Applied Soft Computing 34 (2015) 770–787

Contents lists available at ScienceDirect

Applied Soft Computing

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

An evolving surrogate model-based differential evolution algorithm

Rammohan Mallipeddi, Minho Lee*

School of Electronics Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, Republic of Korea

ARTICLE INFO

Article history:
Received 1 May 2013
Received in revised form 20 April 2015
Accepted 10 June 2015
Available online 19 June 2015

Keywords:
Differential evolution
Global optimization
Surrogate model
Parameter adaptation
Ensemble

ABSTRACT

Differential evolution (DE) is a simple and effective approach for solving numerical optimization problems. However, the performance of DE is sensitive to the choice of mutation and crossover strategies and their associated control parameters. Therefore, to achieve optimal performance, a time-consuming parameter tuning process is required. In DE, the use of different mutation and crossover strategies with different parameter settings can be appropriate during different stages of the evolution. Therefore, to achieve optimal performance using DE, various adaptation, self-adaptation, and ensemble techniques have been proposed. Recently, a classification-assisted DE algorithm was proposed to overcome trial and error parameter tuning and efficiently solve computationally expensive problems. In this paper, we present an evolving surrogate model-based differential evolution (ESMDE) method, wherein a surrogate model constructed based on the population members of the current generation is used to assist the DE algorithm in order to generate competitive offspring using the appropriate parameter setting during different stages of the evolution. As the population evolves over generations, the surrogate model also evolves over the iterations and better represents the basin of search by the DE algorithm. The proposed method employs a simple Kriging model to construct the surrogate. The performance of ESMDE is evaluated on a set of 17 bound-constrained problems. The performance of the proposed algorithm is compared to state-of-the-art self-adaptive DE algorithms: the classification-assisted DE algorithm, regression-assisted DE algorithm, and ranking-assisted DE algorithm.

1. Introduction

Differential evolution (DE) [1] is a fast and simple population-based stochastic search technique that is inherently parallel and performs well on a wide variety of real-world problems [2]. The performance of the DE algorithm is sensitive to the population size (*NP*), mutation and crossover strategies, and their associated control parameters, such as the crossover rate (*CR*) and scale factor (*F*) [3,4]. The best combination of strategies and control parameters can be different for different optimization problems and also for the same functions with different computation times and accuracy requirements. Therefore, to successfully solve a specific optimization problem, it is generally necessary to perform a time-consuming trial-and-error search for the most appropriate combination of strategies and their associated parameter values, which results in high computational costs. In addition, when the population of DE evolves through different regions in the search space, different strategies [5] combined with different parameter settings may be more effective than a single combination of strategies and parameters for the entire search. Therefore, to overcome the trial-and-error search for the appropriate combination of strategies and parameter values and to adapt the combination to different regions in the search space, various adaptation schemes have been proposed [5–10]. Recently, a DE algorithm with ensemble of mutation strategies and parameter values (EPSDE) [10,11] was proposed to overcome the time-consuming trial-and-error procedure of the conventional DE algorithm and has shown significant improvement compared to other state-of-the-art algorithms.

In time-consuming and expensive optimization problems, surrogate models built based on the evolutionary history of the population members have been incorporated into different evolutionary algorithms (EAs) to reduce the number of original fitness evaluations [12–14]. A surrogate model is an estimation model, built by using other functions, which is used to estimate the original objective function. Surrogate models can be built using a variety of techniques, such as radial basis functions [13], artificial neural networks, Kriging models [14], support vector machines, splines, and polynomial approximation models. A classification-assisted DE that incorporates classification for pairwise comparison was proposed in [15] to solve computationally expensive problems. The performance of the support vector classification-assisted DE

* Corresponding author. Tel.: +82 1088596436.
E-mail addresses: mallipeddi.ram@gmail.com (R. Mallipeddi), mholee@knu.ac.kr (M. Lee).

http://dx.doi.org/10.1016/j.asoc.2015.06.010
1568-4946/© 2015 Elsevier B.V. All rights reserved.

(SVC-DE) algorithm has been compared with both the support vector regression-assisted DE (SVR-DE) algorithm and ranking-based support vector machine assisted DE (RankSVM-DE) algorithm.

In this paper, we present an evolving surrogate model-based differential evolution algorithm (ESMDE). In the EMSDE, the DE algorithm is equipped with an evolving surrogate model constructed using the evolving population members. The evolving surrogate model enables the EMSDE algorithm to generate competitive offspring during different stages of the evolution with the assistance of the appropriate strategy and parameter combinations. In the present work, we adopt a surrogate model built using the Kriging technique based on the suggested model discussed in [14]. Kriging is a spatial prediction method based on minimizing the mean squared error. We also apply design and analysis of computer experiments (DACE), which is a parametric regression model that can handle three or more dimensions [14]. The performance of the proposed algorithm has been compared with state-of-the-art self-adaptive algorithms, such as JADE [16], CoDE [17], SaDE [5], and EPSDE [11]. The performance of the proposed algorithm has also been compared to the SVC-DE, SVR-DE, and RankSVM-DE algorithms presented in [15].

The reminder of this paper is organized as follows: Section 2 presents a literature survey regarding DE, surrogate models, and the usage of surrogate models in DE. Section 3 presents the proposed ESMDE algorithm. Section 4 presents the experimental results and discussions, and Section 5 concludes the paper.

2. Literature review

2.1. Differential evolution

Differential evolution (DE) is a parallel direct search method that utilizes *NP* *D*-dimensional parameter vectors, the so-called individuals, which encode the candidate solutions, *i.e.* $\mathbf{X}_{i,G} = \{x^1_{i,G}, \ldots, x^D_{i,G}\}, \quad i = 1, \ldots, NP$. The initial population needs to cover as much of the entire search space as possible by uniformly randomizing the initial individuals within the search space constrained by the prescribed minimum and maximum parameter bounds $\mathbf{X}_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $\mathbf{X}_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$. For example, the initial value of the *j*th parameter of the *i*th individual at generation *G* = 0 is generated by:

$$x^j_{i,0} = x^j_{\min} + rand(0,1).(x^j_{\max} - x^j_{\min}) \quad j = 1, 2, \ldots, D \tag{1}$$

where *rand*(0,1) represents a uniformly distributed random variable within the range [0,1].

After initialization, the DE algorithm employs a mutation operation to produce a mutant vector $\mathbf{V}_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ corresponding to each individual $\mathbf{X}_{i,G}$, the so-called target vector, in the current population. For each target vector $\mathbf{X}_{i,G}$ at generation *G*, its associated mutant vector can be generated via a certain mutation strategy. The most frequently used mutation strategies are:

$$\text{"DE/best/1"[18]}: \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F\cdot(\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G}) \tag{2}$$

$$\text{"DE/best/2"[18]}: \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F\cdot(\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G}) + F\cdot(\mathbf{X}_{r^i_3,G} - \mathbf{X}_{r^i_4,G}) \tag{3}$$

$$\text{"DE/rand/1"[18]}: \mathbf{V}_{i,G} = \mathbf{X}_{r^i_1,G} + F\cdot(\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}) \tag{4}$$

$$\text{"DE/rand/2"[5]}: \mathbf{V}_{i,G} = \mathbf{X}_{r^i_1,G} + F\cdot(\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}) + F\cdot(\mathbf{X}_{r^i_4,G} - \mathbf{X}_{r^i_5,G}) \tag{5}$$

"DE/rand-to-best/1"[18] or"DE/target-to-best/1"[19] :

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + K\cdot(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F\cdot(\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G}) \tag{6}$$

"DE/rand-to-best/2"[5] or"DE/target-to-best/2" :

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + K\cdot(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F\cdot(\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G} + \mathbf{X}_{r^i_3,G} - \mathbf{X}_{r^i_4,G}) \tag{7}$$

$$\text{"DE/current-to-rand/1"[20]}: \mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K\cdot(\mathbf{X}_{r^i_1,G} - \mathbf{X}_{i,G}) + F\cdot(\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}) \tag{8}$$

The indices $r^i_1, r^i_2, r^i_3, r^i_4, r^i_5$ are randomly generated mutually exclusive integers different from index *i* and within the range of [1,*NP*]. These indices are randomly generated once for each mutant vector. The scale factor *F* is a positive control parameter for scaling the difference vector. $\mathbf{X}_{best,G}$ is the best individual vector with the best fitness value in the population at generation *G*. *K* is randomly chosen within the range [0,1].

After the mutation phase, a trial vector $\mathbf{U}_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \ldots, u^D_{i,G}\}$ is generated by the crossover operation on each pair of target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$. The crossover operation speeds the convergence by a constant factor. The DE algorithm generally employs two kinds of crossover methods: binomial (or uniform) and exponential (or two-point modulo). In the basic version, the DE algorithm employs the binomial crossover defined as [1]:

$$u^j_{i,G} = \begin{cases} v^j_{i,G} & \text{if}(rand_j[0,1] \le CR) \text{ or } (j = j_{rand}), \quad j = 1, 2, \ldots, D \\ x^j_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

In (9), the crossover probability, *CR* ∈ [0,1], is a user-specified constant that controls the fraction of parameter values that are copied to the trail vector from the mutant vector. j_{rand} is a randomly chosen integer in the range [1,*D*]. The binomial crossover operator copies the *j*th parameter of the mutant vector $\mathbf{V}_{i,G}$ to the corresponding element in the trial vector $\mathbf{U}_{i,G}$ if $rand_j$ [0,1] or $j = j_{rand}$. Otherwise, it is copied from the corresponding target vector $\mathbf{X}_{i,G}$. The condition $j = j_{rand}$ is introduced to ensure that trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$ by at least one parameter. Here, the crossover operator is uniform in the sense that each parameter of the mutant vector, regardless of its location, has the same probability *CR* of inheriting its value from a given to the trail vector. For this reason, the uniform crossover does not exhibit any representational bias.

In the exponential crossover, an integer *n* ∈ [1,*D*], which acts as a starting point in the target vector, is chosen randomly from the point at which the crossover or exchange of components with the mutant vector starts. *L* ∈ [1,*D*] denotes the number of components that are contributed by the mutant vector to the target vector. Integer *L* is drawn from [1,*D*] depending on the crossover probability (*CR*), according to the following pseudo-code:

```
$L = 0$;
  DO
      $L = L + 1$;
  WHILE ($rand(0,1) < CR$ and $L < D$);
```

After choosing *n* and *L*, the trial vector is obtained by:

$$u^j_{i,G} = \begin{cases} v^j_{i,G}, & \text{for} j = \langle n\rangle_D, \langle n+1\rangle_D, \ldots, \langle n+L-1\rangle_D \\ x^j_{i,G}, & \text{for all other} j \in [1, D] \end{cases} \tag{10}$$

where the angular brackets $\langle\,\rangle_D$ denote a modulo function with modulus *D*.

Table 1
The standard differential evolution algorithm.

```
Step 1 Set the generation number $G = 0$ and randomly initialize a population of
  $NP$ individuals $P_G = \{X_{i,G}, ..., X_{NP,G}\}$ with $X_{i,G} = \{x^1_{i,G}, \ldots, x^D_{i,G}\}$, $i = 1, \ldots, NP$
  uniformly distributed in the range $[X_{\min}, X_{\max}]$, where $X_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$
  and $X_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$
Step 2 WHILE stopping criterion is not satisfied
  DO
  Step 2.1 Mutation step
    /*Generate a mutated vector $V_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ for each target vector $\mathbf{X}_{i,G}$ */
    FOR $i = 1$ to $NP$
      Generate a mutated vector $V_{i,G}$ corresponding to the target vector $\mathbf{X}_{i,G}$ via one of Eqs. (2)–(6).
    END FOR
  Step 2.2 Crossover step
    /*Generate a trial vector $U_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \ldots, u^D_{i,G}\}$ for each target vector $\mathbf{X}_{i,G}$ */
    /*Binomial crossover*/
    FOR $i = 1$ to $NP$
      $jrand = [rand[0,1].D]$
      FOR $j = 1$ to $D$
        $u^j_{i,G} = \begin{cases} v^j_{i,G} & \text{if}(rand_j[0,1] \le CR) \quad \text{or} \quad (j = j_{rand}), \quad j = 1, 2, \ldots, D \\ x^j_{i,G} & \text{otherwise} \end{cases}$
      END FOR
    END FOR
  Step 2.3 Selection step
    /*Selection*/
    FOR $i = 1$ to $NP$
      /* Evaluate the trial vector $U_{i,G}$ */
      IF $f(U_{i,G}) \le f(X_{i,G})$, THEN $X_{i,G+1} = U_{i,G}$, $f(X_{i,G+1}) \le f(U_{i,G})$
        IF $f(U_{i,G}) < f(X_{best,G})$, THEN $X_{best,G} = U_{i,G}$, $f(X_{best,G}) \le f(U_{i,G})$
        /* $X_{best,G}$ is the individual with the best fitness value in generation $G$ */
        ELSE $X_{i,G+1} = X_{i,G}$
        END IF
      END IF
    END FOR
  Step 2.4 Increment the generation count $G = G + 1$
Step 3 END WHILE
```

In the exponential crossover operation, the parameters of trial vector $\mathbf{U}_{i,G}$ are inherited from the corresponding mutant vector $\mathbf{V}_{i,G}$ starting from a randomly chosen parameter index (n) until the first time $rand_j$ (0,1) < CR. The remaining parameters of trial vector $\mathbf{U}_{i,G}$ are copied from the corresponding target vector $\mathbf{X}_{i,G}$. The exponential crossover operator in the DE algorithm is functionally equivalent to the circular two-point crossover operator [5].

During the evolution, mutation, and crossover operations, if the values of some of the parameters from a newly generated trial vector exceed the corresponding upper and lower bounds, then the corresponding parameters are set to the bounds or randomly and uniformly reinitialized within the pre-specified range. The objective function values of all trial vectors are then evaluated. After the evaluation, a selection operation is performed. The objective function value of each trial vector $f(\mathbf{U}_{i,G})$ is compared to that of its corresponding target vector $f(\mathbf{X}_{i,G})$ in the current population. If the trial vector has a less or equal objective function value (in a minimization problem) than the corresponding target vector, the trial vector will replace the target vector and enter the population of the next generation. Otherwise, the target vector will remain in the population for the next generation. The selection operation can be expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if} f(\mathbf{U}_{i,G}) \le f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \tag{11}$$

The three steps, mutation, crossover, and selection, are repeated generation after generation until a termination criterion (reaching the maximum number of function evaluations set) is satisfied. The algorithmic description of the DE method is summarized in Table 1.

From the algorithmic description presented above, it can be observed that the DE algorithm is a floating-point encoded evolutionary algorithm for global optimization over continuous spaces [1]. Although the conventional DE algorithm has recently attracted much attention, its performance depends on the chosen mutation strategy and the associated control parameters. The performance of the DE algorithm becomes more sensitive to the strategy and the associated parameter values when the problem is complex [21]. Inappropriate selection of the mutation strategy or parameters may lead to premature convergence, stagnation, or waste of computational resources [8,19,21–23]. Initially, it was proposed in [1,24] that the control parameters of DE are not difficult to choose. However, due to the complex interaction between control parameters and the performance of DE in hard optimization problems [6], choosing appropriate mutation and control strategies as well as their control parameters requires some expertise. Since the DE algorithm was proposed, various empirical guidelines have been suggested for choosing a mutation strategy with its associated control parameter settings.

In the DE method, the larger the population size, the higher the probability of finding a global optimum for multi-modal problems. However, a larger population implies a slower convergence rate, which would then require a larger number of function evaluations. Therefore, separable and uni-modal functions require smaller population sizes to speed up convergence, whereas parameter-linked multi-modal functions require larger populations to avoid premature convergence. Initially, the population size $NP = 10D$ is considered to be a good choice for the DE method to identify a global optimum [18]. However, to balance speed (the number of function evaluations) and reliability (the ability to find a global optimum), different ranges of NP values, such as $5D$ to $10D$ [1], $3D$ to $8D$ [21], and $2D$ to $40D$ [25], have been suggested. According to [19], $NP = 5D * CR$ is usually a good low-end default setting, whereas for highly multi-modal, parameter-dependent functions, NP may need to be at least $10D$ or higher. However, NP should be larger than a critical value depending on the mutation strategy used [21]. For example, NP must be at least 4 for DE/rand/1/bin and 5 for DE/best/2/bin to ensure that the DE algorithm will have a sufficient number of mutually different vectors. In evolutionary algorithms, a common strategy that is adopted when an algorithm does not perform as well as expected is to increase population size. However, in [26], the authors analytically showed that a large population is not always beneficial.

The classic DE method proposed by Price and Storn uses DE/rand/1/bin, which is most widely used. Later, different DE mutation strategies were proposed [18,27]. In [1,21], two different vector strategies, DE/rand/2/bin and DE/best/2/bin, were shown to be better than either DE/rand/1/bin or DE/best/1/bin due to their ability to improve diversity by producing more trial vectors [23]. DE/best/1/bin and DE/rand-to-best/1/bin are faster for easier optimization problems, but they become unreliable when solving highly multi-modal problems. To balance the exploration and exploitation abilities of DE, a DE/target-to-best/1/bin scheme employing the neighborhood concept of each population member was proposed [28]. However, increasing the population size (NP) had little impact on the convergence probabilities. The classic DE method is slower but still more robust than other strategies that rely on the best-so-far vector. The DE/current-to-rand/1/bin, being a rotation-invariant strategy, can solve rotated problems better than the other strategies [20].

In the DE method, the crossover rate controls which and how many components are mutated in each element of the current population [29]. The crossover rate CR is the probability $0 \le CR \le 1$ of mixing between trial and target vectors. A large CR speeds up convergence [1,18,21]. According to [1], $CR = 0.1$ is a good initial choice, whereas $CR = 0.9$ or 1.0 tend to increase the convergence speed.

According to [21], a good choice for *CR* is between 0.3 and 0.9. For separable problems, a *CR* in the range (0, 0.2) is the best, whereas for multi-modal, parameter-dependent problems, a *CR* in the range (0.9, 1.0) is best [19,25]. Based on these observations, *CR* with non-continuous ranges are also used [30]. When $CR = 1.0$ is chosen, the number of trial solutions will be reduced dramatically, which may lead to stagnation [23,25]. Therefore, $CR = 0.9$ or 0.99 should be used instead.

The scaling factor *F* is usually chosen within [0.5, 1] [18]. According to [1], values of *F* smaller than 0.4 and greater than 1.0 are occasionally effective. The scale factor *F* is strictly greater than zero. A larger *F* increases the probability of escaping from a local optimum [21,25]. While $F > 1$ can solve many problems, decreases in convergence speed make it difficult to converge, as the perturbation is larger than the distance between two members. $F \leq 1$ is usually both faster and more reliable. *F* must be above a certain critical value to avoid premature convergence to a sub-optimal solution [21,25]. However, if *F* becomes too large, the number of function evaluations to find an optimum grows very quickly. According to [1,21], $F = 0.6$ or 0.5 would be a good initial choice, whereas according to [25], $F = 0.9$ would be a good initial choice. Typical values of *F* are 0.4 to 0.95 according to [25]. Zaharie [31] theoretically proved that the DE could converge to a global optimum over a long time if *F* can be transformed into a Gaussian random variable. Despite theoretical proof, it was proven that transforming *F* into a Gaussian random variable does not significantly enhance the performance of the DE [32]. In addition, Price et al. [19] demonstrated that unless the variance of the randomizing distribution is very small, the DE will suffer significant performance loss on highly conditioned non-separable functions [25]. Similar to *CR*, $F = 1.0$ is not recommended since it reduces the number of potential trial solutions and may lead to stagnation.

Even though the above guidelines are useful for choosing individual DE parameters to some extent, the performance of the DE is more sensitive to the combination of mutation strategy and its associated parameters. For a given mutation strategy, a particular value of *CR* makes the parameter *F* sensitive, whereas some other values of *CR* make the same *F* robust [6]. It was recommended in [27] to use the mutation strategy DE/current-to-rand/1/bin and parameter settings $NP = 20D$, $K = 0.5$, and $F = 0.8$. If the DE converges prematurely, one should increase the value of *NP* or *F*, or randomly choose *K* within the range [0,1]. If none of the above configurations work, one may try the strategy DE/rand/1/bin with a small *CR* value.

Taking this all into account, various conflicting conclusions have been drawn regarding the manual parameter tuning of DE, all of which lack sufficient justification. Therefore, to avoid the tuning of parameters by trial-and-error, various techniques have been developed. The scale factor *F* can be linearly reduced from a maximum to a minimum value with increasing generation count [22], or randomly varied in the range (0.5, 1), or a uniform distribution between 0.5 and 1.5 (with a mean of 1) can be employed [33]. Recently, several researchers [8,34–36] have focused on the adaptation of the control parameters *F* and *CR*. FADE adapts the control parameters *F* and *CR* based on fuzzy logic controllers whose inputs are the relative function values and individuals of successive generations [35]. FADE outperforms conventional DE for higher dimensional problems. A parameter adaptation of DE (ADE) based on controlling population diversity and a multi-population approach [8] was proposed, which was later extended to form an adaptive Pareto DE algorithm for multi-objective optimization [36]. Abbass [34] self-adapted the crossover rate of the DE for multi-objective optimization problems by encoding the crossover rater into each individual in order to simultaneously evolve with the other parameters. The scale factor *F* was generated using a Gaussian distribution *N*(0,1) for each individual. Qin et al. [5] proposed a self-adaptive DE algorithm (SaDE), in which the mutation strategies and the respective control parameter are self-adapted based on their previous experiences of generating promising solutions. The scale factor *F* was randomly generated with a mean and standard deviation of 0.5 and 0.3, respectively. Omran et al. [7] introduced a self-adaptation scheme (SDE), in which the *CR* is generated randomly for each individual using a normal distribution *N*(0.5,0.15), whereas the scale factor *F* is adapted analogous to the adaptation of the crossover rate *CR* in [34]. In addition to the control parameters *F* and *CR*, Teo [37] proposed differential evolution with self-adapting populations (DESAP) based on Abbas's self-adaptive Pareto DE. Brest et al. [6] proposed a self-adaptation scheme (JDE), in which the control parameters *F* and *CR* are encoded into the individuals and adjusted by introducing the two new parameters τ_1 and τ_2. Initially, the *F* and *CR* values of each individual were assigned to 0.5 and 0.9, respectively. Depending on a random number (*rand*) uniformly generated in the range of [0,1], the parameters *F* and *CR* are reinitialized to new random values in the ranges [0.1, 1.0] and [0,1], respectively. Re-initialization of *F* and *CR* is done if $rand < \tau_1$ and $rand < \tau_2$, respectively, and are kept unchanged otherwise. In [10,11], the authors proposed a DE algorithm with an ensemble of mutation strategies and parameter values (EPSDE), which consists of a pool of mutation and crossover strategies and their associated parameter values. Initially, the population members randomly pick the strategies and parameter values from the respective pools and produce an offspring population. Depending on the success of the offspring, the corresponding combination of strategies and their associated parameter values is retained or reinitialized. In EPSDE, retaining the combination that produces better offspring and reinitializing the combination that is incapable of producing competitive offspring favor selection of the combination that produces better solutions in the due course of the evolution.

2.2. Surrogate modeling

Surrogate models or meta models are mathematical models that mimic the behavior of a computationally expensive simulation code over the complete parameter space as accurately as possible, using the least amount of data points. In other words, surrogate models are orders of magnitude cheaper to run and can be used in lieu of exact analysis during an evolutionary search [38]. Furthermore, the surrogate model can also yield insights into the functional relationship between the input *x* and output *y*. If the true nature of a computer analysis code is represented as:

$$y = f(x) \tag{12}$$

Then a surrogate model is an approximation of the form:

$$\hat{y} = \hat{f}(x) \tag{13}$$

in such a manner that $y = \hat{y} + \varepsilon$.

A variety of techniques exist for constructing surrogate models, such as rational functions, radial basis functions, artificial neural networks, Kriging models, support vector machines, splines, and polynomial approximation models. There has been a variety of studies where a comparison among different techniques is performed [39,40]. Most of these works make comparisons only among a small group of techniques and test problems, or they take into account only one criterion for selection of the best model. In [14], the authors demonstrated the selection of the meta-modeling technique according to two important factors: dimensionality and suitability of the meta-model to be combined with the evolutionary optimization algorithms. The paper concludes that the best approach to be used in low dimensionality problems is the Kriging or Support Vector Regression method, whereas for high dimensional problems, the best technique is the radial basis function network [14].

Among the different surrogate models, Kriging belongs to the class of Geo-statistical methods and describes spatial and temporal correlations among the values of the attribute [14]. It is a statistically sound alternative for constructing surrogate models of deterministic computer models and is referred to as a design and analysis of computer experiment (DACE) model in the statistics literature [41] as well as a Gaussian process regression in the neural-network literature [42]. In the DACE model, an extension of the Kriging approach, a surrogate model can be expressed as a combination of a parametric model and a non-parametric stochastic process as:

$$y(x) = f(x)^T\beta + z(x) \tag{14}$$

where β is the regression coefficient. $f(x)$ is a polynomial in x, which provides the global approximation of the original function. $z(x)$ is a stochastic process that approximates the localized deviations, such that the Kriging surrogate model can better interpolate the sample points. $z(x)$ is assumed to have a zero mean and covariance given by:

$$Cov(z(x^{(i)}), z(x^{(j)})) = \sigma^2\Re(\theta, x^{(i)}, x^{(j)}) \tag{15}$$

where σ^2 is the process variance and $\Re(\theta, x^{(i)}, x^{(j)})$ is the correlation model with parameters θ. $x^{(i)}$ and $x^{(j)}$ are the ith and$j$th design sites, respectively.

This study employs an evolving surrogate model that is constructed in every generation of the evolution using the population members of the current generation. In other words, the surrogate model is built on the fly as the population members of the new generation are updated, and the built model represents the local model of the current population. Since the surrogate model is built once in every generation during the evolution, computational efficiency is necessary. This consideration motivates the use of Kriging, which can be efficiently applied to approximate multiple-input multiple-output data, particularly when a few hundred data points are used for training [15].

2.3. Surrogate modeling in differential evolution

In complex optimization problems, the performance of DE [1] is sensitive to the selected mutation and crossover strategies and their associated parameter values [21]. Inappropriate selection of strategies and associated parameters may lead to premature convergence, stagnation, or wastage of computational resources [8,19,21–23]. Therefore, in hard optimization problems, due to the complex interactions of the strategies and parameters with the DE performance [6], choosing an appropriate combination of the strategies and control parameters requires some expertise. Since the DE method was introduced, various empirical guidelines have been suggested for choosing the strategies and their associated control parameter settings [1,16,22,25,29,30].

The suggested guidelines are useful in choosing individual DE parameters to some extent; however, it has been shown that DE performance is more sensitive to the combination of mutation and crossover strategies and their associated parameters than the individual settings. In addition, it was also observed that different combinations of strategies and parameters are more effective during the different stages of population evolution than a single combination used for the entire search process [5]. Therefore, to improve the performance of DE, various adaptation techniques have been proposed [5,6,8,10,27,33–37,43,44]. In [10,11], a DE algorithm was proposed with an ensemble of mutation and crossover strategies and parameter values (EPSDE), in which a pool of mutation strategies along with a pool of values corresponding to each associated parameter compete to produce a successful offspring population.

In expensive optimization problems, to reduce the computational costs, surrogate models have been incorporated into the DE method [12–14]. A surrogate model is a computationally inexpensive estimation of the original objective function that significantly reduces the number of expensive fitness evaluations. In [13], the authors proposed a DE algorithm that generates multiple offspring for each parent and chooses the most promising one based on the accuracy and predicted function value of the surrogate model. In [45], the authors built a non-parametric surrogate model of the problem by using the entire search history and applied a gradient descent-like method to mutate solutions in a parameter-less and adaptive manner. In [46], the authors proposed a multi-surrogates assisted memetic algorithm (MSAMA) to solve optimization problems with computationally expensive fitness functions. The backbone of the MSAMA framework is an evolutionary algorithm coupled with local solvers that employs multiple surrogates built using both regression and interpolation to provide diversity in the approximation models.

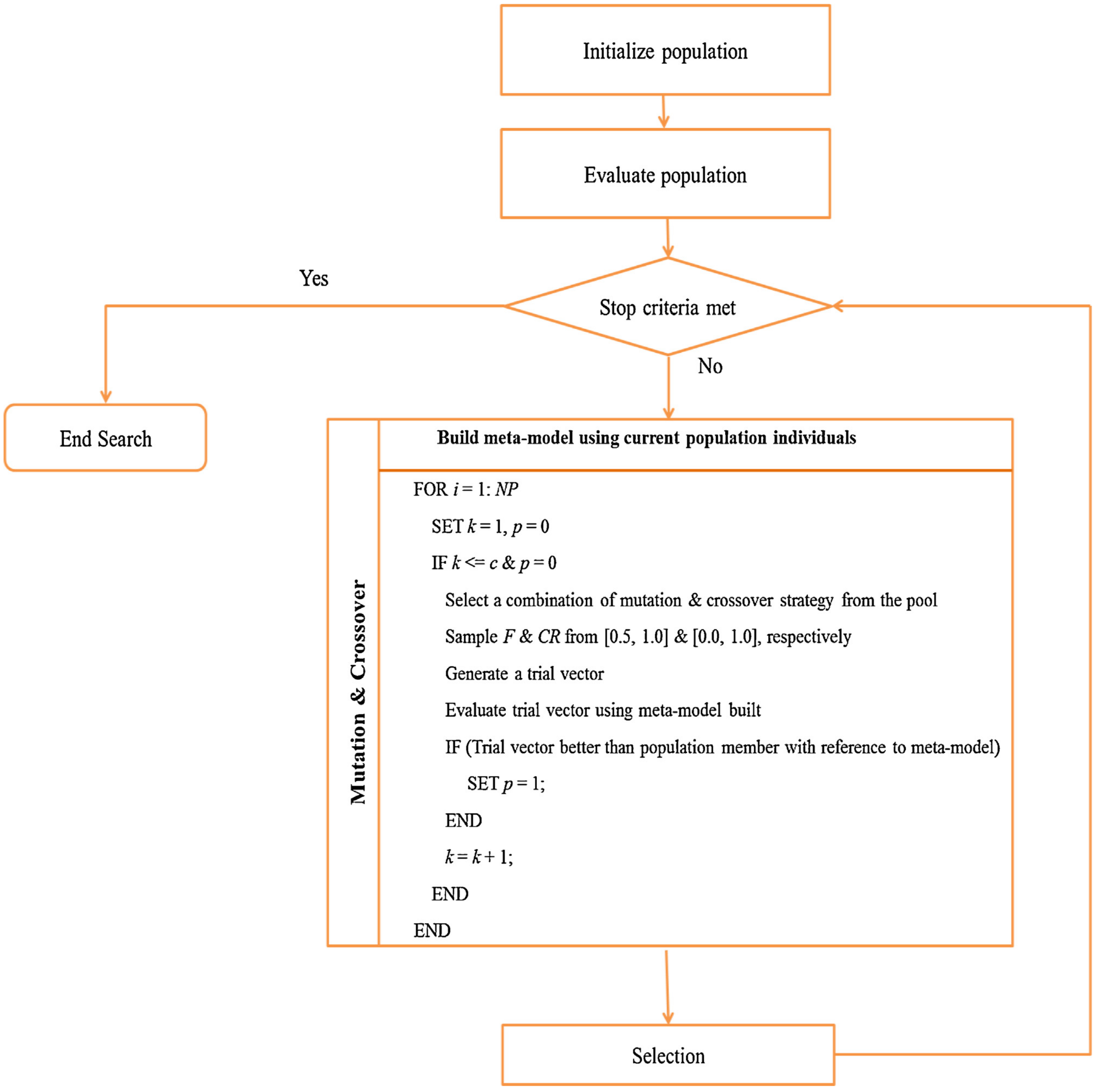
3. The evolving surrogate model-based DE algorithm (ESMDE)

Each optimization problem is unique and requires different mutation and crossover strategies with different parameter values depending on the nature of the problem (uni-modal and multi-modal) and the available computation resources. In addition, to solve a specific problem, different mutation strategies with different parameter settings may be better during different stages of the evolution than the single mutation strategy with unique parameter settings used in conventional DE. To address these issues, the authors of [10,11] proposed an ensemble of parameters, mutation, and crossover strategies (EPSDE).

The EPSDE algorithm consists of pools of mutation and crossover strategies along with a pool of values corresponding to each of the associated parameters that compete to produce a successful offspring population. It is successful to obtain better performance compared to some self-adaptive techniques while overcoming the expensive trial-and-error search for the combination of strategies and parameter values. In the EPSDE algorithm, each member in the initial population is randomly assigned a mutation and crossover strategy and associated parameter values taken from the respective pools. The population members (target vectors) produce offspring (trial vectors) using the assigned mutation strategy and parameter values. If the generated trial vector is better than the target vector, the mutation strategy and parameter values are retained with the trial vector, which becomes the parent (target vector) in the next generation. The combination of mutation strategy and parameter values that produce better offspring than the parent is stored. If the target vector is better than the trial vector, then the target vector is randomly reinitialized with a new mutation strategy and associated parameter values from the respective pools or from the successful combinations stored with equal probability. The process of persisting with the combination of strategies and their associated control parameters that produced successful trial vectors leads to an increased probability of producing offspring individuals in future generations. In the EPSDE algorithm, the candidate pool of mutation and crossover strategies and parameters needs to be restricted in order to avoid the unfavorable influences of less effective mutation strategies and parameters [5]. In addition, the candidate pools need to have diverse characteristics so that they can exhibit distinct performance characteristics during the different stages of the evolution when dealing with a particular problem.

In the EPSDE algorithm, performance degradation can be observed when the population members are assigned a combination of strategies and parameters that produce better offspring but lead to premature convergence in the early generations of the evolution. In this paper, we propose a DE algorithm that can generate competitive offspring during the different stages of the evolution with minimum computational costs with the assistance of a surrogate model built using the evolving population members. The proposed model, referred to as ESMDE, can overcome the drawbacks of the EPSDE algorithm and demonstrates improved performance in terms of convergence and number of function evaluations.

Similar to the EPSDE algorithm, the proposed ESMDE algorithm consists of a pool of mutation and crossover strategies, whereas the associated control parameters are randomly sampled. The ranges of values from which *F* and *CR* are sampled from [0.5, 1.0] and [0.0, 1.0], respectively. In each generation, every population member repeatedly picks up a mutation strategy and a crossover strategy from the respective pools along with randomly sampled *F* and *CR* values from the respective ranges until an offspring (competitive offspring), which is fitter than the parent based on the surrogate model, is generated or a maximum number of chances are reached. Let *c* be the number of chances a population member is given to generate a competitive offspring based on the surrogate model. The outline of the proposed ESMDE algorithm is depicted in Fig. 1. As the surrogate model is an approximate of the original function, the probability of an offspring that is better than the parent based on the surrogate model is high. In every generation, as each parent is given *c* number of chances to produce a successful offspring based on the surrogate model evaluation, the probability increases. In each generation, since every original function evaluation is preceded by a number of surrogate model evaluations, the probability of the population moving to a better search direction is higher with a lower number of original function evaluations. In addition, the burden of selecting an appropriate



```
FOR $i = 1: NP$
  SET $k = 1, p = 0$
  IF $k <= c$ & $p = 0$
    Select a combination of mutation & crossover strategy from the pool
    Sample $F$ & $CR$ from [0.5, 1.0] & [0.0, 1.0], respectively
    Generate a trial vector
    Evaluate trial vector using meta-model built
    IF (Trial vector better than population member with reference to meta-model)
      SET $p = 1$;
    END
    $k = k + 1$;
  END
END
```

Fig. 1. Outline of the proposed ESMDE.

Table 2
The evolving surrogate model-based DE algorithm (ESMDE).

```
Step 1 Set the generation number $G=0$ and randomly initialize a population of
  $NP$ individuals $P_G=\{X_{i,G},\ldots,X_{NP,G}\}$ with $X_{i,G}=\{x^1_{i,G},\ldots,x^D_{i,G}\}$,  $i=1,\ldots,NP$
  uniformly distributed in the range $[X_{\min},X_{\max}]$, where
  $X_{\min}=\left\{x^1_{\min},\ldots,x^D_{\min}\right\}$ and $X_{\max}=\left\{x^1_{\max},\ldots,x^D_{\max}\right\}$
Step 2 Select a pool of mutation and crossover strategies
Step 3 Build a surrogate model with the evaluated population members
Step 4 WHILE stopping criterion is not satisfied
          DO
          FOR $i=1,\ldots,NP$
          WHILE stopping criteria is not satisfied
    Step 4.1 Mutation step
      Generate a mutant vector $V_{i,G}=\{v^1_{i,G},v^2_{i,G},\ldots,v^D_{i,G}\}$ for each target vector
  $\mathbf{X}_{i,G}$ with a randomly selected mutation strategy from the pool and a random
  $F$ value in the range [0.5 1.0].
    Step 4.2 Crossover step
      Generate a trial vector $W_{i,G}=\{w^1_{i,G},w^2_{i,G},\ldots,w^D_{i,G}\}$ for each target vector
  $\mathbf{X}_{i,G}$ with a randomly chosen crossover strategy from the pool and a random
  $CR$ value in the range [0 1.0].
    Step 4.3 Competitive offspring Generation
                IF
                  $f_s(W_{i,G})\le f_s(X_{i,G})$,
                THEN
                  $U_{i,G}=W_{i,G}$,
                END IF
            END WHILE
      END FOR
    Step 4.4 Selection step
            IF
              $f(U_{i,G})\le f(X_{i,G})$
            THEN
              $X_{i,G+1}=U_{i,G}, f(X_{i,G+1})=f(U_{i,G})$
            IF
              $f(U_{i,G})<f(X_{best,G})$
            THEN
              $X_{best,G}=U_{i,G}, f(X_{best,G})=f(U_{i,G})$/* $X_{best,G}$ is the best individual in
  generation $G$ */
            ELSE
              $X_{i,G+1}=X_{i,G}$
            END IF
      END IF
    Step 4.5 Re-build the surrogate model
    Step 4.6 Increment the generation count $G=G+1$
END WHILE
```

combination of strategies and their associated parameter values depending on the nature of the problem and the evolutionary stage can be solved with the assistance of the surrogate model.

In each generation, the surrogate model is constructed with the available population members. Therefore, there is no need for extra memory as compared to some of the surrogate assisted models [13], where the populations members corresponding to earlier generations are stored.

In this paper, the ESMDE algorithm uses the following mutation and crossover strategies:

1) Mutation strategies: DE/rand/1 and DE/current-to-rand/1 [20]
2) Crossover strategies: Binomial crossover and exponential crossover [29]

In the proposed algorithm, the population size ($NP=50$) is maintained constant throughout the evolutionary process. The parameter c is set to 10. The description of the proposed algorithm is depicted in Table 2.

4. Experimental results

In this section, we evaluated and compared the performance of the proposed algorithm with state-of-the-art self-adaptive DE algorithms present in the literature, such as JADE [16], CoDE [17], SaDE [5], and EPSDE [11]. To evaluate the performances of the algorithms,

Table 3
The performance results of different algorithms on 10*D* test functions.

Fcn	10*D*													
	JADE			CoDE			EPSDE			SaDE			ESMDE	
	Mean	Std	h	Mean	Std	h	Mean	Std	h	Mean	Std	h	Mean	Std
f_1	1.15E−43	3.26E−43	1	1.44E−58	4.52E−58	1	1.68E−98	4.51E−98	1	2.48E−101	1.20E−100	1	**5.93E−155**	**1.40E−154**
f_2	1.26E+00	2.14E+00	1	1.73E−26	4.67E−26	1	**0**	**0**	0	1.33E−01	7.28E−15	1	**0**	**0**
f_3	3.55E−15	0	0	3.43E−15	6.49E−16	0	1.42E−15	1.77E−15	0	1.30E−15	1.74E−15	0	3.19E−15	1.12E−15
f_4	5.50E−12	1.21E−11	1	**0**	**0**	0	2.54E−04	1.35E−03	1	**0**	**0**	0	**0**	**0**
f_5	0	0	0		0	0	0	0	0	0	0	0	0	0
f_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_7	3.95E+00	2.16E+01	1	**0**	**0**	0	7.90E+00	3.01E+01	1	**0**	**0**	0	**0**	**0**
f_8	2.85E−59	1.48E−58	1	1.63E−34	3.54E−34	1	3.37E−74	4.88E−74	1	8.72E−77	4.77E−76	1	**2.99E−88**	**5.73E−88**
f_9	1.30E−53	2.30E−53	1	3.40E−30	1.04E−29	1	1.05E−65	3.05E−65	1	1.15E−64	4.74E−64	1	**2.39E−78**	**6.38E−78**
f_{10}	7.07E−39	1.96E−38	1	3.94E−56	5.27E−56	1	2.21E−95	5.11E−95	1	5.60E−99	1.50E−98	1	**7.87E−152**	**4.68E−152**
f_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_{12}	3.68E−22	6.48E−22	1	1.47E−31	1.12E−31	1	4.13E−49	8.94E−49	1	4.03E−52	6.84E−52	1	**2.73E−77**	**1.81E−77**
f_{13}	2.88E−24	4.90E−24	1	5.33E−17	6.53E−17	1	7.93E−22	6.90E−22	1	3.87E−27	7.52E−27	1	**1.54E−34**	**4.72E−35**
f_{14}	1.57E−32	5.57E−48	0	1.57E−32	5.57E−48	0	1.57E−32	5.57E−48	0	1.57E−32	5.57E−48	0	1.57E−32	2.88E−48
f_{15}	1.35E−32	5.57E−48	0	1.35E−32	5.57E−48	0	1.35E−32	5.57E−48	0	1.35E−32	5.57E−48	0	1.35E−32	2.88E−48
f_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_{17}	6.92E+01	1.76E+02	1	2.59E+00	5.60E+00	1	1.25E+02	3.17E+02	1	6.85E+00	6.73E+00	1	**0**	**0**
–		10/7/0			8/9/0			9/8/0			8/9/0			–

The bold items refer to the algorithm that outperforms the other algorithms on a statistically significant basis.

we used the 10D and 30D versions of the test problems presented in Table 6. Details of the problems are presented in the Appendix. The maximum number of function evaluations used was 100,000 and 300,000 for the 10D and 30D problems, respectively. All of the experiments were run 30 times, independently, on each problem. The performances of the different algorithms were compared with that of the proposed algorithm using a statistical t-test with a significance level of 0.05, and the h values are presented. Numerical values $-1, 0, 1$ represent that the proposed ESMDE algorithm is inferior to, equal to, or superior to the algorithm under consideration, respectively.

Population size (NP) was set at 100, 30, 50, and 50 in the JADE, CoDE, SaDE, and EPSDE methods, respectively, for both the 10D and 30D problems. The remaining parameters were similar to the settings found in the original publications [5,11,16,17]. In the ESMDE method, NP was set at 50 for both the 10D and 30D problems. To construct the Kriging surrogate model, we used a zero order polynomial regression function and a Gaussian correlation function.

The results of the algorithms on the 10D and 30D problems are summarized in terms of mean and standard deviation (std.) values in Tables 3 and 4, respectively. The statistical t-test results are also presented in Tables 3 and 4. Figs. 2–14 show the convergence characteristics of the different algorithms for some of the test functions.

From Tables 3 and 4, on both the 10D and 30D versions of $f_1, f_3, f_5, f_8, f_{10}, f_{12}, f_{13}, f_1$, and, f_{15}, the proposed ESMDE showed superior performance. The improved performance of the proposed ESMDE algorithm was due to better convergence speed caused by use of the surrogate model. However, on these problems, all algorithms were able to find the basin of the global optima. This was also observed in the convergence plots in Figs. 5 and 6 where all the curves were able to converge to values less than 10^{-40}. However, the convergence speed of the algorithms varies.

Figs. 3 and 9 present the convergence characteristics of the different algorithms on the 10D and 30D versions of problem f_2. From the convergence plots, it is evident that the convergence speeds of the SaDE and EPSDE algorithms were greater than that of the proposed ESMDE algorithm. However, in the results found in Tables 3 and 4, it is evident that the performance of the ESMDE algorithm was significantly better than those of the JADE, SaDE, and EPSDE algorithms. Thus, the SaDE and EPSDE algorithms showed a speedy convergence but could not find the global optima with consistency as compared to the ESMDE algorithm. From the statistical results and Figs. 3 and 10, a similar kind of algorithmic performance can be observed on problem f_4. However, unlike on f_2, the convergence characteristics of the ESMDE method were greater than those of the other algorithms on f_4. Unlike the EPSDE and SaDE algorithms, the performance of the CoDE method was consistent on f_2 and f_4 but showed an inconsistency on problem f_6, as observed in Fig. 11.

On problem f_7, the performances of the SaDE, CoDE, and ESMDE algorithms were statistically similar and better than those of JADE and EPSDE, which were unable to consistently find the global optima. The premature convergence of the EPSDE algorithm can be observed in Fig. 12 and may be due to the population members sticking to a combination of strategies and parameter values too early in the evolutionary process. On problem f_9, the performances of the CoDE and SaDE algorithms were inconsistent when compared to those of JADE, EPSDE, and ESMDE. Except on the 30D version, where the SaDE method failed, the performances of all algorithms were statistically similar on the 10D and 30D versions of f_{11}.

The ESMDE method showed improved performance compared to those of the other algorithms on the 10D versions of f_{16} and f_{17} (Table 3 and Figs. 7 and 8). However, the ESMDE algorithm did not

Table 4
The performance results of different algorithms on 30D test functions.

Fcn	30D														
	JADE			CoDE			EPSDE			SaDE			ESMDE	
	Mean	Std	h	Mean	Std	h	Mean	Std	h	Mean	Std	h	Mean	Std
f_1	5.30E−123	2.85E−122	1	7.20E−61	1.72E−60	1	4.26E−122	7.61E−122	1	2.12E−137	4.94E−137	1	**6.83E−160**	**6.65E−160**
f_2	5.31E−01	1.38E+00	1	1.15E−07	3.89E−07	1	3.99E−01	1.22E+00	1	9.30E−01	1.72E+00	1	**4.45E−13**	**5.11E−13**
f_3	3.55E−15	0	0	3.55E−15	0	0	3.55E−15	0	0	3.55E−15	0	0	3.55E−15	0
f_4	**0**	**0**	0	**0**	**0**	0	9.04E−04	2.78E−03	1	3.53E−03	7.01E−03	1	**0**	**0**
f_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_6	**0**	**0**	0	3.45E−05	1.06E−04	1	**0**	**0**	0	**0**	**0**	0	**0**	**0**
f_7	3.95E+00	2.16E+01	1	**0**	**0**	0	1.59E+02	1.41E+02	1	**0**	**0**	0	**0**	**0**
f_8	2.27E−48	1.17E−47	1	1.89E−16	2.81E−16	1	2.88E−29	1.15E−28	1	6.85E−22	2.92E−21	1	**3.62E−65**	**9.29E−65**
f_9	3.05E−14	1.59E−13	1	1.06E−03	2.31E−03	1	1.19E−08	3.29E−08	1	6.68E−01	1.22E+00	1	**1.37E−17**	**3.66E−17**
f_{10}	8.39E−108	4.60E−107	1	3.91E−58	8.73E−58	1	9.41E−118	4.70E−117	1	3.98E−133	1.12E−132	1	**1.27E−155**	**1.15E−155**
f_{11}	**0**	**0**	0	**0**	**0**	0	**0**	**0**	0	4.13E−03	1.47E−02	1	**0**	**0**
f_{12}	4.32E−31	1.58E−30	1	4.93E−31	6.90E−31	1	1.28E−62	1.32E−62	1	6.84E−75	1.17E−74	1	**2.68E−81**	**1.05E−81**
f_{13}	3.25E−39	5.61E−39	1	5.47E−11	3.75E−11	1	5.74E−13	4.62E−13	1	4.42E−23	1.51E−22	1	**3.42E−45**	**2.03E−45**
f_{14}	1.57E−32	5.57E−48	0	1.57E−32	5.57E−48	0	1.57E−32	5.57E−48	0	1.57E−32	5.58E−48	0	1.57E−32	2.88E−48
f_{15}	1.35E−32	5.57E−48	0	1.35E−32	5.57E−48	0	1.35E−32	5.57E−48	0	7.333E−04	2.79E−03	0	1.35E−32	2.88E−48
f_{16}	**8.50E−09**	**1.60E−08**	−1	5.04E+02	5.59E+02	1	4.62E+02	4.05E+02	1	1.80E+03	5.07E+02	1	5.44E+00	9.95E+00
f_{17}	7.42E+03	4.37E+03	1	2.71E+03	2.74E+03	0	9.65E+03	1.17E+04	1	**1.62E+03**	**2.25E+03**	−1	2.96E+03	2.08E+03
–		9/7/1			9/8/0			11/6/0			10/6/1			–

The bold items refer to the algorithm that outperforms the other algorithms on a statistically significant basis.

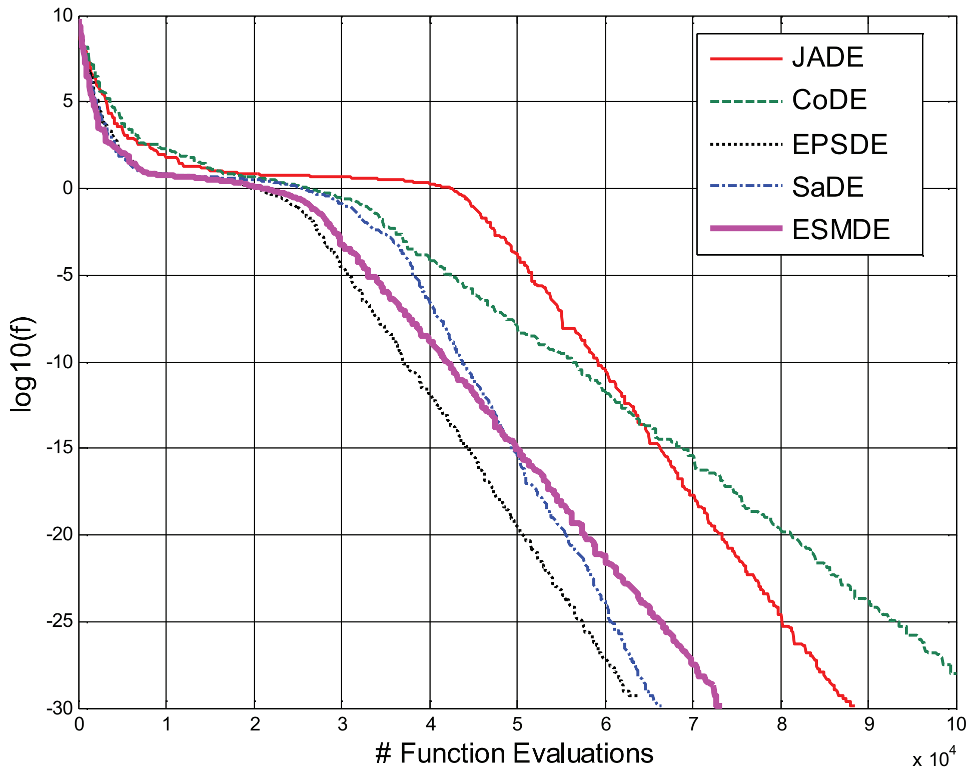


Fig. 2. Convergence Plot of f_2 (10D).

reveal a similarly consistent performance on the 30D versions of f_{16} and f_{17} (Table 4 and Figs. 13 and 14). On f_{16}, performance degradation was due to slow convergence, whereas on f_{17}, performance degradation was due to premature convergence. From the results and the discourse above, the improved performance of the ESMDE algorithm over other algorithms can be attributed to improved convergence speed and consistency in finding the global optima. The faster convergence speed and improved performance of the ESMDE method can be attributed to the surrogate model, which evolves with the population of individuals and assists in producing

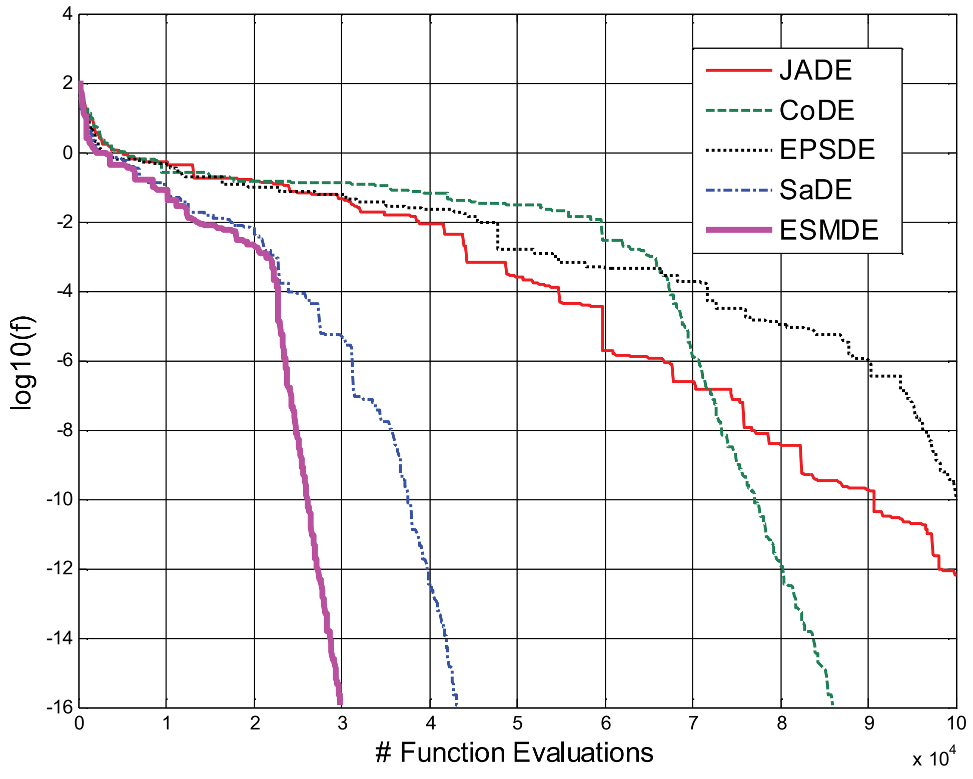


Fig. 3. Convergence Plot of f_4 (10D).

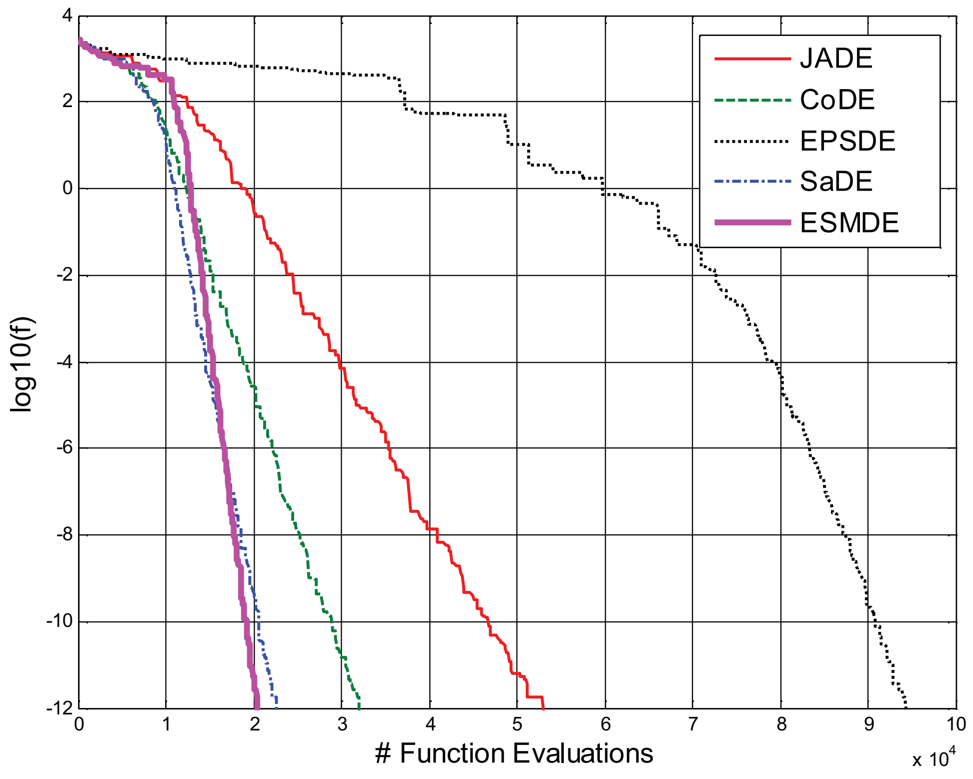


Fig. 4. Convergence Plot of f_7 (10*D*).

competitive offspring corresponding to each parent in every generation of the evolutionary process. However, on problems f_{16} and f_{17}, as dimensionality increased, the proposed ESMDE algorithm failed to show significant improvement due to slow convergence speed and inconsistency. As investigated in [14], the performance of the ESMDE method can be improved by employing a surrogate model that is suitable for high dimensional problems, such as radial basis functions.

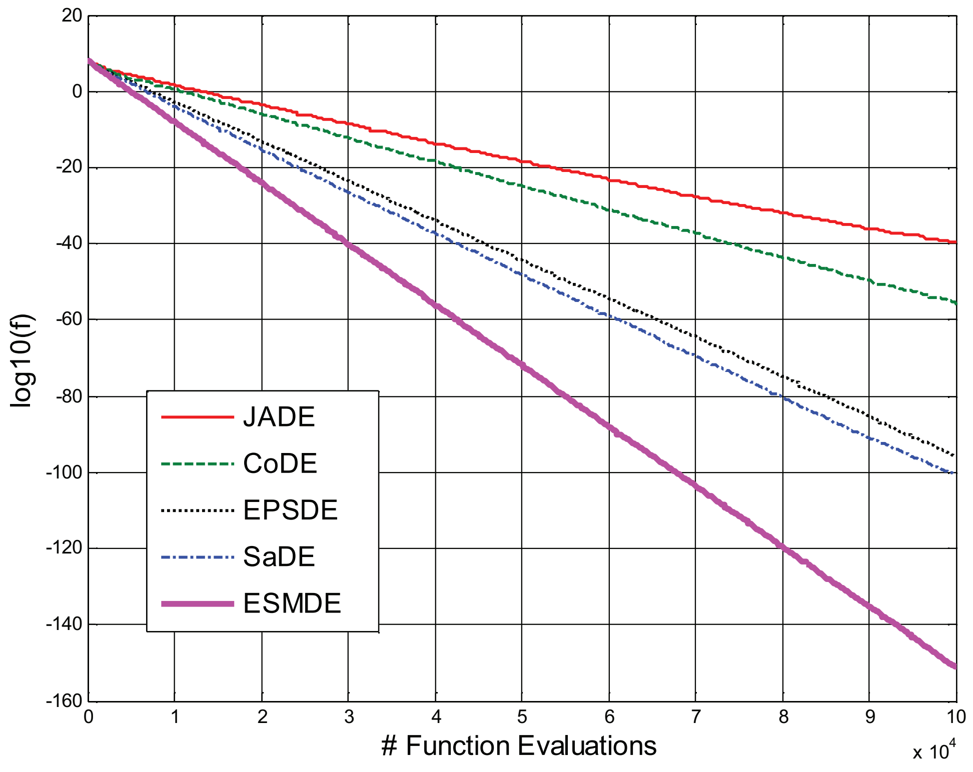


Fig. 5. Convergence Plot of f_{10} (10*D*).

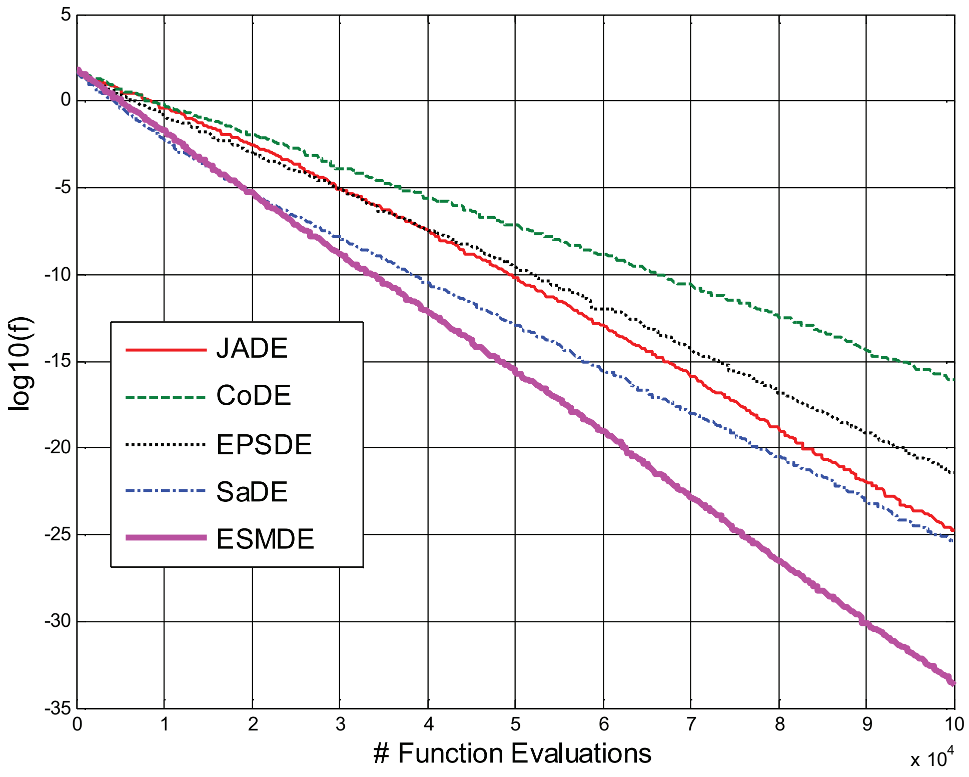


Fig. 6. Convergence Plot of f_{13} (10*D*).

From the results in Table 3, the proposed algorithm was statistically always better than or equal to the other algorithms on 10*D* problems. However, as shown in Table 4, on 30*D* problems, the proposed algorithm could not achieve statistically better or equal results on f_{16} and f_{17} compared to the JADE and SaDE methods, respectively. Degradation in performance of the proposed algorithm from 10*D* to 30*D* on problems f_{16} and f_{17} can be attributed to the selection of the surrogate model. As

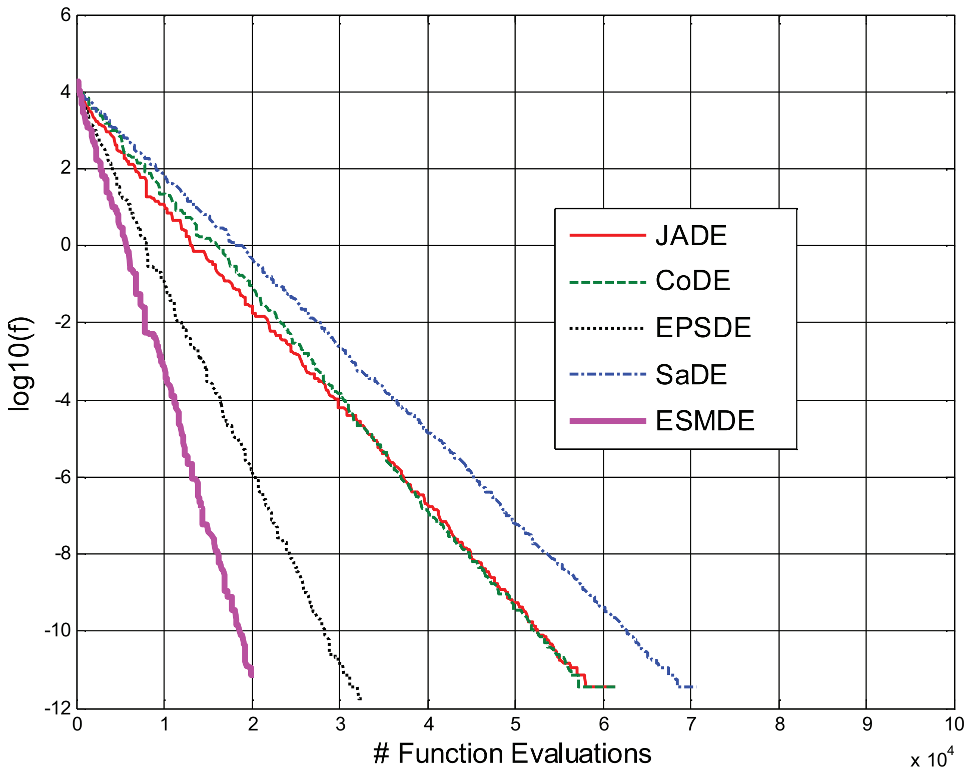


Fig. 7. Convergence Plot of f_{16} (10*D*).

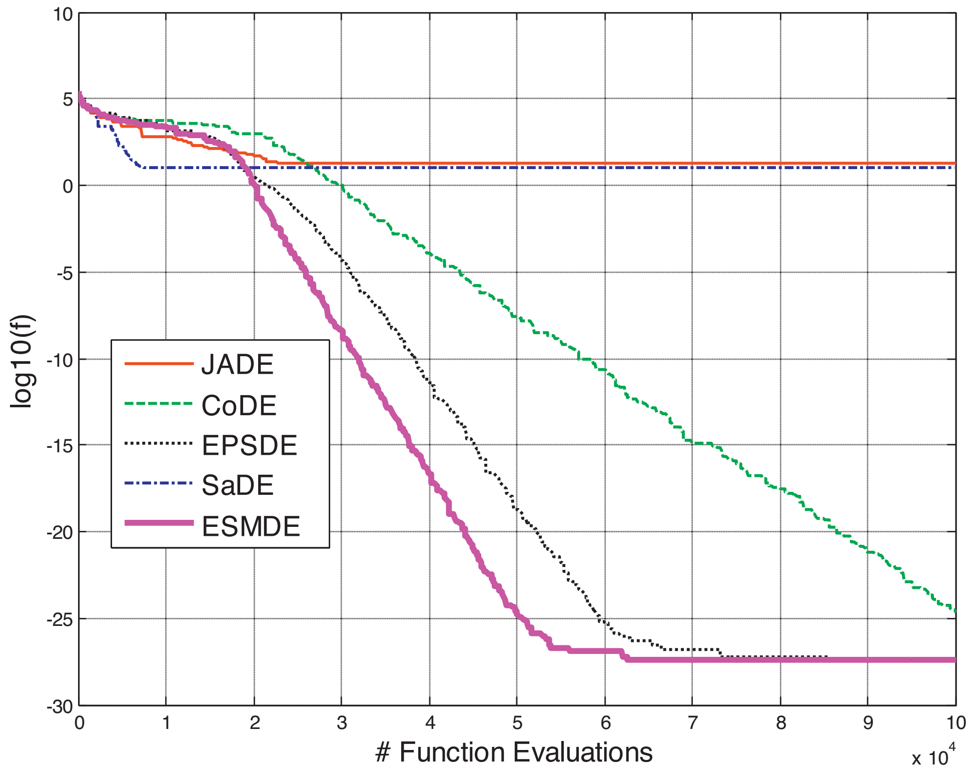


Fig. 8. Convergence Plot of f_{17} (10D).

mentioned by the authors in [14], by using a better surrogate model such as radial basis functions, which perform well on high dimensional problems, better results can be obtained on f_{16} and f_{17}.

We also compared the performance of the proposed algorithm with those of the SVR-DE, RankSVM-DE, and SVC-DE algorithms reported in [15]. For a subset of problems presented in Table 6, the results of the SVR-DE, RankSVM-DE, and SVC-DE algorithms

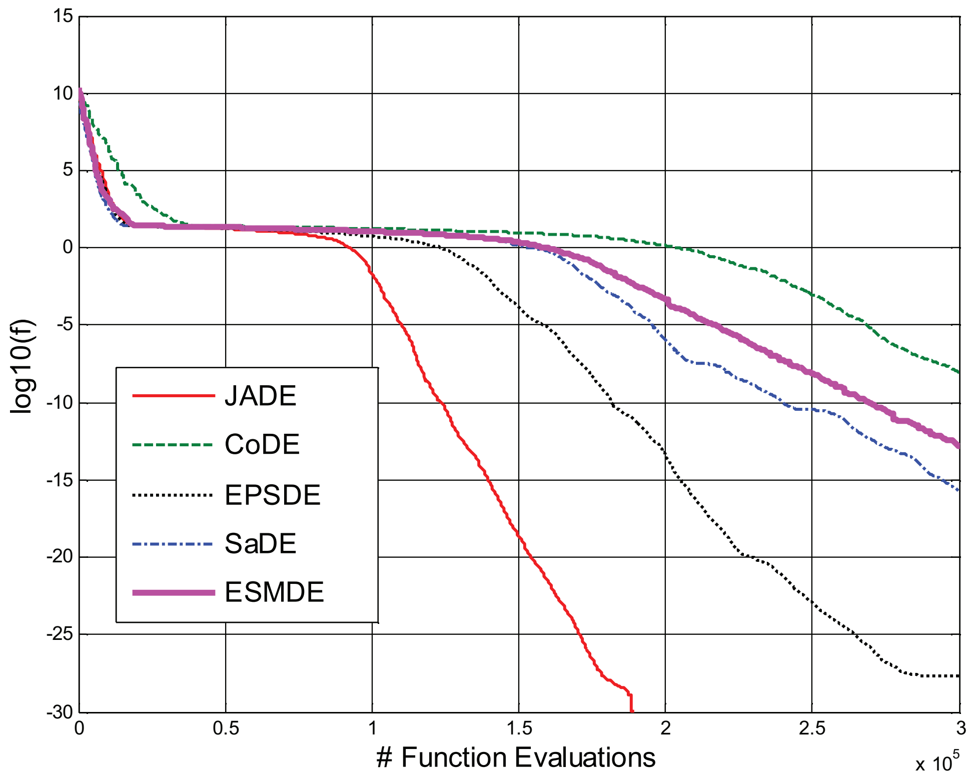


Fig. 9. Convergence Plot of f_2 (30D).

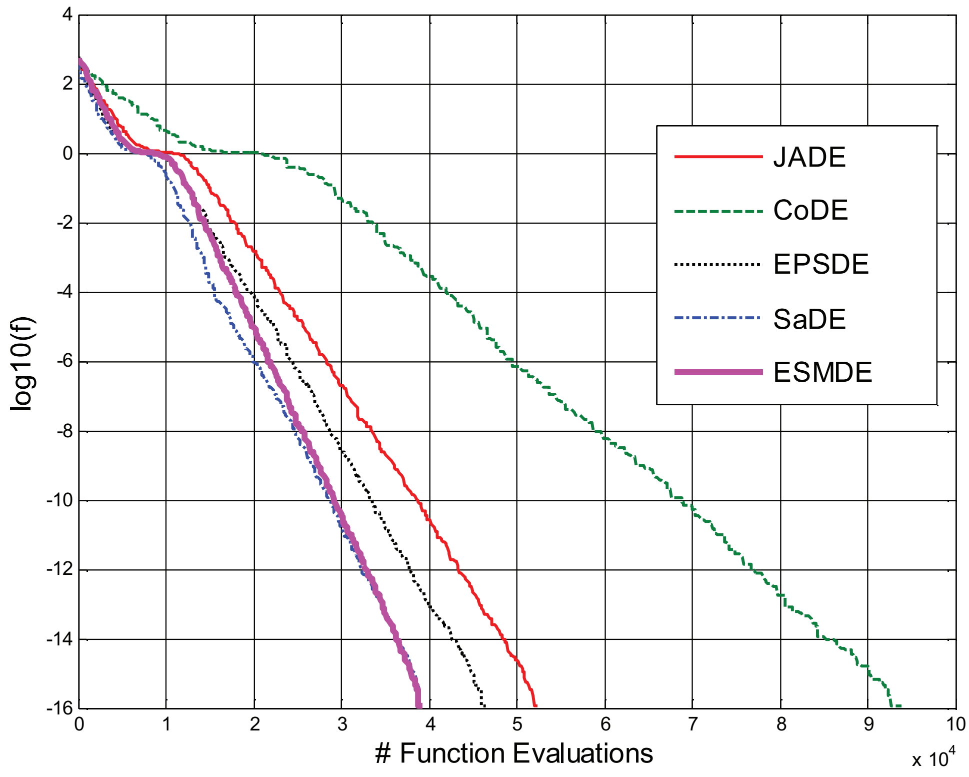


Fig. 10. Convergence Plot of f_4 (30D).

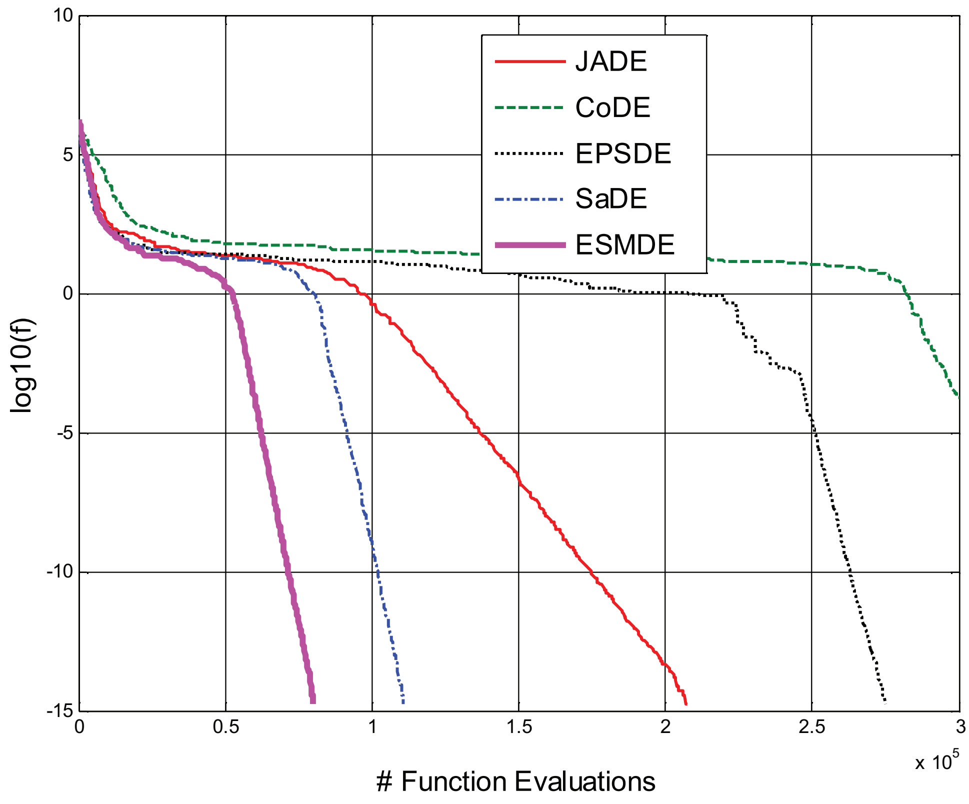


Fig. 11. Convergence Plot of f_6 (30D).

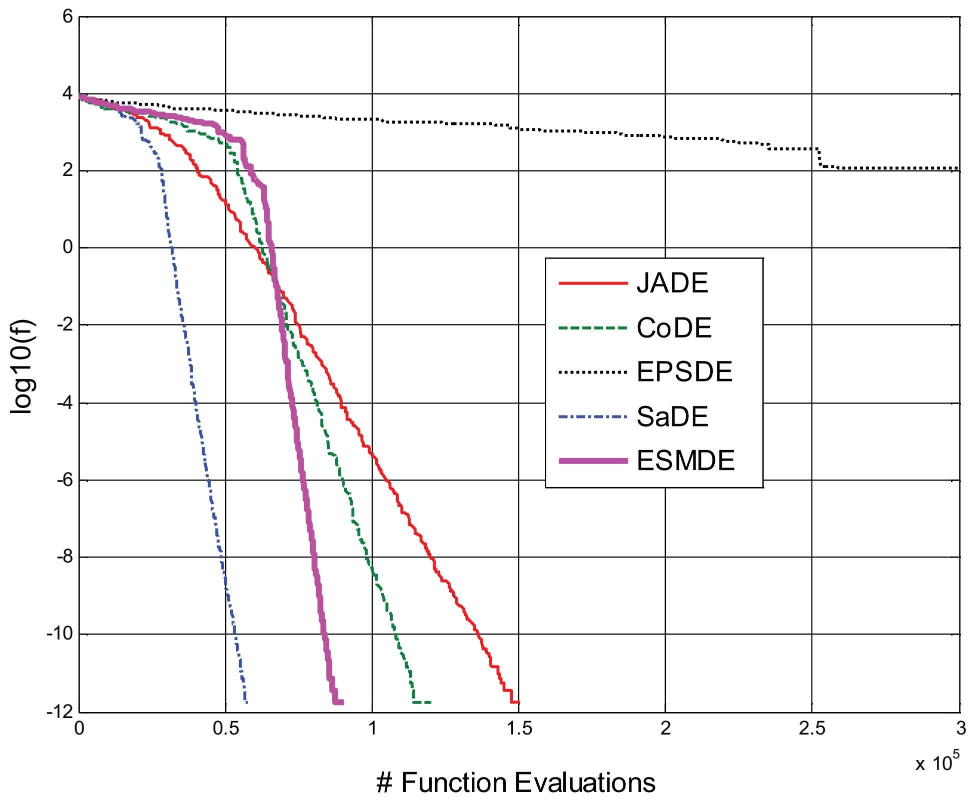


Fig. 12. Convergence Plot of f_7 (30D).

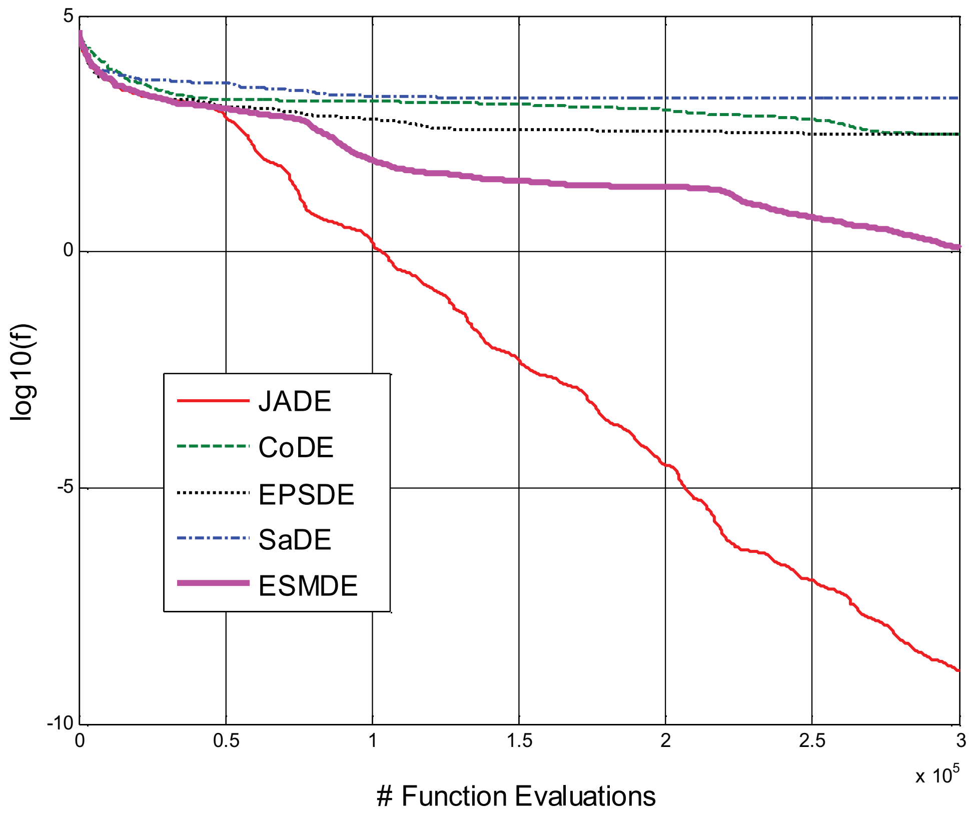


Fig. 13. Convergence Plot of f_{16} (30D).

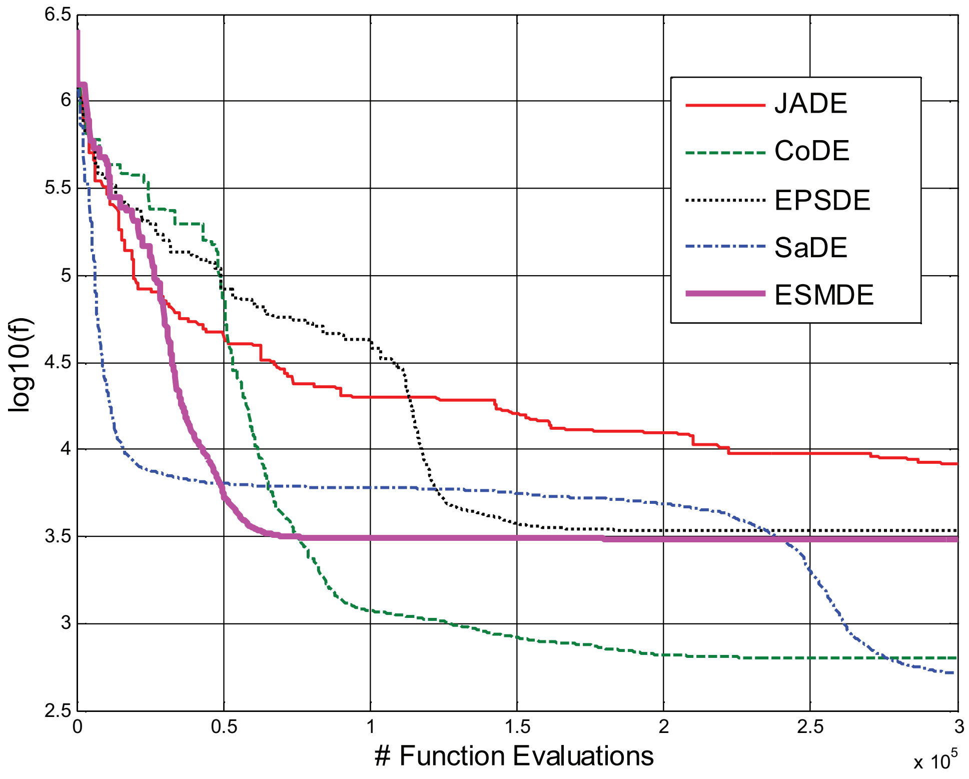


Fig. 14. Convergence Plot of f_{17} (30*D*).

Table 5
The comparison of ESMDE with SVR-DE, RankSVM-DE, and SVC-DE on 30*D* test functions.

Fcn	30*D*							
	SVR-DE		RankSVM-DE		SVC-DE		ESMDE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	6.32E−01	9.19E−02	**5.06E−02**	**1.38E−02**	1.07E−01	3.67E−02	2.91E−01	6.31E−02
f_2	2.32E+07	1.43E+07	1.49E+05	1.33E+05	2.54E+03	3.11E+03	**1.78E+03**	**1.54E+03**
f_5	2.01E+02	1.14E+01	2.18E+02	2.08E+01	2.09E+02	1.31E+01	**9.00E+01**	**6.25E+00**
f_8	1.64E+04	4.87E+03	2.26E+04	4.83E+03	**3.54E+03**	**1.33E+03**	1.65E+04	2.49E+03
f_9	2.70E+04	7.06E+03	3.03E+04	5.38E+03	**7.71E+03**	**2.77E+03**	3.07E+04	3.90E+03
f_{16}	**2.24E+03**	**5.69E+02**	5.96E+03	6.83E+02	2.39E+03	5.71E+02	5.89E+03	7.64E+02

The bold items refer to the algorithm that outperforms the other algorithms on a statistically significant basis.

Table 6
The test functions.

Function		$f(x^*)$	Search range
f_1	Sphere Function [47]	0	$[-100, 100]^D$
f_2	Generalized Rosenbrock Function [47]	0	$[-100, 100]^D$
f_3	Ackley function [47]	0	$[-32, 32]^D$
f_4	Generalized Griewank Function[47]	0	$[-600, 600]^D$
f_5	Generalized Rastrigin Function [47]	0	$[-5.12, 5.12]^D$
f_6	Non-continuous Rastrigin Function [47]	0	$[-500, 500]^D$
f_7	Schwefel's Function 2.26 [47]	0	$[-500, 500]^D$
f_8	Schwefel's Problem 1.2 [47]	0	$[-100, 100]^D$
f_9	Schwefel's Problem 1.2 with Noise in Fitness [47]	0	$[-100, 100]^D$
f_{10}	High Conditioned Elliptic Function [47]	0	$[-100, 100]^D$
f_{11}	Weierstrass's Function [47]	0	$[-0.5, 0.5]^D$
f_{12}	Schwefel's Problem 2.22 [47]	0	$[-10, 10]^D$
f_{13}	Schwefel's Problem 2.21 [47]	0	$[-100, 100]^D$
f_{14}	Generalized Penalized Function 1 [47]	0	$[-50, 50]^D$
f_{15}	Generalized Penalized Function 2 [47]	0	$[-50, 50]^D$
f_{16}	Schwefel's Problem 2.6 with Global Optimum on Bounds [47]	0	$[-100, 100]^D$
f_{17}	Schwefel's Problem 2.13 [47]	0	$[-\pi, \pi]^D$

were taken from [15]. To compare the performance of the proposed algorithm with these algorithms, the maximum number of function evaluations given was set at 10,000, as mentioned in [15]. The results are summarized in Table 5, and the best results are highlighted. The RankSVM-DE and SVR-DE algorithms performed better than the other three algorithms on f_1 and f_{16} problems, respectively. The SVC-DE algorithm performed better than the other algorithms on f_8 and f_9 problems, whereas the ESMDE algorithm performed better on f_2 and f_5 problems. Therefore, from the results, the overall performance of the ESMDE method was comparable to those of the classification-based DE algorithms. However, classification-based DEs employ a computationally intensive database-building phase, which helps build the classification models. In other words, the classification-based DEs require a great deal of memory to store the previous population members and computational times compared to the proposed ESMDE algorithm, where the surrogate model is built with the population members of the current generation.

5. Conclusions

In this paper, we present an evolving surrogate model-assisted DE algorithm (ESMDE). This algorithm makes use of an evolving surrogate model, which helps generate competitive offspring corresponding to each population member in every generation of the evolution based on an appropriate combination of strategies and their associated parameter values. From the results, the performance of the ESMDE method is statistically similar or better than the state-of-the-art self-adaptive DE algorithms in terms of mean and standard deviation values. The improved performance of the proposed ESMDE algorithm is due to the faster convergence speed and better consistency in finding the global optima with the assistance of the surrogate model. In the proposed ESMDE, corresponding to each parent, a few number of trial vectors are generated using different combinations of strategies and parameter values. From the generated offspring members, a competitive offspring is selected based on the surrogate model evaluation. Hence, the proposed method does not require any trial and error approach to find the approximate combination of strategies and their associated parameter values.

The performance of the proposed algorithm depends on the accuracy of the employed surrogate model, and different surrogate models are effective for the different dimensionality and complexity requirements of the problem. The present work employs a surrogate model built using the Kriging technique. In the future, we would like to evaluate the performance of the proposed algorithm employing different surrogate modeling techniques. Our final goal is to develop a parameter-free DE algorithm using surrogate models to solve high dimensional problems with better convergence and consistency.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning (2013R1A2A2A01068687) (50%) and by the Industrial Strategic Technology Development Program (10044009) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) (50%).

Appendix.

(1) Sphere function [47]

$$f_1(x)=\sum_{i=1}^{D}x_i^2$$

(2) Generalized Rosenbrock's function [47]

$$f_2(x)=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$$

(3) Ackley function [47]

$$f_3(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$$

(4) Griewank's function [47]

$$f_4(x)=\sum_{i=1}^{D}\frac{{x_i}^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

(5) Rastrigin's function [47]

$$f_5(x)=\sum_{i=1}^{D}({x_i}^2-10\cos(2\pi x_i)+10)$$

(6) Non-continuous Rastrigin's function [47]

$$f_6(x)=\sum_{i=1}^{D}({y_i}^2-10\cos(2\pi y_i)+10)$$

$$y_i=\begin{cases}x_i & |x_i|<\frac{1}{2}\\ \frac{round(2x_i)}{2} & |x_i|>=\frac{1}{2}\end{cases}\quad i=1,2,..,D$$

(7) Schwefel's Function 2.26 [47]

$$f_8(x)=418.9829\times D-\sum_{i=1}^{D}x_i\sin(|x_i|^{1/2})$$

(8) Shifted Schwefel's Problem 1.2 [47]

$$f_8(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$$

(9) Shifted Schwefel's Problem 1.2 with Noise in Fitness [47]

$$f_9(x)=\left(\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2\right)(1+0.4|N(0,1)|)$$

(10) High Conditioned Elliptic Function [47]

$$f_{10}(x)=\sum_{i=1}^{D}(10^6)^{(i-1)/(D-1}x_i^2$$

(11) Weierstrass's Function [47]

$$f_{11}(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i+0.5))]\right)-D\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k.0.5)]$$
$$a=0.5,\quad b=3,\quad k_{\max}=20$$

(12) Schwefel's Problem 2.22 [47]

$$f_{12}(x)=\sum_{i=1}^{D}|x_i|+\prod_{i=1}^{D}|x_i|$$

(13) Schwefel's Problem 2.21 [47]

$$f_{13}(x)=\max_i\{|x_i|,\ 1\le i\le D\}$$

(14) Generalized Penalized Function 1 [47]

$$f_{14}(x)=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_D-1)^2\right\}+\sum_{i=1}^{D}u(x_i,10,100,4)$$

$$y_i=1+\frac{1}{4}(x_i+1)$$

$$u(x_i,10,100,4)=\begin{cases}k(x_i-a)^m, & x_i>a,\\ 0, & -a\le x_i\le a,\\ k(-x_i-a)^m, & x_i<-a.\end{cases}$$

(15) Generalized Penalized Function 2 [47]

$$f_{15}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_D-1)[1+\sin^2(2\pi x_D)]\right\}+\sum_{i=1}^{D}u(x_i,5,100,4)$$

(16) Schwefel's Problem 2.6 with Global Optimum on Bounds [47]

$$f_{16}(x)=\max_i\{|A_i x-B_i|\},\quad i=1,\ldots,D$$

A is a $D*D$ matrices, a_{ij} are integer random numbers in the range [−500,500], det(A) $\neq 0$

A_i is the ith row of A. $B_i=A_i*$o, o is a $D*1$ vector, o$_i$ are random number in the range [−100,100]

(17) Schwefel's Problem 2.13 [47]

$$f_{17}(x)=\sum_{i=1}^{D}(A_i-B_i(x))^2$$

$$A_i=\sum_{j=1}^{D}(a_{ij}\sin\alpha_j-b_{ij}\cos\alpha_j),$$

$$B_i(x)=\sum_{j=1}^{D}(a_{ij}\sin x_j-b_{ij}\cos x_j),\quad i=1,\ldots,D$$

A, B are two $D*D$ matrices, a_{ij}, b_{ij} are integer random numbers in the range [−100,100]

$\alpha=[\alpha_1,\alpha_2,\ldots,\alpha_D]$, α_j are random numbers in the range [−π,π]

References

[1] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[2] M.K. Venu, R. Mallipeddi, P.N. Suganthan, Fiber Bragg grating sensor array interrogation using differential evolution, Optoelectron. Adv. Mater. – Rapid Commun. 2 (2008) 682–685.
[3] J. Liu, J. Lampinen, On setting the control parameter of the differential evolution method, in: Proc. 8th Int. Conf. Soft Computing, 2002, pp. 11–18.
[4] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (February) (2011) 4–31.
[5] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (April) (2009) 398–417.
[6] J. Brest, S. Greiner, B. Boscovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (December) (2006) 646–657.
[7] M.G.H. Omran, A. Salman, A.P. Engelbrecht, Self-adaptive differential evolution, in: Computational Intelligence and Security, Lecture Notes in Artificial Intelligence, 2005, pp. 192–199.
[8] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: Proceedings of the 9th International Conference on Soft Computing, Brno, 2003, pp. 41–46.
[9] J. Tvrdik, Adaptation in differential evolution: a numerical comparison, Appl. Soft Comput. 9 (June) (2009) 1149–1155.
[10] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (March) (2011) 1679–1696.
[11] R. Mallipeddi, P.N. Suganthan, Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies, in: Swarm Evolutionary and Memetic Computing Conference, Chennai, India, 2010.
[12] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (2003) 3–12.
[13] J. Zhang, A.C. Sanderson, DE-AEC: a differential evolution algorithm based on adaptive evolution control, in: IEEE Congress on Evolutionary Computation, 2007, pp. 3824–3830.
[14] A. Diaz-Manriquez, G. Toscano-Pulido, W. Gomez-Flores, On the selection of surrogate models in the evolutionary optimization algorithms, in: IEEE Congress on Evolutionary Computation, 2011, pp. 2155–2162.
[15] X. Lu, K. Tang, X. Yao, Classification-assisted differential evolution for computationally expensive problems, in: 2011 IEEE Congress on Evolutionary Computation (CEC), 2011, pp. 1986–1993.
[16] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (October) (2009) 945–958.
[17] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (February) (2011).
[18] R. Storn, On the usage of differential evolution for function optimization, in: Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), Berkeley, 1996.
[19] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, Berlin, 2005.
[20] A. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: Australian Conference on Artificial Intelligence, Cairns, Australia, 2004, pp. 861–872.
[21] R. Gämperle, S.D. Müller, P. Koumoutsakos, A parameter study for differential evolution, in: Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, Interlaken, Switzerland, 2002.
[22] S. Das, A. Konar, U.K. Chakraborty, Two improved differential evolution schemes for faster global search, in: Conference on Genetic and Evolutionary Computation, 2005, pp. 991–998.
[23] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: 6th International Mendel Conference on Soft Computing, 2000, pp. 76–83.
[24] R. Storn, K. Price, Differential evolution: a simple evolution strategy for fast optimization, Dr. Dobb's J. 22 (April) (1997) 18–24.
[25] J. Rönkkönen, S. Kukkonen, K.V. Price, Real-parameter optimization with differential evolution, in: IEEE Congress on Evolutionary Computation, 2005, pp. 506–513.
[26] T. Chen, K. Tang, G. Chen, X. Yao, A large population size can be unhelpful in evolutionary algorithms, Theor. Comput. Sci. 436 (2012) 54–70.
[27] K.V. Price, An Introduction to Differential Evolution, McGraw-Hill, London (UK), 1999.
[28] S. Das, A. Abraham, U.K. Chakraborthy, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (June) (2009) 526–553.
[29] D. Zaharie, Influence of crossover on the behavior of differential evolution algorithms, Appl. Soft Comput. 9 (June) (2009) 1126–1138.
[30] E. Mezura-Montes, J. Velazquez-Reyes, C.A. Coello Coello, Modified differential evolution for constrained optimization, in: IEEE Congress on Evolutionary Computation, 2006, pp. 25–32.
[31] D. Zaharie, Critical values for the control parameters of differential evolution, in: Proceedings of MENDEL 2002, Eighth International Conference on Soft Computing, Brno, Czech Republic, 2002, pp. 62–67.
[32] J. Rönkkönen, J. Lampinen, On using normally distributed mutation step length for the differential evolution algorithm, in: Proceedings of MENDEL 2003,Ninth International MENDEL Confernce on Soft Computing, Brno, Czech Republic, 2003, pp. 11–18.
[33] U.K. Chakraborthy, S. Das, A. Konar, Differentail evolution with local neighborhood, in: Proceedings of Congress on Evolutionary Computation, Vancouver, BC, Canada, 2006, pp. 2042–2049.
[34] H.A. Abbass, The self-adaptive pareto differential evolution algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, 2002, pp. 831–836.
[35] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput. 9 (June) (2005) 448–462.

[36] D. Zaharie, D. Petcu, Adaptive Pareto differential evolution and its parallelization, in: Proc. of 5th International Conference on Parallel Processing and Applied Mathematics, Czestochowa, Poland, 2003, pp. 261–268.
[37] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, Soft Comput. 10 (2006) 673–686.
[38] Y.S. Ong, P.B. Nair, A.J. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA J. 41 (April) (2003) 687–696.
[39] W. Carpenter, J.F. Barthelemy, A comparison of polynomial approximation and artificial neural nets as response surface AIAA Technical Report, vol. 92–2247, 1992.
[40] T. Simpson, T. Mauery, J. Korte, F. Mistree, Comparison of response surface and Kriging models for multidisciplinary design optimization AIAA Technical Report, vol. 98–4755, 1998.
[41] J. Sacks, W.J. Welch, T.J. Mitchell, H.P. Wynn, Design and analysis of computer experiments, Stat. Sci. 4 (1989) 409–435.
[42] C.K.I. Williams, C.E. Rasmussen, Gaussian Processes for Regression, MIT Press, Cambridge, MA, 1996.
[43] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: IEEE Congress on Evolutionary Computation, Hong Kong, 2008, pp. 1110–1116.
[44] S.Z. Zhao, P.N. Suganthan, S. Das, Self-adaptive differential evolution with multi-trajectory search for large scale optimization, Soft Comput. (2010).
[45] C.K. Chow, S.Y. Yuen, An evolutionary algorithm that makes decision based on the entire previous search history, IEEE Trans. Evol. Comput. 15 (December) (2011) 741–769.
[46] Z. Zhou, Y.S. Ong, M.H. Lim, B.S. Lee, Memetic algorithm using multi-surrogates for computationally expensive optimization problems, Soft Comput. 11 (2007) 957–971.
[47] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Nanyang Technological University/KanGAL, IIT Kanpur, Singapore/India, May 2005.