Classification-Assisted Differential Evolution for Computationally Expensive Problems

Xiaofen Lu and Ke Tang

NICAL

School of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui 230027, China.

Email: xiaofen@mail.ustc.edu.cn
ketang@ustc.edu.cn

Xin Yao

CERCIA

School of Computer Science
University of Birmingham
Edgbaston Birmingham B15 2TT, U.K.
Email: x.yao@cs.bham.ac.uk

Abstract—Like most Evolutionary Algorithms (EAs), Differential Evolution (DE) usually requires a large number of fitness evaluations to obtain a sufficiently good solution. This is an obstacle for applying DE to computationally expensive problems. Many previous studies have been carried out to develop surrogate-assisted approaches for EAs to reduce the number of real fitness evaluations. Existing methods typically build surrogates with either regression or ranking methods. However, due to the pairwise selection scheme of DE, it is more appropriate to formulate the construction of surrogate as a classification problem rather than a regression or ranking problem. Hence, we propose a classification-assisted DE in this paper. Experimental studies showed that the classification-assisted DE has great potential when compared to the DE that uses regression or ranking techniques to build surrogates.

Index Terms—Surrogate Models, Differential Evolution, Computationally Expensive Problems, Classification.

I. INTRODUCTION

As a population-based Evolutionary Algorithm (EA), Differential Evolution (DE) has achieved great success on many real-world application problems [1]. However, a lot of fitness evaluations are usually needed for DE to achieve near-optimal solutions, which becomes a challenge in solving computationally expensive problems in the real world. Computationally expensive problems arise in complex engineering design problems such as structural design [2], electromagnetics [3] and design of multidisciplinary systems [4]. In such problems, one fitness evaluation can take many hours of computer time. Considering the limited computational resource, it is necessary to develop new strategies to reduce the number of fitness evaluations when DE is employed to solve such problems.

In the literature, there has been increasing interest about how to reduce the number of fitness evaluations in EAs. One popular method is constructing computationally efficient models (usually called surrogates or meta-models) to evaluate individuals instead of carrying out the real evaluations [5]. In the early years, the aim of building surrogates is to predict the fitness of individuals. A surrogate is built on a training set composed of historical evaluated individuals. Usually, part of individuals over some number of generations are added to the training set with real evaluations to update the surrogate

[6]. Commonly-used surrogates include Polynomial Models, Kriging Models, Multi-layer Perceptrons (MLPs), Radial-Basis Function (RBF) Networks, Support Vector Machines (SVMs), Gaussian Processes and a comprehensive description of them can be found in [2]. By using historical data in the optimization process, lots of studies that combine EAs and fitness approximation emerged and an up-to-date survey was presented in [5]. In this paper, this type of methods is referred to as regression-assisted methods. Runarsson [7] stated that only the rank (or partial rank) of individuals is required for EAs that select the best individuals. Based on this, a generic framework of building surrogate ranking to predict the ranks of the individuals in evolutionary computation was presented in [7]. In this framework, the surrogate is built by ordinal regression and a cross-validation is employed to update the surrogate every generation. Following this, the latest work that employs rank-based SVMs in Covariance Matrix Adaption Evolution Strategies (CMA-ES) was presented in [8]. Different from [7], this work does not use the cross-validation process but generates many more pre-children than required in every generation. The surrogate is used to estimate the ranks of the offspring individuals. Then, individuals that undergo real fitness evaluations or enter the next generation directly are chosen based on the estimated ranks with a two-step process. Since the methods in [7] and [8] only make use of surrogate to rank individuals, they are referred to as ranking-assisted methods hereinafter.

In its selection process, DE adopts pairwise comparison. That is, each offspring individual is only compared to the corresponding parent rather than any other individual. In such case, neither the fitness values nor the ranks of offspring individuals are required by the algorithm. Instead, only a surrogate that can distinguish the better one between a parent and its offspring individual is enough. This is in nature a classification problem rather than a regression or ranking one. Although the regression-assisted methods have been applied in DE [9] and ranking-assisted methods are still applicable, they may not fit the framework of DE perfectly. Furthermore, given the same training data, regression and ranking are more complicated tasks than classification. Introducing unnecessarily compli-

cated intermediate problems into the optimization process may deteriorate the quality of the final solution. Having this in mind, we propose incorporating classification techniques into DE to reduce fitness evaluations. For each parent in current population, a training set is chosen from the historical evaluated individuals and then a classifier is built on the training set. After this, the classifier is used to judge whether its offspring individual is better than itself so as to decide whether to evaluate the offspring individual with the fitness function. That means, not all offspring individuals are evaluated with the fitness function. As a result, part of fitness evaluations are saved every generation compared to DE. In the literature, classification techniques have been successfully integrated into EAs to solve specific computationally expensive problems. But the existing work was studied in different context and for different purpose. In [10], [11], classifiers were used to predict whether candidate vectors are defined or not in a pre-selection stage to reduce the consumption of optimization budget. In [12], [13], [14], classification was employed to assist MAs to estimate the feasibility boundary and choose individuals that would experience refinement. So far, however, classification techniques have been employed to compare different individuals in the population of an EA. Experimental comparisons were made between the performance of classification-assisted DE and DE, regression-assisted DE and ranking-assisted DE respectively. The efficacy of the proposed approach has been justified by the results.

This paper is organized as follows. Section II gives a brief introduction to DE. The procedure of classification-assisted DE is described in Section III. In Section IV, experimental studies are presented to investigate the efficacy of classification-assisted DE. Finally, summary and conclusions are given in Section V.

II. DIFFERENTIAL EVOLUTION

Proposed by Storn and Price in [15], DE is a population-based stochastic search algorithm for solving optimization problems in continuous space. A solution of the optimization problem can be represented by a vector of variables. At first, DE initializes a population of solutions with each variable generated according to a uniform distribution between the low and high bound. Then, it uses three reproductive strategies (mutation, crossover and selection strategy) to evolve the population until a stopping criterion is met.

Considering the population $\{\mathbf{x}_{i,g} | i = 1, 2, \dots, \text{popsize}\}$ at a certain generation g , where popsize is the population size and each individual $\mathbf{x}_{i,g}$ is a n -dimensional real-valued vector, mutation vectors are generated according to the typical mutation strategy by adding a weighted difference between two different individuals to another individual. The mutation strategy can be represented by the formula as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i_1,g} + F \cdot (\mathbf{x}_{i_2,g} - \mathbf{x}_{i_3,g}) \quad (1)$$

where $i_1, i_2, i_3 \in [1, \text{popsize}]$ are chosen randomly and meet $i_1 \neq i_2 \neq i_3$ in Eq. (1). $F > 0$ is the scale factor, with 0.9 as

its empirical value. The typical mutation strategy is denoted as 'DE/rand/1'. Other mutation strategies can be found in [15].

After mutation, a binary crossover operation is applied to the mutation vectors according to:

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \cup_j(0,1) \leq CR_i \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \quad (2)$$

where $\cup_j(0,1)$ represents a standard uniform distribution, j_{rand} is a randomly chosen integer $\in [1, n]$. $CR_i \in (0, 1)$ is the crossover rate, with 0.9 as its empirical value. By now, one offspring individual has been generated for each individual in the population.

Last, DE implements the selection strategy by pairwise comparison:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) > f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases} \quad (3)$$

For each parent and its offspring individual, the one with higher fitness value between them enters the next generation. Here f is the fitness function and g denotes the current generation. It is noted that a solution with the maximal fitness value is the best solution.

III. THE PROPOSED ALGORITHM

In this section, a new algorithm, namely classification-assisted DE, is presented. Classification-assisted DE is developed from DE for solving computationally expensive problems. Unlike DE, classification-assisted DE incorporates classification techniques into DE for the sake of reducing fitness evaluations. An overview of classification-assisted DE is given in Algorithm 1. The input parameter f denotes the fitness function of the optimization problem and MaxEval is the maximal number of fitness evaluations. Totally, classification-assisted DE comprises of two phases. One is database building phase (steps 5-10 in Algorithm 1), which begins after a population $\text{pop}_0 = \{\mathbf{x}_{i,0} | i = 1, 2, \dots, \text{popsize}\}$ is initialized. The other is classification employing phase (steps 11-31 in Algorithm 1). Database building phase aims to provide classification employing phase with sufficient evaluated individuals to build classification models. In the database building phase, it generally follows the same procedure of DE. At every generation, DE mutation, crossover and selection strategy are applied to the current population $\text{pop}_g = \{\mathbf{x}_{i,g} | i = 1, 2, \dots, \text{popsize}\}$ to generate a new population $\text{pop}_{g+1} = \{\mathbf{x}_{i,g+1} | i = 1, 2, \dots, \text{popsize}\}$ for the next generation. In addition, all the newly evaluated individuals are archived into a database. In Algorithm 1, MaxGb denotes the count of generations that the database building phase lasts for and DB represents the database that is updated every generation.

At the beginning of every generation in classification employing phase, DE mutation and crossover strategy are applied to the current population $\text{pop}_g = \{\mathbf{x}_{i,g} | i = 1, 2, \dots, \text{popsize}\}$ to generate an offspring population $\text{upop}_g = \{\mathbf{u}_{i,g} | i = 1, 2, \dots, \text{popsize}\}$. After this, classification

Algorithm 1 *Classification-Assisted DE*($\mathcal{F}$, $MaxEval$)

```
1: Set  $g = 0, eval = 0, DB = null, popsize, MaxGb$ 
2: Initialize a population  $\mathbf{pop}_g = \{\mathbf{x}_{i,g} | i = 1, 2, \dots, popsize\}$ 
3: Archive all  $(\mathbf{x}_{i,g}, f(\mathbf{x}_{i,g}))$  into  $DB$ 
4:  $eval = eval + popsize$ 
5: while  $g < MaxGb$  and  $eval < MaxEval$  do
6:    $\mathbf{pop}_{g+1} = DE\text{-}Mutate\text{-}Crossover\text{-}Select(\mathbf{pop}_g)$ 
7:    $g = g + 1$ 
8:   Archive all  $(\mathbf{x}_{i,g}, f(\mathbf{x}_{i,g}))$  into  $DB$ 
9:    $eval = eval + popsize$ 
10: end while
11: while  $eval < MaxEval$  do
12:    $\mathbf{upop}_g = DE\text{-}Mutate\text{-}Crossover(\mathbf{pop}_g)$ 
13:   for each  $\mathbf{u}_{i,g}$  in  $\mathbf{upop}_g$  do
14:      $NB_i = Neighborhood(\mathbf{u}_{i,g}, DB, k)$ 
15:     if  $Mixed(NB_i)$  then
16:        $y_i = SVC(\mathbf{u}_{i,g}, NB)$ 
17:     else
18:        $y_i = BeEvaluated(\mathbf{u}_{i,g}, NB)$ 
19:     end if
20:     if  $y_i == 1$  then
21:       Archive  $(\mathbf{u}_{i,g}, f(\mathbf{u}_{i,g}))$  into  $DB$ 
22:       if  $f(\mathbf{u}_{i,g}) > f(\mathbf{x}_{i,g})$  then
23:          $\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g}$ 
24:       else
25:          $\mathbf{x}_{i,g+1} = \mathbf{x}_{i,g}$ 
26:       end if
27:     else
28:        $\mathbf{x}_{i,g+1} = \mathbf{x}_{i,g}$ 
29:     end if
30:   end for
31: end while
```

techniques are introduced to assist in determining which offspring individuals to evaluate with the fitness function. This process mainly involves three steps. They are training set choosing, classifier training and exact evaluation choosing. In this process, classification techniques are used to train a classifier for each $\mathbf{u}_{i,g}$ to offer estimated comparison between $\mathbf{u}_{i,g}$ and its parent $\mathbf{x}_{i,g}$.

In the training set choosing step, a training set is selected from the database for each $\mathbf{u}_{i,g}$ to build a local model. According to [8], the training set should not lie too far away from $\mathbf{u}_{i,g}$. Therefore, a neighborhood NB_i including k nearest neighbors in DB is identified for each $\mathbf{u}_{i,g}$ (step 14 in Algorithm 1).

For classifier training step, the first important thing is to decide which classification model to use. Inspired from statistical learning theory [16], SVMs are powerful machine learning algorithms with properties of high generalization and kernel trick for nonlinear mapping [17]. SVMs have been extensively used for learning classification functions, which can be called support vector classification (SVC). The SVC usually produces a decision function and employs a threshold between the two classes to handle the classification problem. In

this study, soft-margin SVC [17] is chosen as the classification algorithm. To train a classification model, the training set should be processed to offer two-class information. As the classifier is built to estimate whether $\mathbf{u}_{i,g}$ is better than $\mathbf{x}_{i,g}$, individuals in NB_i should be classified according to their fitness values and the fitness value of $\mathbf{x}_{i,g}$ at first. As a consequence, an individual in NB_i can be a good, bad or equal individual. As only a good offspring individual can replace the parent and enter next generation in the pairwise selection, the equal neighbor is treated as the bad one. Based on this, three types of NB_i can be encountered in building classifiers.

1) *Mixed Neighborhood*: A mixed neighborhood comprises of both good and bad individuals.

2) *Good Neighborhood*: A good neighborhood is composed entirely of good individuals.

3) *Bad Neighborhood*: A bad neighborhood is composed entirely of bad individuals.

Among the three types of neighborhood, only a mixed neighborhood can offer two-class information. Hence, the classification technique (SVC in this study) is only employed to the offspring individual with a mixed neighborhood to train a classifier (step 15-16 in Algorithm 1). However, the number of good individuals and the number of bad individuals might differ from each other in a mixed neighborhood, which means the training set might be imbalanced. Imbalanced training set usually make the trained classifier bias against the minority class. This will have a negative effect on the performance of classification-assisted DE eventually. As threshold-moving [18] is a simple yet effective method in addressing class imbalance problems, it is employed in the classifier training step. Different from [18], the method sets the output threshold so that the G-Mean metric [19] is maximized.

Exact evaluation choosing step determines which offspring individuals to evaluate with the fitness function. As the fitness value of the parent is required in classifying the neighborhood of its offspring individual, all individuals in the current population should have the fitness values. To ensure this, different strategies are applied to different offspring individuals at every generation in the classification employing phase.

For the offspring individual with a mixed neighborhood, whether to evaluate it depends on the classification result given by the trained classifier. If it is classified into the good class, the offspring individual is chosen to evaluate with the fitness function. As a good neighborhood intuitively signifies a better search space and the offspring individual with a good neighborhood is most likely better than its parent, such a offspring individual is selected to evaluate. For the offspring individual with a bad neighborhood, direct intuition advises that the offspring individual is in a worse search space except the offspring individual with the best solution at present as its parent. To make efficient use of computational budget, only this offspring individual is chosen to evaluate with the fitness function. After this, all offspring individuals with exact evaluations are compared with their separate parent and the better one enters the next generation. Moreover, every newly evaluated solution will be archived into the database

TABLE I
CEC2005 TEST FUNCTIONS USED IN THIS STUDY, INCLUDING A SHORT DESCRIPTION, MORE DETAILED DESCRIPTION OF THEM CAN BE FOUND IN [20].

	Test Functions	Characteristics
f_1	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	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	Shifted Rastrigin's Function	multimodal
f_{10}	Shifted Rotated Rastrigin's Function	multimodal

immediately for the sake of building effective classification models next time. The classification employing phase lasts until all fitness evaluations are used up.

IV. EXPERIMENTAL STUDIES

The efficacy of classification-assisted DE has been evaluated by comparing the performance achieved by it and relevant algorithms on benchmark functions. Specifically, the empirical study aimed to check whether introducing classification into DE is capable of improving the performance of DE with a pre-defined number of fitness evaluations.

A. Experimental Setup

1) *Compared Algorithms*: To validate the efficacy of classification-assisted DE, performance comparisons were made between it and each of DE, regression-assisted DE and ranking-assisted DE. To make the comparative study as fair as possible, the following setup was used in the experiments.

First, all the algorithms are built up on the same framework of DE as introduced in Section II. It should be noted that we do not intend to design a new DE variant but a new surrogate-assisted approach that can be applied to any algorithm that employs pairwise comparison. Hence, the original DE is sufficient for our purpose.

Second, as SVC is chosen as the classification algorithm in classification-assisted DE in Section III, the regression and ranking model building are also based on SVM techniques. They are support vector regression (SVR) with ε -insensitive loss function [21] and rank-based SVMs (Rank-SVMs) that were used in [8]. Correspondingly, the three surrogate-assisted DE algorithms are denoted as SVC-DE, RankSVM-DE, SVR-DE hereinafter for convenience.

Third, both the regression-assisted DE and ranking-assisted DE employ the same framework as classification-assisted DE. For regression-assisted DE, a SVM model is trained by SVR with the selected neighborhood for each offspring individual to approximate the real fitness function. If the predicted fitness value of the offspring individual is better than the real fitness value of its parent, it is selected to evaluate with the real fitness function. For ranking-assisted DE, Rank-SVMs are used to learn a ranking model for each offspring individual with a ranked neighborhood. Then the trained model is employed to rank the offspring individual and its parent. Only when the

offspring individual has the higher rank does it undergo exact evaluation.

2) *Parameter Setting*: In this paper, 10 test functions denoted as $f_1 - f_{10}$ were chosen from [20] and detailed in Table I with the configuration of dimension $n = 30$. Every test function is considered as a minimization problem. The control parameters of DE were set as follows: $popsiz$ = 100, $F = 0.5$ and $CR = 0.9$. Other parameters shown in Algorithm 1 were set as: $MaxEval = 10000$ for $f_1 - f_6$ and $f_8 - f_{10}$, $MaxEval = 20000$ for f_7 , $MaxGb = 20$ and $k = 40$. Since the main purpose of the empirical study was to compare the benefits that DE can obtain from the regression, ranking, and classification models, all these parameters were kept the same for each algorithm. The simulation environment is MATLAB with PRTTools [22] used. According to [20], the function error value of a solution equals to the function value of the solution minus the minimal function value. In this paper, the function error value is used as the quality metric of solutions. For every algorithm, the best function error values achieved over 25 runs for the 10 test functions were collected.

B. Experimental Results and Analysis

In Table II, the detailed statistical results of 25 runs for SVC-DE, DE, SVR-DE and RankSVM-DE are presented. Wilcoxon rank-sum tests with a significance level 0.05 were conducted to compare SVC-DE with the other three algorithms and the results are presented in Table III. According to Table III, SVC-DE achieved significantly better solutions than DE on $f_1 - f_7$, f_9 and f_{10} . When compared to SVR-DE, SVC-DE performed much better on $f_1 - f_4$ and $f_6 - f_7$ and comparably on f_5 , f_8 and f_{10} while only a little worse on f_9 . When compared to RankSVM-DE, SVC-DE was the clear winner again. It outperformed RankSVM-DE on $f_2 - f_7$ and $f_9 - f_{10}$ while RankSVM-DE achieved superior performance only on f_1 .

In general, SVC-DE can find better solution than DE on all but one test functions, SVR-DE on most test functions and RankSVM-DE on most test functions. Thus, SVC-DE can reduce fitness evaluations to find the same level of solution found by DE on all but one test functions, found by SVR-DE or RankSVM-DE on most test functions. Through this, SVC-DE is more reliable than SVR-DE and RankSVM-DE. This has, therefore, proved that classification-assisted DE has

TABLE II
STATISTICS OF THE FINAL SOLUTION QUALITY AT THE END OF 10000 EXACT FUNCTION EVALUATIONS OF 25 RUNS ON F1-F10 USING SVC-DE, DE, SVR-DE, RANKSVM-DE.

Algorithm	Statistical Value	Test Function									
		1	2	3	4	5	6	7	8	9	10
DE	Mean	2.16e+03	2.79e+04	1.09e+08	3.49e+04	1.00e+04	6.79e+07	7.86e+01	2.11e+01	2.35e+02	2.64e+02
	Std.Dev.	4.43e+02	4.85e+03	1.91e+07	7.50e+03	1.12e+03	2.59e+07	1.94e+01	5.70e-02	1.48e+01	1.46e+01
	Median	2.11e+03	2.71e+04	1.04e+08	3.53e+04	1.02e+04	6.64e+07	7.89e+01	2.11e+01	2.37e+02	2.64e+02
	Best	1.49e+03	1.81e+04	7.75e+07	1.96e+04	7.76e+03	2.06e+07	4.23e+01	2.10e+01	2.05e+02	2.36e+02
	Worst	3.16e+03	3.77e+04	1.55e+08	4.84e+04	1.20e+04	1.40e+08	1.19e+02	2.12e+01	2.59e+02	2.64e+02
SVR-DE	Mean	6.32e-01	1.64e+04	1.10e+08	2.70e+04	2.24e+03	2.32e+07	1.06e+00	2.11e+01	2.01e+02	2.15e+02
	Std.Dev.	9.19e-02	4.87e+03	2.75e+07	7.06e+03	5.69e+02	1.43e+07	2.35e-02	6.39e-02	1.14e+01	1.29e+01
	Median	6.32e-01	1.59e+04	1.09e+08	2.66e+04	2.16e+03	2.24e+07	1.06e+00	2.11e+01	2.04e+02	2.19e+02
	Best	4.56e-01	6.72e+03	5.82e+07	1.20e+04	7.30e+02	5.11e+06	1.02e+00	2.09e+01	1.79e+02	1.80e+02
	Worst	8.58e-01	2.45e+04	1.68e+08	3.83e+04	3.28e+03	7.16e+07	1.12e+00	2.12e+01	2.17e+02	2.34e+02
RankSVM-DE	Mean	5.06e-02	2.26e+04	9.49e+07	3.03e+04	5.96e+03	1.49e+05	5.42e-01	2.11e+01	2.18e+02	2.36e+02
	Std.Dev.	1.38e-02	4.83e+03	1.95e+07	5.38e+03	6.83e+02	1.33e+05	2.08e-01	5.47e-02	2.08e+01	1.33e+01
	Median	4.79e-02	2.35e+04	9.77e+07	2.97e+04	5.87e+03	1.06e+05	5.32e-01	2.11e+01	2.22e+02	2.36e+02
	Best	2.77e-02	1.43e+04	5.56e+07	2.04e+04	4.53e+03	4.08e+03	1.62e-01	2.10e+01	1.46e+02	2.06e+02
	Worst	7.59e-02	3.07e+04	1.28e+08	3.97e+04	6.98e+03	5.12e+05	9.04e-01	2.12e+01	2.54e+02	2.63e+02
SVC-DE	Mean	1.07e-01	3.54e+03	1.80e+07	7.71e+03	2.39e+03	2.54e+03	4.03e-02	2.08e+01	2.09e+02	2.15e+02
	Std.Dev.	3.67e-02	1.33e+03	5.75e+06	2.77e+03	5.71e+02	3.11e+03	3.15e-02	6.61e-02	1.31e+01	1.37e+01
	Median	9.86e-02	3.45e+03	1.75e+07	7.59e+03	2.22e+03	1.39e+03	2.56e-02	2.11e+01	2.10e+02	2.17e+02
	Best	6.37e-02	1.72e+03	7.38e+06	3.67e+03	1.49e+03	1.08e+02	1.17e-01	2.09e+01	1.84e+02	1.93e+02
	Worst	2.24e-01	7.12e+03	3.42e+07	1.27e+04	3.27e+03	1.04e+04	4.40e-03	2.12e+01	2.27e+02	2.38e+02

TABLE III
RESULT OF WILCOXON RANK-SUM TEST WITH SIGNIFICANCE LEVEL 0.05 COMPARING STATISTICAL VALUES FOR SVC-DE AND THOSE OF DE, SVR-DE, RANKSVM-DE, (w , d AND l STAND FOR "WIN", "DRAW" AND "LOSE", RESPECTIVELY).

	DE	SVR-DE	RankSVM-DE
f_1	w	w	l
f_2	w	w	w
f_3	w	w	w
f_4	w	w	w
f_5	w	d	w
f_6	w	w	w
f_7	w	w	w
f_8	d	d	d
f_9	w	l	w
f_{10}	w	d	w

great potential in reducing fitness evaluations for optimizing computationally expensive problems.

The evolution curves of SVC-DE, DE, SVR-DE, RankSVM-DE are plotted in Figs. 1-10. The X-axis represents the number of fitness evaluations and the Y-axis represents the respective best function error value on the natural log scale. For each test function, the evolution curve of an algorithm was the average over 25 runs. As the optimization algorithm aims to find the solutions with the minimal function error value, an optimization algorithm with one evolution curve lying below the evolution curve of another means that the optimization algorithm can find the better solution with the same number of fitness evaluations. It is evident that the evolution curve of SVC-DE is always beneath the curve of DE on each test function after the database building phase. It is also shown that RankSVM-DE performed well on f_1 but SVC-DE was comparable on f_8 and superior on the remaining 8 test functions. Moreover, SVC-DE was better than SVR-DE on $f_1 - f_4$ and $f_6 - f_8$ from the evolution curves. Among these test functions, the evolution curves of SVC-DE and SVR-DE cross each other on f_1 , f_7

and the evolution curves of SVR-DE are in a lower place before the intersection. We examined the prediction accuracy of the SVR and SVC during the evolutionary process, and found that the prediction accuracy of SVR decreased after the interaction of the two curves. Hence, this might be the reason why SVC-DE outperformed SVR-DE eventually. On the other hand, SVR-DE outperformed SVC-DE on f_5 and $f_9 - f_{10}$. The optimization process of SVC-DE was examined to find out why it performed worse on f_1 , f_5 , f_7 and $f_9 - f_{10}$. It was found that the offspring individual with the best solution at present as its parent is almost always worse than the parent. However, SVC-DE evaluated such offspring individuals directly. As a result, a lot of fitness evaluations were wasted. For such offspring individuals, a better treatment mechanism is necessary. Another reason is that DE generated less good offspring individuals every generation and the training set selected for each offspring individual usually included more bad solutions. Although the imbalanced classification problem is taken into account in the design of SVC-DE, only the simple threshold moving strategy with maximization of G-Mean metric is adopted in

this work. All of these aroused a more challenging scenario for the classification model, which may classify many bad offspring individuals to the good class and thus induced a lot of fitness evaluations when the good individuals are much less than the bad ones in the training set. In other words, a larger part of the limited computational resource has been wasted, which led to the worse performance. The use of more advanced techniques for imbalanced data sets may alleviate this problem.

V. CONCLUSION AND DISCUSSIONS

In this paper, a surrogate-assisted DE, classification-assisted DE, is brought up by incorporating classification rather than ranking or regression for pairwise comparison into DE to solve computationally expensive functions. The efficacy of classification-assisted DE was investigated by using SVMs to learn the surrogate model with a conclusion that classification-assisted DE has great potential in reducing fitness evaluations. Although classification-assisted DE is developed from DE, the classification-based method for building surrogates has its universality in EAs with pairwise selection.

The main weakness of the approach is that imbalanced training set has effect on the classification accuracy. In experimental studies, a simple threshold moving was employed at the expectation of avoiding this. Although this may not solve imbalanced problems completely, the above experimental results have indicated that classification-assisted DE is promising. Moreover, the opportunities for future works have been opened, which may include investigation on incorporating other methods for imbalanced data classification into classification-assisted DE.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China under Grants 60802036, U0835002, and 61028009, and by the Engineering and Physical Sciences Research Council (EPSRC Grant EP/D052785/1).

Fig. 1. Evolution curve on f_1

Fig. 2. Evolution curve on f_2

Fig. 3. Evolution curve on f_3

Fig. 4. Evolution curve on f_4

Fig. 5. Evolution curve on f_5

Fig. 8. Evolution curve on f_8

Fig. 6. Evolution curve on f_6

Fig. 9. Evolution curve on f_9

Fig. 7. Evolution curve on f_7

Fig. 10. Evolution curve on f_{10}

REFERENCES

- [1] K. Price, R. Storn, and J. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Verlag, 2005.
- [2] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 1, pp. 3–12, 2005.
- [3] M. Farina and J. Sykalski, "Comparative study of evolution strategies combined with approximation techniques for practical electromagnetic optimization problems," *IEEE Transactions on Magnetics*, vol. 37, no. 5, pp. 3216–3220, 2002.
- [4] P. Hajela and J. Lee, "Genetic algorithms in multidisciplinary rotor blade design," in *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 36 th, and AIAA/ASME Adaptive Structures Forum, New Orleans, LA*, 1995, pp. 2187–2197.
- [5] L. Shi and K. Rasheed, "A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms," *Computational Intelligence in Expensive Optimization Problems*, pp. 3–28, 2010.
- [6] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.
- [7] T. Runarsson, "Ordinal regression in evolutionary computation," *Parallel Problem Solving from Nature-PPSN IX*, pp. 1048–1057, 2006.
- [8] I. Loshchilov, M. Schoenauer, and M. Sebag, "Comparison-Based Optimizers Need Comparison-Based Surrogates," *Parallel Problem Solving from Nature-PPSN XI*, pp. 364–373, 2010.
- [9] J. Zhang and A. Sanderson, "DE-AEC: A differential evolution algorithm based on adaptive evolution control," in *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*. IEEE, 2008, pp. 3824–3830.
- [10] K. Rasheed, H. Hirsh, and A. Gelsey, "A genetic algorithm for continuous design space search," *Artificial Intelligence in Engineering*, vol. 11, no. 3, pp. 295–305, 1997.
- [11] Y. Tenne, K. Izui, and S. Nishiwaki, "Handling Undefined Vectors in Expensive Optimization Problems," *Applications of Evolutionary Computation*, pp. 582–591, 2010.
- [12] S. Handoko, K. Keong, and O. Soon, "Using classification for constrained memetic algorithm: A new paradigm," in *IEEE International Conference on Systems, Man and Cybernetics, 2008. SMC 2008*. IEEE, 2009, pp. 547–552.
- [13] S. Handoko, C. Kwok, and Y. Ong, "Classification-Assisted Memetic Algorithms for Equality-Constrained Optimization Problems," *AI 2009: Advances in Artificial Intelligence*, pp. 391–400, 2009.
- [14] —, "Feasibility Structure Modeling: An Effective Chaperone for Constrained Memetic Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 740–758, 2010.
- [15] R. Storn and K. Price, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces," *INTERNATIONAL COMPUTER SCIENCE INSTITUTE-PUBLICATIONS-TR*, 1995.
- [16] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [17] H. Yu and S. Kim, "SVM Tutorial: Classification, Regression, and Ranking," *Handbook of Natural Computing*, 2009.
- [18] Z. Zhou and X. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, pp. 63–77, 2006.
- [19] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1263–1284, 2008.
- [20] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *KanGAL Report*, vol. 2005005, 2005.
- [21] S. Gunn, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, 1998.
- [22] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "PRTTools 4.1, A Matlab Toolbox for Pattern Recognition," *Delft University of Technology*, 2007.