Lu XF, Tang K. Classification- and regression-assisted differential evolution for computationally expensive problems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(5): 1024–1034 Sept. 2012. DOI 10.1007/s11390-012-1282-4

Classification- and Regression-Assisted Differential Evolution for Computationally Expensive Problems

Xiao-Fen Lu (陆晓芬) and Ke Tang (唐　珂), *Member*, *IEEE*

Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology
University of Science and Technology of China, Hefei 230027, China

E-mail: xiaofen@mail.ustc.edu.cn; ketang@ustc.edu.cn

Received September 26, 2011; revised March 6, 2012.

Abstract Differential Evolution (DE) has been well accepted as an effective evolutionary optimization technique. However, it usually involves a large number of fitness evaluations to obtain a satisfactory solution. This disadvantage severely restricts its application to computationally expensive problems, for which a single fitness evaluation can be highly time-consuming. In the past decade, a lot of investigations have been conducted to incorporate a surrogate model into an evolutionary algorithm (EA) to alleviate its computational burden in this scenario. However, only limited work was devoted to DE. More importantly, although various types of surrogate models, such as regression, ranking, and classification models, have been investigated separately, none of them consistently outperforms others. In this paper, we propose to construct a surrogate model by combining both regression and classification techniques. It is shown that due to the specific selection strategy of DE, a synergy can be established between these two types of models, and leads to a surrogate model that is more appropriate for DE. A novel surrogate model-assisted DE, named Classification- and Regression-Assisted DE (CRADE) is proposed on this basis. Experimental studies are carried out on a set of 16 benchmark functions, and CRADE has shown significant superiority over DE-assisted with only regression or classification models. Further comparison to three state-of-the-art DE variants, i.e., DE with global and local neighborhoods (DEGL), JADE, and composite DE (CoDE), also demonstrates the superiority of CRADE.

Keywords surrogate model, differential evolution, computationally expensive problem

1 Introduction

Differential Evolution (DE), proposed by Storn and Price in 1995[1], is a simple yet powerful evolutionary algorithm (EA). Like other EAs, one main advantage of DE is that it can search for an optimal or a near optimal solution without the need for gradient information. Hence, DE is applicable to any problem as long as the quality of candidate solutions is measurable, and has achieved great success on a wide variety of real-world problems[2]. However, evaluating the quality of a candidate solution (i.e., fitness evaluation) is not always a trivial task in real-world applications, especially for computationally expensive problems, which broadly exist in complex engineering design fields such as structural design[3-5], electromagnetics[6] and multidisciplinary system design[7]. For such problems, one fitness evaluation may require the simulation of high-fidelity analysis codes, such as Computational Structural Mechanics, Computational Electro-Magnetics, and Computational Fluid Dynamics, which may take from minutes to hours of supercomputer time. As a result, it often becomes prohibitive to use DE to solve computationally expensive problems.

In fact, computationally expensive problems raise challenges not only for DE, but also for all EAs, since all EAs rely on the fitness of solutions to guide their search. Hence, much research has been conducted to make EAs suitable for computationally expensive problems. Among them, the most commonly adopted approach is to construct the so-called surrogate models[8-10]. Briefly speaking, a surrogate model is a computationally efficient model, which is expected to provide guidance to the search in a way that requires less real fitness evaluations. For example, a surrogate model can be a regression model that is used to predict the fitness of candidate solutions. In an ideal scenario where the regression model can predict the fitness of all candidate solutions precisely, there is no need to call the

Regular Paper

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61028009, U0835002, and 61175065, the Natural Science Foundation of Anhui Province of China under Grant No. 1108085J16, and the Open Research Fund of State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing of China under Grant No. 10R04.

real fitness evaluations at all, and thus the computational burden can be greatly reduced. However, in practice, it is non-trivial (if not impossible) to build such an ideal regression model. Therefore, a lot of investigations have been devoted to finding better methods of building surrogate models. Polynomial Models, Kriging Models, Multi-layer Perceptron (MLP), Radial Basis Function (RBF) Networks, Support Vector Regressors (SVRs), Gaussian Processes and ensembles of models have all been employed in the literature, and a comprehensive description of them can be found in [10]. On the other hand, it is well recognized that surrogate models should be used together with real fitness evaluations to limit the negative effect caused by a possibly inaccurate model[3,11-12].

Though regression models might be the most popular type of surrogate model so far, they are not the only one. For specific types of EAs, such as Covariance Matrix Adaption Evolution Strategies (CMA-ES), it has been stated that the prediction of fitness of candidate solutions is unnecessary to be accurate, as long as the surrogate model can always rank candidate solutions correctly[13]. Hence, Rank-based Support Vector Machines (SVMs) have been employed to build ranking models to accelerate CMA-ES[14]. Furthermore, in the context of constrained optimization, classification models have been employed to accelerate constrained Memetic algorithms (MAs)[15-18]. In these studies, classifiers were built with SVMs to estimate the feasibility of individuals. Based on this, MAs would only choose a part of individuals to refine in every generation and thus solve the constrained problems more efficiently.

In recent years, DE has received much attention and a lot of DE variants, such as self-adaptive NSDE (SaNSDE)[19], DE with global and local neighborhoods (DEGL)[20], JADE[21], generalized adaptive DE (GaDE)[22], composite DE (CoDE)[23] and OX-based DE (OXDE)[24] have been proposed to accelerate DE. However, previous work on DE for computationally expensive problems is few. Furthermore, most existing studies employed regression models such as MLP or RBF networks as the surrogate model[25-26]. According to previous studies, different EAs may favor different types of surrogate models[14-15]. Given a specific EA, other types of surrogate models, such as ranking models, can actually lead to even better performance. This implies that, for different EAs, specially designed surrogate models are necessary. Different from many other EAs, each individual in the parent population of DE corresponds to a unique offspring individual. An offspring individual will enter the next generation only if its fitness is higher than its corresponding parent. Therefore, in a recent work, it was pointed out that it is more appropriate to formulate the selection process of DE as a classification problem, rather than regression or ranking problem, and a Classification-Assisted DE (CADE) was proposed[27]. Different from [15-18], classifiers in CADE were built to *compare* different individuals, i.e., for each pair of parent and offspring individuals, SVMs were utilized to train a classifier to identify the better one between them. Experimental studies have shown the advantage of CADE over DE assisted with regression or ranking models.

In brief, the idea behind CADE is to employ a classifier as a *selector*. In the selection process of DE, offspring individuals that are worse than their corresponding parents are discarded directly. Hence, it is unnecessary to actually evaluate the fitness of this type of individuals. In other words, CADE employs a classifier, which is unable to predict the fitness of a solution, to avoid wasting fitness evaluations on those offspring individuals that are worse than their parents. By this means, some fitness evaluations can be saved in every generation. One disadvantage of CADE is that every surviving offspring individual needs a real fitness evaluation, which can still be very costly. On the other hand, a regression model can be used to predict the fitness of individuals, and the predicted fitness of some individuals will be considered as the real fitness in later evolution. To summarize, both classification and regression models would be helpful to reduce the number of real fitness evaluations, but from different perspective. Based on these considerations, a novel approach named Classification- and Regression-Assisted DE (CRADE) is proposed in this paper.

CRADE employs both regression and classification techniques to assist DE in solving computationally expensive problems. Concretely, classifiers are built to check whether an offspring individual is better than its parent. Then, offspring individuals that are considered to be better than their corresponding parents will be evaluated either by a regression model, or by the real fitness function. Moreover, which individuals will undergo real fitness evaluations is determined based on the consistency between classification and regression models. By this means, we have arrived at a surrogate model construction approach that relies on the *synergy* between classification and regression techniques. Although multiple surrogate models have been employed in [28], all of them are regression models, and the purpose of using multiple models is to build an even more accurate regressor. In contrast, CRADE employs two different types of surrogate models, i.e., regression and classification models, and combines them together in DE. No previous work has combined such two different types of surrogate models. Experimental studies show

that CRADE significantly outperforms CADE and DE assisted with only regression models.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to DE. The proposed CRADE is described in detail in Section 3. Experimental results are presented in Section 4 to evaluate the efficacy of CRADE. Finally, Section 5 concludes this paper.

2 Differential Evolution

DE is a population-based stochastic search method for global optimization, especially for dealing with continuous optimization problem[1]. We assume in this paper the objective function to be minimized is $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is a vector of n design variables in a decision space D.

An overview of DE is given in Algorithm 1. The output of the pseudocode is the individual with the smallest objective function value in the final population. Like any other EA, DE starts with a population of *popsize* n-dimensional candidate solutions, where each solution is generated according to a uniform distribution function within the decision space D. We denote the current generation in the evolutionary process of DE by G and its value ranges from 0 to *Gmax*. The notation $P_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, \ldots, popsize\}$ represents the population at the current generation, where $\boldsymbol{x}_{i,G}$ denotes the i-th individual of the population. After initialization, three reproductive strategies (mutation, crossover, and selection strategy) are executed in DE to generate P_{G+1} for next generation from the current population P_G. This process is repeated until a stopping criterion is met. The mutation, crossover, and selection strategy of DE will be briefly described in turn.

Algorithm 1. *DE*$(f, Gmax)$
1: Set $G = 0, popsize$
2: Initialize a population $P_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, \ldots, popsize\}$
3: **while** $G < Gmax$ **do**
4: $P_{G+1} = \varnothing$
5: **for** each $\boldsymbol{x}_{i,G}$ in P_G **do**
6: $\boldsymbol{v}_{i,G} = DE\text{-}Mutate(\boldsymbol{P}_G)$
7: $\boldsymbol{u}_{i,G} = DE\text{-}Crossover(\boldsymbol{v}_{i,G}, \boldsymbol{x}_{i,G})$
8: $\boldsymbol{x}_{i,G+1} = DE\text{-}Select(\boldsymbol{x}_{i,G}, \boldsymbol{u}_{i,G}, f)$
9: $P_{G+1} = P_{G+1} \bigcup \{\boldsymbol{x}_{i,G+1}\}$
10: **end for**
11: $G = G + 1$
12: **end while**

The mutation strategy of DE is carried out by adding a weighted difference between two different individuals to another individual. At each generation, for each individual $\boldsymbol{x}_{i,G}$ of the P_G, three mutually distinct individuals $\boldsymbol{x}_{i_1,G}$, $\boldsymbol{x}_{i_2,G}$ and $\boldsymbol{x}_{i_3,G}$ are randomly selected from the population. Then the mutation strategy is executed as the following formula shows:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i_1,G} + F \cdot (\boldsymbol{x}_{i_2,G} - \boldsymbol{x}_{i_3,G}), \tag{1}$$

where the scale factor F is greater than 0 and 0.5 is recommended as the default value. Actually, the formula only represents the typical mutation strategy "DE/rand/1". Other mutation strategies of DE can be found in [1].

After mutation, for each pair of $\boldsymbol{v}_{i,G}$ and $\boldsymbol{x}_{i,G}$, a binary crossover operation is usually performed using (2):

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } rand_j(0,1) \leqslant CR \text{ or } j = j_{rand}, \\ x_{j,i,G}, & \text{otherwise,} \end{cases} \tag{2}$$

where $u_{j,i,G}$, $v_{j,i,G}$, and $x_{j,i,G}$ represents the j-th variable of $\boldsymbol{u}_{i,G}$, $\boldsymbol{v}_{i,G}$, and $\boldsymbol{x}_{i,G}$, respectively. Meanwhile, $rand_j(0,1)$ represents a number drawn uniformly between 0 and 1 and j_{rand} is a randomly selected integer $\in [1, n]$. The crossover rate $CR \in (0, 1)$ and 0.9 is recommended as the default value. By now, one offspring individual $\boldsymbol{u}_{i,G}$ has been generated for each parent $\boldsymbol{x}_{i,G}$. The offspring population is denoted as $U_G = \{\boldsymbol{u}_{i,G} | i = 1, 2, \ldots, popsize\}$ hereafter.

Last, the selection strategy is implemented in DE by pair-wise comparison:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G}, & \text{if } f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G}), \\ \boldsymbol{x}_{i,G}, & \text{otherwise.} \end{cases} \tag{3}$$

For each pair of $\boldsymbol{x}_{i,G}$ and $\boldsymbol{u}_{i,G}$, the one with lower function value between them is selected to enter next generation. Thus, $P_{G+1} = \{\boldsymbol{x}_{i,G+1} | i = 1, 2, \ldots, popsize\}$ is generated for next generation.

3 CRADE: Classification- and Regression-Assisted DE

Since DE employs pair-wise selection, the replacement of an individual is allowed only if its fitness is lower than that of the corresponding offspring individual. In the rest of this paper, we refer to the offspring individuals that are better than their parents as improved solutions. The number of real fitness evaluations consumed by DE can be reduced from two aspects. First, it would be good not to invest real fitness evaluations into those offspring individuals that are not improved solutions, since they will be discarded directly and will not influence the search later on. Second, the fitness of some improved solutions could be approximated by a surrogate model. For the first case, regression, ranking and classification can all be employed, and classification models have been shown to perform the

best[27]. On the other hand, only regression models are appropriate for the second case. Hence, an intuitive way is to combine both classification and regression models to construct a more effective surrogate model. Considering this, a novel method, called Classification- and Regression-Assisted DE (CRADE) is proposed.

In general, CRADE divides the offspring individuals in every generation into three groups: 1) those not entering the next generation; 2) those entering the next generation and being evaluated with the real fitness function; 3) those entering the next generation with fitness approximated by a surrogate model. In CRADE, the classification model is employed to identify those individuals belonging to the first group. The latter two groups are distinguished by a regression model, i.e., if the fitness of an individual can be well approximated by the regression model, it should fall into the third group. Unfortunately, regression models themselves cannot tell which approximate fitness values they give are accurate. To cope with this problem, CRADE adopts a rule based on the consistency between classification and regression models. That is, for each individual that is classified to be superior to its parent, if the approximate fitness of this individual is better than the fitness of its parent, the fitness of this individual can be considered to be well approximated by the regression model and the individual will enter the next generation with the approximate fitness; otherwise, the individual will be evaluated with the real fitness function and pair-wise comparison will be made between it and its parent.

An overview of CRADE is given in Algorithm 2. The input parameter *MaxEval* equals the maximal number of function evaluations and f denotes the objection function of the optimization problem. Without loss of generality, the term optimization indicates minimization in this section.

Algorithm 2. *CRADE*$(f, MaxEval)$

1: Set $popsize, MaxGb, G = eval = 0, DB = \varnothing$
2: Initialize a population $P_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, \ldots, popsize\}$
3: Archive all $(\boldsymbol{x}_{i,G}, f(\boldsymbol{x}_{i,G}))$ into DB
4: $eval = eval + popsize$
5: **while** $eval < MaxEval$ **do**
6: **if** $G < MaxGb$ **then**
7: $P_{G+1} = DE\text{-}Mutate\text{-}Crossover\text{-}Select(P_G, \mathtt{f})$
8: $G = G + 1$
9: Archive all $(\boldsymbol{x}_{i,G}, f(\boldsymbol{x}_{i,G}))$ into DB
10: $eval = eval + popsize$
11: **else**
12: $P_{G+1} = \varnothing$
13: **for** each $\boldsymbol{x}_{i,G}$ in P_G **do**
14: $\boldsymbol{v}_{i,G} = DE\text{-}Mutate(P_G)$
15: $\boldsymbol{u}_{i,G} = DE\text{-}Crossover(\boldsymbol{v}_{i,G}, \boldsymbol{x}_{i,G})$
16: $NB = Neighborhood(\boldsymbol{u}_{i,G}, DB, k)$
17: Build a regression model R by using NB as the training set
18: Classify individuals in NB into two classes (1: outperform $\boldsymbol{x}_{i,G}$, -1: otherwise)
19: Build a classifier C by using classified NB as the training set
20: $\boldsymbol{x}_{i,G+1} = Evaluate\text{-}Replace(\boldsymbol{x}_{i,G}, \boldsymbol{u}_{i,G}, R, C, f)$
21: $\boldsymbol{P}_{G+1} = \boldsymbol{P}_{G+1} \bigcup \{\boldsymbol{x}_{i,G+1}\}$
22: Archive new real evaluation into DB
23: **end for**
24: **end if**
25: $G = G + 1$
26: **end while**

At the beginning, the initial population $P_0 = \{\boldsymbol{x}_{i,0} | i = 1, 2, \ldots, popsize\}$ is evolved using the mutation, crossover, and selection strategy of DE for $MaxGb$ generations. This process aims to accumulate sufficient evaluated points for training surrogate models and all real evaluations are archived into a database DB.

After this, for every newly generated offspring individual $\boldsymbol{u}_{i,G}$, a neighborhood NB including k nearest points to it in DB is identified. Based on the neighborhood, a regression model and a classifier are built. When building a classifier, points in NB are first categorized into two classes in order to form a training set with class labels. If the fitness of a point is higher than that of $\boldsymbol{x}_{i,G}$, it is labeled as a point in class "+1"; otherwise, it is categorized into the "−1" class.

With the two surrogate models, CRADE differs from existing methods in the way of using the models, i.e., *Evaluate-Replace* procedure in line 20 of Algorithm 2. Given an offspring individual $\boldsymbol{u}_{i,G}$ and its corresponding parent $\boldsymbol{x}_{i,G}$, we denote $C(\boldsymbol{u}_{i,G})$ and $R(\boldsymbol{u}_{i,G})$ as the class label predicted by the classifier and the predicted objective function value on $\boldsymbol{u}_{i,G}$. Further, let $f(\boldsymbol{x}_{i,G})$ be the objective function value of $\boldsymbol{x}_{i,G}$, details of the *Evaluate-Replace* procedure are described below.

• If $C(\boldsymbol{u}_{i,G}) == 1$ and $R(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$, $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$, and $f(\boldsymbol{u}_{i,G})$ is set to $R(\boldsymbol{u}_{i,G})$. Here, $\boldsymbol{x}_{i,G}$ is replaced by $\boldsymbol{u}_{i,G}$ because both C and R predict that $\boldsymbol{u}_{i,G}$ is a better solution than $\boldsymbol{x}_{i,G}$. Moreover, since the prediction made by the regression model is consistent with that made by the classifier, it is more likely that regression model is making an accurate prediction about the objective function value of $\boldsymbol{u}_{i,G}$. Hence, the predicted $R(\boldsymbol{u}_{i,G})$ is used as $f(\boldsymbol{u}_{i,G})$ in the later evolution without evaluating $\boldsymbol{u}_{i,G}$ with the real fitness function.

• If $C(\boldsymbol{u}_{i,G}) == 1$ and $R(\boldsymbol{u}_{i,G}) \geqslant f(\boldsymbol{x}_{i,G})$, $\boldsymbol{u}_{i,G}$ is evaluated with the real objective function. Then, if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$, $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$, otherwise, $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$. Here, the classifier model predicts that $\boldsymbol{u}_{i,G}$ is a better solution than $\boldsymbol{x}_{i,G}$. However, the prediction made by the regression model is inconsistent with that made by the classifier, $R(\boldsymbol{u}_{i,G})$ cannot be directly used as $f(\boldsymbol{u}_{i,G})$ in later evolution, and thus a real function

evaluation needs to be carried out.

• If $C(\boldsymbol{u}_{i,G}) == -1$, $\boldsymbol{u}_{i,G}$ will be discarded directly without undergoing a real function evaluation and $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$.

After the *Evaluate-Replace* procedure, if $\boldsymbol{u}_{i,G}$ has been evaluated with f in the evaluation phase, $(\boldsymbol{u}_{i,G}, f(\boldsymbol{u}_{i,G}))$ is archived into *DB*. The above process iterates until all function evaluations are used up.

4 Experimental Studies

To assess the efficacy of CRADE, performance comparisons have been made between it and relevant algorithms. In this comparative study, 10 low-dimensional test functions, denoted as $f_1 \sim f_{10}$, and 6 high-dimensional test functions, denoted as $f_{11} \sim f_{16}$, were chosen from [29] and [30], respectively. The number of decision variables, n, is set to 30 for $f_1 \sim f_{10}$ and 500 for $f_{11} \sim f_{16}$ in our experiments. A short description of them is given in Table 1 and more detailed description can be found in [29] and [30]. For each algorithm and each test function, the best function error values achieved over 25 runs were collected. The function error value of a solution equals to the function value of the solution minus the minimal function value of the objective function. Each algorithm is assigned with 10 000 fitness evaluations. The simulation environment is MATLAB with PRTools[31].

4.1 Compared with Regression-Assisted DE and Classification-Assisted DE

To investigate whether CRADE can enhance the efficiency of DE, CRADE was compared with DE. Moreover, CRADE was compared with other two surrogate model-assisted DEs, i.e., regression-assisted DE (RADE) and CADE[27], where RADE and CADE only employ regression and classification techniques, respectively. All the four algorithms were built up with the classic DE as introduced in Section 2. The control parameters of DE were set as follows: $F = 0.5$, $CR = 0.9$ and $popsize = 100$. To make a fair comparison, for the other three surrogate model-assisted DEs, $MaxGb$ were set to 20, and k is 120 on low-dimensional problems and 200 on high-dimensional problems. Moreover, soft-margin support vector classification[32] was used to construct classifiers for CADE and CRADE, and support vector regression (SVR) with ϵ-insensitive loss function[33] to build regression models for RADE and CRADE.

In Tables 2 and 3, the average and standard deviation of the function error value of each algorithm over

Table 1. Short Description about the Test Functions Used in This Study

	Test Functions	Characteristics
f_1, f_{11}	Shifted Sphere Function	unimodal
f_2	Shifted Schwefel's Problem 1.2	unimodal
f_3	Shifted Rotated High Conditioned Elliptic Function	unimodal
f_4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	unimodal
f_5	Schwefel's Problem 2.6 with Global Optimum on Bounds	unimodal
f_6, f_{13}	Shifted Rosenbrock's Function	multimodal
f_7	Shifted Rotated Griewank's Function without Bounds	multimodal
f_8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	multimodal
f_9, f_{14}	Shifted Rastrigin's Function	multimodal
f_{10}	Shifted Rotated Rastrigin's Function	multimodal
f_{12}	Shifted Schwefel's Problem 2.21	unimodal
f_{15}	Shifted Griewank's Function	multimodal
f_{16}	Shifted Ackley's Function	multimodal

Table 2. Experimental Results of DE, RADE, CADE, and CRADE on 10 Test Functions of 30 Variables

Function	CRADE	DE	RADE	CADE
f_1	1.80e−04±7.33e−05	2.16e+03±4.43e+02 −	6.32e−01±9.19e−02 −	1.07e−01±3.67e−02 −
f_2	1.09e+03±3.75e+02	2.79e+04±4.85e+03 −	1.64e+04±4.87e+03 −	3.54e+03±1.33e+03 −
f_3	7.93e+06±2.68e+06	1.09e+08±1.91e+07 −	1.10e+08±2.75e+07 −	1.80e+07±5.75e+06 −
f_4	3.97e+03±1.96e+02	3.49e+04±7.50e+03 −	2.70e+04±7.06e+03 −	7.71e+03±2.77e+03 −
f_5	2.38e+03±7.90e+02	1.00e+04±1.12e+03 −	2.24e+03±5.69e+02 ≈	2.39e+03±5.71e+02 ≈
f_6	9.76e+02±2.03e+03	6.79e+07±2.59e+07 −	2.32e+07±1.43e+07 −	2.54e+03±3.11e+03 −
f_7	9.33e−01±1.28e−01	6.70e+02±1.07e+02 −	1.64e+00±2.47e−01 −	1.66e+01±4.11e−01 −
f_8	2.11e+01±5.84e−02	2.11e+01±5.70e−02 ≈	2.11e+01±6.39e−02 ≈	2.08e+01±6.61e−02 ≈
f_9	3.18e+01±8.49e+00	2.35e+02±1.48e+01 −	2.01e+02±1.14e+01 −	2.09e+02±1.31e+01 −
f_{10}	4.27e+01±2.17e+01	2.64e+02±1.46e+01 −	2.15e+02±1.29e+01 −	2.15e+02±1.37e+01 −
−		9	8	8
+		0	0	0
≈		1	2	2

Note: +, − and ≈ denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of CRADE according to the result of the Wilcoxon rank sum test, respectively.

Table 3. Experimental Results of DE, RADE, CADE, and CRADE on 6 Test Functions of 500 Variables

Function	CRADE	DE	RADE	CADE
f_{11}	5.44e+05±4.37e+04	1.52e+06±5.35e+04 −	5.06e+05±3.49e+04 ≈	6.98e+05±4.27e+04 −
f_{12}	1.23e+03±4.40e+02	1.59e+02±2.50e+00 −	1.18e+02±4.06e+00 +	1.19e+02±4.09e+00 ≈
f_{13}	2.15e+11±1.83e+10	7.97e+11±5.39e+10 −	6.96e+11±7.98e+10 −	2.16e+11±1.67e+10 ≈
f_{14}	5.99e+03±2.28e+02	8.73e+03±1.65e+00 −	7.14e+03±1.63e+02 −	7.36e+03±9.87e+01 −
f_{15}	4.31e+03±2.44e+02	1.25e+04±4.57e+02 −	4.23e+03±2.63e+02 ≈	5.63e+03±3.12e+02 −
f_{16}	1.99e+01±1.44e−01	2.10e+01±4.68e−02 −	2.10e+01±3.50e−02 −	1.99e+01±1.40e−01 ≈
−		6	3	3
+		0	1	0
≈		0	2	3

Note: +, − and ≈ denotes that the performance of the corresponding algorithm is better than, worse than, and similar to that of CRADE according to the result of the Wilcoxon rank sum test, respectively.

Table 4. Average Improvement Rates of Evaluation for RADE, CADE, and CRADE over 25 Independent Runs on 10 Test Functions of 30 Variables

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
CRADE	0.9382	0.3471	0.2266	0.3764	0.6787	0.5730	0.8656	0.2551	0.6686	0.6659
RADE	0.5530	0.1309	0.1086	0.0821	0.4074	0.2061	0.5214	0.0424	0.1332	0.1635
CADE	0.6270	0.2319	0.1874	0.1622	0.3064	0.4948	0.5097	0.0417	0.1255	0.1504

25 independent runs on each test function are presented. Wilcoxon rank sum test at a 0.05 significance level was performed between CRADE and the compared algorithms. The results are also presented in Tables 2 and 3. On the 10 low-dimensional test functions, CRADE outperforms DE, RADE, and CADE on 9, 8, and 8 test functions, respectively. None of DE, RADE, and CADE achieves better solutions than CRADE. On the 6 high-dimensional test functions, CRADE is superior to DE, RADE, and CADE on 6, 3, and 3 test functions, respectively. Neither DE nor CADE is better than CRADE on any test function, and RADE outperforms CRADE only on 1 test function. In general, CRADE is the clear winner when compared to any of DE, RADE, and CADE according to the Wilcoxon rank sum test.

Since CRADE intends to reduce the number of real fitness evaluations by identifying the improved solutions with surrogates and only evaluating part of the improved solutions in each generation. It is interesting to verify how many improved solutions were actually selected with this strategy. For this purpose, we define the indicator named Improvement Rate. For each run, the improvement rate is calculated using the following formula:

$$IR_i = NS_i/Eval_i, \tag{4}$$

where NS_i denotes the total number of improved solutions that have been selected by the surrogates to enter the next generation, and $Eval_i$ denotes the total number of fitness evaluations in the run, which is 10 000 in our experiments. Furthermore, we also define the average improvement rate ($\mathrm{IR}_{\mathrm{avg}}$) over all the 25 runs, and it is calculated using (5):

$$IR_{\mathrm{avg}} = \sum_{i=1}^{25} IR_i/25. \tag{5}$$

Table 5. Average Improvement Rates of Evaluation for RADE, CADE, and CRADE over 25 Independent Runs on 6 Test Functions of 500 Variables

Algorithm	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
CRADE	0.7093	0.2640	0.4956	0.4382	0.8345	0.3515
RADE	0.6609	0.1820	0.1608	0.2530	0.6695	0.1144
CADE	0.4784	0.1760	0.4503	0.2418	0.4766	0.3394

It should be noted that the improvement rate does not aim to measure whether CRADE generally outperforms CADE or RADE in terms of solution quality. Instead, it aims to analyze whether the surrogates utilized by CRADE have played their roles as we expected, i.e., spending more real fitness evaluations on improved solutions. Tables 4 and 5 present the average improvement rates of RADE, CADE, and CRADE on all the 16 test functions. On the 16 test functions, CRADE achieved higher improvement rates than RADE and CADE. That means, by combining classification and regression models together, more real fitness evaluations were spent on improved solutions. In other words, CRADE wastes much less fitness evaluations on those offspring individuals that should be directly discarded. This provides a partial explanation to the superior performance of CRADE.

To get a more complete picture of the performance of CRADE, the evolutionary curves of DE, RADE, CADE, and CRADE are plotted in Figs. 1∼3. For each algorithm, the evolutionary curve is obtained by averaging over 25 runs.

For the low-dimensional test functions, it is evident that the evolutionary curves of CRADE are almost always beneath those of DE, RADE, and CADE on each function after the phase that aims at accumulating enough evaluated points.

For the high-dimensional test functions, CRADE

almost always performs better than, or comparable to the other three algorithms on $f_{13} \sim f_{16}$. However, the cases on $f_{11} \sim f_{12}$ are a bit different. On f_{12}, CRADE performs well until the later stage of the evolutionary process (i.e., after 8 000 fitness evaluations have been consumed). This denotes that CRADE has an advantage over other algorithms when the available computational resource is even more limited than the setup of our experiments. On f_{11}, it can be seen that CRADE is almost always inferior to RADE. Taking a closer look at the evolutionary process, it is observed that CRADE actually selects fewer improved solutions to enter the next generation than RADE in the early stage of evolution, although its overall improvement rate is higher. This phenomenon indicates that the combination of classification and regression models failed to provide advantages over a single regression model. In fact, since the neighborhood size was set to 200 for high-dimensional problems in our experiments, there is no guarantee that an accurate model can be built in the 500-dimensional space. For this reason, the combination of two inaccurate models may even deteriorate the performance.

4.2 Compared with Three State-of-the-Art DEs

CRADE was also compared with three state-of-the-art DE variants, i.e., DEGL[20], JADE[21] and CoDE[23]. DEGL employed a hybrid DE-type mutation operator that is a linear combination of an explorative and an exploitive mutation operator. In JADE, the control parameters F and CR were self-adapted during the evolutionary process. In CoDE, three well-studied mutation strategies were combined with three control parameter settings in a random way to generate offspring individuals. The parameter settings for the three DE variants were the same as in their original papers. The number of fitness evaluations was also set 10 000. The average and standard deviation of the function error value of each algorithm over 25 independent runs on each test function were recorded. Wilcoxon rank sum test at a 0.05 significance level was also performed between CRADE and each of DEGL, JADE and CoDE. Tables 6 and 7 summarize the experimental results.

Overall, CRADE outperforms DEGL, JADE, and CoDE according to the Wilcoxon rank sum test. In fact,

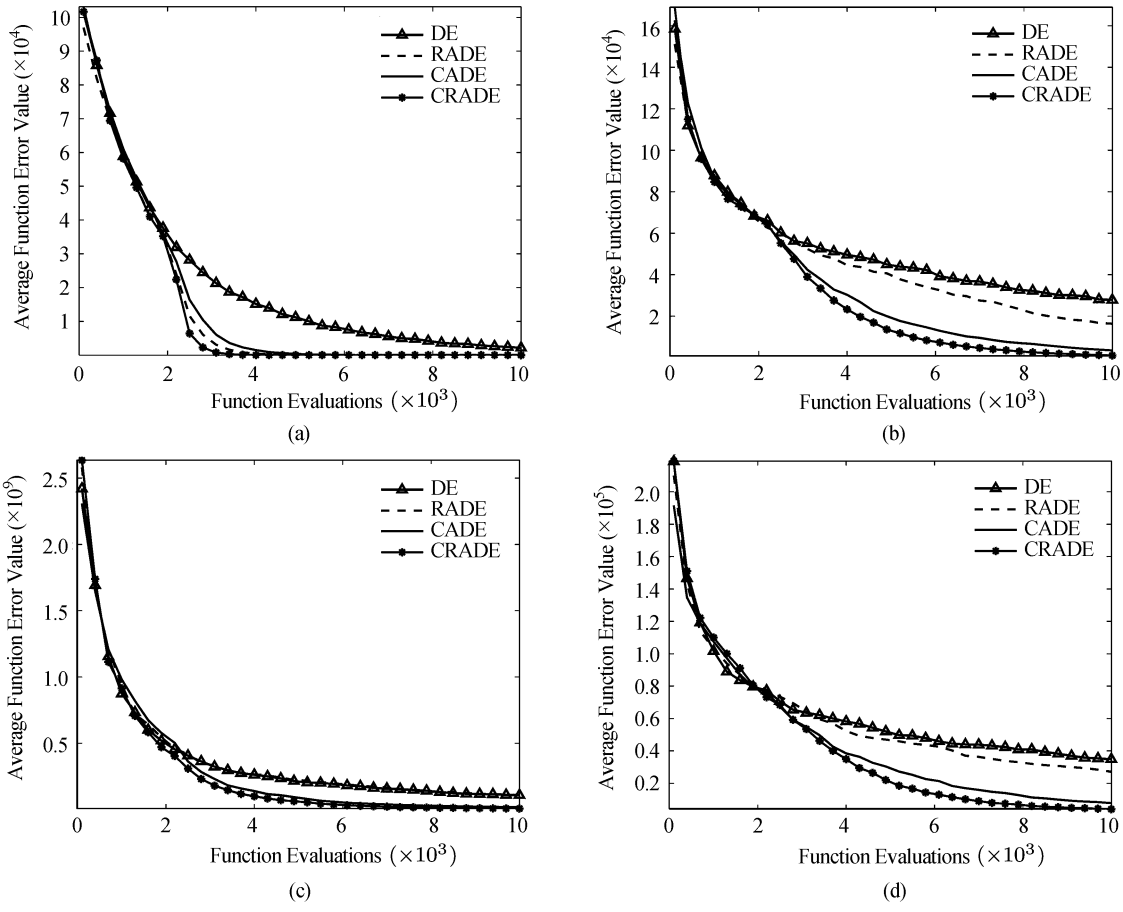

Fig.1. Evolutionary curves for $f_1 \sim f_4$. (a) f_1. (b) f_2. (c) f_3. (d) f_4.

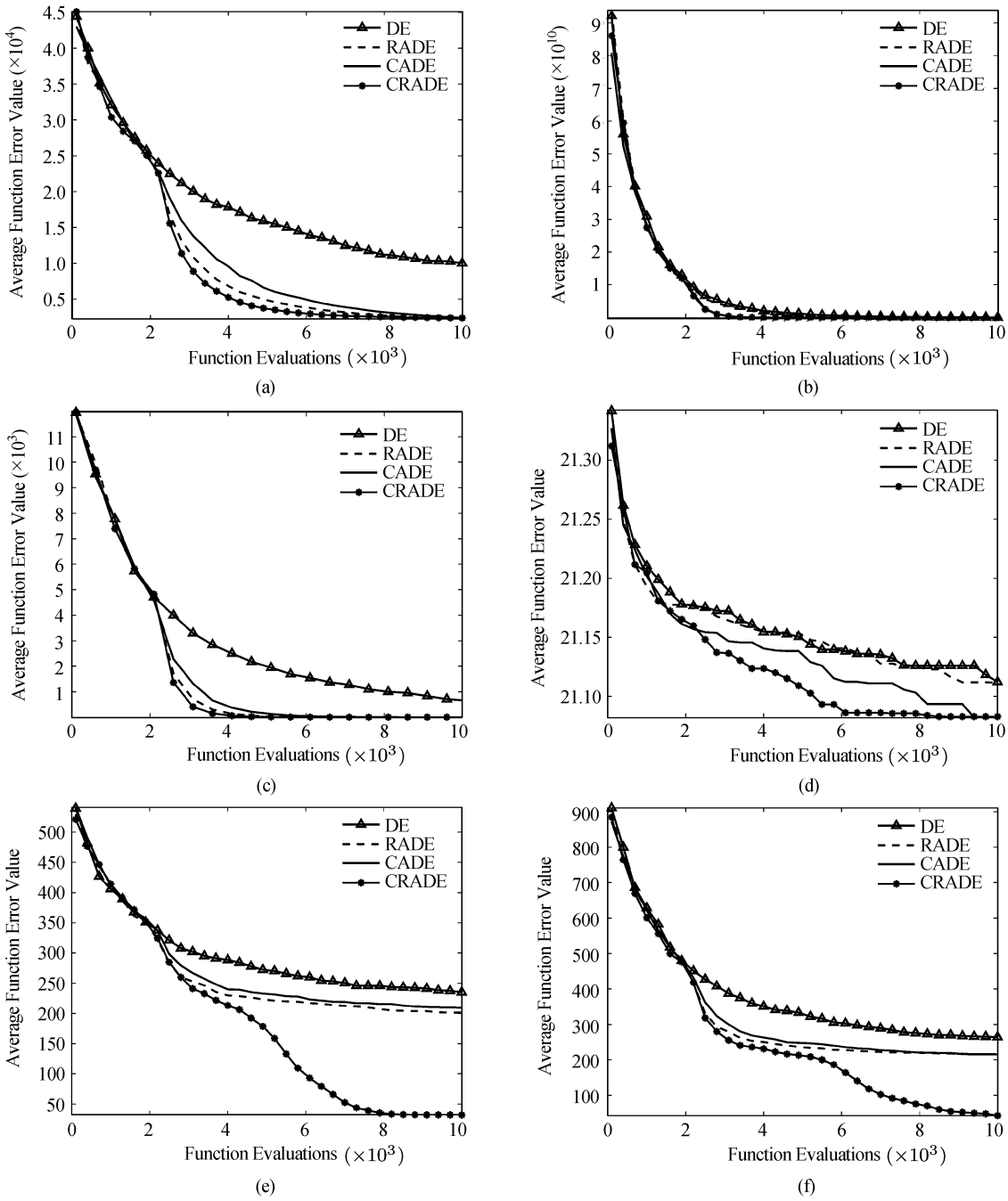

Fig.2. Evolutionary curves for $f_5 \sim f_{10}$. (a) f_5. (b) f_6. (c) f_7. (d) f_8. (e) f_9. (f) f_{10}.

on low-dimensional test functions, CRADE performs better than DEGL, JADE and CoDE on 6, 9, and 9 out of 10 test functions, respectively. DEGL outperforms CRADE only on two test functions, and neither JADE nor CoDE beats CRADE on any test function. On high-dimensional test functions, CRADE achieves better solutions than DEGL, JADE and CoDE on 5, 3, and 5 out of 6 test functions, respectively. DEGL and JADE beat CRADE on 1 and 2 test functions, respectively. CoDE is not superior to CRADE on any test function.

5 Conclusions and Future Work

In this paper, a surrogate model-assisted DE, CRADE, is proposed by incorporating both classification and regression techniques into DE for solving computationally expensive problems. The experimental studies in this paper were carried out on 10 benchmark

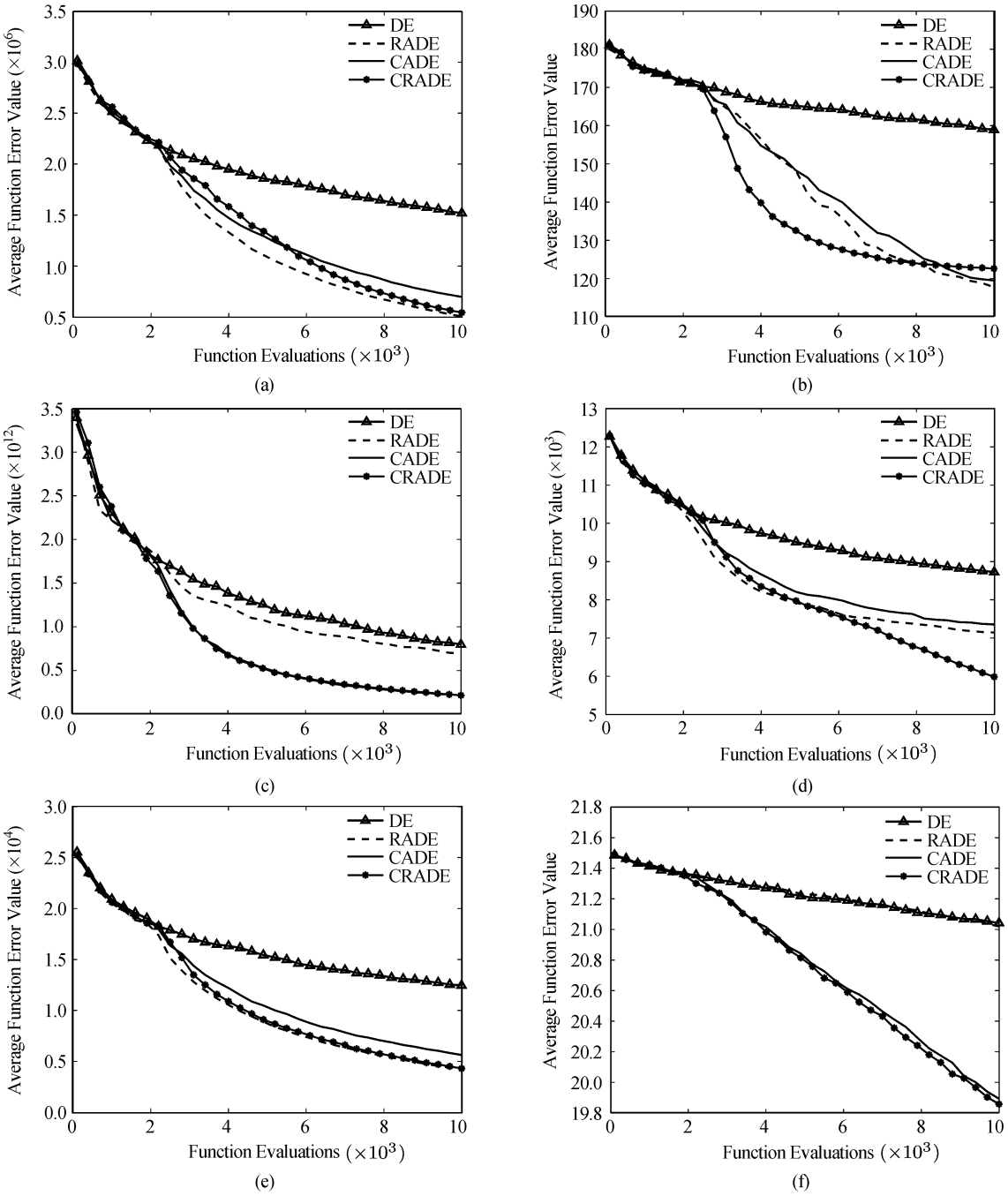

Fig.3. Evolutionary curves for $f_{11} \sim f_{16}$. (a) f_{11}. (b) f_{12}. (c) f_{13}. (d) f_{14}. (e) f_{15}. (f) f_{16}.

functions in the CEC2005 special session on real-parameter optimization and 6 benchmark functions in the CEC2008 special session and competition on large scale global optimization. CRADE was compared with DE, RADE, CADE, and three other state-of-the-art DE variants, i.e., DEGL, JADE, and CoDE. The experimental results suggest that CRADE can enhance the computational efficiency of DE. Assisted by the combination of classification and regression, in general, CRADE achieved better solutions than DE that employs regression or classification technique only. Moreover, its overall performance on the test functions was better than the three state-of-the-art DE variants.

An important issue in surrogate model-assisted EAs is that which individuals should be evaluated with real fitness functions. In previous work, two evolution control rules have been proposed along this direction[3]. However, the ratio of the number of individuals that

Table 6. Experimental Results of DEGL, JADE, CoDE, and CRADE on 10 Test Functions of 30 Variables

Function	CRADE	DEGL	JADE	CoDE
f_1	1.80e−04±7.33e−05	4.31e−02±5.99e−02 −	8.61e+00±2.22e+00 −	1.69e+02±5.26e+01 −
f_2	1.09e+03±3.75e+02	7.03e+02±3.69e+02 +	8.35e+03±2.73e+03 −	8.55e+04±2.30e+03 −
f_3	7.93e+06±2.68e+06	4.77e+06±2.14e+06 +	2.50e+07±5.55e+06 −	2.31e+08±1.08e+07 −
f_4	3.97e+03±1.96e+02	4.84e+03±2.91e+03 ≈	1.49e+04±3.41e+03 −	1.73e+04±4.10e+03 −
f_5	2.38e+03±7.90e+02	3.15e+03±7.10e+02 −	4.73e+03±5.49e+02 −	7.19e+03±7.79e+02 −
f_6	9.76e+02±2.03e+03	4.67e+03±7.82e+03 −	1.29e+04±8.92e+03 −	7.87e+05±3.56e+05 −
f_7	9.33e−01±1.28e−01	1.07e+01±1.34e+01 −	9.02e+00±3.64e+00 −	1.32e+02±5.06e+01 −
f_8	2.11e+01±5.84e−02	2.11e+01±3.85e−02 ≈	2.11e+01±4.61e−02 ≈	2.11e+01±3.41e−02 ≈
f_9	3.18e+01±8.49e+00	2.12e+02±2.15e+01 −	1.55e+02±1.07e+01 −	1.38e+02±1.26e+01 −
f_{10}	4.27e+01±2.17e+01	2.20e+02±1.56e+01 −	2.26e+02±1.24e+01 −	2.49e+02±1.85e+01 −
−		6	9	9
+		2	0	0
≈		2	1	1

Note: +, − and ≈ denotes that the performance of the corresponding algorithm is better than, worse than, and similar to that of CRADE according to the result of the Wilcoxon rank sum test, respectively.

Table 7. Experimental Results of DEGL, JADE, CoDE, and CRADE on 6 Test Functions of 500 Variables

Function	CRADE	DEGL	JADE	CoDE
f_{11}	5.44e+05±4.37e+04	1.00e+06±5.08e+04 −	6.11e+05±3.68e+04 −	1.06e+06±5.82e+04 −
f_{12}	1.23e+03±4.40e+02	1.11e+02±2.71e+00 +	1.20e+02±3.64e+00 ≈	1.20e+02±3.44e+00 ≈
f_{13}	2.15e+11±1.83e+10	3.58e+11±3.10e+10 −	1.75e+11±1.66e+10 +	3.65e+11±2.36e+10 −
f_{14}	5.99e+03±2.28e+02	7.41e+03±6.21e+02 −	7.10e+03±1.11e+02 −	7.60e+03±1.23e+02 −
f_{15}	4.31e+03±2.44e+02	8.12e+03±3.79e+02 −	4.91e+03±2.94e+02 −	8.57e+03±3.83e+02 −
f_{16}	1.99e+01±1.44e−01	2.08e+01±7.00e−02 −	1.96e+01±1.89e−01 +	2.05e+01±8.40e−02 −
−		5	3	5
+		1	2	0
≈		0	1	1

Note: +, − and ≈ denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of CRADE according to the result of the Wilcoxon rank sum test, respectively.

are chosen to evaluate to the number of all individuals is usually hard to determine and disagreement exists in previous work[12]. In this paper, a rule based on the consistency between classification and regression models is made to tell whether the fitness of an individual is well approximated by the regression model. Through this, it can be easily determined which individuals should be evaluated with the real fitness function, thus the determination of the fraction is avoided. According to the experimental results, this rule can also help make efficient use of fitness evaluations.

The experiments were done using 10 000 fitness evaluations. However, even such a number of fitness evaluations may still be computationally prohibitive if one fitness evaluation is too time-consuming in real world. Consequently, in future work, it will be necessary to evaluate CRADE's performance, in comparison to others, using a much smaller number of fitness evaluations. Further, although it is generally difficult to build accurate models in a high-dimensional space, CRADE still shows appealing performance on some high-dimensional problems. Hence, another potential research direction is to further investigate the relationship between the characteristics of problems and surrogate models. Such investigations will provide more insights about under what conditions the combination of several inaccurate models can beat each of the single model in the context of evolutionary optimization, and eventually lead to an even more effective scheme for constructing and combining multiple surrogate models.

References

[1] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997, 11(4): 341-359.

[2] Price K, Storn R M, Lampinen J A. Differential Evolution: A Practical Approach to Global Optimization. New York, USA: Springer-Verlag, 2005.

[3] Jin Y, Olhofer M, Sendhoff B. Managing approximate models in evolutionary aerodynamic design optimization. In *Proc. the 2001 IEEE Congress on Evolutionary Computation* (*CEC2001*), Seoul, Korea, May 2001, pp.592-599.

[4] Ong Y S, Nair P B, Keane A J. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 2003, 41(4): 687-696.

[5] Zhang P, Yao X, Jia L, Sendhoff B, Schnier T. Target shape design optimization by evolving splines. In *Proc. CEC2007*, Singapore, Sept. 2007, pp.2009-2016.

[6] Farina M, Sykulski J. Comparative study of evolution strategies combined with approximation techniques for practical electromagnetic optimization problems. *IEEE Transactions on Magnetics*, 2002, 37(5): 3216-3220.

[7] Hajela P, Lee J. Genetic algorithms in multidisciplinary rotor blade design. In *Proc. the 36th Conference on Structures, Structural Dynamics, and Materials*, New Orleans, USA, April 1995, pp.2187-2197.

[8] Shi L, Rasheed K. A survey of fitness approximation methods applied in evolutionary algorithms. *Computational Intelligence in Expensive Optimization Problems*, 2010, 2(1): 3-28.

[9] Jin Y, Olhofer M, Sendhoff B. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 2002, 6(5): 481-494.
[10] Jin Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 2005, 9(1): 3-12.
[11] Jin Y, Olhofer M, Sendhoff B. On evolutionary optimization with approximate fitness functions. In *Proc. Genetic and Evolutionary Computation Conference*, Las Vegas, Nevada, USA, July 2000, pp.786-793.
[12] Buche D, Schraudolph N N, Koumoutsakos P. Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2005, 35(2): 183-194.
[13] Runarsson T. Ordinal regression in evolutionary computation. In *Proc. the 9th Int. Conf. Parallel Problem Solving from Nature-PPSN IX*, Reykjavik, Iceland, Sept. 2006, pp.1048-1057.
[14] Loshchilov I, Schoenauer M, Sebag M. Comparison-based optimizers need comparison-based surrogates. In *Proc. the 11th Int. Conf. Parallel Problem Solving from Nature-PPSN XI*, Krakov, Poland, September 2010, pp.364-373.
[15] Handoko S D, Kwoh C K, Ong Y S. Using classification for constrained memetic algorithm: A new paradigm. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*. Suntec, Singapore, Oct. 2008, pp.547-552.
[16] Handoko S D, Kwoh C K, Ong Y S. Classification-assisted memetic algorithms for equality-constrained optimization problems. *In Proc. the 22nd AI*, Melbourne, Australia, May 2009, pp.391-400.
[17] Lim D, Ong Y S, Setiawan R, Idris M. Classifier-assisted constrained evolutionary optimization for automated geometry selection of orthodontic retraction spring. In *Proc. the 2010 IEEE Congress on Evolutionary Computation* (*CEC2010*), Barcelona, Spain, July 2010, pp.1-8.
[18] Handoko S D, Kwoh C K, Ong Y S. Feasibility structure modeling: An effective chaperone for constrained memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 2010, 14(5): 740-758.
[19] Yang Z, Tang K, Yao X. Self-adaptive differential evolution with neighborhood search. In *Proc. the 2008 IEEE Congress on Evolutionary Computation* (*CEC2008*), Hong kong, China, June 2008, pp. 1110-1116.
[20] Das S, Abraham A, Chakraborty U K, Konar A. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, 2009, 13(3): 526-553.
[21] Zhang J, Sanderson A C. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 2009, 13(5): 945-958.
[22] Yang Z, Tang K, Yao X. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing*, 2011, 15(11): 2141-2155.
[23] Wang Y, Cai Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 2011, 15(1): 55-66.
[24] Wang Y, Cai Z, Zhang Q. Enhancing the search ability of differential evolution through orthogonal crossover. *Information Sciences*, 2012, 185(1): 153-177.
[25] Zhang J, Sanderson A. DE-AEC: A differential evolution algorithm based on adaptive evolution control. In *Proc. the 2007 IEEE Congress on Evolutionary Computation* (*CEC2007*), Singapore, Sept. 2007, pp.3824-3830.
[26] Wang Y, Shi Y, Yue B, Teng H. An efficient differential evolution algorithm with approximate fitness functions using neural networks. In *Proc. the 2010 Int. Conf. Artificial Intelligence and Computational Intelligence*, Part 2, Oct. 2010, pp.334-341.
[27] Lu X, Tang K, Yao, X. Classification-assisted differential evolution for computationally expensive problems. In *Proc. the 2011 IEEE Congress on Evolutionary Computation* (*CEC2011*), New Orleans, USA, June 2011, pp.1986-1993.
[28] Lim D, Jin Y, Ong Y S, Sendhoff B. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 2010, 14(3): 329–355.
[29] Suganthan P N, Hansen N, Liang J J, Deb K, Chen Y P, Auger A, Tiwari S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore, and KanGAL Report, Kanpur Genetic Algorithms Laboratory, IITkanpur, 2005.
[30] Tang K, Yao X, Suganthan P N, MacNish C, Chen Y P, Chen C M, Yang Z. Benchmark functions for the CEC2008 special session and competition on large scale global optimization. Technical Report, Nature Inspired Comput. Applicat. Lab., USTC, China, 2007. http://nical.ustc.edu.cn/cec08ss.php
[31] Duin R, Juszczak P, Paclik P, Pekalska E, Ridder D de, Tax D M J, Verzakov S. PRTools 4.1, a matlab toolbox for pattern recognition. Delft University of Technology, 2007.
[32] Yu H, Kim S. SVM tutorial: Classification, regression, and ranking. In *Handbook of Natural Computing*, Rozenderg G, Bäck T, Kok J (eds.), Springer 2009.
[33] Gunn S. Support vector machines for classification and regression. Technical Report, University of Southampton, 1998.

Xiao-Fen Lu received the B.Eng. degree in computer science from the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, in 2009. Currently, she is pursuing the Ph.D. degree in NICAL, USTC. Her research interests include evolutionary computation, surrogate-based optimization.

Ke Tang received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002 and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007. In 2007, he joined the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, where he was promoted to Professor in 2011. He is the author/co-author of more than 60 refereed publications. His major research interests include evolutionary computation, machine learning, and their real-world applications. Dr. Tang is an associate editor of IEEE Computational Intelligence Magazine and the Chair of the IEEE Task Force on Collaborative Learning and Optimization. He served as a program co-chair of 2010 IEEE Congress on Evolutionary Computation, held in Barcelona.