62–76, Aug. 2009.

[11] H. S. Bernardino, H. J. C. Barbosa, and L. G. Fonseca, "Surrogate-assisted clonal selection algorithms for expensive optimization problems," *Evol. Intell.*, vol. 4, no. 2, pp. 81–97, 2011. [Online]. Available: http://dx.doi.org/10.1007/s12065-011-0056-1

[12] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, Oct. 2002.

[13] Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee, "Memetic algorithm using multi-surrogates for computationally expensive optimization problems," *Soft Comput.*, vol. 11, no. 10, pp. 957–971, 2007. [Online]. Available: http://dx.doi.org/10.1007/s00500-006-0145-8

[14] X. Y. Sun, D. W. Gong, and X. P. Ma, "Directed fuzzy graph-based surrogate model-assisted interactive genetic algorithms with uncertain individual's fitness," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Trondheim, Norway, 2009, pp. 2395–2402.

[15] R. Mallipeddi and M. Lee, "Surrogate model assisted ensemble differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 1–8.

[16] R. G. Regis, "Particle swarm with radial basis function surrogates for expensive black-box optimization," *J. Comput. Sci.*, vol. 5, no. 1, pp. 12–23, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750313000847

[17] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, 2005.

[18] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 61–70, 2011.

[19] Y. Lian and M.-S. Liou, "Multiobjective optimization using coupled response surface model and evolutionary algorithm," *AIAA J.*, vol. 43, no. 6, pp. 1316–1325, 2005.

[20] M. Farina, "A neural network based generalized response surface multiobjective evolutionary algorithm," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1. Honolulu, HI, USA, 2002, pp. 956–961.

[21] I. Loshchilov, M. Schoenauer, and M. Sebag, "A mono surrogate for multiobjective optimization," in *Proc. 12th Annu. Conf. Genet. Evol. Comput.*, Portland, OR, USA, 2010, pp. 471–478.

[22] M. Herrera, A. Guglielmetti, M. Xiao, and R. F. Coelho, "Metamodel-assisted optimization based on multiple kernel regression for mixed variables," *Struct. Multidiscipl. Optim.*, vol. 49, no. 6, pp. 979–991, 2014.

[23] A. Gaspar-Cunha and A. Vieira, "A hybrid multi-objective evolutionary algorithm using an inverse neural network," in *Proc. Hybrid Metaheuristics*, Valencia, Spain, 2004, pp. 25–30.

[24] A. Gaspar-Cunha and A. Vieira, "A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations," *Int. J. Comput. Syst. Signal*, vol. 6, no. 1, pp. 18–36, 2005.

[25] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA J.*, vol. 41, no. 4, pp. 687–696, 2003.

[26] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 1, pp. 66–76, Jan. 2007.

[27] A. Isaacs, T. Ray, and W. Smith, "An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization," in *Progress in Artificial Life*. Heidelberg, Germany: Springer, 2007, pp. 257–268.

[28] S. Z. Martínez and C. A. C. Coello, "MOEA/D assisted by RBF networks for expensive multi-objective optimization problems," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, Amsterdam, The Netherlands, 2013, pp. 1405–1412.

[29] C. Sun, Y. Jin, J. Zeng, and Y. Yu, "A two-layer surrogate-assisted particle swarm optimization algorithm," *Soft Comput.*, vol. 19, no. 6, pp. 1461–1475, 2015. [Online]. Available: http://dx.doi.org/10.1007/s00500-014-1283-z

[30] D. Buche, N. N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with Gaussian process fitness function models," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 183–194, May 2005.

[31] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2006.

[32] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection," in *Parallel Problem Solving From Nature–PPSN X*. Heidelberg, Germany: Springer, 2008, pp. 784–794.

[33] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 456–474, Jun. 2010.

[34] M. Y. M. Ahmed and N. Qin, "Surrogate-based multi-objective aerothermodynamic design optimization of hypersonic spiked bodies," *AIAA J.*, vol. 50, no. 4, pp. 797–810, 2012.

[35] A. Ratle, "Kriging as a surrogate fitness landscape in evolutionary optimization," *Artif. Intell. Eng. Design Anal. Manuf.*, vol. 15, no. 1, pp. 37–49, 2001.

[36] M. K. Karakasis and K. C. Giannakoglou, "Metamodel-assisted multi-objective evolutionary optimization," in *Proc. Evol. Deterministic Methods Design Optim. Control Appl. Ind. Soc. Problems (EUROGEN)*, Munich, Germany, 2005, pp. 1–11.

[37] M. D. Parno, K. R. Fowler, and T. Hemker, "Framework for particle swarm optimization with surrogate functions," Dept. Comput. Sci., Darmstadt Tech. Univ., Darmstadt, Germany, Tech. Rep. TUD-CS-2009-0139, 2009.

[38] A. G. Di Nuovo, G. Ascia, and V. Catania, "A study on evolutionary multi-objective optimization with fuzzy approximation for computational expensive problems," in *Parallel Problem Solving From Nature-PPSN XII*. Heidelberg, Germany, Springer, 2012, pp. 102–111.
[39] B. Liu, Q. Zhang, and G. G. E. Gielen, "A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 180–192, Apr. 2014.
[40] W. Gong, A. Zhou, and Z. Cai, "A multioperator search strategy based on cheap surrogate models for evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 746–758, Oct. 2015.
[41] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 329–355, Jun. 2010.
[42] S. Z. Martinez and C. A. C. Coello, "Combining surrogate models and local search for dealing with expensive multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Cancún, Mexico, 2013, pp. 2572–2579.
[43] L. G. Fonseca, A. C. C. Lemonge, and H. J. C. Barbosa, "A study on fitness inheritance for enhanced efficiency in real-coded genetic algorithms," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[44] R. E. Smith, B. A. Dike, and S. A. Stegmann, "Fitness inheritance in genetic algorithms," in *Proc. ACM Symp. Appl. Comput.*, Nashville, TN, USA, 1995, pp. 345–350.
[45] T. Hendtlass, "Fitness estimation and the particle swarm optimisation algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 4266–4272.
[46] C. Sun, J. Zeng, J. Pan, S. Xue, and Y. Jin, "A new fitness estimation strategy for particle swarm optimization," *Inf. Sci.*, vol. 221, pp. 355–370, Feb. 2013.
[47] C. Sun, J. Zeng, J. Pan, and Y. Jin, "Similarity-based evolution control for fitness estimation in particle swarm optimization," in *Proc. IEEE Symp. Comput. Intell. Dyn. Uncertain Environ. (CIDUE)*, Singapore, 2013, pp. 1–8.
[48] Y. Tenne and S. W. Armfield, "A framework for memetic optimization using variable global and local surrogate models," *Soft Comput.*, vol. 13, nos. 8–9, pp. 781–793, 2009.
[49] C. A. Georgopoulou and K. C. Giannakoglou, "A multi-objective metamodel-assisted memetic algorithm with strength-based local refinement," *Eng. Optim.*, vol. 41, no. 10, pp. 909–923, 2009.
[50] M. Pilát and R. Neruda, "An evolutionary strategy for surrogate-based multiobjective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 1–7.
[51] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, "A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 3. Edinburgh, U.K., Sep. 2005, pp. 2832–2839.
[52] M. Clerc, "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3. Washington, DC, USA, 1999, pp. 1951–1957.
[53] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.
[54] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, vol. 1. Nagoya, Japan, 1995, pp. 39–43.
[55] J. F. Schutte and A. A. Groenwold, "A study of global optimization using particle swarms," *J. Glob. Optim.*, vol. 31, no. 1, pp. 93–108, 2005.
[56] K. Kameyama, "Particle swarm optimization—A survey," *IEICE Trans. Inf. Syst.*, vol. E92-D, no. 7, pp. 1354–1361, Jul. 2009.
[57] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, Anchorage, AK, USA, 1998, pp. 69–73.
[58] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1. 2000, pp. 84–88.
[59] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[60] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.
[61] R. L. Hardy, "Multiquadric equations of topography and other irregular surfaces," *J. Geophys. Res.*, vol. 76, no. 8, pp. 1905–1915, 1971.
[62] M. J. Er, S. Wu, J. Lu, and H. L. Toh, "Face recognition with radial basis function (RBF) neural networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 697–710, May 2002.
[63] A. Kattan and E. Galvan, "Evolving radial basis function networks via GP for estimating fitness values using surrogate models," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 1–7.
[64] A. B. Rohler and S. Chen, "Multi-swarm hybrid for multi-modal optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 1–8.

Chaoli Sun (M'15) received the B.Sc. and M.Sc. degrees from Hohai University, Nanjing, China, in 2000 and 2003, respectively, and the Ph.D. degree from the Taiyuan University of Science and Technology, Taiyuan, China, in 2011.

She is an Associate Professor with the Department of Computer Science and Technology, Taiyuan University of Science and Technology. Her current research interests include swarm intelligence, particle swarm optimization, surrogate-assisted evolutionary optimization, and their applications to practical engineering problems.

Dr. Sun is an Associate Editor of *Soft Computing* and *Complex and Intelligent Systems*, the Chair of the Task Force on Data-Driven Evolutionary Optimization of Expensive Problems within the Evolutionary Computation Technical Committee of the IEEE Computational Intelligence Society. She has also served as a Reviewer for journals, including IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Soft Computing*, *Natural Computing*, and *Complex and Intelligent Systems*.

Yaochu Jin (M'98–SM'02–F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Bochum, Germany, in 2001.

He is a Professor of Computational Intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He is also a Finland Distinguished Professor funded by the Finnish Funding Agency for Innovation (Tekes) and a Changjiang Distinguished Visiting Professor appointed by the Ministry of Education, Beijing, China. He has (co)-authored over 200 peer-reviewed journal and conference papers and been granted eight patents on evolutionary optimization. He has delivered 20 invited keynote speeches at international conferences. His current research interests include computational intelligence, computational neuroscience, computational systems biology, and nature-inspired and real-world driven problem-solving.

Dr. Jin was a recipient of the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, and the 2014 and 2017 IEEE Computational Intelligence Magazine Outstanding Paper Award. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and the Co-Editor-in-Chief of *Complex and Intelligent Systems*. He is also an Associate Editor or an Editorial Board Member of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON NANOBIOSCIENCE, *Evolutionary Computation*, *BioSystems*, *Soft Computing*, and *Natural Computing*. He is an IEEE Distinguished Lecturer for the period 2013–2015 and 2017–2019, and the past Vice President for the Technical Activities of the IEEE Computational Intelligence Society from 2014 to 2015.

Ran Cheng received the B.Eng. degree from Northeastern University, Shenyang, China, in 2010, and the Ph.D. degree from the Department of Computer Science, University of Surrey, Guildford, U.K., in 2016.

He is a Research Fellow with the CERCIA Group, School of Computer Science, University of Birmingham, Birmingham, U.K. His current research interests include artificial/computational intelligence-based complex system optimization, evolutionary multi/many-objective optimization and decision making, evolutionary multimodal optimization and decision making, model-based evolutionary algorithms, large-scale optimization, data-driven optimization, and particle swarm optimization.

Dr. Cheng serves as a Regular Reviewer for journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS. He is an Editorial Board Member of *Complex and Intelligent Systems*.

Jinliang Ding (M'09–SM'14) received the bachelor's, master's, and Ph.D degrees in control theory and control engineering from Northeastern University, Shenyang, China.

He is a Professor with the State Key Laboratory of Synthetical Automation for Process Industry, Northeastern University. He has authored or co-authored over 90 refereed journal papers and refereed papers at international conferences. He is also the inventor or co-inventor of 17 patents. His current research interests include modeling, plant-wide control and optimization for the complex industrial systems, stochastic distribution control, and multiobjective evolutionary algorithms and its application.

Dr. Ding was a recipient of the Young Scholars Science and Technology Award of China in 2016, the National Science Fund for Distinguished Young Scholars in 2015, the National Technological Invention Award in 2013, two First-Prize of Science and Technology Award of the Ministry of Education in 2006 and 2012, respectively, and the Best Paper Award of 2011–2013 for control engineering practice.

Jianchao Zeng received the B.Sc. degree from Taiyuan Heavy Machinery Institute, Taiyuan, China, in 1982, and the M.Sc. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1985 and 1990, respectively.

He is a Professor with the School of Computer Science and Control Engineering, North University of China, Taiyuan. He has published over 200 international journal and conference papers. His current research interests include modeling and control of complex systems, intelligent computation, swarm intelligence, and swarm robotics.

Dr. Zeng is the Vice President of the North University of China and the Technical Committee on System Simulation of the Chinese Association of Automation, and a Director of the China Simulation Federation.