Swarm and Evolutionary Computation 47 (2019) 21–32

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Evolutionary optimization with hierarchical surrogates

Xiaofen Lu [a], Tao Sun [b,*], Ke Tang [a]

[a] Shenzhen Key Laboratory of Computational Intelligence, University Key Laboratory of Evolving Intelligent Systems of Guangdong Province, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[b] Huawei Technologies, Shenzhen, Guangdong 518129, China

ARTICLE INFO

Keywords:
Evolutionary algorithms
Computationally expensive problems
Surrogate model
Multiple modeling techniques

ABSTRACT

The use of surrogate models provides an effective means for evolutionary algorithms (EAs) to reduce the number of fitness evaluations when handling computationally expensive problems. To build surrogate models, a modeling technique (e.g. ANN, SVM, RBF, etc.) needs to be decided first. Previous studies have shown that the choice of modeling technique can highly affect the performance of the surrogate model-assisted evolutionary search. However, one modeling technique might perform differently on different problem landscapes. Without any prior knowledge about the optimization problem to solve, it is very hard to decide which modeling technique to use. To address this issue, in this paper, we propose a novel modeling technique selection strategy in the framework of memetic algorithm (MA). The proposed strategy employs a hierarchical structure of surrogate models and can automatically choose a modeling technique from a pre-specified set of modeling techniques during the optimization process. A mathematic analysis is given to show the effectiveness of the proposed method. Moreover, experimental studies are conducted to compare the proposed method with two other modeling technique selection methods as well as three state-of-the-art optimization algorithms. Experimental results on the used benchmark test functions demonstrate the superiority of the proposed method.

1. Introduction

Computationally expensive problems (CEPs) widely exist in the engineering filed due to the use of increasingly high-fidelity analysis codes such as computational structural mechanics, computational electro-magnetics and computational fluid dynamics. For a CEP, evaluating the quality of a candidate solution may take from minutes to hours of supercomputer time. For example, in aerodynamic wing design, one function evaluation involving the solution of the Navier-Stokes equations can take many hours of computer time [1]. As evolutionary algorithms (EAs) usually need a lot of fitness evaluations to achieve a satisfying solution, to reduce the number of fitness evaluations, the most popular method is to use surrogate models in lieu of the real fitness function in the evolutionary process [2–4].

Surrogate models are computationally efficient models, which can be interpolation or regression models that are built to approximate the real fitness function based on some input-output pairs that are evaluated with the real fitness function. In the literature, various modeling techniques exist and have been applied in surrogate-assisted evolutionary optimization, including polynomial regression method (PR), radial basis function (RBF) network, Kriging/Gaussian process (GP) and so on [2]. It has been shown that the performance of surrogate-assisted evolution highly depends on the used modeling technique [5]. However, one modeling technique usually models differently on different problem landscapes [6]. As a result, it is almost impossible to know which modeling technique is best beforehand without any knowledge about the fitness landscape of the underlying problem.

To address this issue, some researchers proposed building multiple surrogate models by using different modeling techniques and then selecting the one with the minimum training error or building a more accurate ensemble model based on these models [7–10]. However, the most accurate model might not lead to the best performance [6]. The authors in Refs. [11,12] have shown that smoothing models can help find a better solution on multimodal problems than an exact evaluation model. Thus, the performance metric like training error might not be a good metric for modeling technique selection in surrogate-assisted evolutionary optimization.

Considering this, researchers in Ref. [6] proposed GSM method in which multiple surrogate models are used separately for local optimization in the framework of memetic algorithm (MA) and only the model

* Corresponding author.
E-mail addresses: luxf@sustech.edu.cn (X. Lu), suntao25@huawei.com (T. Sun), tangk3@sustech.edu.cn (K. Tang).

https://doi.org/10.1016/j.swevo.2019.03.005
Received 1 March 2018; Received in revised form 7 March 2019; Accepted 9 March 2019
Available online 16 March 2019

that provides the best solution is selected. Specifically, in GSM, two surrogate models are built using different modeling techniques. Each of the two models undergoes a local search on it to generate an improved solution. The two generated solutions are then evaluated with the real fitness function, and only the one with the better fitness is kept and compared to the individuals in the population. As this method selects a modeling technique according to the true quality of solutions generated on the built models, it can select the best modeling technique each time. However, it costs more than one fitness evaluations in the selection, which is cost ineffective especially when many modeling techniques are considered.

In another work [5], the authors proposed the evolvability learning of surrogates (EvoLS) method in which a novel performance metric was introduced for selecting modeling techniques in surrogate-assisted MA. The performance metric is called the evolvability of modeling technique which indicates the expected fitness improvement that a modeling technique can bring in the local search. The modeling technique that has the best evolvability is selected to build a model for local search each time. To calculate the evolvability of a modeling technique, the quality of the solutions provided by the modeling technique in previous generations are used. Experimental studies in Ref. [5] have shown the superiority of this method over only using one modeling technique. Compared to the GSM method, the advantage of EvoLS is that it only needs one fitness evaluation in the selection of modeling technique. However, the evolvability calculation process in EvoLS has limitations to the type of used evolutionary operators, and thus it can not be applied to any EA. Thus, good modeling techniques are still in demand in the field of surrogate-assisted evolutionary optimization.

Motivated by this, in this paper, we propose a novel modeling technique selection strategy in the framework of MA, which has a two-level hierarchical structure. In this structure, multiple modeling techniques are used to build multiple low-level local models. Then, local search is conducted on each low-level local model to obtain an improved individual. After this, a high-level model is built to select the best one from these individuals. Only the selected individual will be evaluated with the real fitness function and allowed to compete in the evolutionary optimization. The corresponding optimization method is called EHS (evolution with hierarchical surrogates) in this paper. Compared to the selection method employed in GSM, the proposed hierarchical method only needs one fitness evaluation in the modeling technique selection, and thus is more cost effective. Compared to EvoLS, EHS does not have any assumption about the type of evolutionary operators, and thus can be applied to any EA.

The remaining parts of this paper are organized as follows. Section 2 will give a brief introduction to GSM and EvoLS. In Section 3, the proposed hierarchical selection method and EHS along with a mathematic analysis will be detailed. In Section 4, experimental studies will be presented to show the efficacy of EHS. Finally, Section 5 will conclude this paper.

2. Related work

2.1. Memetic algorithm

Memetic algorithm (MA) was proposed by Moscato et al., in 1989 as an optimization framework which aims to combine population-based global search and individual-based local search [13,14]. Algorithm 1 gives the pseudo-code for MA. MA firstly generates an initial population $\boldsymbol{P}_0$ randomly from the search space. At each generation, MA selects a parent population $\boldsymbol{P}_{par}$ from the current population $\boldsymbol{P}_G$, and then generates an offspring population $\boldsymbol{P}_{G+1}$. After evaluating each offspring individual $\boldsymbol{x}_{i,G+1}$ in $\boldsymbol{P}_{G+1}$, MA conducts local search operation on each $\boldsymbol{x}_{i,G+1}$ to get a new individual $\boldsymbol{x}^{opt}_{i,G+1}$. If $\boldsymbol{x}^{opt}_{i,G+1}$ is better than $\boldsymbol{x}_{i,G+1}$, it will enter next generation in replace of $\boldsymbol{x}_{i,G+1}$. Otherwise, $\boldsymbol{x}_{i,G+1}$ will enter next generation. MA iterates the above process until the stopping criteria is satisfied. The best individual that MA meets during the optimization will be output at the end.

2.2. The GSM framework

Within the framework of MA, Lim et al. combined different surrogate modeling techniques and proposed GSM in Ref. [6]. The modeling techniques used by GSM are ensemble technique and second-order PR. Through the use of ensemble technique, GSM aims to reduce the harm caused by the uncertainty of models ("Curse of Uncertainty"). Through the use of second-order PR that can smooth the fitness landscape, GSM aims to get the advantage brought by the uncertainty of models ("Bless of Uncertainty"). GSM conducts local search on both the built ensemble model and smoothing model. The better one between the solutions obtained by the two local searches will be provided to the underlying EA for optimization.

The procedure of GSM is shown in Algorithm 2. The whole process of GSM includes two stages. The first stage (steps 4–8 in Algorithm 2) is called dataset building stage in this paper. This stage is used to accumulate training samples that are needed to build models in the second stage. In this stage, GSM evolves the population as the underlying EA does. At each generation, GSM uses the selection, mutation and crossover operators of the underlying EA to evolve the current population $\boldsymbol{P}_G = \{\boldsymbol{x}_{i,G} \mid i = 1, 2, \ldots, popsize\}$. This stage lasts for *MaxGb* generations. All individuals in this stage are evaluated with the real fitness function, and all real evaluations are saved into a data set *DB*.

The second stage (steps 9–29 in Algorithm 2) is called surrogate model involving stage in this paper. This stage builds surrogate models and uses them for local search. At every generation, GSM conducts selection, mutation and crossover on the current population $\boldsymbol{P}_G$ to generate an offspring population $\boldsymbol{P}_{G+1}$. Then, for each individual $\boldsymbol{x}_{i,G+1}$ in $\boldsymbol{P}_{G+1}$, GSM selects m nearest points from DB and uses them to build an ensemble model M_1 and a smoothing model M_2.

To build M_1, GSM first builds three models on m nearest points by using the interpolating Kriging/GP, interpolating linear RBF and second-order PR. Then, GSM gives each model a weight based on its training error. For each test point, the weighted sum of output of each model is used as the output of the ensemble model. After this, GSM does local search on M_1 and M_2 to obtain new individuals $\boldsymbol{x}^1_{opt}$ and $\boldsymbol{x}^2_{opt}$, respectively. Both of $\boldsymbol{x}^1_{opt}$ and $\boldsymbol{x}^2_{opt}$ will be evaluated with the real fitness function. The one with the better fitness will be compared to $\boldsymbol{x}_{i,G+1}$, and replace $\boldsymbol{x}_{i,G+1}$ to enter next generation if it wins. The second stage will last until all evaluations are used up. During the second stage, all newly evaluated individuals will be added to DB once evaluated with the real fitness function.

When conducting local search on the built surrogate model for the individual $\boldsymbol{x}_c$, GSM uses trust-region method by iteratively using Sequential Quadratic Programming (SQP) [15] to solve the following sub-problem:

$$\begin{aligned} &\min: \widehat{f}_M(\boldsymbol{x}_c^k + \boldsymbol{s}) \\ &\text{subject to: } \|\boldsymbol{s}\| \le \Omega^k \end{aligned} \tag{1}$$

where $\widehat{f}_M$ denotes the built surrogate model, $k = 0, 1, 2, \ldots k_{\max}$ denotes the iteration number, $\boldsymbol{x}_c^k$ denotes the beginning point at the k-th iteration ($\boldsymbol{x}_c^0 = \boldsymbol{x}_c$), Ω^k denotes the radius of the trust region at the k-th iteration, which is adapted during the iteration process.

It can be seen that GSM selects one modeling technique between ensemble technique and second-order PR according to the true quality of solutions obtained by ensemble model and second-order PR model. Therefore, the advantage of GSM is that it always selects the truly best modeling technique. However, to make a perfect selection, GSM needs to truly evaluate each solution provided by the built models. This is cost ineffective especially for CEPs. When many modeling techniques are considered, the selection cost will be prohibitive.

Algorithm 1 Pseudo-code for MA.

Input : Fitness function f, Population size, Maximum generation number

Output: The best individual

```
Initialize a population P_0;
Evaluate P_0 with f;
Set G = 0;
while G < Maximum generation number do
    Select parent population P_par from P_G;
    Generate offspring population P_{G+1} based on P_par;
    Evaluate P_{G+1} with f;
    foreach x_{i,G+1} in P_{G+1} do
        Apply local search to obtain x^opt_{i,G+1};
        Evaluate x^opt_{i,G+1} with f;
        if f(x^opt_{i,G+1}) > f(x_{i,G+1}) then
            x_{i,G+1} = x^opt_{i,G+1};
        end
    end
    Set G = G + 1;
end
Output the best individual met in the whole optimization;
```

2.3. The EvoLS algorithm

Like the GSM framework, EvoLS [5] considers three modeling techniques, the interpolating Kriging/GP, interpolating linear RBF and second-order PR, in building local models for local search within the framework of MA. Differently, for each parent individual $\boldsymbol{x}$ in the population $\boldsymbol{P}_t$ at the t-th generation, EvoLS first selects a modeling technique M from the three used modeling techniques based on their *evolvability*. Then, an offspring individual $\boldsymbol{y}$ is generated for $\boldsymbol{x}$. After this, EvoLS builds one local model with M for $\boldsymbol{y}$ and conducts local search on this model using $\boldsymbol{y}$ as the starting point. For each parent individual $\boldsymbol{x}$ in $\boldsymbol{P}_t$ at generation t, the *evolvability* of a modeling technique M is defined in Ref. [5] as the expectation of fitness improvement that can be achieved over $\boldsymbol{x}$ by doing local search on each possible offspring individual $\boldsymbol{y}$ with the corresponding local model built for $\boldsymbol{y}$ using M, which has the following formulation:

$$\begin{aligned} EV_M(\boldsymbol{x}) &= Exp[f(\boldsymbol{x}) - f(\boldsymbol{y}^{opt}) \mid \boldsymbol{P}_t, \boldsymbol{x}] \\ &= f(\boldsymbol{x}) - \int_{\boldsymbol{y}} f(\varphi_M(\boldsymbol{y})) \times P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x}) d\boldsymbol{y} \end{aligned} \tag{2}$$

where $\boldsymbol{y}^{opt}$ denotes the individual obtained by doing local search on $\boldsymbol{y}$ with the local model built using M. $f(\boldsymbol{x})$ and $f(\boldsymbol{y}^{opt})$ are the function values for $\boldsymbol{x}$ and $\boldsymbol{y}^{opt}$, respectively. $f(\boldsymbol{x}) - f(\boldsymbol{y}^{opt})$ denotes the fitness improvement of $\boldsymbol{y}^{opt}$ over $\boldsymbol{x}$. $Exp[f(\boldsymbol{x}) - f(\boldsymbol{y}^{opt}) \mid \boldsymbol{P}_t, \boldsymbol{x}]$ means the expectation of the fitness improvement obtained on all possible $\boldsymbol{y}$. $\varphi_M(\boldsymbol{y})$ is equal to $\boldsymbol{y}^{opt}$. $P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x})$ denotes the probability density function of generating an offspring individual $\boldsymbol{y}$ based on the parent individual $\boldsymbol{x}$ and the population $\boldsymbol{P}_t$ at generation t. $\int_{\boldsymbol{y}} f(\varphi_M(\boldsymbol{y})) \times P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x}) d\boldsymbol{y}$ is to calculate the expected function value of $\boldsymbol{y}^{opt}$ for all possible $\boldsymbol{y}$.

To calculate $EV_M(\boldsymbol{x})$ in Eq. (2), the authors [5] make use of the historical performance information $((\boldsymbol{y}_j, \varphi_M(\boldsymbol{y}_j)))$ of M to approximate it. Algorithm 3 shows the evolvability calculation process and modeling technique selection process. For each modeling technique M_k, its evolvability is calculated based on the performance information $((\boldsymbol{y}_j, \varphi_{M_k}(\boldsymbol{y}_j)))$ of M_k in previous generations and the probability density $P(\boldsymbol{y}_j \mid \boldsymbol{P}_t, \boldsymbol{x}_{i,G})$ (see Steps 2–10). The $\boldsymbol{y}_j$ denotes individuals for which the modeling technique M_k is used to build local models for local search in previous generations. The $\varphi_{M_k}(\boldsymbol{y}_j)$ denotes the improved individual found by local search on $\boldsymbol{y}_j$ with the model built using M_k. Step 9 in Algorithm 3 tries to approximate the evolvability defined in Eq. (2) for M_k based on the sample points $((\boldsymbol{y}_j, \varphi_{M_k}(\boldsymbol{y}_j)))$ obtained in previous generations.

To obtain the probability density function $P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x})$, the authors in Ref. [5] used uniform crossover and mutation operators in the underlying EA. This can ensure that any generated offspring individual exist in the hyper-rectangle formed by the parent individuals. Furthermore, $P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x})$ satisfies a uniform distribution in the hyper-rectangle, that is:

$$P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x}) = \begin{cases} \dfrac{1}{Vol(\mathbf{R})} & \text{if } \boldsymbol{y} \in \mathbf{R} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{R}$ denotes the hyper-rectangle enclosed by individuals in $\boldsymbol{P}_t$, and $Vol(\mathbf{R})$ denotes the hyper-volume of $\mathbf{R}$. However, it would be not easy to calculate $P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x})$ if more complex crossover and mutation operators (e.g. the hybrid operator) are used in the underlying EA.

EvoLS selects an appropriate modeling technique based on their evolvability before building local model and conducting local search. Compared to GSM, the advantage of EvoLS is that it only needs one fitness evaluation in the selection of modeling technique no matter how many modeling techniques are considered. However, EvoLS needs to calculate the evolvability of each modeling technique, which can be approximated only when $P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x})$ can be calculated. As $P(\boldsymbol{y} \mid \boldsymbol{P}_t, \boldsymbol{x})$ is hard to calculate for EAs with complex crossover and mutation operators, the application range of EvoLS is limited in the field of EAs.

Algorithm 2 The procedure of GSM.

Input : Fitness function f, Population size $popsize$, Maximum evaluation number $MaxEval$, Dataset building generations $MaxGb$

Output: The best individual

Set $G = 0$, $eval = 0$, $DB = \emptyset$;
Initialize and evaluate $\boldsymbol{P}_0 = \{\boldsymbol{x}_{i,0} | i = 1, 2, ..., popsize\}$;
Archive all $(\boldsymbol{x}_{i,0}, f(\boldsymbol{x}_{i,0}))$ into DB;
while $G < MaxGb$ && $eval < MaxEval$ **do**
    $\boldsymbol{P}_{G+1}$ = EA-Select-Mutate-Crossover($\boldsymbol{P}_G$);
    Evaluate $\boldsymbol{P}_{G+1}$,and archive all $(\boldsymbol{x}_{i,G+1}, f(\boldsymbol{x}_{i,G+1}))$ into DB;
    Set $G = G + 1$, $eval = eval + popsize$;
end
while $eval < MaxEval$ **do**
    $\boldsymbol{P}_{G+1}$ = EA-Select-Mutate-Crossover($\boldsymbol{P}_G$);
    Evaluate $\boldsymbol{P}_{G+1}$,and archive all $(\boldsymbol{x}_{i,G+1}, f(\boldsymbol{x}_{i,G+1}))$ into DB;
    foreach $\mathbf{x}_{i,G+1}$ *in* $\mathbf{P}_{G+1}$ **do**
        Select m nearest points as training samples from DB;
        Build ensemble model M_1 and second-order PR model M_2 ;
        Do local search on M_1 and M_2 separately to get $\boldsymbol{x}^1_{opt}$ and $\boldsymbol{x}^2_{opt}$;
        Evaluate $\boldsymbol{x}^1_{opt}$ and $\boldsymbol{x}^2_{opt}$, archive $(\boldsymbol{x}^1_{opt}, f(\boldsymbol{x}^1_{opt}))$, $(\boldsymbol{x}^2_{opt}, f(\boldsymbol{x}^2_{opt}))$ into DB;
        Set $\boldsymbol{x}^{'}_{opt}$ to the one of $\boldsymbol{x}^1_{opt}$ and $\boldsymbol{x}^2_{opt}$ with the better f value ;
        if $f(\mathbf{x}^{'}_{opt}) > f(\mathbf{x}_{i,G+1})$ **then**
            $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}^{'}_{opt}$;
        end
    end
    Set $G = G + 1$;
end

3. The proposed EHS algorithm

The proposed EHS uses a novel modeling technique selection method in the framework of MA. Inspired by hierarchical mixture of expert [16], a two-level hierarchical surrogate structure is introduced in EHS to do modeling technique selection. In the two-level hierarchical structure, the lower level is composed of multiple local models, each built using a different modeling technique. EHS does local search on each low-level model to obtain a new individual, and then builds a high-level model to select the best one from the multiple obtained individuals. Through the selection among the individuals provided by low-level models, EHS also completes the selection among the multiple modeling techniques used to build low-level models.

3.1. The details of EHS

Algorithm 4 gives the framework of EHS. The procedure of EHS also has two stages, the dataset building stage and the surrogate involving stage. Different from GSM and EvoLS, EHS uses the two-level hierarchical surrogate structure to decide which modeling technique to use in the surrogate involving stage. Concretely, at each generation, EHS first builds n low-level local models $M_1, M_2, \ldots, M_n$ for each offspring individual $\boldsymbol{x}_{i,G+1}$ using n modeling techniques. Then, EHS does local search on each $M_j (j = 1, 2, \ldots, n)$ to obtain a new individual $\boldsymbol{x}^j_{opt} (j = 1, 2, \ldots, n)$. After this, EHS builds a high-level ranking model S to select the individual $\boldsymbol{x}'_{opt}$ that has the best rank from $\boldsymbol{x}^j_{opt} (j = 1, 2, \ldots, n)$, and evaluates it with the real fitness function. At the end, $\boldsymbol{x}'_{opt}$ will be compared with $\boldsymbol{x}_{i,G+1}$. The individual $\boldsymbol{x}'_{opt}$ will replace $\boldsymbol{x}_{i,G+1}$ to enter next generation if it has a better f value.

To build low-level models $M_1, M_2, \ldots, M_n$, EHS first selects m points from DB that are nearest to $\boldsymbol{x}_{i,G+1}$ by Euclidean distance, and then builds each low-level model using a different modeling technique with the m points as the training samples. It should be noted that modeling techniques such as PR, RBF, Kriging/GP can be used as the modeling techniques to build low-level models. Besides, the same modeling technique with different parameter values can also be used to build low-level models.

The local search in EHS aims to find a local optimum near $\boldsymbol{x}_{i,G+1}$ on the approximate fitness function $\hat{f}_M(\mathrm{x})$ given by the low-level model M. The sub-problem that EHS needs to solve in the local search can be described as:

$$\begin{aligned} &\min: \hat{f}_M(\mathrm{x}_{i,G+1} + \mathrm{s}) \\ &\text{subject to: } \| \mathrm{x}_{i,G+1} + \mathrm{s} \| \leq \Delta \end{aligned} \tag{4}$$

Algorithm 3 The modeling technique selection process of EvoLS.

Input : Probability density function $P(\boldsymbol{y}|\boldsymbol{P}_t, \boldsymbol{x}_{i,G})$, Parent individual $\boldsymbol{x}_{i,G}$

Output: The selected modeling technique

产生子代个体y的概率密度函数

foreach *modeling technique* M_k **do**
  Obtain previous performance information $\{(\boldsymbol{y}_j, \varphi_{M_k}(\boldsymbol{y}_j))\}$; 局部模型对y进行局部搜索得到的个体
  Calculate the weight $w_j(\boldsymbol{x}_{i,G}) = P(\boldsymbol{y}_j|\boldsymbol{P}_t, \boldsymbol{x}_{i,G})$ for each $\boldsymbol{y}_j$;
  if $\sum w_j(\mathbf{x}_{i,G}) < \epsilon$ **then**
    $w_j(\boldsymbol{x}_{i,G}) = 0$;
    $EV_{M_k}(\boldsymbol{x}_{i,G}) = -\infty$;
  else
    Set $w_j = w_j(\boldsymbol{x}_{i,G}) / \sum w_j(\boldsymbol{x}_{i,G})$;
    $EV_{M_k}(\boldsymbol{x}_{i,G}) = f(\boldsymbol{x}_{i,G}) - \sum f(\varphi_{M_k}(\boldsymbol{y}_j)) \times w_j$;
  end
end
if $EV_{M_k}(\mathbf{x}_{i,G}) < 0$ *is satisfied for each* M_k **then**
  Randomly select M_k;
else
  Select M_k with the maximum $EV_{M_k}(\boldsymbol{x}_{i,G})$;
end

where $\Delta = [\min_{j=1,2,\ldots,n}\{y_k^j\}, \max_{j=1,2,\ldots,n}\{y_k^j\}]_{k=1,2,\ldots,m}$ $(\forall j = 1, 2, \ldots, n)$, and y_k^j denotes the value of j-th decision variable in the k-th sample (n equals to the problem dimension). In this paper, EHS applies SQP that was also used in GSM [6] to solve this sub-problem.

The high-level model in EHS is used to select the best one from individuals provided by low-level models. Previous studies have shown that ranking models are more appropriate for selecting best solutions from multiple candidate solutions [17,18]. Considering this, we apply an efficient ranking method, RankBoost [19], to construct the high-level model. RankBoost was proposed to solve the ranking problem based on boosting. Suppose we have a set of m training samples $\{\boldsymbol{x}_i\,|, i = 1, 2, \ldots, m\}$, each $\boldsymbol{x}_i$ has n ranking features $(rf_1(\boldsymbol{x}_i), rf_2(\boldsymbol{x}_i), \ldots, rf_n(\boldsymbol{x}_i))$, and the ordering information of these samples is given. RankBoost aims to produce a function $H(\boldsymbol{x})$ to score each $\boldsymbol{x}_i$ and enable the ordering of the scores to be consistent with the ordering of $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$.

The key idea of RankBoost is to produce highly correct prediction by combining many weak learners that might have only moderate accuracy. It operates in rounds and produces a weak learner $h_t(\boldsymbol{x})$ on each round (t denotes the round number). Concretely, it maintains a distribution $D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to emphasize different parts of different samples and updates it in every round. $D_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is set to $1/c$ if $\boldsymbol{x}_j$ is ranked above $\boldsymbol{x}_i$, and 0 otherwise. The parameter c is a normalized factor so that $\sum_{\mathrm{x}_i, \mathrm{x}_j} D_1(\mathrm{x}_i, \mathrm{x}_j) = 1$. In each round, a weaker learner $h_t(\boldsymbol{x})$ is constructed to maximize:

$$r = \sum_{\mathrm{x}_i, \mathrm{x}_j} D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(h_t(\mathrm{x}_j) - h_t(\mathrm{x}_i)) \tag{5}$$

In RankBoost, the weak learner $h_t(\boldsymbol{x})$ has the form as follows:

$$h_t(\mathrm{x}) = \begin{cases} 1 & rf_i(\boldsymbol{x}) > \theta \\ 0 & rf_i(\boldsymbol{x}) < \theta \end{cases} \tag{6}$$

The construction process of $h_t(\boldsymbol{x})$ is to find i and θ to maximize r in Eq. (5). Specifically, for each ranking feature i, a θ_i is obtained from the set $(-\infty, rf_i(\boldsymbol{x}_1), rf_i(\boldsymbol{x}_2), \ldots, rf_i(\boldsymbol{x}_m))$ that maximizes r. Then, i is then set to the index of ranking feature that makes the biggest r and θ is set to θ_i.

After the construction of $h_t(\boldsymbol{x})$, $D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is updated according to the following formula:

$$D_{t+1}(x_i, x_j) = \frac{D_t(x_i, x_j) exp(\alpha_t(h_t(x_i) - h_t(x_j)))}{Z_t} \tag{7}$$

where Z_t is a normalization factor equal to:

$$\sum_{x_i, x_j} D_t(x_i, x_j) exp(\alpha_t(h_t(x_i) - h_t(x_j))) \tag{8}$$

and α_t is set to:

$$\alpha = \frac{1}{2} ln(\frac{1+r}{1-r}) \tag{9}$$

Finally, the output ranking function $H(\boldsymbol{x})$ is set to:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{10}$$

where T is the number of rounds.

In this work, when building the ranking model, for each individual $\boldsymbol{x}_{opt}^j (j = 1, 2, \ldots, n)$ provided by the low-level model, EHS first selects m' points that are nearest to $\boldsymbol{x}_{opt}^j$. Then, EHS randomly chooses 80% points among these $n^* m'$ sample points as training samples to build the ranking model. The remaining 20% points are used as test samples to evaluate the accuracy of the ranking model. For the 80% training samples, they are first ranked according to their f values. The one with the highest f value has the rank 1, and the second highest has the rank 2, and so on. The decision variables are treated as ranking features for each training sample after pre-processing. In the pre-processing, for each decision variable, all the training samples are ordered according to the decision variable from smallest to the largest. If the consistency between the ordering for one decision variable and the ranks of f is greater than 0.5, the corresponding decision variable is directly treated

Algorithm 4 The EHS framework.

Input : Fitness function f, Population size $popsize$, Maximum evaluation number $MaxEval$, Dataset building generations $MaxGb$

Output: The best individual

Set $G = 0$, $eval = 0$, $DB = \emptyset$;
Initialize and evaluate $\boldsymbol{P}_0 = \{\boldsymbol{x}_{i,0} | i = 1, 2, ..., popsize\}$;
Archive all $(\boldsymbol{x}_{i,0}, f(\boldsymbol{x}_{i,0}))$ into DB;
while $G < MaxGb$ && $eval < MaxEval$ **do**
    $\boldsymbol{P}_{G+1}$ = EA-Select-Mutate-Crossover($\boldsymbol{P}_G$);
    Evaluate $\boldsymbol{P}_{G+1}$,and archive all $(\boldsymbol{x}_{i,G+1}, f(\boldsymbol{x}_{i,G+1}))$ into DB;
    Set $G = G + 1$, $eval = eval + popsize$;
end
while $eval < MaxEval$ **do**
    $\boldsymbol{P}_{G+1}$ = EA-Select-Crossover-Mutate($\boldsymbol{P}_G$);
    Evaluate $\boldsymbol{P}_{G+1}$,and archive all $(\boldsymbol{x}_{i,G+1}, f(\boldsymbol{x}_{i,G+1}))$ into DB;
    foreach $\mathbf{x}_{i,G+1}$ *in* $\mathbf{P}_{G+1}$ **do**
        Select m nearest points as training samples from DB;
        Build n local models: $M_1, M_2, ..., M_n$ using n modeling techniques;
        foreach M_j **do**
            Do local search on M_j to obtain $\boldsymbol{x}_{opt}^{j}$;
        end
        Select training points from DB and build a ranking model S;
        Use S to rank $\boldsymbol{x}_{opt}^{j} (j = 1, 2, ..., n)$ and select $\boldsymbol{x}_{opt}^{'}$ that has the best rank;
        Evaluate $\boldsymbol{x}_{opt}^{'}$ with f and archive $(\boldsymbol{x}_{opt}^{'}, f(\boldsymbol{x}_{opt}^{'}))$ into DB;
        if $f(\mathbf{x}_{opt}^{'}) > f(\mathbf{x}_{i,G+1})$ **then**
            $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{opt}^{'}$;
        end
    end
    Set $G = G + 1$;
end

as a ranking feature. Otherwise, it is multiplied by −1 before treated as a ranking feature.

After this, RankBoost is carried out on the training samples to produce a ranking model $H(\boldsymbol{x})$. Then, $H(\boldsymbol{x})$ is used to score each test sample. The accuracy of this ranking model is calculated as the consistency between the ordering of the scores and the ranks of f values on the test samples. If the accuracy of the ranking model is larger than 0.5, the one in $\boldsymbol{x}_{opt}^{j} (j = 1, 2, \ldots, n)$ that has the highest $H(\boldsymbol{x}_{opt}^{j})$ score is chosen as $\boldsymbol{x}'_{opt}$. Otherwise, a randomly selected point from $\boldsymbol{x}_{opt}^{j} (j = 1, 2, \ldots, n)$ is used as $\boldsymbol{x}'_{opt}$.

Compared to the selection method employed in GSM, the proposed hierarchical method only needs one fitness evaluation in the modeling technique selection, and thus is more cost effective. Compared to EvoLS, EHS does not have any assumption about the type of evolutionary operators, and thus can be applied to any EA.

3.2. Mathematic analysis

In this part, we will use a simple mathematic analysis to demonstrate the effectiveness of the proposed hierarchical surrogate structure. We would like to prove that the use of hierarchical surrogates has a higher probability to find the optimum of the optimization problem in Eq. (4) than only using one single surrogate model.

Suppose the hierarchical structure consists of m low-level models and a high-level model. If only one low-level model is used for local search, the probability to find the optimum of the optimization problem in Eq. (4) is denoted as pb. We assume all low-level models are mutually independent and has the same value of pb. Note that mutually independent low-level models can be built by randomly choosing training samples from a big set of training samples for each of them. The probability that the high-level model ranks two solutions correctly (i.e. consistent with their true fitness) is denoted as p.

Suppose the high-level model chooses the best ranking solution from the solutions provided by the low-level models through sequentially comparing them, that is, the better one in previous comparison will be compared to the next one. Through the use of hierarchical surrogates, the probability to find the optimum of the optimization problem in Eq. (4) is represented as $pbh(m)$, which is actually a function of pb and p when m is fixed. Then, we have the following theorem:

Theorem 1. If p satisfies $0.5 < p \leq 1.0$ and pb satisfies $0 < pb < 1.0$, then $pbh(m) > pb$ is true for any $m(m > 2)$.

Proof. To prove that the Theorem 1 is true for any $m(m > 2)$, the mathematical induction method is used as follows.

- Suppose there are two low-level models (i.e. $m = 2$). If both low-level models provide the optimum solution of the problem (the probability is $pb*pb$), then the solution selected by the high-level model is surely the optimum solution. If only one low-level model provides the optimum solution of the problem (the probability is $2*pb*(1 - pb)$), then the high-level model has the probability of p to select the optimum solution. If neither of the low-level models gives the optimum solution (the probability is $(1 - pb)*(1 - pb)$), then the high-level model is impossible to select the optimum solution. Thus, the hierarchical surrogates with 2 low-level models can find the optimum solution with the probability of:

$$\begin{aligned} pbh(2) &= pb * pb + 2 * pb * (1 - pb) * p + (1 - pb) * (1 - pb) * 0 \\ &= pb^2 + 2 * pb * (1 - pb) * p \end{aligned} \tag{11}$$

  If pb satisfies $0 < pb < 1.0$, $pbh(2)$ in Eq. (11) is a monotonically increasing function of p. When p is equal to 0.5, we can have $pbh(2) = pb$ in Eq. (11). When $p > 0.5$, we can have $pbh(2) > pb$ in Eq. (11). That is, if both $0 < pb < 1.0$ and $0.5 < p \leq 1.0$ are true, $pbh(m) > pb$ is true for $m = 2$.
- Assume Theorem 1 is true for $m = k(k \geq 2)$, that is, when $0.5 < p \leq 1.0$ and $0 < pb < 1.0$, $pbh(k) > pb$ is true for $m = k$.
- When $m = k + 1$, the high-level model has the probabilities of $pbh(k)$ to select the optimum solution from the first k low-level models and $(1 - pbh(k))$ to select a non-optimum solution. If the high-level model selects the optimum solution from the first k low-level models (the probability is $pbh(k)$), and the $(k + 1)$-th low-level model provides the optimum solution (the probability is pb), then the solution selected by the high-level model from the $(k + 1)$ low-level models is surely the optimum solution. If the high-level model selects the optimum solution from the first k low-level models but the $(k + 1)$-th low-level model does not provide the optimum solution (the probability is $(1 - pb)$), then the high-level model has the probability of p to select the optimum solution from the $(k + 1)$ low-level models. If the high-level model does not select the optimum solution from the first k low-level models (the probability is $(1 - pbh(k))$), but the $(k + 1)$-th low-level model provides the optimum solution (the probability is pb), then the high-level model has the probability of p to select the optimum solution from the $(k + 1)$ low-level models. Otherwise, the high-level model can not select the optimum solution from the $(k + 1)$ low-level models. Thus, the hierarchical surrogates with $(k + 1)$ low-level models can find the optimum solution with the probability of:

$$\begin{aligned} pbh(k+1) &= pbh(k) * pb + pbh(k) * (1 - pb) * p \\ &\quad + (1 - pbh(k)) * pb * p \\ &= p * pb + (p + pb - 2 * p * pb) * pbh(k) \end{aligned} \tag{12}$$

  When pb satisfies $0 < pb < 1$, $pbh(k + 1)$ is a monotonically increasing function of $pbh(k)$. Therefore, when both $0.5 < p \leq 1.0$ and $0 < pb < 1.0$ are true, since $pbh(k) > pb$ is assumed to be true, we can have:

Table 1
Summary of 10 test functions.

Func	Name	Optimum f Value
f_1	Ackley	0.0
f_2	Griewank	0.0
f_3	Rosenbrock	0.0
f_4	Shifted Rotated Rastrigin (F10 in Ref. [21])	−330.0
f_5	Shifted Rotated Weierstrass (F11 in Ref. [21])	90.0
f_6	Shifted Expanded Griewank plus Rosenbrock (F13 in Ref. [21])	−130.0
f_7	Hybrid Composition Function (F15 in Ref. [21])	120.0
f_8	Rotated Hybrid Composition Function (F16 in Ref. [21])	120.0
f_9	Rotated Hybrid Composition Function with Narrow Basin Global Optimum (F19 in Ref. [21])	10.0
f_{10}	Noncontinuous Rotated Hybrid Composition Function (F23 in Ref. [21])	360.0

$$\begin{aligned} pbh(k+1) &= p * pb + (p + pb - 2 * p * pb) * pbh(k) \\ &> p * pb + (p + pb - 2 * p * pb) * pb \\ &= 2 * p * (pb - pb^2) + pb^2 \\ &> 2 * 0.5 * (pb - pb^2) + pb^2 \\ &= pb. \end{aligned} \tag{13}$$

  That is, when $0.5 < p \leq 1.0$ and $0 < pb < 1.0$ are true, if $pbh(k) > pb$ is true for $m = k(k \geq 2)$, then $pbh(k + 1) > pb$ is true for $m = k + 1(k \geq 2)$.
- Since both the base case and the inductive step have been performed in the above statements, according to mathematical induction, we can get that $pbh(m) > pb$ holds for any $m(m > 2)$ when p satisfies $0.5 < p \leq 1.0$ and pb satisfies $0 < pb < 1.0$. The Proof of Theorem 1 is completed. □

According to Theorem 1, when the ranking accuracy of the high-level model is larger than 0.5 (i.e. the accuracy of a random ranking), the probability to find the optimum of the problem in Eq. (4) by using the hierarchical surrogate structure is larger than that of using only one low-level model. This demonstrates that the hierarchical surrogate structure can be beneficial even without a high demand on the high-level model.

4. Experimental studies

To evaluate the efficacy of EHS, experimental studies were conducted to make comparisons between EHS and each of GSM and EvoLS as well as 3 state-of-the-art optimization algorithms. The details of the experimental studies will be presented in this section.

4.1. Benchmark functions

The experiments were conducted on 10 benchmark functions which were also used as test functions in Refs. [5,6]. Table 1 gives the description of these 10 functions. Each of them is a minimization problem. More details about the 10 functions can be found in Refs. [20,21]. In the experiments, the dimension n of each test function was set to 30 and the maximum number of function evaluations was set to 3000.

4.2. Comparison with GSM and EvoLS

In the experiments, three low-level local models were used for EHS. They are Kriging/GP model, interpolating linear RBF model, second-

Table 2
Setup for the set of used surrogates. The DACE [24], RBF [25], SURROGATES [23] were used to run the kriging/GP model, radial basis function, and polynomial regression algorithms, respectively.

Surrogate	Details
Kriging/GP model	Constant trend function and Gaussian correlation
RBF model	Radial basis function: linear spline kernel function
PR model	Second order

Table 3
Parameter settings for EHS.

Population size	100
Selection operator	Elitism and ranking selection
Crossover and mutation	Uniform crossover, uniform mutation
Crossover rate	0.9
Mutation rate	0.1
Local search method	SQP
Maximum number of runs	25

Table 4
Parameter settings for SQP.

MaxFunctionEvaluations	100
HonorBounds	True
SpecifyObjectiveGradient	True
StepTolerance	1e-6
Other parameters	Default

order PR model, which were also used by GSM and EvoLS. Researchers have shown that RBF models are appropriate for high-order nonlinear problems, while Kriging/GP and PR are suitable for low-order nonlinear high-dimensional problems and low-order nonlinear low-dimensional problems [22], respectively. Details about these three models can be found in Ref. [5]. Table 2 gives the setting for the three surrogates used in the experiments. The SURROGATES toolbox [23] was used for easy implementation of the surrogates. To make a fair comparison, GSM also builds three local models (i.e., Kriging/GP model, RBF model, and PR model) and does local search on each of the three local models.

For each of GSM, EvoLS and EHS, the maximum number of function evaluations for the dataset building stage was set to 600. The other parameters of GSM and EvoLS were set the same as in their original papers. Table 3 shows the parameter settings for EHS. Note that EHS used the same underlying EA as well as the same parameter settings as GSM did in Ref. [6]. The running environment is Matlab 2017b. The Matlab optimization toolbox and *fmincon* function was used to implement SQP. Table 4 gives the parameter settings for SQP. More implementation details can be found in Ref. [26]. In the experiments, every algorithm was run 25 times on each test function. In every run, the minimum function values achieved by each algorithm at the cost of 1000, 2000 and 3000 functions evaluations were recorded to evaluate the performance of the algorithm. When making comparisons between different algorithms, the Wilcoxon rank-sum test with a confidence level of 0.5 was applied.

Table 5 summarizes the average of the minimum function values over 25 runs achieved by each of GSM, EvoLS and EHS on each of the 10 test functions at the cost of 1000, 2000 and 3000 function evaluations,

Table 5
The experimental results obtained by GSM, EvoLS and EHS on f_1–f_{10} at the cost of 1000, 2000 and 3000 function evaluations, and the statistical comparison results. +, − and ≈ mean that GSM or EvoLS performed statistically better than, worse than, and similar to EHS, respectively. The best results are marked in **Bold**.

Func	Algorithm	1000 Mean ± Std	2000 Mean ± Std	3000 Mean ± Std
f_1	GSM	1.63e+01 ± 4.75e-01 −	9.03e+00 ± 8.34e-01 −	4.72e+00 ± 5.25e-01 ≈
	EvoLS	1.28e+01 ± 1.13e+00 −	**5.33e+00 ± 4.93e-01** +	**2.72e+00 ± 3.73e-01** +
	EHS	**1.16e+01 ± 7.55e-01**	6.52e+00 ± 4.92e-01	4.45e+00 ± 6.44e-01
f_2	GSM	2.13e+02 ± 2.67e+01 −	1.04e+00 ± 6.35e-01 −	6.93e-01 ± 3.78e-01 −
	EvoLS	9.31e-01 ± 5.88e-01 −	5.90e-01 ± 3.58e-01 −	4.26e-01 ± 2.77e-01 −
	EHS	**1.09e-06 ± 2.60e-06**	**1.09e-06 ± 2.60e-06**	**1.09e-06 ± 2.60e-06**
f_3	GSM	2.02e+03 ± 3.17e+02 −	2.26e+02 ± 9.79e+01 −	1.07e+02 ± 1.65e+01 −
	EvoLS	2.15e+02 ± 4.39e+01 −	7.95e+01 ± 1.32e+01 −	3.39e+01 ± 8.36e+00 −
	EHS	**9.09e+01 ± 1.17e+01**	**2.95e+01 ± 1.45e+00**	**2.81e+01 ± 6.59e-01**
f_4	GSM	1.44e+02 ± 3.29e+01 −	−7.49e+01 ± 1.80e+01 −	−9.95e+01 ± 1.75e+01 −
	EvoLS	−4.95e+01 ± 2.67e+01 −	−1.15e+02 ± 2.68e+01 −	−1.49e+02 ± 3.29e+01 −
	EHS	**−1.73e+02 ± 2.78e+01**	**−2.30e+02 ± 2.64e+01**	**−2.55e+02 ± 2.08e+01**
f_5	GSM	1.36e+02 ± 1.55e+00 −	1.35e+02 ± 1.65e+00 −	1.34e+02 ± 1.52e+00 −
	EvoLS	1.35e+02 ± 1.54e+00 −	1.35e+02 ± 1.58e+00 −	1.34e+02 ± 1.42e+00 −
	EHS	**1.33e+02 ± 2.35e+00**	**1.32e+02 ± 2.31e+00**	**1.31e+02 ± 2.65e+00**
f_6	GSM	1.16e+02 ± 8.16e+01 −	−7.16e+01 ± 9.94e+00 −	−9.05e+01 ± 5.40e+00 −
	EvoLS	−8.31e+01 ± 1.08e+01 −	−9.96e+01 ± 2.23e+00 −	−1.05e+02 ± 2.30e+00 −
	EHS	**−9.91e+01 ± 3.86e+00**	**−1.08e+02 ± 1.98e+00**	**−1.10e+02 ± 1.10e+00**
f_7	GSM	1.10e+03 ± 4.14e+01 −	8.67e+02 ± 3.40e+01 −	7.89e+02 ± 3.50e+01 −
	EvoLS	9.01e+02 ± 7.08e+01 −	7.83e+02 ± 6.96e+01 −	7.25e+02 ± 5.91e+01 −
	EHS	**7.48e+02 ± 5.48e+01**	**6.12e+02 ± 6.59e+01**	**5.56e+02 ± 5.59e+01**
f_8	GSM	7.93e+02 ± 4.84e+01 −	5.00e+02 ± 5.44e+01 −	4.47e+02 ± 5.32e+01 −
	EvoLS	5.35e+02 ± 6.69e+01 −	4.25e+02 ± 4.53e+01 −	3.83e+02 ± 5.75e+01 −
	EHS	**4.43e+02 ± 9.23e+01**	**3.06e+02 ± 9.28e+01**	**2.78e+02 ± 1.01e+02**
f_9	GSM	1.20e+03 ± 2.82e+01 −	1.03e+03 ± 1.31e+01 −	9.90e+02 ± 9.39e+00 −
	EvoLS	1.11e+03 ± 2.66e+01 −	1.01e+03 ± 1.63e+01 −	9.83e+02 ± 1.36e+01 −
	EHS	**1.05e+03 ± 1.79e+01**	**9.96e+02 ± 1.59e+01**	**9.58e+02 ± 4.55e+01**
f_{10}	GSM	1.64e+03 ± 4.32e+01 −	1.33e+03 ± 8.10e+01 −	1.08e+03 ± 7.64e+01 −
	EvoLS	1.50e+03 ± 6.37e+01 −	1.15e+03 ± 1.14e+02 −	9.55e+02 ± 4.44e+01 −
	EHS	**1.19e+03 ± 2.06e+02**	**9.04e+02 ± 2.57e+01**	**8.96e+02 ± 4.81e+00**
In total:	GSM: **0** + /**29** − /**1**≈	EvoLS: **2** + /**28** − /**0** ≈		

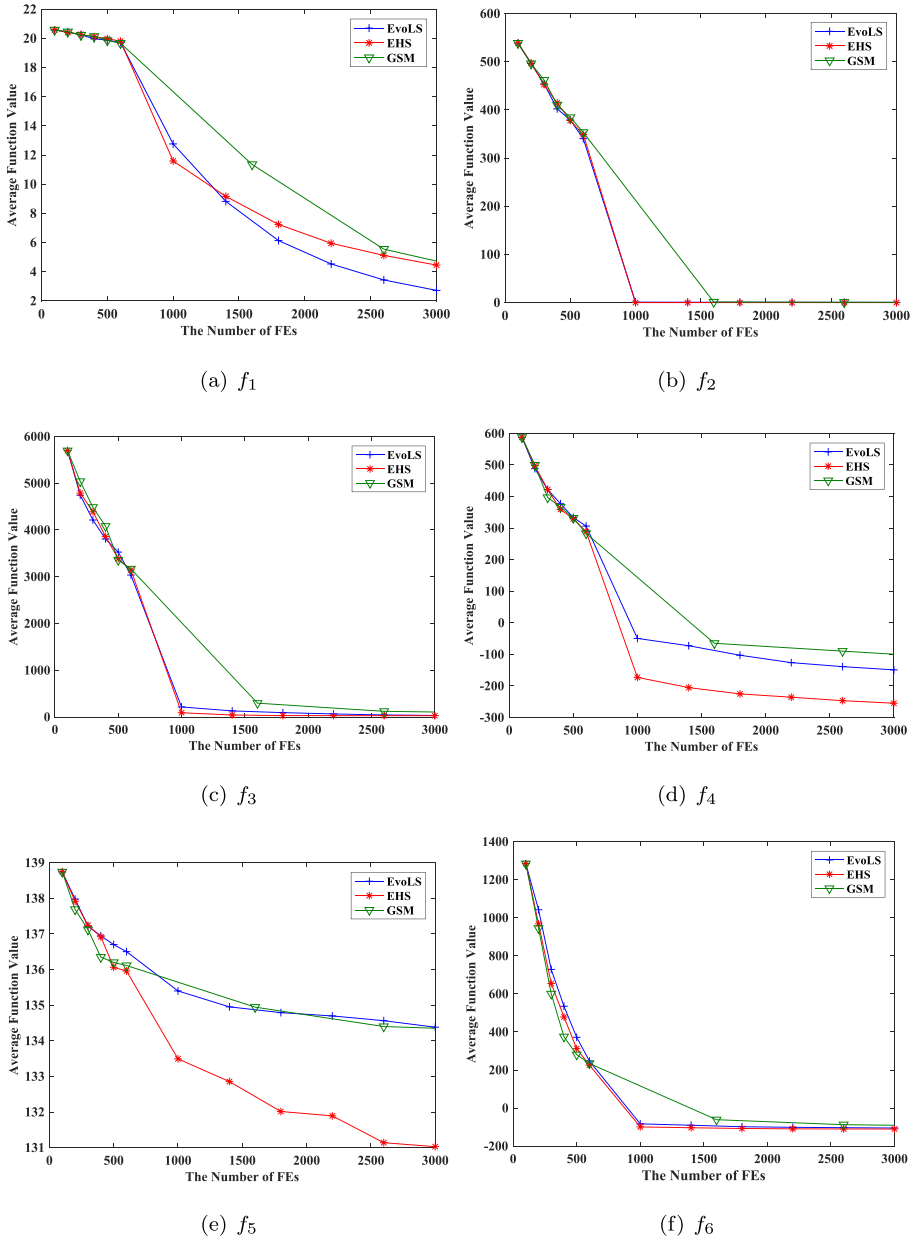

Fig. 1. Evolution curves for GSM, EvoLS and EHS on f_1-f_6.

respectively. Figs. 1 and 2 gives the evolutionary curves of GSM, EvoLS and EHS on each test function. In these two figure, the X-axis denotes the number of function evaluations, and the Y-axis is the average of the minimum function values over 25 runs.

It can be seen from Table 5 that EHS generally outperformed GSM and EvoLS among all 10*3 statistical comparisons. Specifically, EHS achieved better or the same solutions than GSM on every test function in all statistical comparisons at the cost of 1000, 2000 and 3000 function evaluations. When compared to EvoLS, EHS performed better on every test function at the cost of 1000 function evaluations. At the cost of 2000 and 3000 function evaluations, EHS is worse than EvoLS on only 1 test function but better on the other 9 test functions. The reason that EHS did not perform better on f_1 is because that EvoLS employed the trust-region method in the local search as well as a different EA operation to calculate the evolvability.

To investigate why EHS performed better than EvoLS and whether the high-level models built using RankBoost played their roles (i.e. selecting the best solution), we recorded the true rank of the solution selected by the high-level model as well as the true rank of the modeling technique selected by EvoLS each time in the experiments. If the solution selected by the high-level model in EHS is actually the best one among all solutions obtained by doing local search on the low-level models, its true rank is 1. If the modeling technique selected by EvoLS does contribute the best solution among 3 modeling techniques, its true rank is also 1. Table 6 gives the average true rank of the solution selected by the high-level model in EHS and the average true rank of the modeling technique selected by the EvoLS for each test function. The average true rank is calculated by averaging over all individuals first and then averaging over 25 runs.

It can be seen from Table 6 that the high-level models built by RankBoost played well in their roles as the true rank of the selected solution is better than 2 (i.e. the true rank of a random method) on every test function and very close to 1 on 9 of the 10 test functions. Moreover, the average true rank in EHS is better than that in EvoLS on every test

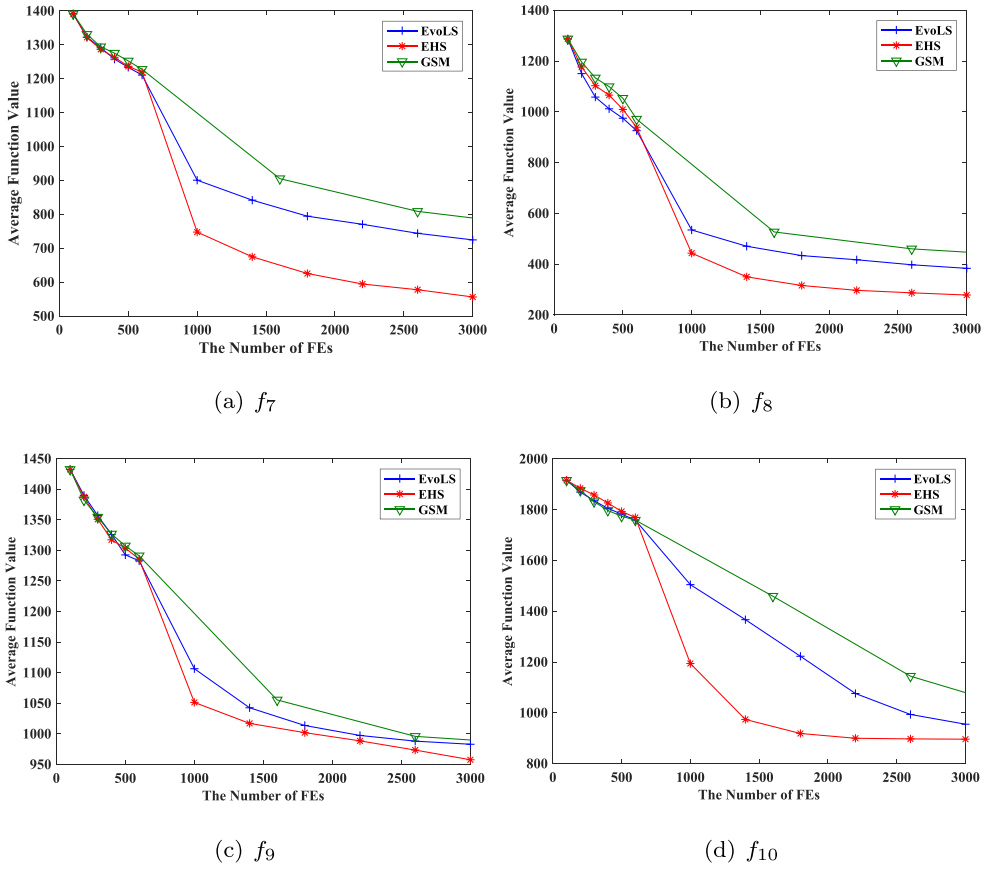

Fig. 2. Evolution curves for GSM, EvoLS and EHS on f_7-f_{10}.

Table 6
The average true rank of the solution selected by the high-level model in EHS and the average true rank of the modeling technique selected by the EvoLS for each test function. The better results are marked in **Bold**.

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
ELS	1.99	2.00	1.71	2.00	1.96	1.97	1.99	2.00	2.00	2.00
EHS	**1.03**	**1.05**	**1.13**	**1.11**	**1.89**	**1.11**	**1.11**	**1.09**	**1.08**	**1.05**

function. This can explained why EvoLS performed worse than EHS on f_2–f_{10}. For f_1, as mentioned above, the reason that EvoLS performed better than EHS is that EvoLS employed a trust-region method in the local search and a different EA operation.

4.3. Performance effect of parameters settings in EHS

In this part, experiments were conducted to study the effect of different parameter settings on the performance of EHS. Two parameters were considered, the population size and the percentage of training samples used to build the high-level model.

In the first experiment, the population size was set to 10, 20, 50 and 100 separately and EHS was run on the 10 test functions for 25 runs with each setting. The other parameters were set the same as in Section 4.2. Table 7 summarizes the average and standard deviation of the function values that EHS obtained with each of the 4 different population size settings. It can be seen from Table 7 that EHS with the 4 population size settings performed similarly on most of the 10 test functions except that EHS with $popsize = 100$ performed relatively worse on f_1, f_5 and f_9. Thus, the population size for EHS can be chosen from the interval of $[10, 50]$.

In the second experiment, the percentage of training samples used to build the high-level model was set to 60%, 70%, 80% and 90% separately and EHS was run on the 10 test functions for 25 runs with each setting. The other parameters were set the same as in Section 4.2. Table 8 summarizes the average and standard deviation of the function values that EHS obtained with each of the 4 different settings. It can be seen from Table 8 that EHS with the percentages of 80% and 90% performed best on all test functions. This is because that the high-level model is more accurate when more training samples are used. However, 100% is not a good choice as there will be no test samples to evaluate the accuracy of the high-level model. Therefore, the percentage of training samples for EHS can be chosen from $[80\%, 90\%]$.

4.4. Comparison with state-of-the-art algorithms

The performance of EHS was further compared with 3 state-of-the-art optimization algorithms. They are MVMO-PHM [27], HS-ES [28] and iDEaSm [29]. Among these 3 algorithms, iDEaSm is a differential evolution algorithm using surrogate models. MVMO-PHM and HS-ES do not use surrogate models but they are the winners in CEC-2016 competition and CEC-2018 competition on real-parameter single objective optimization, respectively.

In the experiments, the population size of EHS was set to 50 and the other parameters of EHS were set the same as in Section 4.2. For

Table 7
Experimental results of EHS with 4 different settings (10, 20, 50, 100) for *popsize* over 25 runs with 3000FEs. The best results are marked in **Bold** according to Wilcoxon rank-sum test at a level of 0.05.

Func	*popsize* = 10	*popsize* = 20	*popsize* = 50	*popsize* = 100
f_1	**2.50e+00 ± 5.48e-01**	**2.59e+00 ± 5.95e-01**	**2.83e+00 ± 7.65e-01**	4.45e+00 ± 6.44e-01
f_2	**3.62e-06 ± 1.05e-05**	**1.39e-03 ± 6.90e-03**	**2.90e-06 ± 6.04e-06**	**1.09e-06 ± 2.60e-06**
f_3	**2.81e+01 ± 2.52e-01**	**2.80e+01 ± 2.33e-01**	**2.80e+01 ± 3.42e-01**	**2.81e+01 ± 6.59e-01**
f_4	**−2.61e+02 ± 2.24e+01**	**−2.58e+02 ± 2.35e+01**	**−2.60e+02 ± 2.07e+01**	**−2.55e+02 ± 2.08e+01**
f_5	**1.23e+02 ± 5.55e+00**	1.27e+02 ± 3.44e+00	1.29e+02 ± 2.68e+00	1.31e+02 ± 2.65e+00
f_6	**−1.11e+02 ± 1.53e+00**	**−1.10e+02 ± 1.73e+00**	**−1.11e+02 ± 1.45e+00**	**−1.10e+02 ± 1.10e+00**
f_7	**5.36e+02 ± 7.49e+01**	**5.15e+02 ± 5.53e+01**	**5.29e+02 ± 4.16e+01**	**5.56e+02 ± 5.59e+01**
f_8	**3.62e+02 ± 1.79e+02**	**3.17e+02 ± 1.38e+02**	**3.12e+02 ± 1.28e+02**	**2.78e+02 ± 1.01e+02**
f_9	9.37e+02 ± 3.85e+01	**9.39e+02 ± 3.94e+01**	**9.27e+02 ± 6.28e+01**	9.58e+02 ± 4.55e+01
f_{10}	**8.95e+02 ± 1.12e+00**	**8.95e+02 ± 2.27e+00**	**8.95e+02 ± 3.46e+00**	**8.96e+02 ± 4.81e+00**

Table 8
Experimental results of EHS with 4 different settings (60%, 70%, 80%, 90%) for the percentage of training samples used to build the high-level model over 25 runs with 3000FEs. The best results are marked in **Bold**.

Func	Percentage = 60%	Percentage = 70%	Percentage = 80%	Percentage = 90%
f_1	**4.42e+00 ± 6.32e-01**	**4.21e+00 ± 8.99e-01**	**4.45e+00 ± 6.44e-01**	**4.56e+00 ± 6.48e-01**
f_2	**3.11e-06 ± 1.33e-05**	**1.35e-04 ± 5.95e-04**	**1.09e-06 ± 2.60e-06**	**2.31e-06 ± 8.50e-06**
f_3	3.06e+01 ± 5.97e-01	3.06e+01 ± 6.30e-01	**2.81e+01 ± 6.59e-01**	**2.79e+01 ± 3.11e-01**
f_4	−2.18e+02 ± 2.95e+01	−2.32e+02 ± 2.49e+01	**−2.55e+02 ± 2.08e+01**	**−2.48e+02 ± 3.52e+01**
f_5	1.32e+02 ± 2.18e+00	1.32e+02 ± 1.50e+00	**1.31e+02 ± 2.65e+00**	**1.31e+02 ± 2.36e+00**
f_6	**−1.10e+02 ± 1.46e+00**	**−1.10e+02 ± 1.80e+00**	**−1.10e+02 ± 1.10e+00**	**−1.10e+02 ± 1.58e+00**
f_7	**5.37e+02 ± 3.92e+01**	**5.57e+02 ± 5.78e+01**	**5.56e+02 ± 5.59e+01**	**5.40e+02 ± 4.65e+01**
f_8	**3.02e+02 ± 7.99e+01**	**2.87e+02 ± 6.15e+01**	**2.78e+02 ± 1.01e+02**	**3.10e+02 ± 1.32e+02**
f_9	**9.50e+02 ± 5.25e+01**	**9.57e+02 ± 4.98e+01**	**9.58e+02 ± 4.55e+01**	**9.64e+02 ± 3.72e+01**
f_{10}	**8.95e+02 ± 2.58e+00**	**8.95e+02 ± 2.33e+00**	**8.96e+02 ± 4.81e+00**	**8.96e+02 ± 3.78e+00**

Table 9
The experimental results obtained by MVMO-PHM, HS-ES, iDEaSm and EHS on f_1–f_{10} at the cost of 3000 function evaluations, and the statistical comparison results. + , − and ≈ mean that the compared algorithm performed statistically better than, worse than, and similar to EHS, respectively. The best results are marked in **Bold**.

Func	MVMO-PHM	HS-ES	iDEaSm	EHS
f_1	1.66e+01 ± 8.95e-01 −	1.17e+01 ± 3.04e+00 −	**3.26e+00 ± 7.46e-01** *a*	**2.83e+00 ± 7.65e-01**
f_2	1.10e-05 ± 7.61e-06 −	8.90e+00 ± 7.37e+00 −	1.03e+00 ± 8.06e-02 −	**2.90e-06 ± 6.04e-06**
f_3	**1.94e+01 ± 2.42e+00** +	1.85e+02 ± 7.90e+01 −	2.92e+01 ± 1.19e+00 −	2.80e+01 ± 3.42e-01
f_4	1.09e+02 ± 7.37e+01 −	−1.73e+01 ± 4.62e+01 −	−1.89e+02 ± 2.95e+01 −	**−2.60e+02 ± 2.07e+01**
f_5	1.34e+02 ± 1.70e+00 −	1.28e+02 ± 2.99e+00 *a*	**1.27e+02 ± 2.78e+00** +	1.29e+02 ± 2.68e+00
f_6	−4.43e+01 ± 6.35e+01 −	−1.09e+02 ± 3.82e+00 *a*	**−1.20e+02 ± 1.73e+00** +	−1.11e+02 ± 1.45e+00
f_7	1.14e+03 ± 1.23e+02 −	7.27e+02 ± 1.02e+02 −	5.58e+02 ± 5.94e+01 −	**5.29e+02 ± 4.16e+01**
f_8	1.01e+03 ± 1.70e+02 −	5.37e+02 ± 1.24e+02 −	3.73e+02 ± 1.18e+02 −	**3.12e+02 ± 1.28e+02**
f_9	1.31e+03 ± 5.73e+01 −	1.01e+03 ± 2.91e+01 −	**9.26e+02 ± 4.94e+00** +	9.27e+02 ± 6.28e+01
f_{10}	1.74e+03 ± 3.38e+01 −	1.35e+03 ± 1.58e+02 −	1.10e+03 ± 1.59e+02 −	**8.95e+02 ± 3.46e+00**
Total	9−/1+/0≈	8−/0+/2≈	6−/3+/1≈	∼

iDEaSm, the maximum population size was set to $18{*}n$ ($n = 30$ is the dimension of the problem) in its original paper [29]. However, this setting might not be appropriate when the allowed number of function evaluations is limited. Considering this, 4 different maximum population sizes, 50, 100, 200 and $18{*}n$ were used for iDEaSm in the experiments. The other parameters of iDEaSm was set the same as in Ref. [29]. We found that iDEaSm performed best when the maximum population size was set to 50. Therefore, iDEaSm with the maximum population size of 50 was used for comparison. The situation is similar for HS-ES. In the experiments, 3 different settings for M and N of HS-ES were used. They are $M = 50, N = 25$, $M = 100, N = 50$ and $M = 200, N = 100$. The other parameters of HS-ES was set the same as in Ref. [28]. We found that HS-ES performed best with $M = 50, N = 25$ and thus HS-ES with such a setting was used for comparison. For MVMO-PHM, we used the same setting as in Ref. [27]. Table 9 summarizes the average of the minimum function values over 25 runs achieved by each of MVMO-PHM, HS-ES, iDEaSm and EHS on each of the 10 test functions at the cost of 3000 function evaluations, respectively.

It can be seen from Table 9 that EHS performed the best compared to MVMO-PHM, HS-ES and iDEaSm. Specifically, EHS is better than MVMO-PHM on 9 test functions but only worse on 1 test function. When compared to HS-ES, EHS performed better or the same on all the 10 test function. When compared to iDEaSm, EHS outperformed iDEaSm on 6 test functions and was defeated on only 3 test functions. The reason that EHS performed worse on some test functions is that each of MVMO-PHM and iDEaSm used a different EA from which EHS used especially that iDEaSm further applied parameter adaptation strategies during the search process.

5. Conclusions and future work

When using EAs to solve computationally expensive optimization problems, surrogate models are usually applied to reduce the number of

true fitness evaluations. In surrogate-assisted evolutionary search, the choice of surrogate modeling technique can highly affect the performance of the search. However, it is not easy to decide which modeling technique to use. To address this issue, we proposed a novel strategy in this paper to do surrogate modeling technique selection in the memetic evolutionary search process. The generated optimization algorithm is called EHS. The proposed modeling technique selection strategy differs from previous approaches in employing a hierarchical structure of surrogates. A mathematic analysis was given in this paper to show that the hierarchical surrogate structure can be beneficial when the accuracy of the high-level model is larger than 0.5. Furthermore, to validate the efficiency of EHS, we conducted experiments on 10 commonly used benchmark functions with a maximum number of fitness evaluations of 3000 and used GSM, EvoLS as well as 3 state-of-the-art optimization algorithms as compared algorithms. The experimental results showed that EHS generally performed better than the compared algorithms. In future work, EHS will be tested on more test functions and real-world application problems. Also, different EAs will be considered in the framework of EHS and further analysis on the behavior of EHS will be conducted.

Acknowledgment

This work was supported in part by the National Key R&D Program of China (Grant No. 2017YFC0804003), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284), the Shenzhen Peacock Plan (Grant No. KQTD2016112514355531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2019.03.005.

References

[1] Y. Ong, P. Nair, A. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA J. 41 (4) (2003) 687–696.
[2] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (1) (2005) 3–12.
[3] Y. Ong, P. Nair, A. Keane, K. Wong, Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems, in: Knowledge Incorporation in Evolutionary Computation, vol. 167, Springer, 2005, pp. 307–331.
[4] T. Chugh, K. Sindhya, J. Hakanen, K. Miettinen, A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms, Soft Comput. (2017) 1–30.
[5] M. Le, Y. Ong, S. Menzel, Y. Jin, B. Sendhoff, Evolution by adapting surrogates, Evol. Comput. 21 (2) (2013) 313–340.
[6] D. Lim, Y. Jin, Y. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comput. 14 (3) (2010) 329–355.
[7] N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. Tucker, Surrogate-based analysis and optimization, Prog. Aero. Sci. 41 (1) (2005) 1–28.
[8] Y. Tenne, S. Armfield, Metamodel accuracy assessment in evolutionary optimization, in: Proc. of the 2008 IEEE Congress on Evolutionary Computation (CEC'08), IEEE, 2008, pp. 1505–1512.
[9] T. Goel, R. Haftka, W. Shyy, N. Queipo, Ensemble of surrogates, Struct. Multidiscip. Optim. 33 (3) (2007) 199–216.
[10] E. Acar, M. Rais-Rohani, Ensemble of metamodels with optimized weight factors, Struct. Multidiscip. Optim. 37 (3) (2009) 279–294.
[11] K.-H. Liang, X. Yao, C. Newton, Combining landscape approximation and local search in global optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), vol. 2, IEEE, Piscataway, NJ, USA, 1999, pp. 1514–1520.
[12] K.-H. Liang, X. Yao, C. Newton, Evolutionary search of approximated n-dimensional landscapes, Int. J. Knowl. Based Intell. Eng. Syst. 4 (3) (2000) 172–183.
[13] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: towards Memetic Algorithms, Tech. rep., California Institute of Technology, Pasadena, CA, USA, 1989.
[14] P. Moscato, C. Cotta, A gentle introduction to memetic algorithms, in: Handbook of Metaheuristics, Springer-Verlag, Berlin/Heidelberg, Germany, 2003, pp. 105–144.
[15] C.T. Lawrence, A.L. Tits, A computationally efficient feasible sequential quadratic programming algorithm, SIAM J. Optim. 11 (4) (2001) 1092–1118.
[16] M. Jordan, R. Jacobs, Hierarchical mixtures of experts and the em algorithm, Neural Comput. 6 (2) (1994) 181–214.
[17] T. Runarsson, Ordinal Regression in Evolutionary Computation, Proc. of the Parallel Problem Solving from Nature-PPSN IX, 2006, pp. 1048–1057.
[18] I. Loshchilov, M. Schoenauer, M. Sebag, Comparison-based optimizers need comparison-based surrogates, in: International Conference on Parallel Problem Solving from Nature, Springer-Verlag, Berlin/Heidelberg, Germany, 2010, pp. 364–373.
[19] Y. Freund, R. Iyer, R. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, J. Mach. Learn. Res. 4 (2003) 933–969.
[20] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (4) (2001) 481–506.
[21] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, KanGAL Report 2005005, 2005.
[22] R. Jin, W. Chen, T.W. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, Struct. Multidiscip. Optim. 23 (1) (2001) 1–13.
[23] F. Viana, Surrogates Toolbox User's Guide, 2011, https://sites.google.com/site/srgtstoolbox/.
[24] S. Lophaven, H. Nielsen, J. Sondergaard, Dace - a Matlab Kriging Toolbox, Tech. rep.. Technical University of Denmark, Denmark, 2002, http://www2.imm.dtu.dk/hbn/dace/.
[25] G. Jekabsons, Rbf: Radial Basis Function Interpolation for Matlab/octave, 2009, http://www.cs.rtu.lv/jekabsons/regression.html.
[26] MATLAB, Constrained Nonlinear Optimization Algorithms, 2017b (2017) https://cn.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html.
[27] J.L. Rueda Torres, I. Erlich, Solving the Cec 2016 Real-Parameter Single Objective Optimization Problems through Mvmo-Phm, 2016, http://resolver.tudelft.nl/uuid:03415237-0448-48db-8cba-889c3d34d1d3.
[28] G. Zhang, Y. Shi, Hybrid sampling evolution strategy for solving single objective bound constrained problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC'18), IEEE, 2018, pp. 1–7.
[29] N.H. Awad, M.Z. Ali, R. Mallipeddi, P.N. Suganthan, An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization, Inf. Sci. 451 (2018) 326–347.