2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Locally Weighted Metamodel for Pre-selection in Evolutionary Optimization

Qiuxiao Liao
Department of Computer Science and Technology
East China Normal University
Shanghai, China, 200241
Email: qxliao@student.ecnu.edu.cn

Aimin Zhou
Department of Computer Science and Technology
East China Normal University
Shanghai, China, 200241
Email: amzhou@cs.ecnu.edu.cn

Guixu Zhang
Department of Computer Science and Technology
East China Normal University
Shanghai, China, 200241
Email: gxzhang@cs.ecnu.edu.cn

***Abstract*—The evolutionary algorithms are usually criticized for their slow convergence. To address this weakness, a variety of strategies have been proposed. Among them, the metamodel or surrogate based approaches are promising since they replace the original optimization objective by a metamodel. However, the metamodel building itself is expensive and therefore the metamodel based evolutionary algorithms are commonly applied to expensive optimization. In this paper, we propose an alternative metamodel, named locally weighted metamodel (LWM), for the pre-selection in evolutionary optimization. The basic idea is to estimate the objective values of candidate offspring solutions for an individual, and choose the most promising one as the offspring solution. Instead of building a global model as many other algorithms do, a LWM is built for each candidate offspring solution in our approach. The LWM based pre-selection is implemented in a multi-operator based evolutionary algorithm, and applied to a set of test instances with different characteristics. Experimental results show that the proposed approach is promising.**

I. Introduction

Many real world applications involve optimizing an objective or a function. In this paper, we consider the following *box-constrained continuous optimization problem*.

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \Omega, \end{aligned} \tag{1}$$

where $x = (x^1, x^2, \cdots, x^d)^T$ is a decision variable vector, $\Omega = [a^i, b^i]^d$ is the feasible region of the decision space, where $a^i < b^i$, $a^i \in R$ and $b^i \in R$ are the lower and upper boundaries of the decision space in the ith dimension, respectively. $f(x) : \Omega \to R$ is a continuous mapping from the decision space to the objective space, and R is the objective space.

According to the properties of the problems to tackle, a variety of methods, such as linear programming, least square optimization, convex optimization, etc., have been proposed [1]. However in practice, the properties of the problems are unknown or even there do not exist mathematical formulations of the problems.Thus physical experiments or computational simulations are applied to calculate the objective values instead. In such cases, the traditional methods may not be applicable and many researchers resort to heuristic optimization methods. Evolutionary Algorithms (EAs) are such kind of methods [2]. The major differences between EAs and conventional methods include (a) EAs use a population of candidate solutions to search for the optimal solution, and (b) EAs rely on the objective function values to guide the search. Furthermore, EAs are easy to implement and are able to converge close to the global optimum. For these reasons, EAs have been successfully applied to different fields.

However, EAs do not work without any cost. They are usually criticized for their slow convergence. A major reason is that EAs usually require a large number of objective function evaluations. To deal with this problem, a variety of strategies have been proposed in the last decades. Among them, the metamodel or surrogate based approaches are promising ones [3], [4], [5], [6], [7]. The basic idea behind a metamodel based EA is that it builds a metamodel or a surrogate model based on the obtained solutions and their objective values, and replaces the original objective function by the metamodel sometimes in the running process. To use a metamodel based EA, some issues should be considered, such as which kind of metamodel to use, and how to combine the metamodel with an EA more efficiently. A comprehensive survey of the metamodel based evolutionary optimization can be found in [8], [9], [10].

In the metamodel based evolutionary optimization, it is usually assumed that modeling is much cheaper compared to the objective function evaluation. For this reason, the metamodel based approaches are usually applied to expensive optimization. It is arguable that the metamodel can be also utilized to handle general optimization problems if some cheap models could be found. Following this idea, some models based on nonparametric density estimation techniques have been proposed [11], [12]. In this paper, an alternative metamodel, named locally weighted metamodel (LWM) is proposed, for the pre-selection in evolutionary optimization. Instead of building a global model as many metamodel based evolutionary algorithms do, a LWM is built for each candidate offspring solution in this approach. In the reproduction process, a set of candidate offspring solutions are generated for each parent individual, and only the best one according to the LWM is chosen as the offspring solution.

The rest of this paper is organized as follows. Section II briefly reviews some widely used metamodels used in evolutionary optimization. Section III presents a new metamodel

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

based method for optimization. The algorithm framework, the reproduction operator, and the new metamodel are introduced in detail. The experimental results are reported in Section IV. Finally, this paper is concluded in Section V.

II. Related Work

The metamodels are actually the regression or interpolation models. Therefore, the regression or interpolation methods from the community of statistical and machine learning can be naturally applied as metamodels and used in evolutionary optimization [13], [14].

Let $\{x_i, f(x_i)\}, i = 1, \cdots, N$ be a set of N training data points where $x_i \in \mathbb{R}^d$, $f(x_i) \in \mathbb{R}$. The target of metamodeling is to find $\hat{f}(x)$ which can fit $f(x)$ well for all the given training points. Some widely used models are summarized as follows.

- Polynomial Regression (PR) [9]: It is a form of linear regression which fits a linear or non-linear model to a training set. A PR model that is linear in β can be expressed as follows
$$\hat{f}(x) = \hat{x}^T \beta, \tag{2}$$
where $\hat{x}$ is a basis vector based on the decision variable vector x, and β is the coefficient vector, defining the complexity of the model. The least square method can be used to fit this model.
- Gaussian Processes (GP) [15], [16]: It is also known as Kriging regression. It assumes that the function value is from a Gaussian distribution as follows
$$\hat{f}(x) \sim N(\mu, \sigma^2), \tag{3}$$
where μ denotes the mean value, and σ^2 is the variance. Both μ and σ^2 are estimated based on the given training data set. A major advantage of GP over other regression methods is that it offers not only the estimated objective value but also the error of the estimation. To have a high quality estimation, it may need to optimize some parameters which might be time-consuming.
- Artificial Neural Network (ANN) [17]: An ANN consists of a system of interconnected nodes, called neurons. Each neuron is a computational model that computes values from inputs. When fixing the form of the neurons and the connection structure of the ANN, the target is to fit the following model by the given training data set.
$$\hat{f}(x) = ANN(x, w), \tag{4}$$
where w denotes the weights between the neurons. The multilayer perceptron (MLP) is widely used and the activation function or neuron is often defined by a logistic sigmoid function $f_a(a) = 1/(1 + exp(-a))$.
- Radial Basis Function (RBF) [18]: RBF is a special case of ANN. It usually builds up an approximation function in the form of
$$\hat{f}(x) = \sum_{i=1}^{N_{RBF}} w_i \phi(||x - y_i||), \tag{5}$$
where N_{RBF} denotes the number of radial basis functions, and each basis function is with a center y_i and a weight w_i. The centers are learned from the training data set, and the weights w_i can be estimated using the least square method.
- Support Vector Regression (SVR) [19]. The SVR is an extension of support vector machine. A general form of SVR can be defined as follows
$$\hat{f}(x) = \sum_{i=1}^{N} w_i k(x_i, x)) + b, \tag{6}$$
where w_i are the coefficients, b is the basis, and $k(\cdot, \cdot)$ is a kernel function. A major advantage with SVR is that the parameters can be optimized by solving a quadratic problem.
- Nonparametric Density Estimation: It is proved that minimizing an objective function is equal to maximizing a corresponding density function [12]. Therefore, we can replace the objective function values by estimating the density of the solutions. A nonparametric density estimation method was thus introduced as follows [11].
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1/\bar{f}(x_i)}{\sum_{j=1}^{N} 1/\bar{f}(x_j)} \frac{1}{w} \varphi \left(\frac{||x - x_i||}{w} \right) \right), \tag{7}$$
where
  - w is the window width and it is estimated as
$$w = \left(\frac{1}{d} \sum_{j=1}^{d} \left(\bar{a}^j - \underline{b}^j \right)^2 \right)^{1/2},$$
where $\bar{a}^j = \arg \max_{i=1,\cdots,n} x_i^j$ and $\underline{b}^j = \arg \min_{i=1,\cdots,n} x_i^j$;
  - $\bar{f}(x_i) = f(x_i) + H$ where H is a positive number making $\bar{f}(x_i) > 0$;
  - $\varphi(u)$ is a window function which is defined as
$$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}.$$

  The major advantage of nonparametric density estimation is that the density function has no need to be chosen beforehand.

III. A New Metamodel based Evolutionary Algorithm

Most of the above mentioned models are directly borrowed from statistical and machine learning community and do not consider the properties of evolutionary algorithms. Furthermore, it is time-consuming to train those models. Considering these two issues, a locally weighted metamodel (LWM) is proposed in this paper for evolutionary optimization. Based on the LWM, a pre-selection strategy is applied to filter 'bad' solutions prior to real function evaluations.

In the following, we firstly present the algorithm framework, then introduce the reproduction operator, and finally propose the new metamodel in detail.

$$y_1^j = \begin{cases} x_{r1}^j + F \cdot (x_{r2}^j - x_{r3}^j) & \text{if } rand < C_r \text{ or } j = j_{rnd} \\ x^j & \text{otherwise} \end{cases}$$

$$y_2^j = \begin{cases} x_{r1}^j + F \cdot (x_{r2}^j - x_{r3}^j) + F \cdot (x_{r4}^j - x_{r5}^j) & \text{if } rand < C_r \text{ or } j = j_{rnd} \\ x^j & \text{otherwise} \end{cases}$$

$$y_3^j = \begin{cases} x^j + rand \cdot (x_{r1}^j - x^j) + F \cdot (x_{r2})^j - x_{r3}^j) & \text{if } rand < C_r \text{ or } j = j_{rnd} \\ x^j & \text{otherwise} \end{cases}$$

Fig. 1. Multiple reproduction operators

A. Algorithm Framework

Denote the locally weighted metamodel based evolutionary algorithm as LWM-EA. In each generation, the LWM-EA maintains

- a set of N solutions $\{x_1, x_2, \cdots, x_N\}$,
- their objective values $\{f(x_1), f(x_2), \cdots, f(x_N)\}$.

The framework of LWM-EA is shown as follows.

Algorithm 1: Procedure of LWM-EA

```
1  Initialize the population {x_1, ..., x_N} and evaluate them;
2  while not terminate do
3      foreach x ∈ {x_1, ..., x_N} do
4          Generate M trial points y_1, ..., y_M;
5          For each y_i, build a LWM f̂ and estimate its
           value f̂(y_i), i = 1, ..., M;
6          Let y* = arg min_{y∈{y_1,y_2,...,y_M}} f̂(y) be the offspring
           solution of x;
7          Evaluate y*;
8          if f(y*) < f(x) then
9              Replace x by y*;
10         end
11     end
12 end
```

We would like to explain the algorithm as follows.

- *Initialization:* The initial population is uniformly randomly sampled from the search space in *Line 1*.
- *Termination Condition:* The algorithm terminates when a given maximum number of generations is reached in *Line 2*.
- *Reproduction Procedure:* For each solution, M trial offspring solutions are generated in *Line 4* and the details will be discussed in the following section.
- *Selection Procedure:* It should be noted that the selection procedure is performed in *Lines 5-10*. Firstly, the M trial solutions are evaluated by a metamodel in *Line 5*, the most promising one is chosen in *Line 6*, and finally it will replace the parent solution x if it has better quality according to the original objective function in *Line 9*. The details of the metamodel building will be discussed shortly in the following section.

B. Offspring Reproduction

Since a pre-selection strategy is used to filter 'bad' offspring solutions, a multiple candidate offspring solutions need to be generated for each parent solution. In our previous work [11], multiple solutions are sampled from the probability distribution model. In this paper, we use a multi-operator search strategy. The reproduction operator introduced in CoDE [20] is used for this purpose.

For a parent solution x, we generate $M = 9$ trial offspring solutions y_1, $\cdots$, y_9 by the three reproduction operators in Fig. 1. Each operator uses three types of control parameters: $[F = 1.0, C_r = 0.1]$, $[F = 1.0, C_r = 0.9]$, and $[F = 0.8, C_r = 0.2]$.

It has been shown that the multi-operator search strategy in CoDE is promising for dealing with continuous optimization problems and more details are referred to [20]. It should be noted that (a) all the trial offspring solutions are evaluated by the objective functions in CoDE while only one in our approach, and (b) any multi-operator search strategies can be applied here in our framework.

C. Locally Weighted Metamodel

It is reasonable to assume that the adjacent individuals have similar fitness values for continuous optimization problems. The fitness values of a new individual can thus be estimated by the objective values of the nearby points. Following this idea, we propose a locally weighted metamodel (LWM) in this section.

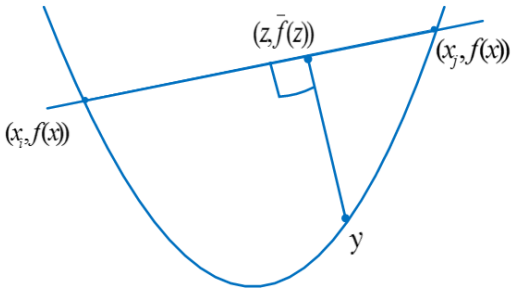


Fig. 2. Illustration of the relationship between y, x_i, x_j, and z.

TABLE I
THE 12 TEST INSTANCES USED IN COMPARISON STUDY

Test Instance	Search Space
$f_1(x) = \sum_{i=1}^{d} (x^i)^2$	$[-100, 100]^d$
$f_2(x) = \sum_{i=1}^{d} \lvert x^i \rvert + \prod_{i=1}^{d} \lvert x^i \rvert$	$[-10, 10]^d$
$f_3(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x^j \right)^2$	$[-100, 100]^d$
$f_4(x) = \max \{ \lvert x^i \rvert \}$	$[-100, 100]^d$
$f_5(x) = \sum_{i=1}^{d-1} \left[100(x^{i+1} - (x^i)^2)^2 + (x^i - 1)^2 \right]$	$[-30, 30]^d$
$f_6(x) = \sum_{i=1}^{d} \lfloor x^i + 0.5 \rfloor^2$	$[-100, 100]^d$
$f_7(x) = \sum_{i=1}^{n} i(x^i)^4 + rand[0, 1)$	$[-1.28, 1.28]^d$
$f_8(x) = \sum_{i=1}^{d} \left[(x^i)^2 - 10\cos(2\pi x^i) + 10 \right]$	$[-5.12, 5.12]^d$
$f_9(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}(x^i)^2} \right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x^i) \right) + 20 + e$	$[-32, 32]^d$
$f_{10}(x) = \frac{1}{4000}\sum_{i=1}^{d}(x^i)^2 - \prod_{i=1}^{d}\cos\left(\frac{x^i}{\sqrt{i}}\right) + 1$	$[-600, 600]^d$
$f_{11}(x) = \frac{\pi}{d}\left\{ 10\sin^2(\pi y^i) + \sum_{i=1}^{d-1}(y^i - 1)^2[1 + 10sin^2(\pi y^i + 1)] + (y^d - 1)^2 \right\} + \sum_{i=1}^{d} u(x^i, 10, 100, 4)$ where $y^i = 1 + \frac{1}{4}(x^i + 1)$, and $u(x^i, a, k, m) = \begin{cases} k(x^i - a)^m & x^i > a \\ 0 & -a \le x^i \le a \\ k(-x^i - a)^m & x^i < -a \end{cases}$	$[-50, 50]^d$
$f_{12}(x) = 0.1\left\{ sin^2(3\pi x^i) + \sum_{i=1}^{d-1}(x^i - 1)^2[1 + \sin^2(3\pi x^{i+1})] + (x^d - 1)^2[1 + \sin^2(2\pi x^d)] \right\} + \sum_{i=1}^{d} u(x^i, 5, 100, 4)$ where $u(\cdot)$ is the same as in f_{11}.	$[-50, 50]^d$

For a new trial solution y, K nearest solutions in the current population are chosen to estimate its objective value. Without loss of generality, the K nearest solutions are denoted as $x_1, x_2, \cdots, x_K$. Let x_i and x_j be any two nearest solutions, and we can draw a line $\overline{x_i x_j}$ between the two points. Let z be the projection of x in line $\overline{x_i x_j}$. We can estimate the value of z by a linear interpolation as follows.

$$\bar{f}(z) = f(x_i) + (f(x_j) - f(x_i))\frac{(z - x_i)^T(x_j - x_i)}{(x_j - x_i)^T(x_j - x_i)} \quad (8)$$

Fig. 2 illustrates the idea to project x in line $\overline{x_i x_j}$.

For the given K nearest solutions, we can find $\frac{1}{2}K(K-1)$ such lines and project points. Then we use these project points to estimate the value of y in this equation.

$$\hat{f}(y) = \frac{\sum_{i,j,i\neq j} w(z)\bar{f}(z)}{\sum_{i,j,i\neq j} w(z)} \quad (9)$$

where $w(z)$ denotes the weight of z.

To have a high quality estimation, the points y, x_i, x_j, and z should be close to each other. To this end, the weight is defined as follows.

$$w(z) = \frac{1}{\|x_i - z\| + \|x_j - z\| + \|x - z\|} \quad (10)$$

We would make the following comments on the LWM.

- This model considers the property of the population of

TABLE II
MEAN±STD. VALUES OF THE RESULTS OBTAINED BY THE TWO COMPARISON ALGORITHMS AFTER $1,500$ AND $3,000$ GENERATIONS OVER 50 RUNS FOR ALL THE TEST INSTANCES.

	$generation = 1,500$		$generation = 3,000$	
instance	LWM-EA	nEDA	LWM-EA	nEDA
f_1	2.762e-31±1.3668e-30	**1.583e-71**±1.2350e-71	7.219e-64±3.5726e-63	**3.380e-89**±3.3860e-89
f_2	3.833e-18±1.8969e-17	**1.563e-38**±6.8690e-39	3.580e-36±1.7715e-35	**8.816e-51**±6.1250e-51
f_3	1.154e-05±5.7127e-05	**3.137e-19**±8.7260e-19	1.759e-12±8.7050e-12	**5.702e-41**±1.5410e-40
f_4	1.562e-08±7.7320e-08	**2.147e-31**±3.8790e-32	2.526e-16±1.2501e-15	**3.004e-34**±1.0180e-34
f_5	**6.964e-04**±3.4465e-03	1.142e+01±2.7740e+01	**4.440e-06**±2.1970e-05	3.452e+00±8.0280e+00
f_6	**0.000e+00**±0.0000e+00	**0.000e+00**±0.0000e+00	**0.000e+00**±0.0000e+00	**0.000e+00**±0.0000e+00
f_7	**9.751e-05**±4.8255e-04	3.022e-04±2.8830e-04	**5.803e-05**±2.8716e-04	1.790e-04±1.4140e-04
f_8	**9.218e-02**±4.5619e-01	1.450e+01±3.5500e+00	**7.278e-02**±3.6018e-01	1.121e+00±3.1850e+00
f_9	**1.599e-16**±7.9116e-16	3.713e-15±9.7360e-16	**1.776e-17**±8.7907e-17	3.357e-15±1.3790e-15
f_{10}	**2.056e-03**±1.0176e-02	1.945e-01±1.1870e-01	**8.540e-04**±4.2263e-03	1.515e-01±1.1000e-01
f_{11}	**1.336e-30**±6.6094e-30	1.096e-20±9.1200e-36	**1.885e-33**±9.3266e-33	1.096e-20±9.1200e-36
f_{12}	**1.723e-30**±8.5261e-30	3.485e-21±1.5200e-36	**5.399e-34**±2.6719e-33	3.485e-21±1.5200e-36

an evolutionary algorithm: as the running process, the population will converge and the solutions will hopefully be more close to each other. Therefore, the estimation quality of LWM will increase during the running process.
- The cost of LWM comes from two aspects: (a) the choosing of K nearest solutions, and (b) the objective estimation in (9). The time complexity of the two procedures is on $O(N^2)$.

It should be noted that our LWM is similar to the locally weighted regression (LWR) method proposed in [21]. The major difference is that the LWM does not directly use the function values of the neighbor points but uses the values of the projected points.

IV. EXPERIMENTAL RESULTS

A. Test Instances and Parameter Settings

In this section, we apply the proposed LWM-EA to 12 widely used test instances from [22]. The details of these test instances are listed in Table I. The nonparametric estimation of distribution algorithm (nEDA) [11] is used for comparison study. nEDA and LWM-EA share the similar idea, and the major differences between the two approaches are (a) nEDA uses a Gaussian model to sample new trial solutions, and (b) nEDA utilizes a nonparametric density estimation method as a metamodel. More details of nEDA are referred to [11].

The parameters for experimental study are as follows.

- The dimension of the test instance is $d = 10$ for all test instances.
- The population size for both algorithms is $N = 100$.
- The maximum evolutionary generations is $3,000$.
- In LWM-EA, the number of candidate offspring solutions is $M = 9$, and the number of nearest solutions is $K = 5$.
- In nEDA, the number of candidate offspring solutions is $M = 10$ as used in [11].
- The experimental results are based on 50 independent executions of the two algorithms for each instance.

B. Comparison Results and Analysis

LWM-EA is compared with nEDA on the 12 test instances listed in Table I. The statistical results of the mean and std. values of the two algorithms after $1,500$ and $3,000$ generations are shown in Table II.

As shown in Table II, the statistical results after $1,500$ and $3,000$ generations are consistent with each other. It is obvious that LWM-EA and nEDA perform similarly on f_6, LWM-EA outperforms nEDA on f_5, f_7-f_{12}, while LWM-EA performs worse than nEDA on f_1-f_4.

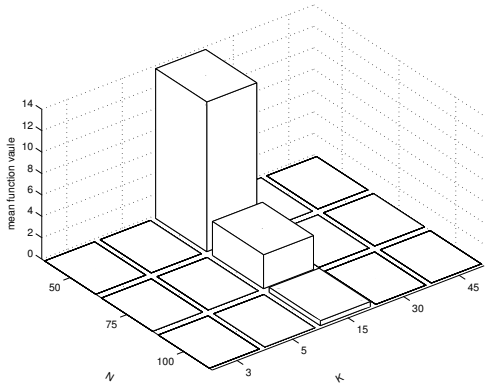
It should be noted that although LWM-EA does not work as well as nEDA on f_1-f_4, it still can obtain good results. However on f_5, f_8, and f_{10}, nEDA fails to achieve good results. Therefore, we may safely draw a conclusion that LWM-EA performs better than nEDA on these test instances.

C. Sensitivity to Control Parameters

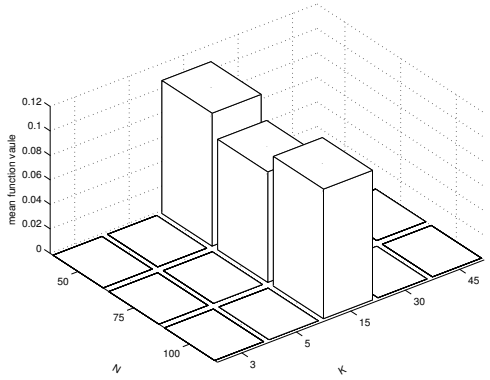
The proposed LWM-EA has a major control parameter, i.e., the number of nearest solutions K for LWM model building. This section studies the sensitivity to this control parameters.

The influence of the population size N is also taken into consideration.

We set $N = 50, 75$, and 100, and $K = 3, 5, 15, 30$, and 45 respectively. The other algorithm parameters are the same as in the previous section. f_1 and f_2 are chosen for the study.



(a) f_1



(b) f_2

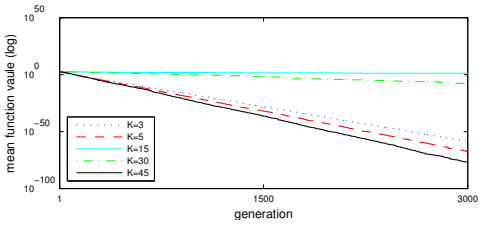
Fig. 3. The average function values obtained with different control parameters on (a) f_1, and (b) f_2.

Fig. 3 plots the average function values obtained by LWM-EA with different control parameters on f_1 and f_2. It is clear that when $K = 15$, LWM-EA achieves the worst results on both the test instances, while when K is less or more, LWM-EA can get relatively better performance with different parameter settings of the population size.

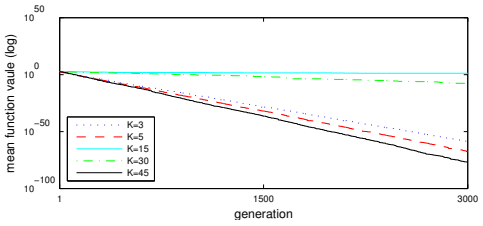
Figs. 4 and 5 show the average objective values versus generations on f_1 and f_2 respectively with different settings of N and K. The results are very consistent with those in Fig. 3 that when $K = 3, 5$, and 45, LWM-EA performs better than LWM-EA with $K = 15$ and 30. Furthermore, we can see that when $N = 50$, LWM-EA with $K = 45$ performs the best while when $N = 75$ and 100, LWM-EA with $K = 5$ performs the best slightly.

It is not clear why LWM-EA performs the worst with median settings of K. The reason might be that the quality of the LWM is not high with median sizes of training points, and it is worth for further investigation in the future.

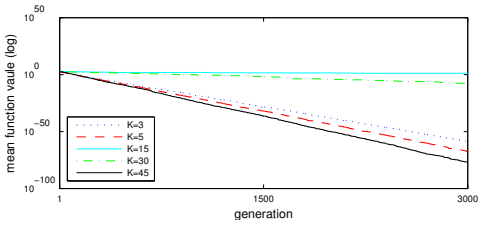
When giving a large size of K, the complexity of LWM modeling process will increase. Therefore, we choose a small



(a) $N = 50$

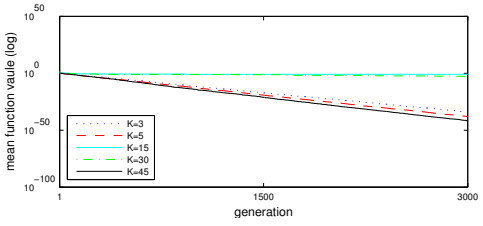


(b) $N = 75$

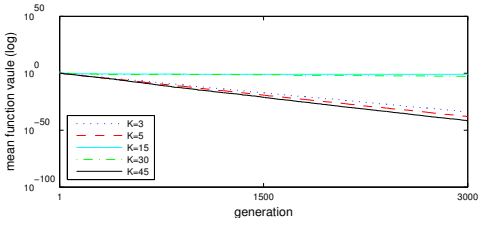


(c) $N = 100$

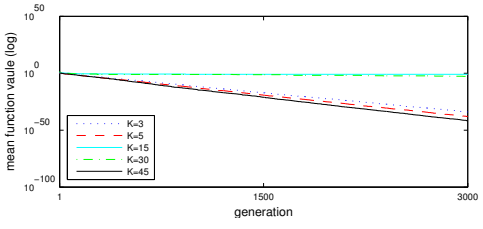
Fig. 4. The average objective values versus generations on f_1 with population size (a) $N = 50$, (b) $N = 75$, and (c) $N = 100$.



(a) $N = 50$



(b) $N = 75$



(c) $N = 100$

Fig. 5. The average objective values versus generations on f_2 with population size (a) $N = 50$, (b) $N = 75$, and (c) $N = 100$.

TABLE III
MEAN±STD. VALUES OF THE RESULTS OBTAINED BY LWM-EA AFTER $3,000$ GENERATIONS OVER 50 RUNS FOR ALL THE TEST INSTANCES WITH VARIABLE DIMENSION $d = 10, 15, 20, 25$,AND 30.

	$d = 10$	$d = 15$	$d = 20$	$d = 25$	$d = 30$
f_1	7.219e-64±3.5726e-63	5.335e-27±2.6400e-26	4.001e-13±1.9800e-12	1.167e-06±5.7768e-06	6.957e-03±3.4429e-02
f_2	3.580e-36±1.7715e-35	4.283e-11±2.1193e-10	4.489e-08±2.2216e-07	1.842e-06±9.1176e-06	3.507e-05±1.7353e-04
f_3	1.759e-12±8.7050e-12	2.220e+00±1.0987e+01	9.153e+01±4.5298e+02	4.020e+02±1.9892e+03	6.765e+02±3.3480e+03
f_4	2.526e-16±1.2501e-15	3.343e-04±1.6542e-03	7.747e-02±3.8336e-01	5.084e-01±2.5162e+00	1.502e+00±7.4311e+00
f_5	4.440e-06±2.1970e-05	2.226e+00±1.1016e+01	3.846e+00±1.9034e+01	2.940e+00±1.4548e+01	5.526e+00±2.7346e+01
f_6	0.000e+00±0.0000e+00	0.000e+00±0.0000e+00	0.000e+00±0.0000e+00	0.000e+00±0.0000e+00	0.000e+00±0.0000e+00
f_7	5.803e-05±2.8716e-04	2.766e-04±1.3689e-03	5.238e-04±2.5920e-03	1.077e-03±5.3307e-03	2.920e-03±1.4448e-02
f_8	7.278e-02±3.6018e-01	1.066e-01±5.2750e-01	1.104e+00±5.4636e+00	2.475e+00±1.2249e+01	6.889e+00±3.4090e+01
f_9	1.776e-17±8.7907e-17	8.436e-13±4.1746e-12	8.715e-07±4.3128e-06	2.762e-04±1.3668e-03	1.107e-02±5.4784e-02
f_{10}	8.540e-04±4.2263e-03	3.455e-03±1.7097e-02	1.190e-02±5.8898e-02	3.882e-02±1.9209e-01	7.799e-02±3.8593e-01
f_{11}	1.885e-33±9.3266e-33	1.552e-12±7.6812e-12	4.100e-05±2.0291e-04	2.157e-01±1.0672e+00	2.133e+00±1.0555e+01
f_{12}	5.399e-34±2.6719e-33	6.357e-08±3.1457e-07	6.620e-03±3.2760e-02	7.295e-01±3.6102e+00	7.297e+02±3.6112e+03

one in the other experiments.

D. Scalability of LWM

Usually, the model quality will decrease as the dimension of the problems increase. In this section, we investigate the scalability of the LWM. For this purpose, LWM-EA is applied to the 12 test instances with dimension $d = 10, 15, 20, 25$, and 30. The mean and std. deviation values of the algorithm over 50 runs for the test instances after $3,000$ generations are shown in Table III.

It is clear from Table III that on most of the test instances, the performance of LWM-EA decreases as the variable dimension increases. The reason is that the quality of the LWM decreases as the variable dimension increases. It is similar to that in nEDA [11].

To build a high quality model in high dimensional space, we may need (a) more training points, and (b) the training points are similar with the given points. However, these two requirements are hard to be satisfied since the population in the high dimensional space is sparse and might be far away from each other. Therefore, it might not be suitable to apply metamodel directly to high dimensional problems. How to use metamodel more efficiently for high dimensional problems is worth for further investigation in the future.

V. CONCLUSIONS

In this paper, we proposed a locally weighted metamodel (LWM) to do pre-selection in an evolutionary algorithm. A new evolutionary algorithm, combing the LWM and a multi-operator search strategy, called LWM-EA, was thus proposed. In each generation, nine candidate offspring solutions are generated for each parent solution by the multi-operator, and only the most promising one according to the LWM is chosen as the offspring solution. LWM-EA was compared with an estimation of distribution algorithm based on nonparametric density estimation (nEDA) on 12 selected test instances. The experimental results indicated that LWM-EA outperformed nEDA on 7 out of the 12 test instances.

We also empirically studied the influences of the number of points to build the LWM and the scalability of the LWM. The experimental results suggested that (a) the LWM is sensitive to median sizes of the training points, and (b) the LWM does not work well when the variable dimension increases. Therefore, it is worth to investigate how to choose a proper parameter to build the LWM, and how to apply the LWM to high dimensional problems in the future. Furthermore, the comparison between different metamodels is another direction worth for future work.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation of China (No.61273313).

REFERENCES

[1] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[2] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*, 1997.

[3] Y. S. Ong, K. Lum, P. Nair, D. Shi, and Z. Zhang, "Global convergence of unconstrained and bound constrained surrogate-assisted evolutionary search in aerodynamic shape design," in *IEEE Congress on Evolutionary Computation (CEC 2003)*, 2003, pp. 1856–1863.

[4] Z. Zhou, Y. S. Ong, and P. B. Nair, "Hierarchical surrogate-assisted evolutionary optimization framework," in *IEEE Congress on Evolutionary Computation (CEC 2004)*, 2004, pp. 1586–1593.

[5] Y. S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392–404, 2006.
[6] D. Lim, Y. Jin, Y. S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.
[7] I. Loshchilov, "Surrogate-assisted evolutionary algorithms," Ph.D. dissertation, Université Paris Sud-Paris XI, 2013.
[8] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.
[9] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, "A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 2832–2839.
[10] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.
[11] L. Zhou, A. Zhou, G. Zhang, and C. Shi, "An estimation of distribution algorithm based on nonparametric density estimation," in *IEEE Congress on Evolutionary Computation (CEC 2011)*, 2011, pp. 1597–1604.
[12] W. Gong, A. Zhou, and Z. Cai, "A multi-operator search strategy based on cheap surrogate models for evolutionary optimization," 2014, submit.
[13] T. Hastie, R. Tibshirani, and J. J. H. Friedman, *The elements of statistical learning*. Springer New York, 2001.
[14] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. springer New York, 2006.
[15] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.
[16] A. Zhou and Q. Zhang, "A surrogate-assisted evolutionary algorithm for minimax optimization," in *IEEE Congress on Evolutionary Computation (CEC 2010)*, 2010, pp. 1–7.
[17] L. Marim, M. Lemes, and A. Dal Pino Jr, "Neural-network-assisted genetic algorithm applied to silicon clusters," *Physical Review A*, vol. 67, no. 3, pp. 1–8, 2003.
[18] S. Zapotecas Martínez and C. A. Coello Coello, "MOEA/D assisted by RBF networks for expensive multi-objective optimization problems," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, 2013, pp. 1405–1412.
[19] Y. S. Liau, K. C. Tan, J. Hu, X. Qiu, and S. B. Gee, "Machine learning enhanced multi-objective evolutionary algorithm based on decomposition," in *Intelligent Data Engineering and Automated Learning–IDEAL 2013*, 2013, pp. 553–560.
[20] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.
[21] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: an approach to regression analysis by local fitting," *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 596–610, 1988.
[22] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.