Surrogate model assisted cooperative coevolution for large scale optimization

Zhigang Ren1,2 · Bei Pang1 · Muyi Wang1 · Zuren Feng2 · Yongsheng Liang1 · An Chen1 · Yipeng Zhang1

Published online: 5 September 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
It has been shown that cooperative coevolution (CC) can effectively deal with large scale optimization problems (LSOPs) through a ‘divide-and-conquer’ strategy. However, its performance is severely restricted by the current context-vector-based sub-solution evaluation method, since this method needs to invoke the original high dimensional simulation model when evaluating each sub-solution, thus requiring many computation resources. To alleviate this issue, this study proposes a novel surrogate model assisted cooperative coevolution (SACC) framework. SACC constructs a surrogate model for each sub-problem and employs it to evaluate corresponding sub-solutions. The original simulation model is only adopted to reevaluate a small number of promising sub-solutions selected by surrogate models, and these really evaluated sub-solutions will in turn be employed to update surrogate models. By this means, the computation cost could be greatly reduced without significantly sacrificing evaluation quality. By taking the radial basis function (RBF) and the success-history based adaptive differential evolution (SHADE) as surrogate model and optimizer, respectively, this study further designs a concrete SACC algorithm named RBF-SHADE-SACC. RBF and SHADE have only been proved to be effective on small and medium scale problems. This study scales them up to LSOPs under the SACC framework, where they are tailored to a certain extent for adapting to the characteristics of LSOPs and SACC. Empirical studies on IEEE CEC 2010 benchmark functions demonstrate that SACC can significantly enhance the sub-solution evaluation efficiency, and even with much fewer computation resources, RBF-SHADE-SACC can find much better solutions than traditional CC algorithms.

Keywords Cooperative coevolution (CC) · Large scale optimization problem (LSOP) · Surrogate model · Radial basis function (RBF) · Success-history based adaptive differential evolution (SHADE)

1 Introduction

Nowadays, large scale optimization problems (LSOPs) are becoming more and more popular in scientific research and engineering applications with the rapid development of big data techniques [1, 2]. As this kind of problem generally possesses black-box characteristics, the gradient-free evolutionary algorithms (EAs) are often employed to tackle them. However, the performance of traditional EAs rapidly deteriorates with the increase of the problem dimension. This is the so-called ‘curse of dimensionality’ [3, 4], the main reason for which consists in that the solution space of a problem exponentially grows with the increase of its dimension and traditional EAs cannot adequately explore the solution space of a LSOP within acceptable computation time.

Taking the idea of ‘divide-and-conquer’, cooperative coevolution (CC) provides a natural way for solving LSOPs [5]. For a LSOP to be tackled, CC first decomposes it into several smaller and simpler sub-problems, and then tries to solve it by cooperatively optimizing all the sub-problems with a traditional EA. It is understandable that decomposition plays a fundamental role in CC. A right decomposition can greatly reduce the optimization difficulty of a LSOP without changing its theoretical optimum. Therefore, most research efforts on CC were put into designing various types of decomposition methods in recent years, and by now several efficient decomposition
algorithms have been developed [6]. We will specially review them in Section 2.1.

By contrast, another important algorithmic operation in CC, i.e., the evaluation of the solutions to each sub-problem, which also has an important influence on the efficiency of CC, is neglected. It is known that CC mainly focuses on black-box LSOPs and generally evaluates their solutions by simulation. This means that all the sub-problems obtained via decomposition do not own separate or explicit objective functions. To evaluate sub-solutions, now all the CC algorithms adopt a context-vector-based method [4]. This method takes a complete solution of the original LSOP as the context vector. For a sub-solution to be evaluated, the method first inserts it into the corresponding positions in the context vector, and then achieves evaluation by indirectly evaluating the modified context vector with the simulation model of the original LSOP. This method seems straightforward and reasonable. It is also via it that different sub-problems cooperate with each other. Nevertheless, it inevitably brings some issues. First, the simulation process is very time-consuming and it consumes most of the available computation resources, thus a practical LSOP generally allows to simulate a very limited number of solutions. Then the number of simulations assigned to each sub-problem becomes much smaller. With so few fitness evaluations (FEs), it is challenging for the optimizer in CC to generate high-quality sub-solutions. As a result, the quality of the final solution to the original LSOP can hardly be guaranteed. Second, for an incorrectly decomposed LSOP, it is theoretically impossible to find its global optimum based on a single context vector [7]. Multi-context-vector-based evaluation method is likely to remedy this defect [7–9], but it further raises solution simulation requirement. Therefore, now it is very significant to develop an efficient sub-solution evaluation method which depends less on the original simulation model.

As explained above, LSOPs can be regarded as computationally expensive black-box optimization problems. To deal with these kind of problems, some researchers developed surrogate model assisted EAs (SAEAs) [10–12]. Their key idea is to construct a calculable surrogate model for the computationally expensive simulation model and to mainly rely on the surrogate model to evaluate solutions. Only some good solutions picked out by the surrogate model are allowed to access the original simulation model. By this means, the number of real evaluations can be greatly reduced. So far, several types of surrogate models have been proposed and integrated with a variety of EAs [11, 12]. However, these SAEAs only take effect on low and medium dimensional problems since a high dimensional problem often requires too many really evaluated solutions to build an accurate enough surrogate model.

Aiming at the sub-solution evaluation issue in CC, this study makes the following beneficial attempts:

1) Develop a novel surrogate model assisted CC (SACC) framework. According to the characteristics of the sub-problems in CC and the relative merits of SAEAs, SACC constructs and maintains a surrogate model for each sub-problem. Based on these models, it completes the main sub-solution evaluation task and simultaneously finds out a small number of promising sub-solutions, which after undergoing the real evaluations will in turn be employed to refine the corresponding surrogate models. Consequently, the evaluation accuracy of surrogate models could be continually improved and the solution simulation requirement could be greatly reduced.

2) Different from the traditional CC which evaluates a solution in terms of its fitness value (FV), SACC achieves this by investigating the fitness improvement of a solution to the context vector. In this way, the evaluation result on a solution of the current sub-problem can get rid of the influence of the other sub-problems. This facilitates improving the evaluation accuracy of surrogate models and also avoids reevaluating sub-solutions with the original simulation model when the context vector changes.

3) Design a concrete SACC algorithm named RBF-SHADE-SACC by taking the radial basis function (RBF) [13] and the success-history based adaptive differential evolution (SHADE) [14] as surrogate model and optimizer, respectively. RBF and SHADE have only been proved to be effective for small and medium scale problems, but have seldom been employed to solve LSOPs. This study scales them up to LSOPs of 1000 dimensions under the SACC framework. Instead of directly integrating RBF and SHADE into SACC, this study tailors them to a certain extent such that they can adapt well to the characteristics of LSOPs and SACC. As a result, the performance of RBF-SHADE-SACC could be guaranteed.

The rest of this paper is organized as follows: Section 2 presents the related work on CC and surrogate models. Section 3 describes SACC in detail, including the framework of SACC, the tailored RBF and SHADE, and the final RBF-SHADE-SACC algorithm. Section 4 reports experimental settings and results. Finally, conclusions are drawn in Section 5.

2 Related work

This section first reviews existing CC algorithms, mainly concerning representative decomposition algorithms and
2.1 Cooperative coevolution

CC can effectively tackle LSOPs by cooperatively optimizing the lower dimensional subproblems obtained through decomposition. Therefore, decomposition plays a pivotal role in CC and many decomposition algorithms have been developed by now [6]. These algorithms can generally be divided into three types, including static decomposition, random decomposition, and learning-based decomposition. Static decomposition is the simplest type of decomposition strategies since it directly divides the decision variables into a certain number of fixed groups. Potter and De Jong [5] proposed the first CC algorithm, which partitions an n-dimensional problem into n independent 1-dimensional sub-problems. They also developed a splitting-into-half decomposition strategy, which decomposes an n-dimensional problem into two fixed n/2 dimensional sub-problems [15]. More generally, Van den Bergh and Engelbrecht [4] suggested grouping an n-dimensional problem into k s-dimensional sub-problems for some \( k \cdot s = n \) and \( s \ll n \). These static decomposition strategies perform well on separable problems, but usually show poor performance on nonseparable problems as they take no account of variable interactions.

To remedy this defect, some random decomposition strategies were developed. Yang et al. [16] proposed the first random decomposition strategy and named it random grouping. This strategy randomly allocates all the decision variables into \( k \) \( s \)-dimensional sub-problems in each coevolution cycle instead of using a static grouping. To tackle the issue that it is difficult to specify a value for \( s \), Yang et al. [17] further developed a multilevel CC framework, which probabilistically selects a value for \( s \) from a candidate value pool for each new coevolution cycle. If a value helps CC achieve greater performance improvement in the past cycles, it will be selected with a higher probability. Omidvar et al. [18] indicated that with random grouping, the probability of grouping all the interacting variables into one sub-component dramatically decreases as the number of interacting variables increases, and suggested increasing the frequency of random grouping by reducing the iterations within a cycle.

In recent years, some learning-based decomposition methods were developed. They focus on making near optimal decomposition by explicitly detecting the interdependencies among variables. Delta grouping [19] can be regarded as an early representative of this kind of methods. It calculates the variation of each variable in two consecutive cycles and divides the variables with similar variations into the same sub-problem. It was shown that delta grouping can outperform random grouping on a variety of LSOPs, but often loses its efficiency on the problems of more than one group of interacting variables [19, 20]. To overcome this limitation, Omidvar et al. [20] proposed the differential grouping (DG) method. DG regards two variables as separable if one variable does not influence the change of the FV caused by the change of the other variable. However, the original DG ignores indirect interdependencies among variables and may group some interacting variables into different sub-problems. Aiming at this shortcoming, global DG [21] was proposed. It explicitly detects the interdependency between each pair of variables. Consequently, the decomposition accuracy is greatly improved at the cost of consuming much more computation resources. Recently, Omidvar et al. [22] developed a new version of DG which reduces the computation resource requirement by reusing some samples. Ren et al. [23] further improved the decomposition efficiency of DG by detecting interdependencies from the point view of vectors.

After decomposition, CC needs a specific algorithm to optimize the obtained sub-problems. By now, almost all kinds of EAs, including genetic algorithm (GA) [5], particle swarm optimization (PSO) [4, 24], and differential evolution (DE) [7, 8, 16–20], etc., have been employed as optimizer in CC, where an improved DE variant called the self-adaptive differential evolution with neighborhood search (SaNSDE) [25] was most frequently employed. No matter which kind of EA is used, the candidate solutions obtained for each sub-problem need to be evaluated. As indicated in the introduction section, each sub-problem does not have a separate or explicit objective function, and its solutions also cannot be directly evaluated by the simulation model of the original LSOP since they only reflect part of dimensions of the total solution space. Now all the existing CC algorithms adopt a context-vector-based method to evaluate sub-solutions [4]. This method takes a complete solution of the original LSOP, which is generally the best solution obtained so far, as the context vector. For a sub-solution to be evaluated, this method first inserts it into the corresponding positions in the context vector, and then achieves the evaluation by indirectly evaluating the modified context vector with the original simulation model. It has been verified that the context-vector-based method really takes effect [4, 6], but it requires many computation resources since it needs to invoke the original simulation model when evaluating each sub-solution.

Besides, this method is generally performed based on a single context vector, which may hinder the optimizer from finding the optimum for an incorrectly decomposed
LSOP [7]. To alleviate this issue, some multi-context-vector-based evaluation methods were developed. Wu et al. [7] and Tang et al. [8] proposed two multi-context mechanisms of a similar idea. Both mechanisms maintain a context set which contains a certain number of complete solutions. When evaluating a sub-solution, they randomly select a solution from the context set and take it as the final context vector. After each coevolution cycle, a crossover operation may be performed on the worst context vector to improve its quality. Different from Wu et al. [7] and Tang et al. [8], Peng et al. [9] suggested solving each sub-problem in CC with a multi-population scheme and constructing multiple context vectors with the optima achieved by the multiple populations. For a sub-solution to be evaluated, all the context vectors are used and the quality of the sub-solution is finally quantified by the best fitness obtained. There is no doubt that the multi-context-vector-based evaluation methods have some theoretical advantages over the single-context mechanism. Nevertheless, if straightforwardly performed, they are more likely to consume much more computation resources. Therefore, it becomes increasingly important to develop more efficient sub-solution evaluation methods.

### 2.2 Surrogate models

It is known that most EAs require a large number of FEIs before capturing the global optimal or near-optimal solution. However, some computationally expensive optimization problems do not support enough number of FEIs. To tackle this kind of problem, SAEAs were developed [10, 11]. They build surrogate models for the computationally expensive simulation models and employ the obtained surrogate models to evaluate candidate solutions, by which means, the number of the required real FEIs can be greatly reduced. SAEAs received increasing attentions in recent years, and several types of surrogate models, including polynomial regression (PR), support vector regression (SVR), Gaussian process (GP) regression, and RBF, were developed [10-13, 26]. Among these models, PR is easy to train but is generally of low estimation accuracy, SVR is able to relieve the ‘curse of dimensionality’ but has difficulty in tackling large scale samples, while GP can fit complex response surface well but asks long training time. Besides, GP shows dramatic performance deterioration as the problem dimension increases. As for RBF, it is easy to train and is relatively robust to the change of problem dimension [12]. Profiting from this excellent performance, now RBF is most widely employed by SAEAs.

No matter which type of surrogate models are adopted, it is essentially impossible to build a fixed surrogate model that can correctly evaluate all the solutions generated during the whole optimization process due to the possible high dimensionality, ill distribution, and the limited number of training samples. To weaken the negative influence of the evaluation error, it is necessary to reevaluate part of solutions with the real simulation model. Based on these solutions, the surrogate model could be refined and the optimizer is more likely to be guided into promising solution regions. This brings the so-called model management problem, i.e., determining which solutions should be reevaluated by the real simulation model. Now two classes of model management strategies have been developed. One is the generation-based strategy which employs the real simulation model for fitness evaluation at some specified generations [27], and the other is the individual-based strategy which employs the real simulation model to evaluate some selected individuals at every generation [28]. These generations and individuals are generally specified according to a static rule, a random rule, or even an adaptive rule. It is revealed that the individual-based model management strategy tends to be more suited to steady state evolution or generational evolution implemented on a single machine. By contrast, the generation-based model management strategy is better for parallel implementation on heterogeneous machines of different speeds [11].

According to the solution regions covered by surrogate models, existing SAEAs can be divided into three categories, including global-surrogate assisted EAs [26], local-surrogate assisted EAs [29], and ensemble-surrogate assisted EAs [13]. The global surrogate model tries to model the whole solution space and can generally endow SAEA with strong exploration ability. However, it cannot provide accurate enough fitness estimations. The local surrogate model aims at the current search region of EA. Compared with the global surrogate model, it is more likely to produce more accurate fitness estimations, but can hardly help EA escape from local optima. By contrast, the ensemble surrogate model takes the advantages of both the global and the local surrogate models by integrating them together and has been shown to be able to outperform a single kind of model in most cases.

It has been proved that SAEAs really take effect on computationally expensive optimization problems. However, they were mainly verified on small scale problems of dimensions lower than 30 [10, 13]. In recent years, some researchers tried to scale up surrogate models to medium scale problems. Liu et al. [26] proposed a GP assisted EA which achieves good performance on problems of 50 dimensions with the help of a dimension reduction technique. To the best of our knowledge, the highest dimension...
tackled by SAEAs is 100. The corresponding algorithm is surrogate-assisted cooperative swarm optimization [30], which cooperatively employs a surrogate-assisted PSO and a surrogate-assisted social learning based PSO to search the global optimum. However, with respect to large scale problems of dimensions up to 1000, all the existing surrogate models will lose their efficiency if they are applied in a straightforward way. This study makes the first attempt to scale up surrogate models to LSOPs of 1000 dimensions by introducing them into CC. Within the CC framework, it is possible to construct sufficiently accurate surrogate models for lower dimensional sub-problems obtained through decomposition, and these surrogate models can be expected to significantly improve the evaluation efficiency of a vast number of sub-solutions generated during the CC process.

3 Description of SACC

Many real-world LSOPs are difficult to tackle but possess an appealing feature, i.e., separability, where partially additive separability is the most common type and is also most extensively studied in the CC research field [20–23]. The definition of additive separability can be described as follows:

Definition 1 A function is partially additively separable if it has the following general form:

\[ f(\mathbf{x}) = \sum_{g=1}^{k} f_g(\mathbf{x}_g), \]  

where \( \mathbf{x} = (\mathbf{x}_1^T, \mathbf{x}_2^T, ..., \mathbf{x}_T^T)^T \) is a global decision vector of \( n \) dimensions, \( \mathbf{x}_g \) is a column decision vector of \( f_g(\cdot) \) with different \( \mathbf{x}_g (g = 1, 2, \ldots, k) \) being mutually exclusive, and \( k \) is the number of independent sub-components [20].

CC solves a high dimensional problem \( f(\mathbf{x}) \) by first identifying all the mutually exclusive sub-components \( \mathbf{x}_g (g = 1, 2, \ldots, k) \) and then cooperatively optimizing the resulting lower dimensional sub-problems \( f_g(\mathbf{x}_g) \). Figure 1 shows the general framework of the traditional CC, where it sequentially initializes the parameters of the chosen optimizer, the best overall solution \( \mathbf{x}^* \) obtained so far, and the population \( P_g \) for each sub-problem after the decomposition operation, then it enters a cycle process. During each cycle, a sub-problem \( g \) is first selected generally in a round-robin fashion, then a child population \( U_g \) is generated for the selected \( g \)th sub-problem based on its current population \( P_g \). After evaluation, some good solutions in \( U_g \) are finally employed to update \( P_g \) and \( \mathbf{x}^* \). This process of selection, generation, evaluation, and updating is cycled until all the available computation resources are exhausted.

It is notable that the expression of each \( f_g(\cdot) \) is unknown as \( f(\cdot) \) is a black-box optimization problem in the context of CC. To evaluate the solutions of each sub-problem, the traditional CC takes a context-vector-based method, where the context vector is generally set as the best overall solution \( \mathbf{x}^* \) [4]. More concretely, it evaluates a sub-solution \( \mathbf{x}_g^* \) by means of \( f(\mathbf{x}_g^*|\mathbf{x}_g) \), where \( \mathbf{x}_g^*|\mathbf{x}_g \) denotes the complete solution that replaces the corresponding sub-component of \( \mathbf{x}^* \) with \( \mathbf{x}_g^* \). It is obvious that this method needs to access the original high dimensional simulation model of \( f(\cdot) \) when evaluating each low dimensional sub-solution thus requiring many computation resources. On the other side, the available computation resources are very limited for real word LSOPs. Therefore, the traditional CC can hardly obtain high-quality solutions for them. Besides, when updating the population for a sub-problem, some optimizers, such as the commonly-used DE, need to compare the quality of the current individuals with that of the corresponding child individuals. In general, the current individuals have been evaluated at past iterations, when the context sub-vectors provided by the other sub-problems may be different from the new ones (denote the corresponding old context vector and the new one as \( \mathbf{x}^* \) and \( \mathbf{x}^{*'} \), respectively). In this case, the FV of a current individual \( f(\mathbf{x}_g^*|\mathbf{x}_g) \) and that of the corresponding child individual \( f(\mathbf{x}_g^{*'}|\mathbf{x}_g^*) \) are incommensurable. A straightforward way to tackle this issue is to reevaluate the current individual \( \mathbf{x}_g^* \) by computing \( f(\mathbf{x}_g^{*'}|\mathbf{x}_g) \), but this will consume extra computation resources. Aiming at this issue, the traditional CC persistently optimizes a selected sub-problem for a certain number of consecutive iterations, during which the context sub-vectors provided by the other sub-problems remain unchanged [6, 16]. However, this approach still needs to reevaluate the individuals of a sub-problem at the first iteration when it is selected to be optimized in each cycle. Moreover, it also reduces the interaction frequency among different sub-problems, which is adverse to improving the performance of CC [18].

3.1 The framework of SACC

From the above explanations, it can be known that how to efficiently evaluate sub-solutions has become a key issue in CC. This issue owns the following two characteristics: 1) Compared with the original LSOP, the sub-problems obtained via decomposition are usually small and medium
scale problems, but the evaluations of their solutions are still computationally expensive; 2) For a sub-problem, the goal of evaluation is not to get the accurate FV of each solution, but to identify which solution is better among a certain number of candidates. It has been shown that surrogate models fit well to these kind of problems, but they have never been applied to LSOPs. As the first attempt, SACC can be expected to scale up surrogate models to LSOPs.

Figure 2 presents the framework of SACC, from which it can be seen that SACC inherits the main framework of the traditional CC, but introduces a surrogate model construction (updating) module and a related model management module. Concretely, SACC specially maintains an external archive $D_g$ for each sub-problem $g (g = 1, 2, \ldots, k)$ to record a certain number of really evaluated sub-solutions and the corresponding evaluation values. Based on these sub-solutions, SACC constructs a surrogate model $S_g$ and employs it to preliminarily evaluate all the solutions generated for the $g$th sub-problem at each iteration. Then SACC selects a small number of promising sub-solutions according to the preliminary evaluation results and the model management strategy. After undergoing real evaluation, these promising sub-solutions are in turn utilized to update the corresponding external archive $D_g$.

It is by this iterative process that the surrogate model for each sub-problem become more and more accurate and the number of real evaluations required by CC could be greatly reduced.

By comparing Figs. 1 and 2, it can be observed that SACC cancels the inner loop customized for the traditional CC. As explained above, the main goal of introducing the inner loop into the traditional CC is to avoid reevaluating the solutions to a sub-problem with the original simulation model when the context sub-vectors provided by the other sub-problems change. SACC achieves this in a simpler way, i.e., evaluating a solution by investigating its fitness improvement to the best overall solution $x^*$ instead of its fitness value itself. Taking the minimization problem as an example, SACC defines the fitness improvement of a solution $x_g$ to the $g$th sub-problem as $e(x_g) = f(x^*) - f(x^*|x_g)$. According to the property of additive separability defined by (1), it can be known that $e(x_g)$ eliminates the influence of the other sub-problems and can remain fixed even when the other sub-components in $x^*$ change. Benefiting from this property, the interaction frequency among different sub-problems can be arbitrarily increased without reevaluating sub-solutions with the original simulation model. SACC adopts the highest interaction frequency, i.e., just allowing a selected sub-problem to undergo a single iteration during each coevolution cycle,
since a high interaction frequency contributes to enhancing the performance of CC [18]. Besides, it will be revealed below that the new evaluation indicator is also beneficial to improving the evaluation accuracy of surrogate models.

SACC provides an open framework that makes surrogate model and CC complement each other. To implement a concrete SACC algorithm, it is necessary to specify the type of surrogate model, the model management strategy, and the optimizer for sub-problems. This study develops a SACC algorithm named RBF-SHADE-SACC by integrating the well known RBF model [13] and the efficient SHADE [14] into SACC. Next, we will sequentially present the tailored RBF, SHADE, and the procedure of RBF-SHADE-SACC in the following three sub-sections.

### 3.2 RBF model for SACC

As reviewed in Section 2.2, several types of surrogate models have been developed by now, among which RBF is easier to train, more robust to different problem dimensions, and is also more widely applied in the traditional SAEAs [13, 31–33]. Considering all these advantages, this study employs the RBF model given in [31] as the surrogate model for SACC. RBF is essentially an instance-based learning method. Given \( d \) training samples \( t_1^g, t_2^g, \ldots, t_d^g \in \mathbb{R}^s \) for a sub-problem \( g \) of \( s \) dimensions, the evaluation value provided by RBF for a new sub-solution \( x^g \in \mathbb{R}^s \) can be represented as

\[
\hat{e}(x^g) = \sum_{i=1}^{d} \omega_i \phi(||x^g - t_i^g||) + \beta^T x^g + \alpha, \tag{2}
\]

where \( ||x^g - t_i^g|| \) denotes the Euclidean distance between the two sub-solutions, \( \phi(\cdot) \) denotes the basis function, and \( (\omega_1, \omega_2, \ldots, \omega_d)^T = \omega \in \mathbb{R}^d, \beta \in \mathbb{R}^s, \) and \( \alpha \in \mathbb{R} \) are corresponding parameters. There are several different choices for the basic function \( \phi(\cdot) \), such as the cubic basis function, the Gaussian basis function, and the multi-quadric function [32]. This study adopts the cubic basis function, i.e., \( \phi(r) = r^3 \), since it was shown to be more suitable for SAEAs than other functions. As for \( \beta^T x^g + \alpha \), it is a polynomial tail appended to the standard RBF.

With the given training samples, the parameters \( \omega \in \mathbb{R}^d, \beta \in \mathbb{R}^s, \) and \( \alpha \in \mathbb{R} \) can be obtained by solving the following linear system of equations:

\[
\begin{pmatrix}
\Phi & Q \\
Q^T & 0
\end{pmatrix}
\begin{pmatrix}
\omega \\
\gamma
\end{pmatrix}
=
\begin{pmatrix}
e \\
0
\end{pmatrix}, \tag{3}
\]

where \( \Phi \) is a \( d \times d \) matrix with \( \Phi_{ij} = \phi(||t_i^g - t_j^g||) \), \( \gamma = (\beta^T, \alpha)^T \), \( e = (e(t_1^g), e(t_2^g), \ldots, e(t_d^g))^T \) with each \( e(t_i^g) \) being the real evaluation value of \( t_i^g \), and

\[
Q^T = \begin{pmatrix}
t_1^g & t_2^g & \cdots & t_d^g \\
1 & 1 & \cdots & 1
\end{pmatrix}. \tag{4}
\]

It can be proved that the square matrix in the left hand of (3) is invertible if and only if \( \text{rank}(Q) = s + 1 \) [31]. In this case, the linear system of equations has a unique solution, then (2) can be used as the approximate
model to evaluate a new sub-solution \( \mathbf{x}_g \in \mathbb{R}^d \). To satisfy this condition, the number of training samples \( d \) must be greater than or equal to \( s + 1 \). How to set \( d \) based on this basic rule affects the performance of SACC much, since SACC usually involves many sub-problems of different dimensions. For a given sub-problem, a larger \( d \) generally leads to a more accurate RBF model, but it also asks for more computation resources and may result in overfitting. To weaken the sensitivity of RBF to \( d \), SACC does not build a fixed RBF model for each sub-problem, but first constructs an initial model and then continually updates it with new samples. To achieve this, SACC utilizes the external archive \( D_g \) for each sub-problem \( g \) to store its newest \( d \) sub-solutions evaluated by the original simulation model. \( D_g \) is generally initialized with \( d \) randomly generated sub-solutions and is continually updated with some new sub-solutions generated at each iteration. Once \( D_g \) is updated, the RBF model will be retrained accordingly. By this means, SACC can get a coarse global RBF model for each sub-problem at its initial search stage which could filter out some local optima, and gradually make the model become more and more accurate as new samples are introduced.

When evaluating a solution with surrogate model, the traditional SAEAs usually directly estimate its FV. However, this straightforward approach makes against constructing accurate surrogate models for the sub-problems in SACC. The main reasons consist in that: 1) each sub-problem does not have an explicit objective function, and 2) the indirect FVs \( f(x^*|x_g) \) of sharply different solutions \( x_g \) to the same sub-problem generally show small differences, since a \( f(x^*|x_g) \) adds the FVs of the solutions to all the sub-problems together. To alleviate this issue, SACC measures the quality of each training sample \( t_g \) according to its fitness improvement to the best overall solution \( x^* \), i.e., \( \varepsilon(t_g) = f(x^*) - f(x^*|t_g) \). The fitness improvements enlarge the relative difference among the quality of different training samples of a sub-problem as it eliminates the influence of other sub-problems. Accordingly, the evaluation value \( \varepsilon(x_g) \) provided by the trained RBF as in (2) predicts the fitness improvement of a new sub-solution \( x_g \) to the best overall solution \( x^* \). Obviously, a sub-solution of larger fitness improvement can be considered better.

### 3.3 SHADE for SACC

SHADE is an excellent DE variant developed in recent years [14]. It inherits the efficient ‘current-to-pbest/1’ mutation operator from the classic JADE algorithm [34], but further improves JADE with a novel parameter adaptation mechanism. Instead of employing a single pair of parameter means to generate the new mutation factor and crossover rate as JADE, SHADE maintains a diverse set of means for each parameter to guide its adaptation. In this way, the negative impact of some poor parameter values can be reduced. SHADE and its variants achieved great success in solving small and medium scale optimization problems [35], but have never been adopted to solve LSOPs. This study scales up SHADE to LSOPs under the SACC framework by employing it as optimizer for the lower dimensional sub-problems obtained through decomposition. SACC not only keeps the key features of SHADE, but also tailors it to a certain extent such that it can adapt to the characteristics of CC and surrogate model. For the convenience of description, we name the tailored SHADE as \( rSHADE \).

For each sub-problem \( g \), \( rSHADE \) maintains a population \( P_g \), which contains \( p \) really evaluated individuals. As the original SHADE, \( rSHADE \) produces a trial vector \( \mathbf{u}_g^t \) (\( i = 1, 2, \cdots, p \)) for each individual \( \mathbf{x}_g^i \) in \( P_g \) at each generation according to the following ‘current-to-pbest/1’ mutation operator and the binomial crossover operator:

\[
\mathbf{v}_g^i = \mathbf{x}_g^i + F_i \left( \mathbf{x}_g^{pbest} - \mathbf{x}_g^i \right) + F_i \left( \mathbf{x}_g^{1} - \mathbf{x}_g^{2} \right),
\]

\[
\mathbf{u}_g^{i,j} = \begin{cases} 
\mathbf{v}_g^{i,j} , & \text{if } \text{rand}(0, 1) \leq CR_i \text{ or } j = j_{\text{rand}} \\
\mathbf{x}_g^i , & \text{otherwise}
\end{cases}
\]

where \( \mathbf{v}_g^i \) denotes the mutation vector of \( \mathbf{x}_g^i \), \( \mathbf{u}_g^{i,j} \) denotes the \( j \)th element of \( \mathbf{u}_g^i \), and all the other symbols have the same meanings as the corresponding ones in the original SHADE. It is notable that the mutation factor \( F_i \) in (5) and the crossover rate \( CR_i \) in (6) are individual-dependent. They are generated based on a historical memory \( M_g \) which contains a certain number of pairs of their means. Besides, \( rSHADE \) also maintains an external archive \( A_g \) which preserves some inferior solutions and together with \( P_g \) provides candidates for \( \mathbf{x}_g^{2} \). The updating of \( M_g \) and \( A_g \) depends heavily on the relative quality of each pair of \( \mathbf{x}_g^i \) and \( \mathbf{u}_g^i \). At each generation, an entry selected in a round-robin fashion from \( M_g \) is updated by the respective means of all the effective \( F_i \) and \( CR_i \) (\( i = 1, 2, \cdots, p \)) that help \( \mathbf{u}_g^i \) outperform the corresponding \( \mathbf{x}_g^i \). On the other side, all the failing \( \mathbf{x}_g^i \) are employed to fill or update \( A_g \).

At this moment, the key issue is to properly judge whether a \( \mathbf{u}_g^i \) is better than the corresponding \( \mathbf{x}_g^i \). For the original SHADE, this can be easily achieved by directly comparing the FVs of the two solutions. Nevertheless, this approach is unacceptable for \( rSHADE \) since SACC does not allow \( rSHADE \) to really evaluate each \( \mathbf{u}_g^i \). On the contrary, if we compare all pairs of \( \mathbf{u}_g^i \) and \( \mathbf{x}_g^i \) completely relying on the constructed RBF model, then the comparison result will be affected too much by the evaluation error of RBF, especially at the initial search stage of SACC. \( rSHADE \)
Table 1: Classification of IEEE CEC2010 benchmark functions

| Functions | Separability     | No. of separable variables | No. of nonseparable variables |
|-----------|------------------|-----------------------------|------------------------------|
| F1−F3    | Separable        | 1000                        | 0 × 50                       |
| F4−F8    | Partially separable | 950                        | 1 × 50                       |
| F9−F13   | Partially separable | 500                        | 10 × 50                      |
| F14−F18  | Partially separable | 0                         | 20 × 50                      |
| F19−F20  | Fully nonseparable | 0                         | 1 × 1000                     |

fulfills this task in a two-step manner. First, it compares each pair of \(u_i^g\) and \(x_i^g\) based on the evaluation results \(\bar{e}(u_i^g)\) and \(\bar{e}(x_i^g)\) provided by RBF. Second, it picks out \(q\) (\(q \ll p\)) best trial vectors, reevaluates them with the original simulation model, and adjusts the comparison results on the corresponding population individuals according to the new evaluation results. This two-step manner avoids most of the real evaluations on the one hand, and also ensures the evaluation accuracy on high-quality sub-solutions on the other hand.

As for the population updating operation, in contrast to the original SHADE which always replaces a population individual with the corresponding successful trial vector, \(t\) SHADE neglects the relationship between a population, individuals and its trial vector, and updates some bad individuals with the \(q\) reevaluated trial vectors if the latter achieve larger fitness improvements. By this means, the population of each subproblem essentially keeps the \(p\) best sub-solutions that are really evaluated by the original simulation model, and the risk brought by the evaluation error of RBF can be greatly reduced. As indicated in 3.2, the RBF archive \(D_g\) of each sub-problem \(g\) also needs to be updated in time to learn a more and more accurate RBF model. SACC achieves this by replacing the \(q\) oldest individuals in \(D_g\) with the \(q\) new sub-solutions which are really evaluated at each generation.

According to the updating strategies for each population \(P_g\) and archive \(D_g\), it can be known that the sub-solutions therein are generally introduced at different generations, during which the context vectors, i.e., \(x^*\), may change. Without fine intervention, the fitness improvements of these sub-solutions are still incommensurable since they may take different baselines. To solve this issue, once the best overall solution \(x^*\) is improved by a better solution \(x'^*\), SACC updates the fitness improvements of all the sub-solutions in \(D_g\) and \(P_g\) as follows:

\[
\text{if } \Delta f = f(x^*) - f(x'^*) > 0, \text{ then } x^* \leftarrow x'^* \text{ and } e(x_g) \leftarrow e(x_g) - \Delta f, \text{ for } \forall x_g \in D_g \cup P_g.
\]

It is also in this way that the reevaluation issue concerned in the traditional CC can be avoided.

### 3.4 The procedure of RBF-SHADE-SACC

By integrating RBF and \(t\) SHADE into the SACC framework, we can get the procedure of RBF-SHADE-SACC as presented in Algorithm 1. Steps 2-7 mainly perform initialization operations, where step 5 initializes the parameter memory \(M_g\) with the same rule as the one in the original SHADE [14], but initializes the external archive \(A_g\) in a different way. The original SHADE initializes its external archive with an empty set and continually fills or updates it with population individuals which are worse than the corresponding trial vectors. This asks SHADE to perform different operations according to the real number of solutions in the archive during the whole evolution process, although the archive will be definitely fully filled after several generations. To simplify these operations, Step 5 directly initializes \(A_g\) with a specified number of random sub-solutions. This minor change significantly simplifies the implementation of SHADE without affecting its performance. As for the external archive \(D_g\) of RBF and the
population $P_g$ of tSHADE, they require $d$ and $p$ really evaluated sub-solutions for initialization, respectively. To reduce the number of real evaluations, step 6 directly generates $\max(d, p)$ sub-solutions which are further assigned to $D_g$ and $P_g$ in step 7.

After the initialization operations, step 8 selects a sub-problem to optimize according to a specified rule. The selection rule essentially determines the computation resource allocation among different sub-problems. Although several different selection rules have been proposed by now

### Algorithm 1 RBF-SHADE-SACC

1. Generate a decomposition $x \rightarrow \{x_1, x_2, \ldots, x_k\}$;
2. Initialize the best overall solution $x^*$ with a randomly generated complete solution;
3. for sub-problem $g = 1: k$ do
   4. Initialize the parameters of RBF and tSHADE, including $d$, $p$, and $q$;
   5. Initialize all the entries in $M_g$ to 0.5 and initialize $A_g$ with $p$ random sub-solutions;
   6. Randomly generate $\max(d, p)$ sub-solutions $x_g$ and evaluate them with $e(x_g)$;
   7. Initialize $D_g$ and $P_g$ with $d$ and $p$ generated sub-solutions, respectively;
8. Select a sub-problem $g$ to optimize according to a specified rule;
9. Build a RBF model for the $g$th sub-problem with $D_g$ according to (2)-(4);
10. for each sub-solution $x'_g \in P_g$ do
11.     Generate a pair of control parameters $F_i$ and $CR_i$ based on $M_g$;
12.     Generate a trial vector $u'_g$ according to the “current-to-pbest/1/bin” rule presented by (5)-(6);
13.     Evaluate $x'_g$ and $u'_g$ with $\bar{e}(x'_g)$ and $\bar{e}(u'_g)$ provided by RBF, respectively;
14.     Select $q$ best trial vectors from the group of $u'_g, i = 1, 2, \ldots, p$ and store them into $Q_g$;
15.     Reevaluate each trial vector $u'_g \in Q_g$ with $e(u'_g)$ and modify $\bar{e}(u'_g) \leftarrow e(u'_g)$;
16.     for each sub-solution $x'_g \in P_g$ do
17.         if $\bar{e}(u'_g) > \bar{e}(x'_g)$ then
18.             Replace a randomly selected sub-solution in $A_g$ with $x'_g$;
19.             Record the corresponding control parameters $F_i$ and $CR_i$;
20.     Update an entry in $M_g$ based on the recorded successful control parameters;
21.     Update the $q$ oldest sub-solutions in $D_g$ with the trial vectors in $Q_g$;
22.     for each $u'_g \in Q_g$ do
23.         Find out the worst sub-solution $x'_g$ in $P_g$;
24.         if $e(x'_g) < e(u'_g)$ then
25.             Delete $x'_g$ from $P_g$ and insert $u'_g$ into $P_g$;
26.     Find out the best sub-solution $x'_g$ in $P_g$;
27.     if $e(x'_g) > 0$ then
28.         Update $f(x^*) \leftarrow f(x^* | x'_g)$, $x^* \leftarrow x^* | x'_g$;
29.         for each sub-solution $x'_g \in D_g \cup P_g$ do
30.             Update $e(x'_g) \leftarrow e(x'_g) - e(x'_g)$;
31.     if termination condition is not met then goto step 8;
32. Output $x^*$, $f(x^*)$. 
Simulation model. As a result, nearly evaluating for a sub-problem at each iteration, but it only requires CC algorithms, RBF-SHADE-SACC generates p solutions. From Algorithm 1, it can be seen that as the traditional CC algorithms, RBF-SHADE-SACC generates p solutions for a sub-problem at each iteration, but it only requires evaluating q \( q \ll p \) sub-solutions with the original simulation model. As a result, nearly \( (p - q)/p \times 100\% \) of computation resources can be saved since the simulation process consumes most of the computation resources provided to the whole optimization algorithm. Besides, RBF-SHADE-SACC is designed strictly according to the SACC framework shown in Fig. 2. It scales up RBF and SHADE to LSOPs on the one hand, and can be utilized to verify the effectiveness of SACC on the other hand.

4 Experimental Studies

4.1 Experimental settings

The IEEE CEC2010 benchmark suite [38] which contains 20 LSOPs was employed in our experiments. All these benchmark functions are minimization problems of 1000 dimensions. Table 1 presents their classification in terms of their separability, where the nonseparable sub-problems in partially separable functions all involve 50 decision variables. For more details about these functions, readers can refer to [38]. From [38], it can be known that \( F_{10} \) and \( F_{20} \) are fully nonseparable functions. All kinds of CC algorithms have no advantage on them in comparison with the traditional EAs, therefore they were excluded from our experiments.

In order to perform an unbiased analysis on RBF-SHADE-SACC, ideal decomposition was employed in our experiments, which means that all the decision variables of a benchmark function were manually grouped into some sub-problems according to the prior knowledge of the function. As suggested by [38], most existing CC algorithms take a maximum number of \( 3.0 \times 10^6 \) real FEs as the termination condition of a run. To show the superiority of RBF-SHADE-SACC, each of our experiments only utilized 10 percent of the suggested computation resources, i.e., a maximum number of \( 3.0 \times 10^5 \) real FEs, as the default termination condition of a run. Unless otherwise mentioned, the results of each algorithm on a function was calculated based on 25 independent runs.

4.2 Parameter settings

There are several parameters in RBF-SHADE-SACC. Besides the archive size \( d \) of RBF, the population size \( p \) of \( r \)SHADE, and the number of elitist sub-solutions selected at each generation \( q \), it also needs another parameter \( s \) to specify how to divide separable variables concerned in separable and partially separable functions such as \( F_1-F_3 \). It is understandable that separable variables could be divided in any way without affecting their theoretical optima. However, if all the separable variables are grouped into a single large scale sub-problem, then the advantage of CC will be weakened and this will also raise the difficulty in building an accurate enough RBF model. On the contrary, if all the separable variables are independently treated, the difficulty in building RBF models will be alleviated, but the limited number of real FEs will have to be assigned to so many sub-problems that RBF-SHADE-SACC will be most likely not to converge on some sub-problems. From above analysis, it can be known that the key of dividing separable variables is to balance the difficulty in constructing RBF models and the quantity of computation resources assigned to each sub-problem. To numerically investigate the influence of \( s \), we tested RBF-SHADE-SACC with different \( s \) values selected from \{10, 20, 50, 100, 200\}.

As for the archive size \( d \) of RBF, its influence has been extensively analyzed in Section 3.2. Since RBF-SHADE-SACC continually updates the RBF model constructed for each sub-problem in an iteration-wise way, its sensitivity to \( d \) could be greatly weakened. According to the suggestion given in [33], we set \( d = 5s \) for the RBF related to each sub-problem. As a common parameter, the population size \( p \) has been investigated much in the original SHADE [14], and it is revealed that the algorithm performs well on most small and medium scale problems when the population size is set to 100. This conclusion was also verified by our pilot experiments. Accordingly, \( p \) was
fixed to 100 for different sub-problems concerned in RBF-SHADE-SACC. As for \( q \), it is a new parameter introduced by RBF-SHADE-SACC. The larger \( q \) is, the more easily an accurate RBF model can be obtained since the more really evaluated sub-solutions will be employed to update the RBF archive at each generation, but the faster the available computation resources will be exhausted, which may be adverse to the convergence of RBF-SHADE-SACC. To numerically investigate the influence of \( q \), we tested RBF-SHADE-SACC with different \( q \) values selected from \{1, 5, 10, 15, 20\}.

Taking the separable functions \( F_1 \) and \( F_3 \) and the partially separable functions \( F_{10} \) and \( F_{13} \) as examples, Fig. 3 shows the average FVs obtained by RBF-SHADE-SACC when \( s \) varies and \( q \) is fixed to 10. It can be observed that, with the variation of \( s \), RBF-SHADE-SACC shows sensitive performance on the two separable functions \( F_1 \) and \( F_3 \). On the whole, a relatively small \( s \) helps RBF-SHADE-SACC find superior solutions. However, an extremely small \( s \) may destroy its performance on \( F_1 \). Due to this reason, we recommend to set \( s = 20 \) for separable functions. For each of the partially separable functions \( F_{10} \) and \( F_{13} \), RBF-SHADE-SACC needs to group the 500 separable variables therein according to \( s \). Although its performance on these two functions is not affected too much by \( s \) as that on \( F_1 \), it demonstrates completely opposite performance trends on these two functions when \( s \) varies. Based on this observation, we recommend to set \( s = 100 \) for partially separable functions to balance the performance of RBF-SHADE-SACC on this kind of functions.

By keeping \( s \) at the recommended value, Fig. 4 presents the influence of \( q \). It is surprising to see that, for each of the test functions, the performance of RBF-SHADE-SACC generally improves with the increase of \( q \) if it deteriorates with the increase of \( s \), and vice versa. By comparing the basic functions concerned in these test functions, it can be revealed that the settings of \( q \) and \( s \) on LSOPs are directly affected by the landscape complexity of the basic functions. For example, the separable function \( F_3 \) takes Ackley function as its basic function which itself is a complicated multimodal function [38]. To generate a RBF model of acceptable accuracy for each sub-problem, RBF-SHADE-SACC has to reduce the dimension of each sub-problem and really evaluate more sub-solutions at each generation. On the contrary, the partially separable function \( F_{13} \) takes Sphere function and Rosenbrock function as the basic functions for the separable and the nonseparable variables, respectively [38]. The landscapes of both basic functions are so simple that RBF-SHADE-SACC tends to increase \( s \) and decrease \( q \), so that it can undergo more generations without significantly sacrificing the accuracy of each RBF model. Despite this meaningful conclusion, we do not know the basic functions concerned in a black-box LSOP, not to mention their landscapes. For this reason, this study sets \( q \) to a fixed value of 10 to balance the performance of RBF-SHADE-SACC on different functions.

### 4.3 Comparison between RBF-SHADE-SACC and other CC algorithms

To evaluate the performance of RBF-SHADE-SACC, we compared it with four CC algorithms. One is an existing CC algorithm developed in [36] with a name of CC-I. It takes the traditional CC framework and employs another efficient DE variant called SaNSDE as optimizer [25], and is used as the baseline for comparison. The other three CC algorithms are named SHADE-CC, SaNSDE-CC, and RBF-SaNSDE-SACC. They were specifically designed to investigate the effectiveness of each algorithmic component in RBF-SHADE-SACC. The difference between SHADE-CC and RBF-SHADE-SACC only lies in that the former removes RBF model and evaluates all the sub-solutions with the original simulation model. SaNSDE-CC and RBF-SaNSDE-SACC can be regarded as the counterparts of SHADE-CC and RBF-SHADE-SACC, respectively. They take SaNSDE as optimizer instead of SHADE. To ensure the fairness of the comparison, the parameters of SaNSDE in SaNSDE-CC and RBF-SaNSDE-SACC were set according to the recommendation in the original paper [25], and the other parameters therein and the ones in SHADE-CC were set to the same values as the corresponding ones in RBF-SHADE-SACC.

Table 2 reports the results obtained by SaNSDE-CC, RBF-SaNSDE-SACC, SHADE-CC, and RBF-SHADE-SACC with \( 3.0 \times 10^5 \) and \( 1.0 \times 10^5 \) real FEs and the results obtained by CC-I with \( 3.0 \times 10^6 \) real FEs. It is necessary
Table 2 The average FVs ±standard deviations obtained by CC-I, SaNSDE-CC, RBF-SaNSDE-SACC, SHADE-CC, and RBF-SHADE-SACC on IEEE CEC2010 benchmark functions over 25 independent runs

| Functions | CC-I          | SaNSDE-CC     | RBF-SaNSDE-SACC | SHADE-CC       | RBF-SHADE-SACC |
|-----------|---------------|---------------|-----------------|----------------|----------------|
|           | 3.5e + 11 ± 2.0e + 10^- | 1.35e + 08 ± 1.50e + 07^- | 6.86e + 03 ± 6.02e + 03^- | 1.34e + 07 ± 7.70e + 05^- | 4.33e + 03 ± 3.88e + 03 |
| F2        | 9.4e + 03 ± 2.1e + 02^- | 4.98e + 03 ± 6.92e + 01^- | 1.50e + 03 ± 7.24e + 01^- | 4.84e + 03 ± 6.17e + 01^- | 1.29e + 03 ± 5.40e + 01 |
| F3        | 2.0e + 01 ± 4.4e - 02^- | 1.58e + 01 ± 1.32e - 01^- | 1.30e + 01 ± 1.83e - 01^- | 1.48e + 01 ± 1.66e - 01^- | 1.28e + 01 ± 1.83e - 01 |
| F4        | 3.4e + 14 ± 7.5e + 13^- | 9.73e + 13 ± 1.98e + 13^- | 2.68e + 12 ± 1.43e + 12^- | 2.13e + 13 ± 5.86e + 12^- | 7.42e + 11 ± 2.47e + 11 |
| F5        | 4.9e + 08 ± 2.4e + 07^- | 3.89e + 08 ± 2.25e + 07^- | 1.45e + 08 ± 2.86e + 07^- | 3.34e + 08 ± 1.81e + 07^- | 1.13e + 08 ± 2.31e + 07 |
| F6        | 1.1e + 07 ± 7.5e + 05^- | 2.45e + 06 ± 1.73e + 05^- | 1.98e + 06 ± 1.06e + 06^- | 3.04e + 03 ± 6.22e + 02^- | 4.14e + 05 ± 6.35e + 05 |
| F7        | 7.7e + 10 ± 9.6e + 09^- | 2.70e + 10 ± 4.31e + 09^- | 3.05e + 08 ± 6.17e + 08^- | 7.67e + 09 ± 2.52e + 09^- | 6.23e + 07 ± 1.17e + 08 |
| F8        | 1.8e + 14 ± 9.3e + 13^- | 9.32e + 09 ± 5.49e + 09^- | 6.87e + 07 ± 4.77e + 07^- | 7.61e + 07 ± 4.38e + 07^- | 3.99e + 07 ± 5.57e + 07 |
| F9        | 9.4e + 08 ± 7.1e + 07^- | 1.29e + 09 ± 8.32e + 07^- | 2.79e + 07 ± 3.39e + 06^- | 3.41e + 08 ± 2.92e + 07^- | 1.19e + 07 ± 1.48e + 06 |
| F10       | 4.8e + 03 ± 6.7e + 01^- | 8.26e + 03 ± 7.76e + 01^- | 3.10e + 03 ± 1.29e + 02^- | 7.82e + 03 ± 9.34e + 01^- | 2.69e + 03 ± 1.28e + 02 |
| F11       | 4.1e + 01 ± 1.5e + 00^- | 5.97e + 01 ± 1.04e + 00^- | 3.60e + 01 ± 2.64e + 00^- | 1.66e + 01 ± 2.43e - 01^- | 2.28e + 01 ± 1.92e + 00 |
| F12       | 4.9e + 05 ± 3.4e + 04^- | 4.85e + 05 ± 2.17e + 04^- | 1.21e + 04 ± 2.81e + 03^- | 1.85e + 05 ± 1.14e + 04^- | 1.89e + 03 ± 1.43e + 03 |
| F13       | 1.5e + 07 ± 4.1e + 06^- | 4.47e + 06 ± 6.43e + 05^- | 9.94e + 02 ± 4.41e + 02^- | 4.14e + 03 ± 2.25e + 03^- | 6.32e + 02 ± 3.05e + 02 |
| F14       | 2.7e + 07 ± 2.1e + 06^- | 4.33e + 09 ± 2.12e + 08^- | 8.07e + 07 ± 5.65e + 06^- | 1.36e + 09 ± 6.37e + 07^- | 3.59e + 07 ± 2.79e + 06 |
| F15       | 4.0e + 03 ± 1.6e + 02^- | 8.86e + 03 ± 1.07e + 02^- | 2.92e + 03 ± 1.46e + 02^- | 8.04e + 03 ± 9.38e + 01^- | 2.18e + 03 ± 7.89e + 01 |
| F16       | 2.0e + 01 ± 4.0e + 00^- | 1.70e + 02 ± 6.84e + 00^- | 4.14e + 01 ± 2.67e + 00^- | 2.33e + 01 ± 1.23e + 00^- | 1.18e + 01 ± 3.28e + 00 |
| F17       | 2.2e + 01 ± 3.7e + 01^- | 1.31e + 06 ± 5.05e + 04^- | 5.88e + 04 ± 9.12e + 03^- | 5.56e + 05 ± 1.68e + 04^- | 1.38e + 04 ± 4.30e + 03 |
| F18       | 1.0e + 03 ± 1.7e + 02^- | 1.76e + 08 ± 1.44e + 07^- | 2.18e + 03 ± 6.24e + 02^- | 9.31e + 04 ± 1.87e + 04^- | 1.43e + 03 ± 4.37e + 02 |
| No.of +/− | 3/0/15        | 0/0/18        | 0/4/14          | 2/0/16          | -              |
Table 2  (continued)

| Functions | CC-I     | SaNSDE-CC  | RBF-SaNSDE-SACC | SHADE-CC   | RBF-SHADE-SACC   |
|-----------|----------|------------|-----------------|------------|-----------------|
| No. of real FEs = 1.0e + 05  | 3.5e + 11±2.0e + 10  | 5.06e + 09 ± 4.68e + 08  | 1.06e + 07 ± 1.27e + 06  | 1.65e + 09 ± 1.12e + 08  | 6.89e + 06 ± 1.03e + 06  |
| F1  |         |            |                 |            |                 |                      |
| F2  | 9.4e + 03±2.1e + 02  | 8.71e + 03 ± 9.98e + 01  | **1.61e + 03±7.47e + 01**  | 7.10e + 03 ± 9.90e + 01  | 1.81e + 03 ± 1.46e + 02  |
| F3  | 2.0e + 01±4.4e − 02  | 1.86e + 01 ± 8.67e − 02  | 1.46e + 01 ± 1.65e − 01  | 1.70e + 01 ± 1.46e − 01  | 1.42e + 01 ± 1.98e − 01  |
| F4  | 3.4e + 14±7.5e + 13  | 4.63e + 14 ± 7.10e + 13  | 1.30e + 13 ± 5.69e + 12  | 1.79e + 14 ± 2.37e + 13  | 5.09e + 12 ± 1.77e + 12  |
| F5  | 4.9e + 08±2.4e + 07  | 4.78e + 08 ± 1.81e + 07  | 1.45e + 08 ± 2.86e + 07  | 4.31e + 08 ± 2.29e + 07  | 1.13e + 08±2.31e + 07   |
| F6  | 1.1e + 07±7.5e + 05  | 1.25e + 07 ± 7.08e + 05  | 1.98e + 06 ± 1.06e + 06  | 4.58e + 06 ± 2.39e + 05  | 4.14e + 05±6.35e + 05   |
| F7  | 7.7e + 10±9.6e + 09  | 7.69e + 10 ± 7.66e + 09  | 1.10e + 10 ± 3.49e + 09  | 4.41e + 10 ± 7.21e + 09  | 5.73e + 09±1.92e + 09   |
| F8  | 1.8e + 14±9.3e + 13  | 2.85e + 14 ± 6.59e + 13  | **1.02e + 08±5.40e + 07**  | 1.43e + 12 ± 7.51e + 11  | 6.78e + 07±5.64e + 07   |
| F9  | 9.4e + 08±7.1e + 07  | 8.99e + 09 ± 3.89e + 08  | 1.81e + 08 ± 1.96e + 07  | 2.90e + 09 ± 3.13e + 08  | 9.02e + 07±9.08e + 06   |
| F10 | 4.8e + 03±6.7e + 01  | 1.07e + 04 ± 8.28e + 01  | 3.11e + 03 ± 1.31e + 02  | 9.46e + 03 ± 1.15e + 02  | 2.69e + 03±1.28e + 02   |
| F11 | 4.1e + 01±1.5e + 00  | 1.83e + 02 ± 2.00e + 00  | 3.71e + 01 ± 2.51e + 00  | 9.94e + 01 ± 1.71e + 00  | 2.41e + 01±1.97e + 00   |
| F12 | 4.9e + 05±3.4e + 04  | 1.48e + 06 ± 4.05e + 04  | 2.39e + 05 ± 2.21e + 04  | 7.76e + 05 ± 2.47e + 04  | 1.45e + 05±1.80e + 04   |
| F13 | 1.5e + 07±4.1e + 06  | 1.26e + 10 ± 11.12e + 09  | 2.07e + 03 ± 5.68e + 02  | 2.30e + 08 ± 3.14e + 07  | 1.84e + 03±1.05e + 03   |
| F14 | **2.7e ± 07±2.1e + 06**  | 1.58e + 10 ± 5.55e + 08  | 5.86e + 08 ± 4.78e + 07  | 6.70e + 09 ± 3.52e + 08  | 2.80e + 08±2.02e + 07   |
| F15 | 4.0e + 03±1.6e + 02  | 1.09e + 04 ± 9.21e + 01  | 2.93e + 03 ± 1.46e + 02  | 9.46e + 03 ± 7.77e + 01  | 2.18e + 03±7.89e + 01   |
| F16 | 2.0e + 01±4.0e + 00  | 3.74e + 02 ± 2.60e + 00  | 4.17e + 01 ± 2.64e + 00  | 2.35e + 02 ± 3.14e + 00  | 1.18e + 01±3.28e + 00   |
| F17 | **2.2e ± 01±3.7e + 01**  | 2.64e + 06 ± 7.18e + 04  | 7.06e + 05 ± 5.80e + 04  | 1.65e + 06 ± 4.23e + 04  | 4.66e + 05±5.34e + 04   |
| F18 | **1.0e ± 03±1.7e + 02**  | 7.53e + 10 ± 4.00e + 09  | 3.55e + 04 ± 1.13e + 04  | 3.40e + 09 ± 2.96e + 08  | 1.45e + 04±6.39e + 03   |

No. of +/- ≈ /−  3/0/15  0/0/18  1/1/16  0/0/18

\(^a\) ‘+’, ‘≈’, and ‘−’ denote that the performance of the corresponding algorithm is better than, similar to, and worse than that of RBF-SHADE-SACC, respectively

\(^b\) The results marked in bold indicate they are the best
to mention that the results of CC-I are directly taken from [36]. To statistically analyze the performance of the five competitors, we employed Cohen’s $d$ effect size [39] to quantify the difference among the average FVs obtained by them. Cohen’s $d$ effect size is independent of the sample size and is generally considered ‘small’, ‘medium’, and ‘large’ if its absolute value belongs to $[0.2, 0.3)$, $[0.3, 0.8)$, and $[0.8, +\infty)$, respectively. According to this rule, if a result in Table 2 is judged to be better than, worse than, or similar to the corresponding one obtained by RBF-SHADE-SACC, it is marked with ‘$+$’, ‘$-$’, and ‘$\approx$’, respectively. Besides, if a result is the best among the ones obtained by the five algorithms, it is marked in bold. To show the comparison more clearly, Fig. 5 graphically compares the average FVs achieved by the five algorithms on each benchmark function.

From Table 2 and Fig. 5, it can be seen that RBF-SHADE-SACC demonstrates excellent performance. When a maximum number of $3.0 \times 10^5$ real FEs is allowed, it outperforms CC-I, SaNSDE-CC, RBF-SaNSDE-SACC, and SHADE-CC on 15, 18, 14, and 16 out of total 18 benchmark functions, respectively. The success of RBF-SHADE-SACC profits from its three algorithmic components, i.e., the RBF model, the SHADE optimizer, and the SACC framework. The effect of RBF can be revealed by comparing the results obtained by SHADE-CC and RBF-SHADE-SACC and the ones obtained by SaNSDE-CC and RBF-SaNSDE-SACC, since whether employing RBF is the only difference between each pair of algorithms. From the reported results, it can be observed that RBF-SHADE-SACC surpasses SHADE-CC on all the benchmark functions expect $F_6$ and $F_{11}$. Especially on $F_1$, $F_4$, $F_7$, $F_{12}$, and $F_{14}$, it yields better solutions by two orders of magnitude in terms of the average FV. As for $F_6$ and $F_{11}$, they take Ackley function as basic functions, whose fitness landscape is like a plateau in the solution region close to the global optimum, which is located in a very narrow region near the origin [38]. Then it is very difficult to build accurate enough RBF models for the sub-problems in $F_6$ and $F_{11}$ using a limited number of samples, which restricts the performance of RBF-SHADE-SACC on them. As for SaNSDE-CC and RBF-SaNSDE-SACC, it can be seen from Table 2 and Fig. 5 that the latter has an edge over the former on all the benchmark functions. All these comparison results indicate that RBF really takes a positive effect on the performance of RBF-SHADE-SACC and RBF-SaNSDE-SACC. The underlying reasons are twofold: 1) By taking RBF as surrogate model, the two SACC algorithms are allowed to generate much more candidate solutions when a limited quantity of computation resources are available, and consequently can explore the solution space more thoroughly; 2) RBF is relatively easy to train and tends to become more and more accurate with the updating of the model archive, and thus can efficiently evaluate sub-solutions and filter out the inferior ones.

From the comparison between RBF-SHADE-SACC and RBF-SaNSDE-SACC, it can be concluded that SHADE has an edge over SaNSDE for LSOPs under the SACC framework since RBF-SHADE-SACC differs from RBF-SaNSDE-SACC only in terms of the employed optimizer and it outperforms the latter on 14 out of a total of 18 functions and obtains similar results on the rest 4 functions. This conclusion can also be verified by the comparison between SHADE-CC and SaNSDE-CC. The comparison results show that by employing SHADE as optimizer instead of SaNSDE, SHADE-CC improves SaNSDE-CC on all 18 functions. From the reported results, it can also be observed that although SaNSDE-CC is defeated by the other three new algorithms, it surpasses CC-I on more than half of the test functions. This observation is exciting because...
SaNSDE-CC only consumes 10% of the real FEs consumed by CC-I. SaNSDE-CC achieves this mainly by benefiting from another effective strategy in SACC, i.e., evaluating each solution according to its fitness improvement to the best solution obtained so far. It enables SaNSDE-CC to avoid reevaluating solutions with the original simulation model and also significantly increases the interaction frequency among sub-problems.

When the allowed maximum number of real FEs is reduced to $1.0 \times 10^5$, RBF-SHADE-SACC and RBF-SaNSDE-SACC demonstrate more obvious advantages. They both completely surpass their respective counterparts, i.e., SHADE-CC and SaNSDE-CC, and outperform CC-I on 15 and 14 out of total 18 benchmark functions, respectively. This result is very exciting since RBF-SHADE-SACC and RBF-SaNSDE-SACC only consume 1/30 of real FEs consumed by CC-I. It also indicates that the two SACC algorithms are more robust to the decrease of computation resources than SHADE-CC and SaNSDE-CC. As for the comparison between RBF-SHADE-SACC and RBF-SaNSDE-SACC, the former performs no worse than the latter on 17 out of 18 total benchmark functions.

To further verify the efficiency of RBF-SHADE-SACC, we conducted another experiment which counts the average number of real FEs required by SHADE-CC to obtain the same average FV with RBF-SHADE-SACC. Table 3 reports the results when RBF-SHADE-SACC is allowed to undergo $1.0 \times 10^5$ or $3.0 \times 10^5$ real FEs. To make a more clear comparison, Fig. 6 graphically shows the ratio between the number of real FEs required by SHADE-CC and RBF-SHADE-SACC. It can be seen from Table 3 and Fig. 6 that, to achieve the same result, RBF-SHADE-SACC generally requires much fewer real FEs than SHADE-CC. When RBF-SHADE-SACC is allowed to undergo $1.0 \times 10^5$ real FEs, SHADE-CC demands three times the computation resources at least to achieve similar results on all the benchmark functions except $F_6$ and $F_{11}$. Especially on $F_2$, $F_3$, $F_{10}$, and $F_{15}$, it consumes more than ten times of computation resources. When a maximum number of $3.0 \times 10^5$ real FEs is taken as the termination condition for RBF-SHADE-SACC, the inferiority of SHADE-CC is alleviated to a certain extent, but it still requires three times of computation resources at least on 10 out of total 18 benchmark functions and two times of computation resources at least on 5 out of the other 8 benchmark functions.

In order to examine the evolution characteristics of RBF-SHADE-SACC, Fig. 7 compares the evolution curves of the average FVs obtained by SaNSDE-CC, SHADE-CC, RBF-SaNSDE-SACC, and RBF-SHADE-SACC, where functions $F_1$, $F_3$, $F_{10}$, and $F_{13}$ are taken as examples. From Fig. 7, three phenomena can be observed. First, RBF-SHADE-SACC and RBF-SaNSDE-SACC perform

| FEs No. for RBF-SHADE-SACC | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ | $F_9$ | $F_{10}$ | $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ | $F_{15}$ | $F_{16}$ | $F_{17}$ | $F_{18}$ |
|-----------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| $1.0 \times 10^5$           |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
|                             | $3.32 \times 10^5$ | $4.38 \times 10^5$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ | $1.37 \times 10^6$ | $1.12 \times 10^6$ |
| $3.0 \times 10^5$           |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
Surrogate model assisted cooperative coevolution...  

The case when RBF-SHADE-SACC undergoes $1.0 \times 10^5$ real FEs  

The case when RBF-SHADE-SACC undergoes $3.0 \times 10^5$ real FEs

Fig. 6 The ratio between the number of real FEs required by SHADE-CC and RBF-SHADE-SACC to achieve similar results

Slightly worse than SHADE-CC and SaNSDE-CC at the initial search stage. The reason mainly consists in that RBF-SHADE-SACC and RBF-SaNSDE-SACC generate and really evaluate much more random sub-solutions to construct RBF model for each sub-problem during their initialization operations, while SHADE-CC and SaNSDE-CC quickly move to the optimization stage after the simple initialization of each sub-population. Second, RBF-SHADE-SACC and RBF-SaNSDE-SACC yield better solutions than SHADE-CC and SaNSDE-CC after several hundreds of real FEs and keep better evolution trends until all the available computation resources are

Fig. 7 The evolution trends of average FVs obtained by SaNSDE-CC, SHADE-CC, RBF-SaNSDE-SACC, and RBF-SHADE-SACC on $F_1$, $F_3$, $F_{10}$, and $F_{13}$
exhausted. Finally, RBF-SHADE-SACC and SHADE-CC always surpass RBF-SaNSDE-SACC and SaNSDE-CC, respectively, during the whole evolution process.

To sum up, RBF-SHADE-SACC, followed by RBF-SaNSDE-SACC, performs best among all the five tested algorithms. The two SACC algorithms definitely defeat the other three algorithms. Besides the superior performance of themselves, their success also verifies that SACC really is an excellent framework that can make surrogate model and CC complement each other, thus reducing the training difficulty of surrogate model and increasing the sub-solution evaluation efficiency in CC. Furthermore, it indicates that SACC is compatible with different optimizers and SHADE is more efficient than SaNSDE in solving LSOPs within the SACC framework.

5 Conclusion

In this paper, a novel CC framework named SACC is proposed for LSOPs. SACC is mainly characterized by employing a surrogate model assisted sub-solution evaluation method. Different from the traditional CC which evaluates each sub-solution based on a context vector and the original time-consuming simulation model, SACC builds a calculable surrogate model for each sub-problem and employs it to filter out most of the inferior sub-solutions. It does not invoke the original simulation model except for evaluating some high-quality sub-solutions. As a result, its requirement on computation resources can be greatly reduced. This paper also designs a concrete SACC algorithm by introducing RBF and SHADE into the SACC framework. To make RBF and SHADE adapt to the characteristics of LSOP and SACC, some modifications are conducted on them. Experimental results on IEEE CEC2010 benchmark suite demonstrate that SACC can significantly improve the sub-solution evaluation efficiency, and compared with the traditional CC algorithms, the resultant RBF-SHADE-SACC can generate highly competitive solutions even with much fewer computation resources.

The SACC algorithm presented in this paper employs the same surrogate model and the same optimizer for all the sub-problems which may have strikingly different characteristics. Our future work will focus on developing SACC algorithms which can adaptively configure surrogate models and optimizers according to the characteristics of sub-problems. Furthermore, the superiority of SACC is only verified on benchmark functions now. We will also investigate the efficiency of SACC on some real-world LSOPs, where different levels of noises may be concerned.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 61873199, in part by the Postdoctoral Science Foundation of China under Grants 2014M560784 and 2016T90922, and in part by the Fundamental Research Funds for the Central Universities of China.

References

1. Zhou ZH, Chawla NV, Jin Y, Williams GJ (2014) Big data opportunities and challenges: Discussions from data analytics perspectives. IEEE Comput Intell Mag 9(4):62–74
2. Sabar NR, Abawajy J, Yearwood J (2017) Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems. IEEE Trans Evol Comput 21(2):315–327
3. Bellman RE (1957) Dynamic Programming, ser. Dover Books on Mathematics. Princeton University, Princeton
4. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
5. Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: Proceedings of the 3rd Conference on Parallel Problem Solving Nature, pp 249–257
6. Mahdavi S, Shiri ME, Rahnamayan S (2015) Metaheuristics in large-scale global continues optimization: A survey. Inf Sci 295:407–428
7. Wu Z, Zhao M (2016) Leap on large-scale nonseparable problems. In: Proceedings of the IEEE Congress on Evolutionary Computation CEC, pp 1808–1814
8. Tang RL, Li X (2018) Adaptive multi-context cooperatively coevolving in differential evolution. Appl Intell 48(9):2719–2729
9. Peng X, Liu K, Jin Y (2016) A dynamic optimization approach to the design of cooperative co-evolutionary algorithms. Knowl-Based Syst 109(2):174–186
10. Lim D, Jin Y, Ong YS,Sendhoff B (2010) Generalizing surrogate-assisted evolutionary computation. IEEE Trans Evol Comput 14(3):329–355
11. Jin Y (2011) Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm & Evol Comput 1(2):61–70
12. Diaz-Manriquez A, Toscano G, Coello CAC (2017) Comparison of metamodeling techniques in evolutionary algorithms. Soft Comput 21:5647–5663
13. Sun C, Jin Y, Zeng J, Yu Y (2015) A two-layer surrogate-assisted particle swarm optimization algorithm. Soft Comput 19(6):1461–1475
14. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation CEC, pp 71–78
15. Potter MA, De Jong KA (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29
16. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999
17. Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation CEC, pp 1663–1670
18. Omidvar MN, Li X, Yang Z, Yao X (2010) Cooperative coevolution for large scale optimization through more frequent random grouping. In: Proceedings of the IEEE Congress on Evolutionary Computation CEC, pp 1754–1761
19. Omidvar MN, Li X, Yao X (2010) Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation CEC, pp 1–8
20. Omidvar MN, Li XD, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393
21. Mei Y, Omidvar MN, Li X, Yao X (2016) A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. ACM Trans Math Softw 42(2):13:1–24
22. Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) DG2: A faster and more accurate differential grouping for large scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942
23. Ren Z, Chen A, Wang L, Liang Y, Pang B (2017) An efficient vector-growth decomposition algorithm for cooperative coevolution in solving large scale problems. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp 41–42
24. Li X, Yao X (2012) Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans Evol Comput 16(2):210–224
25. Yang Z, Tang K, Yao X (2008) Self-adaptive Differential Evolution with Neighborhood Search. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp 1110–1116
26. Liu B, Zhang Q, Gielen GGE (2014) A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Trans Evol Comput 18(2):180–192
27. Jin Y, Olhofer M, Sendhoff B (2008) A framework for evolutionary optimization with approximate fitness functions. IEEE Trans Evol Comput 6(5):481–494
28. Branke J, Schmidt C (2005) Fast convergence by means of fitness estimation. Soft Comput 9(1):13–20
29. Sun C, Zeng J, Pan J, Jin Y (2013) Similarity-based evolution control for fitness estimation in particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp 1–8
30. Sun C, Jin Y, Cheng R, Ding J, Zeng J (2017) Surrogate-assisted cooperative Swarm optimization of high-dimensional expensive problems. IEEE Trans Evol Comput 21(4):644–660
31. Powell MJD (1992) The theory of radial basis function approximation in 1990. Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions. Oxford University Press, Oxford, pp 105–210
32. Wild SM, Shoemaker C (2013) Global convergence of radial basis function trust region derivative-free algorithms. SIAM J Optim 21(3):761–781
33. Wang H, Jin Y, Doherty J (2017) Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. IEEE Trans Cybern 47(9):2664–2677
34. Zhang J, Sanderson AC (2009) JADE: Adaptive Differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
35. García-Martínez C, Gutiérrez PD, Molina D, Lozano M, Herrera F (2017) Since CEC 2005 competition on real-parameter optimisation: A decade of research, progress and comparative analysis’s weakness. Soft Comput 21:5573–5583
36. Yang M, Omidvar MN, Li C, Li X, Cai Z, Kazimipour B (2017) Efficient resource allocation in cooperative co-evolution for large-scale global optimization. IEEE Trans Evol Comput 21(4):493–505
37. Mahdavi S, Rahnamayan S, Shiri ME (2017) Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization. Appl Intell 47:888–913
38. Tang K, Li X, Suganthan PN, Yang Z, Weise T (2010) Benchmark functions for the CEC’2010 special session and competition on large scale global optimization. Technical report Nature Inspired Computation and Applications Laboratory (NICAL), USTC, China
39. Cohen J (1988) Statistical power analysis for the behavioral sciences, 2nd edn. Lawrence Earlbaum Associates, Hillsdale